SAPIENZA
UNIVERSITÀ DI ROMA

AI Methods for Interactive Systems: Implicit Behaviour Detection and Large Language Model–Based Design Evaluation

Faculty of Information Engineering, Informatics, and Statistics
Department of Computer Science
PhD in Computer Science, Cycle XXXVIII

Stefano Zeppieri
ID number 1793449

Advisor
Prof. Emanuele Panizzi

Academic Year 2025/2026

AI Methods for Interactive Systems: Implicit Behaviour Detection and Large Language Model–Based Design Evaluation

Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: zeppieri@di.uniroma1.it

Abstract

Understanding and improving how people interact with digital systems requires both reliable signals about user context and practical methods for evaluating interface quality. This thesis investigates how two complementary forms of intelligence, implicit sensing and Large Language Models (LLMs), can support these goals across real-time interaction and design-time analysis.

Part I focuses on *AI for interaction*, studying implicit interaction mechanisms in which smartphones act as sensing and inference devices that reduce the need for explicit user input. The thesis introduces and evaluates mobile sensing techniques capable of recognizing car-related events and behaviors, including parking and unparking transitions and cruising-for-parking patterns, under constraints of deployability and low resource consumption. It further explores how short-range wireless cues, such as Bluetooth Low Energy, can complement motion and location traces to enable context-aware support in mobility settings. Together, these contributions show how implicit sensing can reduce user effort and enable proactive assistance, while also surfacing the interaction challenges introduced by uncertainty and misclassification.

Part II examines LLMs for interaction engineering and design evaluation. Rather than treating LLMs as general-purpose chat interfaces, the thesis studies them as components within interactive pipelines that generate inspectable intermediate artifacts. Through a set of empirical and methodological studies, it investigates LLM-supported evaluation (including walkthrough-inspired critique generation and adaptations of heuristic evaluation for proactive and implicit systems), LLM adoption in early-stage need finding, and the automation of structured tasks such as transforming unstructured descriptions into form-ready input. The findings highlight a recurring duality: LLMs can accelerate work and produce useful, context-sensitive outputs, but they can also introduce plausible yet incorrect content and inconsistent reasoning. The thesis therefore treats fallibility as a design constraint, examining workflows and interaction patterns that make uncertainty visible, support verification, and enable effective correction and repair while preserving user control and trust.

Overall, this thesis presents a unified view of AI-augmented interaction that spans low-level sensing of human behavior and high-level reasoning about interfaces and user tasks. It argues that the practical value of AI in interactive systems depends less on fully eliminating errors and more on designing mechanisms that keep humans informed, in control, and able to recover when automation fails.

Contents

List of Figures xi

List of Tables xiii

1 Introduction 1
- 1.1 AI for interaction: implicit interaction methods and smart mobility case studies . . . 2
- 1.2 LLMs for interaction engineering and design evaluation . . . 3
- 1.3 Research goals and contributions . . . 4
- 1.4 Thesis structure . . . 5
- 1.5 Declaration of original work and publications . . . 6
  - 1.5.1 Publications related to Part I: AI for interaction . . . 6
    - 1.5.1.1 Publications about Implicit Interaction in the parking domain . . . 6
    - 1.5.1.2 Publications about Implicit Interaction supported through BLE . . . 7
    - 1.5.1.3 Indoor Localization Using Large Language Models . 7
    - 1.5.1.4 Publications about Supporting User Input with LLMs 8
  - 1.5.2 Publications related to Part II: LLMs for interaction engineering and design evaluation . . . 8
    - 1.5.2.1 Designing for Fallible Automation . . . 8
    - 1.5.2.2 Enhancing Interface Design with AI . . . 8
    - 1.5.2.3 Large Language Models Adoption in Need Finding . 8
    - 1.5.2.4 Heuristic Evaluation of Implicit Interactions in Proactive Systems . . . 9
    - 1.5.2.5 Transforming Interactive Systems with Large Language Models . . . 9
    - 1.5.2.6 Memory Augumented Generation . . . 9
  - 1.5.3 Technology Transfer and Entrepreneurship: Patent and University Startup . . . 9
    - 1.5.3.1 Patent: autonomous localization for privacy-preserving contexts. . . . 9
    - 1.5.3.2 University startup: NITSY s.r.l. and user-centered digital processes. . . . 10

I AI for Interaction 11

2 Implicit Interaction in the parking domain 15
2.1 Introduction . . . 15
2.2 Related Works . . . 16
2.2.1 Smart parking tasks and interfaces . . . 16
2.2.2 Automatic detection of parking and unparking events . . . 17
2.2.3 Cruising-for-parking detection . . . 18
2.2.4 Summary and positioning . . . 19
2.3 Implicit Interaction Approach for Car-related Tasks On Smartphone Applications . . . 20
2.3.1 Design of the implicit interaction . . . 21
2.3.1.1 Definitions . . . 22
2.3.2 Implementation . . . 23
2.3.2.1 Flow and Architecture . . . 23
2.3.2.2 Approaching the Parking Location . . . 24
2.3.3 Discussion . . . 24
2.4 Publications on Implicit Interaction Approach for Car-related Tasks On Smartphone Applications - A Demo . . . 26
2.4.1 The Prototype . . . 26
2.4.1.1 User Interface . . . 27
2.4.2 Demo . . . 27
2.4.3 Discussion . . . 28
2.5 Cruising-for-Parking Detection on the Smartphone Based on Implicit Interaction and Machine Learning . . . 30
2.5.1 Approach . . . 31
2.5.1.1 Cruising For Parking . . . 31
2.5.1.2 Cruising for parking indicators . . . 32
2.5.1.3 Double cross . . . 33
2.5.1.4 Time in range . . . 33
2.5.1.5 Previous Parking Spots . . . 34
2.5.2 Experiment . . . 36
2.5.2.1 Collection of Data . . . 36
2.5.2.2 Data labelling . . . 37
2.5.2.3 Ground truth . . . 37
2.5.2.4 Evaluation of the Heuristics . . . 39
2.5.3 Machine Learning Models . . . 40
2.5.3.1 Defining the Cruising for Parking Problem . . . 40
2.5.3.2 Observations on collected data . . . 40
2.5.3.3 Adopted Machine Learning Model . . . 41
2.5.4 Final Cruising for Parking Detection System . . . 42
2.5.4.1 Thresholds and Final System . . . 42
2.5.4.2 Final Cruising Detection System Evaluation . . . 43
2.5.5 Discussion . . . 44
2.6 An approach to leverage Artificial Intelligence for car-parking related mobile applications . . . 46
2.6.1 Engineering . . . 47

2.6.2 Approach . . . 47
2.6.2.1 Identifying the task . . . 47
2.6.2.2 Identifying the context . . . 48
2.6.2.3 Gather Data . . . 49
2.6.2.4 Model training and distribution . . . 50
2.6.2.5 Release, feedback, and Human-AI collaboration . . . 51
2.6.3 Updating models . . . 52
2.6.4 Discussion . . . 53

3 Implicit Interaction supported through BLE **55**
3.1 Introduction . . . 55
3.2 Related Works . . . 56
3.2.1 Inertial and motion-based TMD . . . 56
3.2.2 Location and trajectory-based approaches . . . 57
3.2.3 Sensor fusion and non-traditional modalities . . . 57
3.2.4 BLE visibility for co-presence and context-aware interaction . . . 58
3.2.5 Summary and positioning . . . 58
3.3 Towards Context-Aware UX in Automated Mobility: BLE Based Passenger Detection via Smartphones . . . 59
3.3.1 Methodology and Data Collection . . . 59
3.3.2 Model and Evaluation . . . 61
3.3.2.1 Feature Design and Labeling . . . 61
3.3.2.2 Model Performance . . . 61
3.3.2.3 Average Performance Metrics . . . 62
3.3.2.4 Interface Integration and Feedback . . . 63
3.3.2.5 Key Takeaways . . . 63
3.3.3 User Interface Design . . . 64
3.3.4 Human Automated Vehicle Use Cases . . . 65
3.3.5 Discussion . . . 67
3.4 Detecting Human Presence via Smartphone BLE Beaconing: Preliminary Investigations . . . 70
3.4.1 Methodology . . . 71
3.4.2 Preliminary Observations & Challenges . . . 71
3.4.3 Applications & Future Directions . . . 72

4 Indoor Localization Using Large Language Models **75**
4.1 Introduction . . . 75
4.2 Related Works . . . 76
4.3 Methodology . . . 76
4.3.1 Study Setup . . . 76
4.3.2 Study Locations and Evacuation Maps . . . 77
4.3.3 LLM Selection and Prompt . . . 77
4.3.4 Data Collected throughout the study . . . 80
4.4 Study Outcomes . . . 80
4.4.1 OU Study . . . 80
4.4.2 Sapienza Study . . . 81
4.5 Discussion . . . 82

5 Supporting User Input with LLMs 85
5.1 Introduction . . . 85
5.2 Related Works . . . 86
5.2.1 From rule-based assistance to LLM-driven structured input . 86
5.2.2 Human–AI interaction principles and mixed-initiative workflows 87
5.2.3 Evidence from deployment-oriented studies and application domains . . . 87
5.2.4 Gaps and positioning of our work . . . 88
5.3 Engineering Large Language Model Agents for Transforming Unstructured Descriptions into Structured Input . . . 89
5.3.1 Methodology . . . 90
5.3.1.1 AI Agent Design . . . 90
5.3.1.2 Integration with Existing Infrastructures . . . 91
5.3.1.3 Improving Accuracy and Handling Ambiguity . . . 91
5.3.1.4 User-Centered Evaluation . . . 91
5.3.2 Engineering and Modular Pipeline . . . 92
5.3.2.1 Pipeline Overview . . . 92
5.3.2.2 Agent Design and Prompt Logic . . . 94
5.3.2.3 Implementation and Feasibility Tests . . . 95
5.3.3 Discussion . . . 96
5.4 Automating Form Completion with Large Language Models . . . 98
5.4.1 System Overview . . . 98
5.4.2 Case Study and Evaluation . . . 98
5.4.3 Results . . . 99
5.4.4 Discussion . . . 99

II LLMs for interaction engineering and design evaluation 101

6 Designing for Fallible Automation 105
6.1 Introduction . . . 105
6.1.1 To err is AI . . . 106
6.1.1.1 When it is a good idea? . . . 106
6.1.1.2 When it is not a good idea? . . . 107
6.1.2 Driving examples . . . 107
6.1.3 Chapter Outline . . . 108
6.2 Related Works . . . 109
6.2.1 Levels of automation . . . 109
6.2.2 Intelligent interactions . . . 109
6.2.3 Errors in human-AI systems . . . 110
6.2.4 Human and system error and repair . . . 111
6.3 Define errors and failures . . . 113
6.3.1 Observable errors . . . 113
6.3.2 AI errors of commission and omission . . . 114
6.3.3 Users perception of errors, changing behavior and expectations 115
6.4 Error detection and repair to prevent failures . . . 117
6.4.1 Importance of detection . . . 117

6.4.2 Detection and Repair - who does what? . . . 117
6.5 Sensing errors . . . 118
6.5.1 Times and information . . . 119
6.5.2 Post-hoc estimation – traces and inconsistency . . . 120
6.5.3 Levels of ambiguity . . . 121
6.5.4 Using traces and ambiguity for post-hoc detection . . . 122
6.5.5 Dealing with sensor-layer failure . . . 124
6.6 Design implications . . . 124
6.6.1 Helping early detection . . . 124
6.6.2 Helping communicate . . . 125
6.6.3 System Reliability - when accuracy may become a problem . 125
6.6.3.1 Low reliability . . . 125
6.6.3.2 High reliability . . . 126
6.6.3.3 Responsibility for detection . . . 126
6.7 Discussion . . . 126

7 Enhancing Interface Design with AI 129
7.1 Introduction . . . 129
7.2 Related Works . . . 130
7.3 CWGPT . . . 131
7.3.1 CWGPT evaluation process . . . 131
7.4 Exploratory Study . . . 133
7.4.1 Comparison of human and CWGPT evaluations (RQ1) . . . 133
7.4.1.1 Methodology . . . 133
7.4.1.2 Results . . . 133
7.4.2 User study of CWGPT (RQ2) . . . 134
7.4.2.1 Methodology . . . 134
7.4.2.2 Results . . . 135
7.5 Discussion . . . 136

8 Large Language Models Adoption in Need Finding 139
8.1 Introduction . . . 139
8.2 Related Works . . . 140
8.3 Methodology . . . 141
8.4 An example of Direct prompts vs. LLM Assisted Need-finding . . . 142
8.5 Engineering . . . 145
8.5.1 Integrating Different Engineering Approaches . . . 145
8.5.2 Addressing Non-Deterministic Aspects of AI . . . 145
8.5.3 Ensuring Humans Remain in the Loop . . . 146
8.5.4 Guaranteeing Positive Aspects for Human Users . . . 147
8.5.5 Assessing Risk vs. Value in Specific Domains . . . 148
8.6 Discussion . . . 148

9 Heuristic Evaluation of Implicit Interactions in Proactive Systems 151
9.1 Introduction . . . 151
9.2 Related Works . . . 152
9.3 Implicit interaction with Proactive Agents . . . 153

9.4 Applicability of Nielsen's Heuristics . . . 155
9.4.1 Visibility of System Status . . . 155
9.4.2 Match Between the System and the Real World . . . 156
9.4.3 User Control and Freedom . . . 156
9.4.4 Consistency and Standards . . . 157
9.4.5 Error Prevention . . . 158
9.4.6 Recognition Rather than Recall . . . 159
9.4.7 Flexibility and Efficiency of Use . . . 159
9.4.8 Aesthetic and Minimalist Design . . . 160
9.4.9 Help Users Recognize, Diagnose, and Recover from Errors . . . 160
9.4.10 Help and Documentation . . . 161
9.5 Discussion . . . 161

10 Transforming Interactive Systems with Large Language Models 163
10.1 Introduction . . . 163
10.2 Related Works . . . 164
10.3 Methodology . . . 165
10.4 Ongoing Work . . . 167
10.5 Research Objectives, Contributions, and Dissertation Goals . . . 168
10.6 Discussion . . . 169

11 Memory Augmented Generation 171
11.1 Introduction . . . 171
11.2 Related Works . . . 172
11.3 Memory Taxonomy . . . 173
11.3.1 Conversational Memory . . . 174
11.3.2 Long-Term User Memory . . . 174
11.3.3 Episodic and Event-Linked Memories . . . 174
11.3.4 Sensory and Context-Aware Memory . . . 175
11.3.5 Short-Term Working Memory . . . 175
11.4 Coordination Across Memory Types . . . 176
11.4.1 Interaction Between Memory Modules . . . 176
11.4.2 Prioritization and Conflict Resolution . . . 177
11.5 Implementation Considerations . . . 177
11.6 User Studies . . . 179
11.6.1 Use Case . . . 179
11.6.2 Evaluation Strategies . . . 180
11.7 Discussion . . . 181

12 Conclusions 183
12.1 What this thesis contributed . . . 183
12.2 A unifying lesson: design for fallible automation . . . 184
12.3 Limitations . . . 185
12.4 Future work . . . 185
12.5 Closing remarks . . . 186

Bibliography 187

List of Figures

2.1 Prototype App 25
2.2 Prototype Interface 28
2.3 Detected user and vehicle status examples. 29
2.4 Heuristics Algorithms. The final step ("User is probably looking for a parking spot") can be triggered by any of the heuristics, even simultaneously. 35
2.5 Cruising for parking detection delays 44
2.6 A visualization of our learning loop 48
2.7 Example screens from the data collection app to recognize the user's means of transport. In this version, the app does not yet implement a recognition model. Therefore, the labeling of each trip is done manually by the user. On the left: "You are not in a vehicle". The user can assign a vehicle type to each of their previous trips. Center: "Select the vehicle". Upon selecting a trip from the history, the trace of collected GPS locations is shown to the user, along with the day and time it took place, as a reminder. Right: "You were in a car". The user labeled the collected data by selecting the means of transport. 50
2.8 Example screen from the released parking app. On the left: "Type of parking". The app predicted the type of parking "angle". On the right: the user can correct the prediction by selecting a different type of parking, "perpendicular", "angle", or "parallel". 51

3.1 Trip history interface showing mode classification, timestamps, and quick visual summaries. Translation: *"Non sei su un veicolo"* = "You are not in a vehicle", *"Auto"* = "Car", *"Bus"* = "Bus", *"Metro"* = "Subway", *"Tram"* = "Tram", *"A piedi"* = "On foot", *"Viaggi Precedenti"* = "Previous trips", *"Tutti"* = "All". Dates and times (e.g., *"martedì, 28 mag"*) indicate the day (Tuesday, 28 May). . . . 65
3.2 The user interface for viewing trip history, with a filter option to select different transportation modes 65
3.3 Completed trip view with density-based heatmap reflecting BLE activity along the route. Translation: *"Dispositivi"* = "Devices", *"venerdì, 19 apr"* = "Friday, 19 April", *"Eri sul bus"* = "You were on the bus", *"Auto"* = "Car", *"Bus"* = "Bus", *"Metro"* = "Subway", *"Tram"* = "Tram", *"A piedi"* = "On foot". 66

3.4 Completed trip view with a density-based heatmap displaying BLE activity along the route. 66
3.5 Main view of the app showing current detection status and manual override controls. Translation: *"Seleziona il veicolo"* = "Select the vehicle", *"Dispositivi"* = "Devices", *"Sei a piedi"* = "You are on foot", *"Sei sul tram"* = "You are on the tram", *"Sei sulla metro"* = "You are on the subway", *"Sei sul bus"* = "You are on the bus", *"Sei in auto"* = "You are in a car", *"Auto"* = "Car", *"Bus"* = "Bus", *"Metro"* = "Subway", *"Tram"* = "Tram", *"A piedi"* = "On foot". 69
3.6 Main view of the application showing current vehicle selected for transport mode detection. 69

4.1 Evacuation map of the chosen floor in UO. The red X marks the study location. 78
4.2 Evacuation map of the chosen floor in Sapienza. The red X marks the study location. 79

5.1 The LLM receives both natural language descriptions and uploaded documents to generate structured output for form completion. 92
5.2 System Workflow pipeline 93

6.1 Ideal case – perfect sensing and an event is generated at the moment the user's activity changes 108
6.2 Error Timeline 113
6.3 Two times: time of estimation vs. time about which we are estimating 119
6.4 Unambiguous sensing 121
6.5 Ambiguous and incoherent sensing 122
6.6 Difficult case 123

7.1 Example of interaction with CWGPT. 132

9.1 Human-system interaction for sensing and supporting tasks 154
9.2 Sensed and supported tasks 155

10.1 Example of a prompt with CWGPT to evaluate an interface [29] . . 166

11.1 Taxonomy of memory types in MMAG, showing the mapping between human cognitive psychology and technical components for LLM-based agents. 176

List of Tables

2.1 Trips per user 37
2.2 Ground Truth Evaluation 38
2.3 Heuristics Evaluation 39
2.4 Heuristics Evaluation - Comparison 39
2.5 Datasets 41
2.6 Boosted Tree Classifier Evaluation 42
2.7 Boosted Tree Classifier Confusion Matrix - Testing 42
2.8 Boosted Tree Classifier Confusion Matrix - Precision and Recall . . 42
2.9 Final Cruising Detection System Evaluation 43

3.1 Classification Report 62
3.2 A table showing classification performance for four transport modes: Bus-Tram, Car, Subway, and Walk. Metrics include Precision, Recall, F1 Score, and Support. Car has the highest F1 score of 0.87 and the largest support (77). Subway shows poor performance with an F1 score of 0.29. The overall classification accuracy is 0.74, based on 109 samples. 62
3.3 Macro and Weighted Averages 63
3.4 A Table which shows the macro and weighted average precision, recall, and F1 scores for the classification task. 63

6.1 AI actions based on design expectations 115

7.1 Overall agreements between experts and CWGPT answers 134

Chapter 1

Introduction

Interactive systems are increasingly expected to adapt to users, anticipate needs, and provide meaningful support in contexts where attention is scarce and mistakes have real consequences. Two technological shifts are making this expectation more realistic, but also more demanding from a design and engineering standpoint. First, mobile devices have become pervasive sensing platforms that can infer relevant aspects of a user's situation without requiring continuous explicit input, enabling forms of *implicit interaction* that move effort from the user to the system [185]. Second, large language models (LLMs) have made natural language a viable interface, not only for operating systems but also for supporting core activities in Human–Computer Interaction such as design and evaluation [189]. This thesis studies how these capabilities can be engineered and evaluated in a human-centered way, with the goal of reducing user effort while preserving transparency, control, and trust.

A recurring tension in interaction design is that the most valuable moments for support often happen when explicit interaction is least feasible. In mobility scenarios, attention is constrained and even simple interactions may be unsafe or impractical [39]. In administrative and professional work, user effort is instead absorbed by repeated translations between informal information (notes, emails, documents) and rigid system inputs (forms, schemas, configuration fields), where mixed-initiative interaction can reduce friction while keeping the human in the loop [90]. Despite their differences, these settings share the same underlying question: how can systems take on more of the work without becoming opaque, brittle, or unaccountable, and without turning inevitable automation errors into user-facing failures [35].

This thesis addresses this question through contributions that span two complementary directions. Part I focuses on *AI for interaction*, investigating how sensing, inference, and lightweight AI techniques can be treated as interaction mechanisms that enable implicit input and proactive support, with mobility serving as a demanding testbed and application domain [39]. Part II focuses on *LLMs for interaction engineering and design evaluation*, studying how LLMs can be embedded into interac-

tive pipelines that generate inspectable intermediate artifacts, and how these systems can support evaluation, early-stage design work, and reliable task automation under human oversight [189, 90]. Together, these contributions frame automation not as a replacement for interaction, but as a design material whose value depends on how well uncertainty is managed and how effectively users can understand, verify, and correct the system's behavior [35].

1.1 AI for interaction: implicit interaction methods and smart mobility case studies

The first line of work investigates how AI can be embedded into interaction itself, enabling systems that reduce user effort by relying on implicit signals rather than continuous explicit input [39, 33]. The core idea is to treat sensing and inference as interaction primitives: the system observes everyday traces (e.g., motion activity, location patterns, and short-range wireless cues) and turns them into user-facing effects that remove manual steps, reminders, and repetitive reporting. This perspective is particularly valuable in contexts where explicit interaction is costly or unsafe, and where intent is often expressed through behavior rather than articulated through commands [185].

A first cluster of contributions grounds these ideas in the parking domain, used as a demanding real-world setting to develop and validate implicit interaction techniques. The thesis introduces smartphone-based approaches for detecting car-related events under practical constraints, emphasizing lightweight computation, low battery impact, and deployability [39]. It complements these methods with an interactive demonstrator that illustrates how implicit sensing can be surfaced as actionable interface behavior, helping users complete car-related tasks with minimal manual input [33]. The work then extends from discrete event recognition toward higher-level driving behaviors that enable proactive support, including cruising-for-parking detection [40]. Finally, it reflects on how AI techniques can be operationalized within mobile applications for car-parking tasks, consolidating methodological choices and design implications emerging from these studies [32].

A second cluster expands implicit interaction beyond motion and location traces by leveraging Bluetooth Low Energy (BLE) as an additional source of context. Here, the thesis explores how smartphone BLE beaconing can be used to infer human presence and support more context-aware experiences in automated mobility settings [62, 61]. These studies treat BLE not as a standalone sensing solution but as a complementary interaction signal, contributing evidence about feasibility, limitations, and the kinds of user-facing support that such inferred context can responsibly enable.

A third cluster shifts the focus from sensing people's context to interpreting their environment through LLMs, investigating how language models can be exploited "out of the box" for indoor localization from evacuation maps and other spatial artifacts. The thesis explores the extent to which LLMs can transform map content into actionable localization cues, and what kinds of interaction scaffolding are needed when the underlying reasoning is uncertain or approximate [201]. This line of work broadens the thesis perspective on context awareness, moving from implicit sensing signals to language-mediated spatial understanding while still targeting practical, safety-relevant scenarios.

Finally, a fourth cluster connects implicit interaction to user input support through LLMs, focusing on the transformation of unstructured user descriptions into structured representations that interactive systems can act upon. The thesis investigates LLM-based agentic pipelines that generate intermediate artifacts (such as structured fields or form-ready inputs) that users can inspect and correct, aiming to reduce friction in workflows where the bottleneck is often the translation from natural language to structured data [224, 5]. Together, these contributions broaden the scope of *AI for interaction* from sensing-based implicit input to language-based structuring support, while maintaining a consistent emphasis on interaction mechanisms that keep users in control of automation outputs.

Across these studies, a recurring interaction challenge is that uncertainty and misclassification are unavoidable, whether the system infers context from sensors or derives structure from language. The thesis therefore treats fallibility as a first-class design constraint, linking technical limitations to interface questions about how systems should communicate confidence, support verification, and recover gracefully when the inference is wrong [36].

1.2 LLMs for interaction engineering and design evaluation

The second, and main, line of work examines how large language models can support the engineering and evaluation of interactive systems, treating LLMs as components embedded within workflows rather than as standalone chat interfaces [190]. In this framing, models produce intermediate artifacts (critiques, rationales, structured notes, candidate requirements, or reusable context) that users can inspect, validate, and refine. The goal is to combine automation with the designer's agency, so that LLMs accelerate design work while preserving human control and accountability [91].

The thesis starts from the premise that LLM-based assistance is a form of fallible automation that must be designed for. In *Designing for Fallible Automation*, the

work frames model errors, overconfidence, and variability as predictable interaction properties, focusing on how interfaces should communicate uncertainty, set expectations, and support recovery when AI output is wrong or misleading [36]. This perspective motivates the remaining contributions, which treat LLM outputs as suggestions to be verified and refined, not decisions to be accepted.

In *Enhancing Interface Design with AI*, the thesis investigates LLM-supported design evaluation through a cognitive walkthrough inspired tool, examining how model-generated critiques can be produced from interface evidence and scenarios, and how useful and trustworthy such feedback is in practice when compared with human-led evaluation [30]. Complementing this, *Heuristic Evaluation of Implicit Interactions in Proactive Systems* extends the evaluation focus to proactive and implicit interaction, analyzing how traditional heuristic evaluation needs reinterpretation when system behavior is driven by inferred context rather than explicit commands [37].

Part II also addresses earlier stages of user-centered design. In *Large Language Models Adoption in Need Finding*, the thesis studies how LLMs can support need finding and requirements articulation under time and resource constraints, while characterizing risks such as plausible but ungrounded outputs and the interaction practices needed to keep human judgment in the loop [42].

These themes are brought together in *Transforming Interactive Systems with Large Language Models*, which positions LLMs as accelerators within interactive pipelines for design and evaluation, and outlines methodological and practical implications for building systems around model-generated artifacts [223]. Finally, in *Memory Augumented Generation*, the thesis explores how memory-augmented generation can improve LLM applications by enabling controlled reuse of context, discussing implications for reliability, transparency, and user trust when past information is incorporated into model outputs [222].

Across Part II, the thesis returns to a consistent interaction goal: making LLM support effective in real settings by designing workflows that surface uncertainty, encourage verification, and enable efficient correction and repair when model outputs fail [36].

1.3 Research goals and contributions

The overarching goal of this thesis is to provide methods and evidence for building interactive systems where AI reduces user effort without undermining user agency [189]. Concretely, the thesis contributes:

- Smartphone-based implicit interaction techniques for recognizing mobility context and enabling proactive support in car-related scenarios [39, 33].

- Models and interaction strategies for detecting higher-level behaviors relevant to smart parking, including approaches aimed at near-real-time operation on-device [40].
- LLM-based tooling and workflows that support usability evaluation activities, with adaptations that make walkthrough-style practices operational through conversational and mixed-initiative interaction [166, 29, 90].
- Evidence and design implications for integrating LLM assistance in early user-centered design phases, with attention to verification needs and failure modes [41, 35].
- A unified perspective on error detection and repair in AI-supported interaction, bridging sensing-based uncertainty in mobility with generative uncertainty in LLM-based systems [35].

1.4 Thesis structure

The remainder of this thesis is organized into two main parts, preceded by background chapters that introduce the conceptual and methodological foundations, and complemented by dedicated sections that map the dissertation to the corresponding peer reviewed outputs.

The thesis first motivates the overall research problem, positioning AI as a dual contribution to interactive systems: on the one hand, as a means to make interaction more context-aware and less demanding through sensing and inference; on the other hand, as a tool that can support and accelerate HCI activities such as interface design, documentation, and evaluation. These introductory chapters frame the central thread of the dissertation, namely how to benefit from automation while keeping interaction and design processes robust when AI is uncertain or fallible.

Part I investigates how AI can be embedded into the interaction itself to reduce user effort and enable proactive support. It focuses on implicit interaction in mobility, where smartphones act as sensing and inference devices to detect car related events and behaviors without requiring continuous explicit input. The part introduces the mobility setting and the sensing rationale, then presents the main technical and interaction contributions grouped around three themes: (i) implicit interaction for parking related tasks, (ii) implicit interaction enabled by Bluetooth Low Energy sensing for context aware mobility, and (iii) LLM based support for transforming user descriptions into structured input and assisting form completion. Beyond reporting models and systems, the chapters in this part discuss what these capabilities imply for real deployments, including reliability, user trust, and the design of proactive behaviors that avoid distracting or over-automating the user. The publications associated with Part I are summarized in Section 1.5.1.

Part II shifts the perspective from AI in the interface to AI as a tool for the people who design and evaluate interfaces. This part examines methods, systems, and studies that use LLMs to augment HCI work, with a particular focus on interaction engineering and design evaluation. The part first introduces the methodological challenges of using generative models in structured workflows, including reproducibility, grounding, and assessing output quality. It then presents approaches that leverage LLMs to support usability evaluation activities and early stage design tasks, discussing both the benefits and the practical failure modes that emerge when AI generated assistance is incomplete, inconsistent, or incorrect. The publications associated with Part II are summarized in Section 1.5.2.

Chapter 12 synthesize then the two parts through a human centered AI lens. They highlight a shared theme across sensing driven interaction and LLM based workflow support: automation can be valuable even when it is fallible. The thesis therefore emphasizes strategies to keep both interactive systems and design processes robust under uncertainty, combining proactive capabilities with transparency, user control, and recovery mechanisms. Finally, the thesis summarizes the main contributions and outlines directions for future work, including broader validation, deployment oriented evaluation, and extensions toward more adaptive and context-aware human AI interaction.

1.5 Declaration of original work and publications

This thesis presents original research conducted during my PhD at Sapienza University of Rome. The work is my own, and it was developed in a collaborative research setting through ongoing discussions, feedback, and input from colleagues and supervisors.

Several parts of the thesis build on results that I previously published and presented at international conferences and workshops. These contributions informed the research questions, guided the experimental work, and influenced the findings reported in the following chapters. In total, the thesis draws on 15 publications (14 of them peer reviewed); in 12 of these, I am the first author or co-first author. It also builds on 1 patent [155] and 1 Sapienza University Startup [153].

1.5.1 Publications related to Part I: AI for interaction

1.5.1.1 Publications about Implicit Interaction in the parking domain

1. A. Bisante, E. Panizzi, and **S. Zeppieri***, *"Implicit Interaction Approach for Car-related Tasks On Smartphone Applications"* in Proceedings of the 2022

*First or co-first author.

International Conference on Advanced Visual Interfaces, 2022, pp. 1–5. doi: 10.1145/3531073.3531173. [39]

2. A. Bisante, V. S. V. Datla, **S. Zeppieri***, and E. Panizzi, *"Implicit Interaction Approach for Car-related Tasks On Smartphone Applications - A Demo"* in Proceedings of the 2022 International Conference on Advanced Visual Interfaces, 2022, pp. 1–3. doi: 10.1145/3531073.3534465. [33]

3. A. Bisante, E. Panizzi, and **S. Zeppieri**, *"Cruising-for-Parking Detection on the Smartphone Based on Implicit Interaction and Machine Learning"* in Proceedings of the 15th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, 2023, pp. 93–102. doi: 10.1145/3580585.3607162. [40]

4. A. Bisante, V. S. V. Datla, G. Trasciatti, **S. Zeppieri***, and E. Panizzi, *"An Approach to Leverage Artificial Intelligence for Car-Parking Related Mobile Applications"* in Engineering Interactive Computer Systems. EICS 2023 International Workshops and Doctoral Consortium, M. Harrison, C. Martinie, N. Micallef, P. Palanque, A. Schmidt, M. Winckler, E. Yigitbas, and L. Zaina, Eds., Cham: Springer Nature Switzerland, 2024, pp. 63–71. doi: https://doi.org/10.1007/978-3-031-59235-5_7. [31]

1.5.1.2 Publications about Implicit Interaction supported through BLE

5. V. S. V. Datla, **S. Zeppieri***, A. Aiuti, A. Bisante, G. Trasciatti, and E. Panizzi, *"Towards Context-Aware UX in Automated Mobility: BLE Based Passenger Detection via Smartphones"* in Proceedings of the 16th Biannual Conference of the Italian SIGCHI Chapter, in CHItaly '25. New York, NY, USA: Association for Computing Machinery, 2025. doi: 10.1145/3750069.3750132. [63]

6. V. S. V. Datla, A. Aiuti, A. Bisante, G. Trasciatti, **S. Zeppieri**, and E. Panizzi, *"Detecting Human Presence via Smartphone BLE Beaconing: Preliminary Investigations"* in Proceedings of the 24th International Conference on Mobile and Ubiquitous Multimedia, in MUM '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 483–485. doi: 10.1145/3771882.3773958. [61]

1.5.1.3 Indoor Localization Using Large Language Models

7. G. Trasciatti, **S. Zeppieri**, V. S. V. Datla, A. Chambers, and E. Panizzi, *"Investigating out-of-the-box Indoor Localization Using Large Language Models and Evacuation Maps"* (accepted at PERCOM) 2026. [201]

*First or co-first author.

1.5.1.4 Publications about Supporting User Input with LLMs

8. **S. Zeppieri***, A. Aiuti, A. Bisante, V. S. V. Datla, G. Trasciatti, and E. Panizzi, *"Engineering Large Language Model Agents for Transforming Unstructured Descriptions into Structured Input"* in Proceedings of the EISEAIT 2025 Workshop (co-located with EICS 2025), in Lecture Notes in Computer Science (LNCS). Springer, 2025. [225]

9. A. Aiuti, **S. Zeppieri***, A. Bisante, V. S. V. Datla, G. Trasciatti, E. Panizzi et al. *"Automating Form Completion with Large Language Models"* in Proceedings of the 16th Biannual Conference of the Italian SIGCHI Chapter, in CHItaly '25. New York, NY, USA: Association for Computing Machinery, 2025. doi: 10.1145/3750069.3755963. [5]

1.5.2 Publications related to Part II: LLMs for interaction engineering and design evaluation

1.5.2.1 Designing for Fallible Automation

10. A. Bisante, A. Dix, E. Panizzi, and **S. Zeppieri***, *"To Err is AI"* in Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter, in CHItaly '23. New York, NY, USA: Association for Computing Machinery, 2023. doi: 10.1145/3605390.3605414. [35]

1.5.2.2 Enhancing Interface Design with AI

11. A. Bisante, V. Datla, E. Panizzi, G. Trasciatti, and **S. Zeppieri***, *"Enhancing Interface Design with AI: An Exploratory Study on a ChatGPT-4-Based Tool for Cognitive Walkthrough Inspired Evaluations"* in Proceedings of Advanced Visual Interfaces 2024, in AVI '24. New York, NY, USA: Association for Computing Machinery, 2024. doi: 10.1145/3656650.3656676. [29]

1.5.2.3 Large Language Models Adoption in Need Finding

12. A. Bisante, **S. Zeppieri***, V. S. V. Datla, G. Trasciatti, and E. Panizzi, *"Assessing Large Language Models Adoption in Need Finding: an Exploratory Study"* 2024. [41]

*First or co-first author.

1.5.2.4 Heuristic Evaluation of Implicit Interactions in Proactive Systems

13. A. Bisante, A. Dix, E. Panizzi, and **S. Zeppieri***, *"Implicit Interactions in Proactive Systems: Evaluation Challenges and Adaptations for Nielsen's Heuristics."* 2025. [38]

1.5.2.5 Transforming Interactive Systems with Large Language Models

14. **S. Zeppieri***, *"Transforming Interactive Systems with Large Language Models: Accelerating Interface Design and Evaluation"* in Proceedings of the CHItaly 2025 Doctoral Consortium, in CEUR Workshop Proceedings. 2025. [223]

1.5.2.6 Memory Augumented Generation

15. **S. Zeppieri***, *"MMAG: Mixed Memory-Augmented Generation for Large Language Models Applications."* 2025. doi: https://arxiv.org/abs/2512.01710. [222]

1.5.3 Technology Transfer and Entrepreneurship: Patent and University Startup

Beyond peer-reviewed publications, this PhD has also produced technology-transfer outcomes that consolidate the research direction into deployable artifacts. These results reflect a recurring theme across the dissertation: designing intelligent, user-centered systems that reduce user effort while preserving efficiency, robustness, and privacy, and translating research prototypes into solutions with practical applicability.

1.5.3.1 Patent: autonomous localization for privacy-preserving contexts.

The patent by Panizzi et al. [155] represents a technology-transfer outcome aligned with the thesis' broader focus on reducing user effort through intelligent, context-aware mobile systems. The invention targets localization scenarios where traditional infrastructures are unavailable or undesirable, emphasizing robustness and privacy. Rather than relying on external signals, it enables a mobile device to estimate its position using on-device sensing and lightweight matching strategies, with the goal of remaining practical on resource-constrained devices and suitable for indoor, outdoor, underground, or remote environments.

*First or co-first author.

1.5.3.2 University startup: NITSY s.r.l. and user-centered digital processes.

The university startup initiative [153] is a Sapienza University spin-off that operationalizes the broader vision of leveraging AI and HCI to simplify complex workflows, with a specific focus on bureaucratic and business processes in both public and private organizations. The company's objective includes the end-to-end design and delivery of digital platforms offered as services (e.g., through usage or rental contracts), combining scalable software engineering practices (microservices, containerization, standardized APIs, federated authentication) with careful interface design grounded in Human-Computer Interaction, Software Engineering, and Business Process Management. Within this scope, LLM-based assistance and other AI components are positioned as enabling technologies to reduce friction in data-intensive tasks, support interaction, and improve the usability and efficiency of process-driven applications, while extending to applied domains such as sustainable mobility, geolocalization, smart city services, and smart parking.

Part I

AI for Interaction

Overview

This part introduces the first research stream of this thesis, which investigates how AI can become part of the interaction mechanism itself. The focus is on reducing user effort by shifting from continuous explicit input to implicit, naturally occurring signals [39, 33]. Instead of asking users to repeatedly report information or complete manual steps, the system treats sensing and inference as first-class interaction primitives: it monitors lightweight traces of everyday activity such as movement, mobility and location regularities, and short-range wireless cues, and translates them into user-facing actions. This approach is especially relevant in settings where explicit interaction is burdensome or unsafe, and where users' goals are more often revealed through behavior and context than through direct commands [185].

Chapter 2

Implicit Interaction in the parking domain

2.1 Introduction

Smartphone applications are now a common companion to driving. Beyond navigation, many tools support everyday car-related tasks such as tracking consumption and maintenance, paying for parking, or remembering where the car is parked [13]. Despite their usefulness, these apps often fall short from an interaction perspective: they demand attention at the wrong time (potentially increasing distraction while driving) and they frequently rely on repeated manual input to keep information up to date. In the parking domain this tension is particularly evident, because the key information needed by many services (where and when the car is parked, whether the driver is about to leave, and when a new trip begins) is also the information users are least willing or able to enter consistently.

This section introduces our work on *implicit interaction* for smart parking, where the system infers relevant events and intentions from smartphone context data, rather than requesting explicit actions from the driver. Parking search in large cities, especially for on-street parking, is widely recognized as stressful and inefficient, with negative effects on congestion and emissions; estimates reported in the literature indicate that drivers spend several minutes searching and that cruising can account for a substantial portion of urban traffic [192, 87]. Implicit interaction is a natural fit here: if a smartphone can detect when the user approaches, enters, exits, parks, and starts driving again by leveraging location and motion sensing, then core parking-related tasks can be supported without burdening the user or introducing additional distraction. Throughout these contributions, we also treat practical constraints as first-class requirements, aiming to avoid manual configuration, external sensors, and excessive battery consumption.

The four papers in this subsection form a coherent progression from foundational interaction design to real-time behavioral inference and AI-enabled services. We first propose an implicit interaction approach for car-related tasks on smartphones, centered on automatically inferring basic smart-parking actions from parking and unparking events. We then present a working prototype that demonstrates the approach end-to-end, making the interaction model and sensing pipeline concrete in a deployable application. Building on this capability, we tackle a key mobility behavior that directly impacts parking efficiency, namely *cruising for parking* [192, 214], and study whether it can be detected in real time from smartphone traces to enable timely, proactive assistance. Finally, we broaden the perspective by discussing how to systematically leverage AI in car-parking mobile applications, framing design and implementation challenges (feature selection, data quality, on-device constraints) and illustrating how context data and lightweight models can transform parking-related interactions into a more seamless experience [203].

2.2 Related Works

Research on interactive systems in the driving context has grown steadily in recent years, with a strong emphasis on preserving driver safety while facilitating time-sensitive tasks such as navigation and parking. Smartphones are frequently adopted as the main platform because they are affordable, powerful, and typically available to the driver both inside and outside the vehicle. In the smart parking domain, prior work spans three closely related areas: (i) the tasks and interfaces offered by consumer-facing parking applications and larger smart parking systems, (ii) methods for automatically detecting parking-related events (e.g., entering/exiting a vehicle, parking/unparking), and (iii) approaches to detect the parking search phase, often referred to as *cruising for parking* [192, 214]. In the following, we review these strands and position our contributions with respect to them.

2.2.1 Smart parking tasks and interfaces

Smart parking systems commonly aim to support drivers in finding a suitable space near a destination, often at low cost, and in some cases reserving a space in advance. For broad overviews of goals, architectures, and techniques, several recent review papers summarize the landscape of intelligent parking solutions [77, 49, 228, 174]. A recurring distinction concerns how parking availability is sensed. Many proposals rely on external infrastructure, including cameras, infrared sensors, beacon-like devices, and more generally IoT deployments [149]. While such approaches can deliver accurate availability estimates in controlled or delimited areas, they are harder to scale to city-wide on-street parking due to deployment and maintenance costs, and they often assume explicit user actions (e.g., requesting a reservation and

specifying an arrival time), as reflected in common app interfaces such as JustPark and RingGo [99, 170].

From an interaction point of view, a key set of user-facing tasks in mainstream apps concerns (i) saving the last parking location, (ii) managing parking time and payments, and (iii) sharing parking availability signals with others. Many widely used applications implement these functions through explicit input. For example, saving the parking location is often done via a button that stores the current GPS coordinates, as in Google Maps, EasyPark, PayByPhone, JustPark, RingGo, PassportParking, Parkster, and Parkopedia [119, 74, 163, 99, 170, 158, 159, 157]. Although this interface is simple, it is also easy to forget and can become tedious when repeated frequently. Some popular applications attempt to reduce explicit input by leveraging implicit signals, for instance Apple Maps and Waze, which can infer parking status using Bluetooth connectivity or trip-related cues, and apps such as FindMyParkedCar that also employ Bluetooth-based detection [11, 211, 135].

A similar tension between explicit and implicit interaction emerges in parking payment. Several services ask users to specify the expected end time and then manually extend or terminate the session (e.g., EasyPark, PayByPhone, JustPark) [74, 163, 99]. In contrast, in private or controlled areas, systems can infer entry and exit through external sensing (e.g., plate-recognition at barriers), enabling time-based charging without opening the app [74]. Overall, the literature and deployed products highlight a central usability challenge: many valuable parking services depend on knowing *when* and *where* parking and unparking happen, yet current solutions frequently obtain this information through repeated explicit input.

2.2.2 Automatic detection of parking and unparking events

A foundational research direction focuses on sensing and recognizing vehicle-related states using smartphones, often as part of transportation mode detection. PhonePark [195], building on prior work on mode inference [196], represents an early attempt to detect parking and unparking using GPS, accelerometer signals, and Bluetooth connectivity, modeling state changes in user mobility. ParkSense [136] explored a different assumption: inferring only unparking events by combining increasing speed patterns with Wi-Fi observations, but requiring the user to manually provide the parking location. ParkHere [176] proposed a supervised approach based on accelerometer and gyroscope sampling with a Random Forest classifier, detecting parking/unparking via transitions between *walking* and *automotive* states, while still relying on Bluetooth-assisted vehicle association and fixed-rate sensing.

ParkUs [50] is one of the closest works to our goals in terms of interaction requirements and energy constraints. It targets low user burden and limited calibration while maintaining low energy consumption, using magnetometer and accelerometer

data and an ad-hoc state detection approach that identifies a dedicated sensor-fusion feature. Complementary efforts have investigated broader frameworks for car-related event sensing. For example, Panizzi et al. proposed a mobile framework that combines Bluetooth connectivity with smartphone sensing to detect events such as parking/unparking, parked-car location, and trip data [154]. More recent work by Lee et al. [105] reported high accuracy for in/out car classification using magnetometer and orientation sensors with a Convolutional Neural Network, but it was presented within a larger infrastructure-supported smart parking system for delimited areas, limiting comparability and transferability.

Across these proposals, two issues recur. First, several systems depend on in-car Bluetooth connectivity or require explicit configuration to associate a vehicle, which can hinder adoption and can behave inconsistently across car models and infotainment implementations. Second, many approaches rely on custom sensing pipelines and per-study models trained on limited datasets, with energy trade-offs that can be challenging in real deployments. These considerations motivate solutions that can (i) reduce or remove Bluetooth dependence, (ii) exploit operating-system level activity inference when possible, and (iii) support background operation without excessive battery drain, while still enabling parking services that would otherwise require explicit user input.

2.2.3 Cruising-for-parking detection

While the smart parking literature is extensive, research specifically aimed at detecting the parking search phase during a trip is comparatively more limited. The phenomenon is commonly referred to as *cruising for parking* [192] and is studied as a form of excess travel associated with parking search [214]. Much of the earlier and parallel work estimates cruising costs and rates rather than enabling real-time detection, focusing on additional time or distance traveled, the share of traffic involved, and the impact on congestion and emissions [192, 87, 60].

Hampshire et al. summarized common methodological families for identifying parking search behavior [87]. These include direct observation (often with cameras), interviews and surveys, controlled field tests, probabilistic reasoning from vehicle counts and fixed-camera streams, trajectory analysis from visual tracking, and GPS-based trajectory analysis possibly supported by machine learning. Variants of image-based approaches track vehicles and trajectories to infer cruising patterns [168, 150], and some work has explored in-car camera signals pointed at the driver for accurate start-time identification, at the cost of portability and deployment complexity [86]. GPS-based work includes early methods that recognize cruising evidence from recorded paths even when the search start time is not explicitly labeled [131, 130]. Other studies infer search onset through empirical speed thresholds [68, 209] or compute *excessive routing* by comparing driven trajectories to shortest

paths, sometimes combined with machine learning to classify whether trips include search behavior [213, 126]. Related contributions address commercial fleets and incorporate historical traffic conditions [60] or estimate expected search time given a destination by recognizing spiral-like patterns and routing inefficiency [118].

Only a smaller subset of approaches targets real-time cruising detection using smartphone-collectable data. ParkUs [97] is particularly relevant in this respect, as it deploys a model based on smartphone sensors to detect cruising in real time. However, it assumes additional user input (e.g., specifying the destination), whereas the broader goal of implicit interaction in this domain is to avoid tedious or distracting actions while driving. This line of work is closely aligned with context-aware services for driving environments [92, 117] and with the motivation to reduce distraction through adaptive or implicit interaction strategies [101].

2.2.4 Summary and positioning

Taken together, prior work shows that smart parking systems can be effective but often trade usability and scalability for sensing accuracy, either by relying on external infrastructure [149] or by shifting burden to users through repeated explicit inputs [119, 74, 163, 99, 170, 158, 159, 157]. Smartphone-based event detection research demonstrates that parking-related states can be inferred from sensors, but frequently introduces configuration steps (often Bluetooth-based) or depends on custom pipelines with unclear cross-platform or background-operation properties [195, 176, 50, 154]. Cruising detection research, meanwhile, is rich in offline estimation methods and GPS-based analyses, yet real-time smartphone-only approaches remain relatively scarce [87, 131, 213, 97]. These gaps motivate the implicit-interaction approach adopted in our work: minimizing explicit user input and distraction while leveraging smartphone context data to infer parking/unparking and cruising behavior in real time, under practical constraints typical of mobile deployment (limited configuration, low energy use, and background operation).

2.3 Implicit Interaction Approach for Car-related Tasks On Smartphone Applications

Many car drivers use apps on their smartphones to get support in typical tasks related to car usage. Navigators and maps are a primary example, but many other apps exist to track consumption, expenses, maintenance, pay parking lots or keep track of where the car is parked [13]. Nevertheless, some of the available apps lack user interaction under two main aspects: first, they require the user's attention and might be a distraction while driving; secondly, they rely on the user inputting data repetitively.

Implicit interaction is a possible solution to improve the user experience of typical car-related interfaces. The less the user is required to interact with the smartphone while driving, the safer. The less the user needs to input data manually, the better.

Our work focuses on smart parking, which is recently receiving increasing attention in the literature. Searching for parking in large cities, in particular on-street parking, is a stressful activity for the driver and has significant impacts such as increased traffic and pollution. Literature has estimated that, on average, looking for a place to park the car takes 8 minutes and affects about 34% of urban traffic [192], [87]. Smart parking systems aim to help find a parking space, remind users of where they parked, facilitate parking payment, and much more. These solutions have often been developed for smartphones, as they are within everyone's reach inside and outside the car. The primary information needed to complete smart parking tasks is where and when the car is parked and unparked, and existing apps generally require that the user inputs these information explicitly.

The main goal of this work is to present an implicit interaction approach to automatically detect when the user enters and exits their vehicle, hence the occurrence of parking and unparking events. To detect such occurrences, we leverage the the smartphone's sensing capability of users' locations and motion activities.

The proposed approach eases the implementation of the following tasks on a smartphone application:

$T1$ user reports parking their car in a specific position

$T2$ user declares their willing to free a parking spot

$T3$ user declares that a new trip with the car has begun, thus the parking spot is free

These tasks allow some of the main smart parking features, like informing users of available parking spots, reminding drivers of where they parked their car, compute usage statistics and so on. However, it is not easy for a user to complete these tasks explicitly, as they may forget to do so or they may be driving.

The objective is also to avoid manual configuration of the system on the user's part, as this worsens the user experience and may even hinder the use of the system. In addition, as non-functional requirements, the proposed system avoids requiring external sensors and prevents intensive battery usage.

This work contributes in three ways:

1. It shows how to design a context-aware implicit interaction for three daily tasks related to car usage.
2. It allows collecting data about driving events, which can be used to train any machine learning models or can be leveraged to detect driver behaviors.
3. It lays the foundation for future works that will exploit it to provide users with new, useful features.

The remainder of this paper is organized as follows. In section (2.3.1 and 2.3.2), a high-level description of the approach and its implementation are presented.

Finally (2.3.3), the conclusions drawn from our study and possible future research topics will be illustrated.

2.3.1 Design of the implicit interaction

Our main goal was to design an implicit user interface for the three tasks:

$T1$ user reports parking their car in a specific position

$T2$ user declares their willing to free a parking spot

$T3$ user declares that a new trip with the car has begun, thus the parking spot is free

We do not expect the user to open the app and directly operate its interface. The user interaction we designed is implicit because, according to [185], its input-effect relationship is *unintentional*: the system responds to the user actions with effects that go beyond what the user has intended. According to Dix [69], the designed interaction is also *expected*, as the user is aware and expects the possible outcomes of its actions, yet they perform them without the intention to cause any output.

For example, when the user parks their car and exits it, our app updates the car status to *parked*, storing the latitude, longitude, and timestamp of the parking, thus inferring that the user has performed task $T1$. This effect happens even if the app is in the background or the smartphone is locked.

Similarly, the app tries inferring the implicit execution of task $T2$ when the user walks towards the parked car. We know that the user may walk to their car for

other reasons than unparking it, such as when they just left something in the car and go back to take it. We allow for some errors in this task and plan to analyze the error rate when we have collected sufficient data.

Finally, when the user starts driving a car from the position of their parked car, the app toggles the car status to *unparked*. The user implicitly performed task $T3$ while their main intention was to use their car.

2.3.1.1 Definitions

Let us define an Event as a triple:

$$E(a, t, l) \tag{2.1}$$

where a is a motion activity that can be either *walking* or *automotive*; t and l are a time and a location indicating when and where the activity took place.

We can sample the user's motion with a temporally ordered series of events:

$$History = E_1(a_1, t_1, l_1), E_2(a_2, t_2, l_2), ..., E_n(a_n, t_n, l_n) \tag{2.2}$$

where E_i precedes E_{i+1} if $t_i < t_{i+1}$.

Two or more consecutive events may have the same activity. However, the most relevant events in the $History$ have activities different from the preceding event. They represent the transitions between *walking* and *automotive* and vice-versa.

We call $ParkingEvent$ P an event E_i which has a *walking* activity:

$$P(t, l) = E_i(a_i = walking, t_i, l_i) \mid \exists E_{i-1}(a_{i-1} = automotive, t_{i-1}, l_{i-1}) \tag{2.3}$$

Similarly an $UnParkingEvent$ U is an event E_i which has *automotive* activity:

$$U(t, l) = E_i(a_i = automotive, t_i, l_i) \tag{2.4}$$

such that

$\exists E_{i-1}(a = walking, t_{i-1}, l_{i-1}) \wedge$

$\exists P(t, l) \wedge$

$|l_i - l| \leqslant \epsilon$

where ϵ is an empirically chosen distance.

Finally, an $ApproachingEvent$ for a given parking $P(t, l)$ is an event

$$A_P = E_i(a_i = walking, t_i, l_i) \tag{2.5}$$

such that

$$\nexists U(t_j, l),\ t < t_j < t_i \wedge$$

$$\exists E_{i-1}(a = walking, t, l_{i-1}) \wedge$$

$$|l_i - l| \leqslant \epsilon \wedge |l_{i-1} - l| > \epsilon$$

where ϵ is an empirically chosen distance.

So we can formulate the three tasks in terms of the events described above:

$T1$ Occurrence of a ParkingEvent $P(t, l)$

$T2$ Occurrence of an ApproachingEvent A_P

$T3$ Occurrence of an UnParkingEvent $U(t, l)$

2.3.2 Implementation

We implemented the interface both on an iOS and an Android app, which exploit the respective Operating System's motion and location libraries. The two implementations differ slightly due to the architectural differences of these libraries, and we refer here to the iOS implementation, which requires dealing with more stringent privacy and battery consumption constraints.

A central issue about data collection is the impossibility of constantly requesting information. Having as much data as possible would allow a higher level of accuracy to make detections. However, it is better not to request continuous updates on location and motion activities for both privacy and battery consumption, especially when the app is in the background.

2.3.2.1 Flow and Architecture

The two sources of location and motion activity data are not directly related. The location data does not include a motion activity, and the motion data do not contain the location at which the motion event took place. However, both of them include a timestamp, so we can try to match location and motion to create events.

The operating system wakes up the application when a new location is available. Therefore, we can collect the user's locations in an ordered sequence and trigger a motion activity request when a new location arrives. Then, when we receive the activity report, we can create events with location and motion information, using a couple of matching algorithms based on linear interpolation, and append them to a `History` array.

When we append a new event to the `History`, we check if it has the same activity value as the latest event. If not, we generate a new `Parking` or `Unparking` event

and change the car status.

The app interface provides a map with green pins representing the *walking* locations and red pins representing the *automotive* locations. The `Unparking` position is the first red pin in a sequence, while the `Parking` position is the latest red pin (Figure 2.1).

2.3.2.2 Approaching the Parking Location

As users approach their parked vehicle, they will likely enter the car and free a parking spot. Hence, it is valuable information to store. According to formula 2.5, it would be possible to compute the distance from the last parking at each new location received. However, such approach is highly energy-consuming. We adopted the following OS available service as a more battery-saving alternative.

To implicitly detect ApproachingEvents, when a ParkingEvent *P(t,l)* occurs, we set a circular `Region` centred on the location of the latest parking, `l`. This way, when the user enters the region's boundaries, the operating system wakes up the app to check if the user can unpark the car.

As reported in formula 2.5, approaching a parked vehicle is a valid ApproachingEvent only if the user is *walking*. This condition is a consequence of the issue described in the previous section. Indeed, the *automotive* state can be triggered when using different vehicles. For instance, the user might pass by the parked car while in another vehicle; in such cases, they are not willing to free the parking spot.

To avoid considering as an ApproachingEvent the user getting near an old parking position, we remove the `Region` when an UnParkingEvent occurs; thus, an ApproachingEvent can not take place when the car has already been unparked.

2.3.3 Discussion

The work presented in this research proposes an entirely implicit interaction-based system to detect when the user enters and exits their car automatically. It contributes in three ways: *(i)* it shows how to design a context-aware implicit interaction for three daily tasks related to car usage; *(ii)* it allows collecting data about driving events, which can be used to train any machine learning models or can be leveraged to detect driver behaviors; *(iii)* it lays the foundation for future works that will exploit it to provide users with new, useful features.

Figure 2.1. Prototype App

2.4 Publications on Implicit Interaction Approach for Car-related Tasks On Smartphone Applications - A Demo

Basic user tasks for many smart-parking applications are *(i)* reporting parking the car in a specific position, *(ii)* declaring that the user will soon free a parking spot and *(iii)* that a new trip with the car has begun (thus, a parking spot became free). Existing apps generally require that the user inputs this information explicitly (e.g. [99], [170], [158], [159], [157]); on the contrary, our approach is totally implicit. We detect when users move toward, enter, or exit their car by leveraging the smartphone's sensing capabilities of users' locations and motion activities. From such information, the execution of the reported basic tasks can be inferred; thus, the user is not burdened by unnecessary explicit interactions.

This section presents a prototype smartphone application that implements the proposed approach and automatically detects when users enter and exit their vehicle. Among our objectives, we aimed at avoiding any manual configuration of the system on the user's part, as this worsens the user experience and may even hinder the use of the system. In addition, as non-functional requirements, the proposed system avoids requiring external sensors and prevents intensive battery usage.

2.4.1 The Prototype

The presented prototype application was implemented on iOS.

By exploiting the smarpthone's motion sensor, we can detect when the user's motion activity status changes from *walking* to *automotive*, hence infer an `UnParking` event. Similarly, when a change in motion is detected from *automotive* to *walking* status, a `Parking` event is inferred. We can also locate the events by combining such information with GPS data. In this way, we can store the `Parking` events' coordinates and show the users the parking position on a map, to help them find their parked car. Also the location of `UnParking` events is relevant, as we compare it with the previous `Parking` event location to strengthen the validity of our detections.

Unfortunately, these two sources of data are not directly related. The location data does not include a motion activity, and the motion data do not contain the location at which the motion event took place. However, both have a timestamp, so we match location and motion to create events.

A central issue about data collection is the impossibility of continuously getting information from the smartphone's operating system services, especially when the app is in the background. Although it could allow a higher level of accuracy to make detections, it drains battery and raises privacy concerns. Consequently, a crucial

part of this study was to compensate for this inability by filling in the gaps through an interpolation algorithm.

2.4.1.1 User Interface

We do not expect the user to open the app and directly operate its interface. When the user parks their car and exits it (i.e., a `Parking` event is detected), our app updates the car status to *parked*, storing the coordinates and the timestamp of the parking. The detection occurs even if the app is in the background or the smartphone is locked. Similarly, when the user walks towards the parked car and starts a new trip, the application recognizes the `UnParking` event and toggles the car status to *unparked*.

As we described in [39], the user interaction we designed is implicit, as the system responds to the user actions with effects that go beyond what the user has intended (i.e. *unintentional* interaction, [185]). According to Dix [69], the designed interaction is also *expected*, as the user is aware and expects the possible outcomes of its actions.

The interface of our prototype app presents a full view map that shows the user's changing locations with red pins for *automotive* locations and green pins for *walking* locations (Figure 2.2). The car's parking location corresponds to the last red pin of a sequence. On top, two icons represent the current user's motion activity and car status, respectively (Figures 2.3a and 2.3b).

2.4.2 Demo

We organized a demo to allow conference participants to test the app and discuss its approach and details. Due to space limitations in the conference area, we will use a bicycle in place of a car; such change entailed slightly modifying the detection algorithm by replacing the *automotive* activity detection with *cycling* activity detection. We installed the application on an iPhone 13 Pro, which we will give to testers to participate in the demo.

Once the demo starts, we ask the tester to take a walk with the smartphone in their hand to appreciate if it triggers the motion sensor, thus switching to the *walking* status. The collected walking locations are stored and displayed on the map with green pins.

Then, we ask the participant to jump on the bike and take a ride; the changing locations and the motion sensor will trigger the *cycling* status. At this point, the user will be able to see that the vehicle has been *unparked*. The user should travel a minimum of 10 meters for the smartphone to start sensing and changing the vehicle's current status to *unparked*. All automotive locations are stored and displayed on the map using red pins.

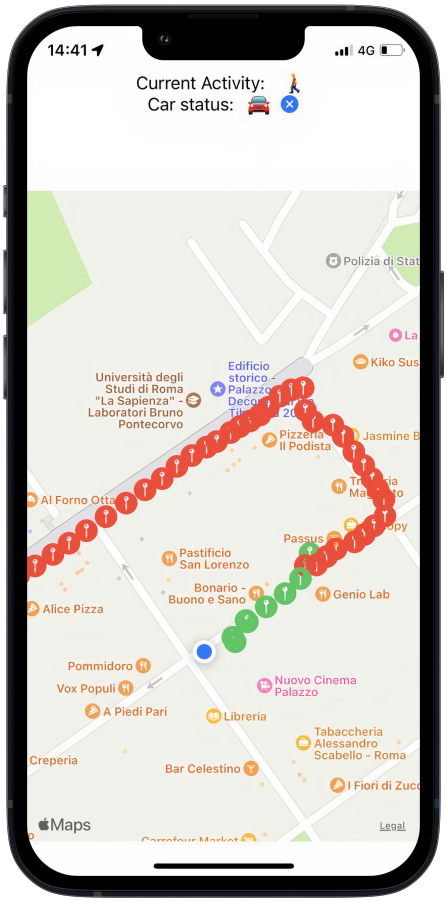


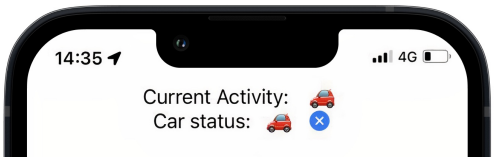
Figure 2.2. Prototype Interface

The app shows the *parked* status when the user stops the ride and takes a few steps. It takes around 10 minutes for a user to run this demo, starting from the first *walking* status to the final *parked* status.

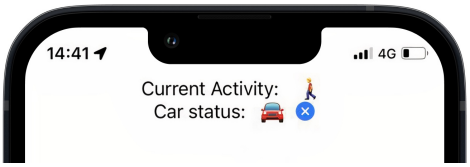
2.4.3 Discussion

In this section, we presented an entirely implicit interaction-based application to automatically detect when the user enters and exits their car. The system is based on detecting changes between the two motion activities *walking* and *automotive*. Once a change in motion has been detected, the *Parking* and *UnParking* events are inferred and localized through the GPS sensor of the smartphone.

To test the app's performance, we proposed a demo in which, due to physical space limitations, we expect to use a bicycle in place of a car. The interface of the app shows to the tester the changing motion states, the status of the vehicle (i.e.,



(a) User *automotive*, car *unparked*



(b) User *walking*, car *parked*

Figure 2.3. Detected user and vehicle status examples.

parked or *unparked*) and records their locations on a map in real-time.

2.5 Cruising-for-Parking Detection on the Smartphone Based on Implicit Interaction and Machine Learning

Finding on-street parking in a city is often difficult, as several factors can make it challenging to find a spot, such as limited parking spaces, high demand for parking, or restrictions on when and where it is possible to park. Searching for parking is done by sight: often, space becomes available in a nearby street, which the motorist has just crossed a few seconds or minutes before, but the driver is unaware of that. When the driver returns to the street, someone searching in the same area has already occupied that place. In the end, most drivers find a spot but at the cost of wasted of time, fuel consumed, increased traffic, and frustration. Wandering with a car in an urban area looking for an available parking spot is often called *cruising for parking*, [192], and constitutes excess travel due to parking search [214]. Cruising for parking drivers participate in traffic congestion and are partly responsible for the rising rates of harmful emissions [192, 87]. To address this issue, literature proposes smart parking systems as a way to provide drivers with information about available parking spots in an area.

With this work, we aim to address the following research questions:

- RQ1: Is it possible to detect in real-time if a driver is cruising for parking from smartphone context data, without any explicit hint by the driver?
- RQ2: How fast can we do that since the driver starts looking for parking?

This problem appears significant because knowing that a driver is cruising for parking would allow a system to provide timely information about available parking spots or parking restrictions in their area. The system could leverage implicit interaction, i.e., knowing that the driver began cruising, it could start looking for the relevant information to provide without requiring any explicit action on the driver's part. Implicit interaction would help avoid driver distraction and dramatically enhance the usability of a smart parking system. Finally, seamlessly and unobtrusively offering parking services will enhance the acceptability of the smart parking system and help reduce cruising, air and noise pollution, and driver frustration.

We provide the following contributions:

1. We provide an extensive and specific literature review of previous works on cruising for parking detection.
2. We report on several users' conscious and unconscious behaviors while searching for parking that we observed as the results of an interview-based study to analyze driver habits, proposing our cruising behavior modeling approach.

3. We designed a system based on a Boosted Tree (BT) classifier to real-time detect cruising for parking. Our method involves analyzing the drivers' GPS traces and feeding the classifier with the driver's speed, changing locations, and previous parking spots as features. The classifier was trained and tested on real data (615 car trips) collected by 9 test users. We describe the data collection phase, the classifier training, and validation and report the results showing that the model can detect cruising with very high accuracy.

Roadmap. In section 2.5.1, we describe our approach to analyzing the problem. In section 2.5.2, we present the experiment and the data collection, and In section 2.5.3, we dive deep into the BT model training. We describe the results In section 2.5.4. In section 2.5.5, we present the results discussion, the conclusions of the research.

2.5.1 Approach

2.5.1.1 Cruising For Parking

In this work, we focus on on-street parking. By on-street parking, we mean public parking places in urban areas that any driver can use, not reserved for a specific user. To park on-street, drivers generally need to look for an available parking spot, typically (especially in crowded urban areas and peak hours) traveling through different streets near their destination until they find a place: they need to *cruise for parking*. Thus, private garages, off-street parking lots, and any situations where the driver can park without cruising are outside this work's scope.

We approached this problem by analyzing the behavior of drivers cruising for parking. In 2018 we ran an informal study consisting of several discussions on this topic with students, colleagues, and other people outside our university. We discussed only with drivers, some of whom had little driving experience (less than three years), and some were experienced (more than 15 years of driving experience). We conducted the discussions without a predefined set of questions. However, in each discussion, we tried to elicit 1) how they would describe themselves while cruising for parking and 2) how they could understand if another driver was cruising for parking based on their driving behavior.

With the first question, we collected many interesting driver habits that we report below:

1. Drivers tend first to reach the destination, and only afterward do they start cruising for parking;

2. Drivers generally search in streets known to have higher parking availability;

3. Drivers also search among less known streets, such as secondary and (if accessible) private streets;
4. Drivers tend to look on the same roads near their destination again and again, trying to step away as little as possible: therefore, they follow concentric paths around the destination;
5. Sometimes, if they are aware that it is difficult to find a parking spot at their destination, drivers start directly to look in an adjacent area with more free parking possibilities.

Many of us share these habits when looking for a parking spot. They all derive from our desire to park as close as possible to the destination, to save time, reduce walking or make less effort. Moreover, knowing the destination area might help the parking search task: the driver knows the secondary streets and which streets have higher availability and search there first; they can avoid reaching their exact destination first (point 1) and start searching when they approach the area, to reduce the cruising and continue on-foot if they find parking; they may even search in a different area in some cases (habit 5). As the second question, we asked them to share how they can recognize when a car in front of them is cruising for parking. The most frequent answers are:

1. The car slows down;
2. The car has an anomalous behavior (for example, it slows down until it stops, then it starts moving again, it picks up speed, it stops again, and so on);
3. The car turns on the direction lights, even though it does not turn;
4. The car tends to pull over to the side of the road.

These behaviors are fascinating to observe as they make us recognizable among other drivers while cruising.

Precedents in the literature support the results of our informal survey. For example, the distinguishing habits of a user looking for parking are compatible with those of Polak et al. [165] and Benson et al. [25].

2.5.1.2 Cruising for parking indicators

Based on the observations presented In section 2.5.1, we focused on some information that can help understand if the driver is cruising for parking:

1. if the driver traveled twice or more times on the same road;

2. if the driver slows down and moves in the same area longer than necessary;
3. if the driver approaches an area where they are used to parking.

Thus, our first practical approach to modeling a cruising for parking detection system involved analyzing and comparing users' GPS locations. Such hints are supported by precedents in the literature, as Montini et al. [131] [130], De Waerden et al. [68] [209] and Mannini et al. [118].

Note that information (1) and (2) are different, even if they overlap in some circumstances; for example, a driver may drive on the same road multiple times and remain in the same area range. Indeed, they indicate two different aspects of cruising behavior that can help cruising detection.

Based on these considerations, we defined two heuristics for our detection model. We describe them in the following subsubsections.

2.5.1.3 Double cross

Our first heuristic determines if a driver passed over the same road or intersubsection more than once.

By collecting GPS data, we can track the car's position with high frequency and suitable accuracy and compare subsequent positions to assess if they overlap.

More specifically, starting from the beginning of the trip, we store the car position (latitude and longitude), assuming that it is the center of a 50-meter radius circle. Then, every 5 seconds, we compare each new position with the previous ones, performing the following checks:

- if the current position is outside the latest circle, a new circle is set around the current position;
- if the current position is inside one of the previous circles, then the heuristic *triggers, i.e., the driver passed twice on the same road or intersubsection.*

Once a certain amount of time and a distance threshold have been exceeded, going over the same road twice is no longer a strong indicator of cruising for parking. For this reason, we keep only the latest 50 circle centers at 12 samples per minute, which also helps to make a low number of comparisons at each new position, which results in speeding up the process.

2.5.1.4 Time in range

The second heuristic attempts to establish whether the user spends more time than needed to cross a zone. As for the first heuristic, we monitor the car changing

positions. However, we also take into account the time factor in this case.

Starting from the first position of the trip, we center a circular range around the current position. The circle is bigger than the one used for the first heuristic (we now use a 200-meter radius). In addition to that, we set a timer for 75s. While the car is moving and the timer is running, we compare each new position to the latest circle, carrying out the following checks:

- if the car position is outside the latest circle, we set a new circular range in the current position, invalidate the current timer, and start a new one;
- if the timer fires and the car position is still within the latest circular range, the heuristic *triggers*, meaning that the driver slowed down and spent more time than needed in that area.

We computed the 75s duration of the timer by averaging the speed of cars in an urban context, considering an approximate speed of 30 km/h (8.3 m/s; consistent also with the data we collected, see Table 2.2); hence, a car should take about 24 seconds to travel a distance of 200 meters. Allowing for a tripled duration, which we arbitrarily chose, we can safely assume that the user is outside the circle within that time if the car is proceeding on an almost straight path. On the other hand, if the timer fires before exiting the circle, it is likely that the user slowed down, made some turns, and headed back, or started to follow concentric paths. As previously mentioned, such behavior is typical of those cruising for parking.

We considered that the driver might waste time in traffic or stop at traffic lights: it is not reasonable to assume that the car is continuously moving. We addressed this issue by tracking the car's speed. Using a minimum speed threshold, we could detect when the user slowed down or made a stop, and we paused the timer resuming it when a higher speed was detected.

2.5.1.5 Previous Parking Spots

As observed In section 2.5.1, users' previous parking spots may play a key role in determining whether they are cruising for parking. Indeed, one of the central aspects that emerged by analyzing the beta testers' trips was the presence of a parking routine. Each user tended to visit the same places often and, if necessary, always search for parking in the same way, following repetitive patterns. Thanks to the prototype app we developed, it was possible to easily collect, for each monitored user, the coordinates of places where they parked their cars. Secondly, we processed those coordinates to infer useful information. More in detail, we treated each parking by the same user as a graph node; we placed an edge between pairs of nodes only where the two associated parking places were less than 50m apart. The edges were weighted

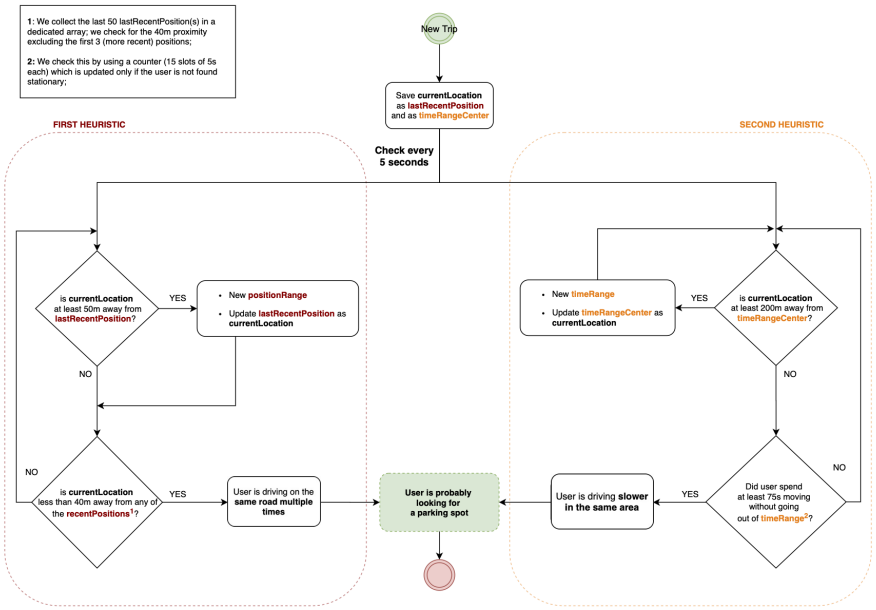


Figure 2.4. Heuristics Algorithms. The final step ("User is probably looking for a parking spot") can be triggered by any of the heuristics, even simultaneously.

inversely to the distance between the associated parking spaces. In addition, we gave greater importance to the connection between parking spots carried out in the same time slot during the day.

Once we identified the Parking Clusters for each user, we had to develop a criterion to distinguish the clusters that represented a private parking zone and clusters that represented parking spots found on the street. This distinction is important because it affects how users behave when they drive toward a destination: users will not look for parking if there is a private parking zone. Otherwise, they will likely start to do so (it may always happen that the driver gets lucky and immediately finds a place to park). We consider private any parking place that does not require looking for parking, be it a private parking lot or an area where locating a parking spot is very fast, such as a dedicated parking facility or a countryside area.

We established the distinction criterion by calculating the average distance between the cluster's parking spaces and their midpoint. Indeed, we obtained a sufficiently descriptive value (called *avDistance*) of the type of cluster. By comparing clusters of parking spaces known to be private and clusters that were not, we found that private clusters tended to have an *avDistance* value smaller than 30m, primarily due to GPS location accuracy. This criterion is based on the observation that if the type of parking lot is private, the user will tend to park in the same spot, with minimal variation in the distance between one parking and another. However, we also had to add the number of parking spaces in the same area as a threshold. In fact, where we recorded very few parking spaces, the distance between them could

be accidentally very small and mislead the identification. The classification work between private and public clusters has been more thoroughly depicted in Panizzi et al. [152].

2.5.2 Experiment

2.5.2.1 Collection of Data

We developed a mobile application on iPhone to collect data during car trips. The app detects automatically when the user enters the car (*unpark*) and exits it (*park*) by detecting the Bluetooth connection between the smartphone and the car. It collects smartphone positions and speeds at a high rate with GPS precision (multiple GPS locations per second), from *unpark* to *park*, plus other user-generated data. In addition, the app stores all the necessary data to run the heuristics described above.

We distributed the app to **9 drivers** who own an iPhone and a Bluetooth-equipped car. The user group consisted of 5 women and 4 men; the age of the participants is heterogeneous and includes 6 users between 22 and 25 years old and 3 users over 45 years old (mean=33.11; SD=13.48; min=22; max=55). 5 out of 9 testers own a private parking spot, either at their home, their place of work, or both. All drivers have more than a year of driving experience; people aged 40+ have more than 15 years of driving experience. Before the beginning of the trial, all drivers claimed to drive as part of their daily routine to reach destinations of interest, such as work or study places.

We conducted a longitudinal study, logging user activity from September 2020 to January 2021. The users were asked to install the app and pair their smartphones with their cars through Bluetooth. The app monitored all car trips for each user, tracing their paths and collecting all the data. The app works in the background and does not require interaction to collect inertial data. Nevertheless, users can state when they start looking for a parking spot by tapping a button in the main view. The app records the timestamp of this action, i.e., the cruising phase start moment, together with the trip data. We asked users to collect such information whenever they could, but this task was not mandatory. As for the rest, we encouraged users to avoid possible interaction with the application since it could result in behavioral changes, potentially altering the collected data.

During the data collection phase, the app recorded **615** car trips made by the nine users, despite the decrease in mobility due to the pandemic[1]. Users 0 and 1 continued to collect data for two additional months besides the three months we asked all participants. Thus the study initially planned for three months ended in January 2021 for these users. As instructed, users generally kept the app in

[1]COVID-19 Pandemic

background mode, where it silently gathered information. Only some users declared explicitly they were starting cruising; we collected only 19 records of the cruising start moment. It is possible that users have not provided this information because they systematically tended to forget to do so while driving, as distracted from the task in progress (search for parking). As presented in Table 2.1, users 5 and 7 collected drastically fewer trips with respect to the others. This data indicates that they used their car less than they claimed before the study (once per month, on average). Excluding users 5 and 7, the average number of trips recorded indicates that the users who took part in the test use their car at least three times a week. Users 0 and 1 stand out for the number of collected trips, but it was an expected outcome as they collected data for longer.

User #	0	1	2	3	4	5	6	7	8	TOTAL
Total trips	184	104	70	52	68	7	40	6	84	**615**
Cruising trips	75	7	9	3	38	2	2	2	33	**171**
On-street trips	59	38	46	27	30	5	8	4	51	268
Private trips	50	59	15	22	0	0	30	0	0	176

Table 2.1. Trips per user

2.5.2.2 Data labelling

We visually inspected each retrieved trip, plotting the recorded locations on a map to label the collected data as *cruising* or *non-cruising trips*. During this phase, we were aided by the testers in order to identify recurring trip destinations (e.g., home, place of work, and other relevant places of interest) and to identify private parking locations (e.g., private garages). To speed up this phase, we used the two heuristics *Double Cross* and *Time in Range* for the first screening. From this step, we have identified **171** trips during which at least one of the two heuristics was triggered. We labeled these trips as *cruising trips*. We manually inspected the remaining **444** trips and found that they did not show typical cruising behavior, as the users proceeded directly to their destinations. A further distinction was then made within such trips, as they may end in an on-street parking without a cruising phase (**268** *on-street trips*) or end on a private parking spot (i.e. a parking spot included in a Private Parking Cluster; these **176** trips were labeled as *private trips*).

A summary breakdown of the collected trips is presented in Table 2.1.

2.5.2.3 Ground truth

In order to evaluate the *Double cross* and *Time in range* heuristics (and the following ML models), we needed a ground truth that distinguishes the two phases of any car trip, *travel* or *cruising*. After a few attempts, we defined a criterion,

looking at journeys already completed in their entirety and consulting the tester drivers. Given an ended trip, we draw a circle of 200 meters radius centered on the parking position; then, we look at the trip data to determine the moment the user first entered the circle. That moment represents the cruising for parking phase start time, while we consider all the previous trip points as belonging to the travel phase.

This criterion is chosen due to the observation mentioned In section 2.5, according to which users tend to approach their destination first, and only then start looking for parking, if necessary. Moreover, questioning the tester drivers, we individually identified for each of them the areas within which they would consider acceptable to park when approaching their most frequent destinations. Combining the collected information resulted in the average value of 200 meters radius. To support our choice, we also used the data collected during the 19 trips where users explicitly marked the beginning of the cruising phase. The distance from the parking spot of such trips averages 158.21m, which is compatible with our 200m assumption. Moreover, as shown by Table 2.2, we obtained comparable figures even considering the cruising phase's average duration and user speed.

Finally, our approach is also supported by previous literature. As hinted by Montini et al. [131] [130], without explicit indication from the driver, detecting the exact moment of cruising start has to be done by estimation. For this reason, previous studies also propose a personal evaluation metric based on empirical trials, and there is no fixed common guideline. Montini et al. [130] proposed a range of 800 meters, supporting the choice as an overestimation of the previous boundaries of parking search of 350m found by Weinberger et al. [213]. Millard et al. [126] did not need a radius estimation, as they performed the cruising detection based on the ratio between excessive travel and the shortest path to the destination. However, their results suggest that the average cruising area begins 400m before the destination. Similarly, in modeling the cruising behavior, Levy et al. [109] and Benson et al. [25] conclude that the search starts between the 100m and 400m from the destination. Also, Waerden et al. [209] did not use a distance range but set the starting point of searching when the speed meets certain thresholds estimated during previous studies [68]. Our heuristic of 200 meters falls in the range of previous estimations (100-800m).

Trip Phases	Average Duration	Average Speed
Complete Trip	948.45 s	27.70 km/h
(200-m-distance) Cruising Phase	497.36 s	16.96 km/h
(User-declared) Cruising Phase	298.42 s	16.22 km/h

Table 2.2. Ground Truth Evaluation

2.5.2.4 Evaluation of the Heuristics

Tables 2.3 and 2.4 summarize the results of the heuristics evaluation. We value a heuristic detection as a True Positive (TP) when it triggers within the 200m ground truth, even if another TP detection already occurred during the trip. On the other hand, a detection is considered a False Positive (FP) if it occurs too far from the final parking location, or too far in time. We observed the latter circumstance with users with long trips ending near the starting point. The performance of the two heuristics counted no False Negatives. As mentioned In section 2.5.2.2, we manually inspected the trips where the heuristics were not triggered, confirming that those were not *cruising trips*. Another positive outcome is given by the Average Distance from Parking values (Table 2.4), which corresponds to the average distances between the final parking of the trip and the point where was triggered for the first time one of the two heuristics. Indeed, the reported average distances are less than 200 m (the assumed ground truth), and they are also informative of the users' tendency to remain very close to the destination when looking for parking.

An interesting piece of information, already anticipated, that can be deduced from Table 2.3 is the fact that the trips during which both heuristics were triggered are 73.1% of all the trips with a cruising instance, which demonstrates the similarity between the two heuristics but also confirms the fact that it was the right call to divide them.

Finally, looking at Table 2.4, it can be noticed that the *Double Cross* heuristic tends to be triggered more than once per trip, a phenomenon which does not often occur in the case of the *Time in Range* heuristic. In percentage, the latter heuristic is even more prone to false positives.

Total number of trips	615
Number of trips with at least one TP*	171
Number of trips where at least one heuristic was triggered	222
Number of trips where both heuristics were triggered	125

*Where at least one of the two heuristics detected a cruising for parking instance correctly.

Table 2.3. Heuristics Evaluation

	Double Cross	Time in Range
Number of trips where it was triggered	197	150
Times it was triggered	322	191
True Positives	281	155
False Positives	41	36
Average Detection Delay	265.86 s	264.59 s
Average Distance from Parking	173.8 9m	166.46 m

Table 2.4. Heuristics Evaluation - Comparison

2.5.3 Machine Learning Models

2.5.3.1 Defining the Cruising for Parking Problem

By analyzing the results obtained by the two heuristics (Tables 2.3 and 2.4), it was noted that although the number of false positives was low and no false negatives were counted, the detections, despite being accurate, arrived with a not negligible delay (about 3.5 minutes on parking searches that lasted an average of 8 minutes, see Table 2.2). As a consequence, it was decided to leverage a machine learning-based algorithm.

The cruising for parking detection was modeled as the problem of having to establish whether, given a portion of the car trip, that portion was part of a parking search or not. In our model, a car trip is defined as a sequence of **trip points**, and a portion of it is a single trip point, which corresponds to approximately a second of travel.

2.5.3.2 Observations on collected data

A significant flaw in the data collected concerns the imbalance between the number of trips carried by each user. Indeed, 2 out of 9 users contributed to more than half of the trips monitored. The remaining 7 users are divided into 5 users who have contributed similarly, and two whose trips make up less than 2% of the total dataset.

The main disadvantage, however, derived from how the cruising detection problem was modeled, as by considering the trip points of a given trip separately, the number of negative examples (all those trip points belonging to the *travel* phase) is significantly higher than that of positive examples (trip points of the *cruising* phase); this phenomenon is due both to the fact that the traveling phase in generally longer than the cruising one, and also to the scarcity of cruising trips collected (Table 2.1).

To alleviate the imbalance between *false* and *true* cruising instances, it was decided to pick, for the training (Table 2.5), **200 trips**, of which almost half (**98**) were *cruising trips*, whilst the remaining were divided between *on-street* and *private trips*. The *cruising trips* were randomly picked from the users who collected the most trips. In addition to that, the start of each trip was truncated in order to obtain a number of *false* cruising instances equal or less than that of *true* cruising instances.

For the testing phases, on the other hand, a separate dataset was used containing the remaining trips (total of **415** trips, **73** of which were *cruising trips*). By testing the models on users never seen before (assuming that they have habits that tend to be different from the users used during the training phase), it is possible to have a more meaningful evaluation of performance. The main goal for the near future is

to expand the training set as much as possible (also in terms of data variation) to improve the models' overall performance, but the accuracy of **85%** obtained in the meantime is still very promising.

Dataset Type	Samples	CruisingFalse	CruisingTrue	Trips
Training	90737	53023	37714	200
Testing	118572	90433	28139	415

Table 2.5. Datasets

2.5.3.3 Adopted Machine Learning Model

Given the formalization of the problem, the best choice resulted in adopting a Boosted Tree Classifier. To achieve such a decision, different machine learning models and datasets were defined and compared.

To this matter, it is worth reporting that, while designing the first models, it occurred that classifying single trip points may lack the time dimension, i.e. the searching status of each trip point was computed independently from the status of the immediately preceding instants. Indeed, the cruising for parking detection problem may be interpreted as a Time Series Forecasting problem, which can be modeled with a Convolutional Neural Network. Consequently, several CNNs were built based on different combinations of design choices which concerned the number of features selected and the length of the time samples. However, although they were more sophisticated models than the BTs, and hence better outcomes were expected, the accuracy of the CNNs-based systems never exceeded 65%. We believe that these unsatisfactory results are due to the scarcity of data we had available. Therefore, we defer to future works a possible refinement of the detection system based on CNNs.

The proposed Boosted Tree Classifier leverages the following features:

- **date - Float64**

  Float64 parameter based on the timestamp describing the time of collection. This information was used and manipulated in different ways depending on the ML modeling choices.

- **avSpeed - Float64**

  Average speed in m/s computed over the last 60 trip points.

- **doubleCross - Boolean**

  A parameter that indicates the results of the first heuristic. It is set to 1 from the instant in which the heuristic detected a double-cross (if any). This parameter divides the trip into two phases and it resulted to be the most significant during the learning process.

- **timeInRange - Int32**

  A parameter that describes the number of seconds spent in the ranges monitored by the second heuristic.

- **clusterProximity - Double**

  A custom computed value describing the user's location with respect to their Parking Clusters.

The Boosted Tree Classifier achieved 88.45% accuracy during training and **84.8% accuracy** during testing (Table 2.6). The detail of the evaluation is shown in the confusion matrix in Table 2.7. Table 2.8 present the evaluation details regarding precision and recall.

Features	Training Accuracy	Test Accuracy
date, avSpeed, clusterProximity, doubleCross, timeInRange	88.45%	84.8%

Table 2.6. Boosted Tree Classifier Evaluation

2.5.4 Final Cruising for Parking Detection System

2.5.4.1 Thresholds and Final System

True/Prediction	CruisingFalse	CruisingTrue
CruisingFalse	80264	10169
CruisingTrue	8473	19666

Table 2.7. Boosted Tree Classifier Confusion Matrix - Testing

	Precision	Recall
CruisingFalse	90.45%	88.76%
CruisingTrue	65.92%	69.89%

Table 2.8. Boosted Tree Classifier Confusion Matrix - Precision and Recall

The ML Model chosen to be the base of the final cruising for parking detection system is a Boosted Tree Classifier. During testing, it reached an accuracy of nearly **85%**. However, although it may seem a high percentage, the model performed poorly during real-time tests, resulting in an unacceptably high number of false positives. In fact, an average car drive lasts around 15 minutes, i.e. about 900 trip points whose status needs to be predicted by the model. Being 85% accurate means that, on average, the model will rightly classify around 765 seconds, leaving 135 possibilities of error in just one trip.

Understandably, it was necessary to establish a threshold of minimum consecutive detection predictions and with certain confidence above which the parking search detections made by the model were considered to be true. Therefore, the performance of the model was evaluated by trying all possible combinations between confidence thresholds (parameter **confidenceThreshold**, tested on a [0.5,1.0] range) and a minimum number of consecutive prediction with such confidence (parameter **predictionsThreshold**, tested on a [0,20] range). The trips used for this testing phase were those already used in the testing dataset (Table 2.5), excluding 39 trips that were too short (less than 3.5 minutes). Indeed, 376 trips were used, which counted 72 *cruising trips*, 108 *private trips*, and 196 *on-street trips*.

The adopted evaluation parameters were precision, recall, and accuracy.

Finally, the best couple of parameters resulted in being **confidenceThreshold = 0.95** and **predictionThreshold = 0**, which have raised the value of the accuracy (calculated on the totality of the trip) up to **93%** (Table 2.9).

Number of trips	376
Expected True Positives	72
Expected True Negatives	304
True Positives	58
True Negatives	294
False Negatives	14
False Positives	26
Trips containing FP	12
Average Detection Delay	177.43s
Recall	83%
Precision	81%
Accuracy	93%

Table 2.9. Final Cruising Detection System Evaluation

2.5.4.2 Final Cruising Detection System Evaluation

As shown in the graph pictured in Table 2.5, in the majority of cases, the detection system captures a cruising for parking instance with a delay between 100 and 200 seconds. On average, as reported in Table 2.9, the detection delay counts 177.43s (a significant saving with respect to the 265s obtained by the two heuristics, Table 2.4).

As for the accuracy evaluation, it should be noted that, given the imbalance of the testing dataset (i.e. the large presence of trips without parking search), high accuracy values are easily reached even if the model always guesses for non-cruising. However, by observing the low percentage of false positives, the even lower number of trips in which they are gathered, and, on the other hand, the good percentage of

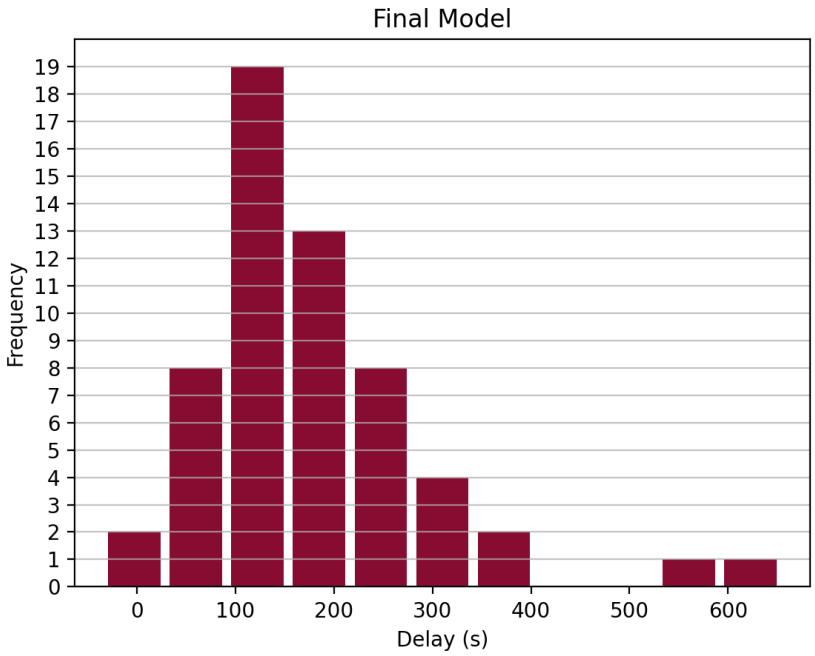


Figure 2.5. Cruising for parking detection delays

true positives (80% over expected result), we believe that the obtained performance is very promising. Indeed, the model tends not to make negative mistakes, and especially with respect to the intended use of this research, it tends not to give rise to false parking searches.

2.5.5 Discussion

This section presents a context-aware approach for detecting cruising for parking in real-time using smartphone sensors. We have identified several indicators of cruising behavior, such as the heuristics Double-cross and Time in range, and collected data to train and evaluate our machine learning models. Our final cruising detection system achieved high accuracy and can be used to provide drivers with parking availability information and reduce traffic congestion.

As main contributions, we provided an extensive literature review on relevant works. Then, we reported on several users' conscious and unconscious behaviors while searching for parking that we observed as the results of an interview-based study to analyze driver habits, which is supported by previous literature. We proposed an approach to model cruising for parking behavior that involves analyzing the drivers' GPS traces and referred to previous literature to support its validity. Finally, we designed a system based on a Boosted Tree (BT) classifier to near-real-time detect whether the driver is cruising for parking.

The classifier's features include the driver's speed, changing locations, and previous parking spots, addressing RQ1. The final cruising for parking detection system can recognize, with a high level of accuracy (**93%**) and a nominal delay rate (177.43s), when a driver is looking for parking, addressing RQ2.

The main limitations of our work are the use of a dataset of only 9 users and the lack of the timestamp of the beginning of the cruising phase for each trip considered. However, these limitations are shared by other precedents in the literature, including Montini et al. [130] for the definition of the ground truth and Waerden et al. [209] for the experiment carried out on a small number of users. Our proposal to alleviate the missing ground truth issue is estimating the start of the cruising area 200 m before the destination, in line with previous studies that averagely reported radiuses between 100 and 400 m. Regarding the possibility of having a higher number of users, we postpone a more extensive study to future works.

Another limitation of our work is contextualizing the experiment in a specific geographical area (Rome, Italy). However, precedents in the literature often restrict the experiments as well (examples are Montini et al. [130], Waerden et al. [209], Dalla et al. [60]). Furthermore, our approach is based on an analysis of cruising behavior that generalizes from the specific context, which is indeed supported by previous studies. In detail, none of the model's features is context-dependent. However, the training was conducted on users that traveled the same areas, possibly indicating a chance of over-fitting. Moreover, the last feature implicitly requires using the model extended over time, as it is based on previous parking lots.

2.6 An approach to leverage Artificial Intelligence for car-parking related mobile applications

Mobile applications have become an integral part of our daily lives [203], and the potential for leveraging artificial intelligence (AI) to facilitate and enhance user experiences within these applications is vast. AI can support or replace repetitive and demanding tasks, offering users a more varied and efficient interaction with the application [15]. However, despite the increasing attention and interest in AI across various fields, integrating AI-based functionality into mobile applications poses unique design and implementation challenges, such as individuating the correct set of features that can benefit from AI and the gathering of quality data for training. Moreover, the limited processing power and memory must be considered, as well as battery life.

Our research mainly focuses on context-aware smartphone apps designed for the smart parking context. This domain presents a compelling opportunity to harness the power of AI to provide valuable features and services to users, who will no longer need to enter data by hand but can rely on AI to understand their needs. Many drivers rely on their smartphones to assist them in performing tasks while on the road, but this can pose significant safety risks as actively using the smartphone distracts them from driving. We aim to mitigate distractions and enhance the driving experience by leveraging AI-based features in context-aware parking applications. Such apps can provide relevant and timely suggestions by intelligently adapting to the user's situation, making parking-related tasks a seamless experience.

Our group also used this approach in other contexts, such as more critical situations, such as analyzing the smartphone microphone to understand whether a person around it is breathing in post-earthquake scenarios [22].

This section presents our approach to addressing the challenges of integrating AI into mobile applications. We strongly focus on smartphones, which are highly versatile and powerful enough to collect and process contextual data. Once a user need is identified, we use context data to train dedicated machine learning models. The models, running directly on the smartphone, help predict the user's needs, facilitate the execution of tasks, and sometimes directly replace the user. Examples of parking-related features we addressed by adopting the described approach are identifying events such as parking, un-parking, and approaching a parked car [39][33]; classifying parking types (parallel, diagonal or perpendicular parking; private vs public parking) [23, 152]; detecting cruising for parking [40]; estimating the level of parking availability on a street-level scale [20], etc.

2.6.1 Engineering

This work presents the development of an interactive system designed for the domain of parking, which we also experimented with in the context of earthquakes [24]. In order to minimize distractions, our smart system can intelligently adapt to the user's situation, automatically understanding the context (driving, looking for parking, parked, etc.) and providing parking recommendations. In our system, we prioritize the user's involvement, creating a collaborative environment that provides a positive user experience and continuous system improvement. Making an AI-powered application where the user feels in control is not an easy task; in the following sections, we report our experience on how users respond to different methods of feedback requests from the system and what we found to be both engaging for the user and feasible for the system.

Moreover, we present a systematic approach to creating AI smartphone applications that describe the entire process, from identifying automatable features to data gathering, model learning, and release.

2.6.2 Approach

Our approach can be summarized in four steps:

1. identify the task that can be automated and the degree of AI involvement. In order to select the right task, the users' behavior and needs must be thoroughly analyzed to understand where automation is functional. On the other hand, AI involvement depends on the type of available data and the capabilities of devices.

2. Data collection for training. We release a companion application that collects users' feedback on the task, creating the first dataset used to train the machine learning model.

3. Train a machine learning model over gathered data. This passage can be iterated when a significant new amount of data is available and if the application has a personalized approach and thus must adapt to users' habits.

4. Release the app and allow user feedback to refine the data labeling. This allows for continuous updates to the running model.

2.6.2.1 Identifying the task

Identifying tasks that can be automated while ensuring a positive user experience can be challenging [122]. It is crucial to strike a balance where users feel in control of the system while benefiting from AI-powered automation.

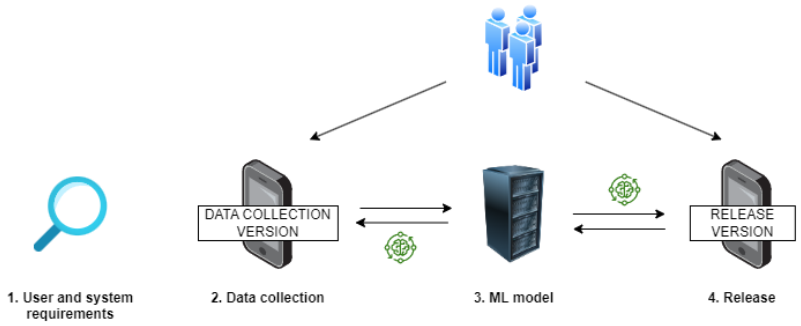


Figure 2.6. A visualization of our learning loop

During the identification phase, it is essential to begin by understanding the user's needs. This step involves gaining insights into their habits, needs, challenges, and areas where automation could enhance their experience. By empathizing with the user and comprehending their requirements, we can lay the foundation for effective AI integration.

Subsequently, it is vital to determine to which extent AI can intervene by identifying the appropriate model and the necessary features and data. This step entails evaluating the task's complexity and the feasibility of using AI.

For example, a significant user need is getting help to find a parking spot. We thoroughly analyzed how drivers look for on-street parking places and identified average behaviors in cruising for parking [40]. AI can help detect when a driver is looking for parking and prompt the system to look up a database for a suitable parking spot, anticipating an explicit request, hence a distraction on the user's part and possibly saving cruising time. We designed an interaction that keeps the user in control by not forcing them to accept parking spot suggestions and letting them ignore any requests on the AI part (e.g., a notification advising the user that a nearby parking facility has available spots).

2.6.2.2 Identifying the context

Our approach to developing context-aware mobile apps for the driving context emphasizes assisting users as naturally and seamlessly as possible. Leveraging the power of smartphones, we capitalize on their capability to collect a diverse range of contextual data and computing power for model processing. Important data that can be gathered from smartphones include readings from inertial sensors (such as the accelerometer, gyroscope, magnetometer, and GPS), weather conditions, temperature, time zone, lighting conditions, noise, and specific user-related information

like visited places, calendar events, and nearby users.

In order to tailor our services specifically to the driving context, we have developed an implicit interaction approach [39, 33] to identify when the user enters and exits a vehicle. Our solution leverages the smartphone's sensing capability of users' locations and motion activities and merges them to infer parking and un-parking events, avoiding any explicit interaction on the user's part. More in detail, such an approach relies on the machine learning models integrated into the smartphone, enabling recognition of the user's activity type (driving or walking) and simple decision algorithms to refine the models' detection, further enhancing their accuracy and augmenting the activity detected with a location and a timestamp. By isolating the driving context, we can provide personalized and relevant services accordingly, triggering even more context-specific features. For example, the above-cited cruising detection feature is triggered only if the user is detected to be driving. Currently, we are developing another AI that focuses on recognizing the user's means of transport, such as cars, buses, trams, or metro. Such an AI can augment and refine the detection capabilities currently available on standard smartphones, based on the device bluetooth connections.

2.6.2.3 Gather Data

Training machine learning models necessitates a substantial amount of data. Consequently, our standard practice is to develop a companion version of the target application to facilitate the collection and labeling of data.

By distributing a companion app, we can streamline the data collection process. Such apps incorporate HCI principles [132], ensuring ease of understanding, intuitive interactions, and a visually appealing interface. Adhering to these guidelines enhances user engagement, encourages user participation, and ultimately contributes to the quality of the collected dataset. Indeed, when users find the application easy to understand and navigate, they are more likely to provide accurate and meaningful labels, enhancing the overall effectiveness of the training dataset.

For the means of transport recognition project, the data collection app we developed warns users when it detects that they have entered a state of motion on a motorized vehicle. Upon receiving the notification, users can respond promptly by selecting the specific type of transport used, enabling the collection of labeled training data. However, developing this app presented its challenges. One of the critical difficulties we encountered was that users occasionally overlooked the notification and needed to input the data manually later. To mitigate this, we introduced an editable past travel history feature in the data-collection app that allows users to label the collected data after the trip is finished (see Figure 2.7). This enhancement lets users label their data retrospectively, ensuring a more complete and accurate

dataset.

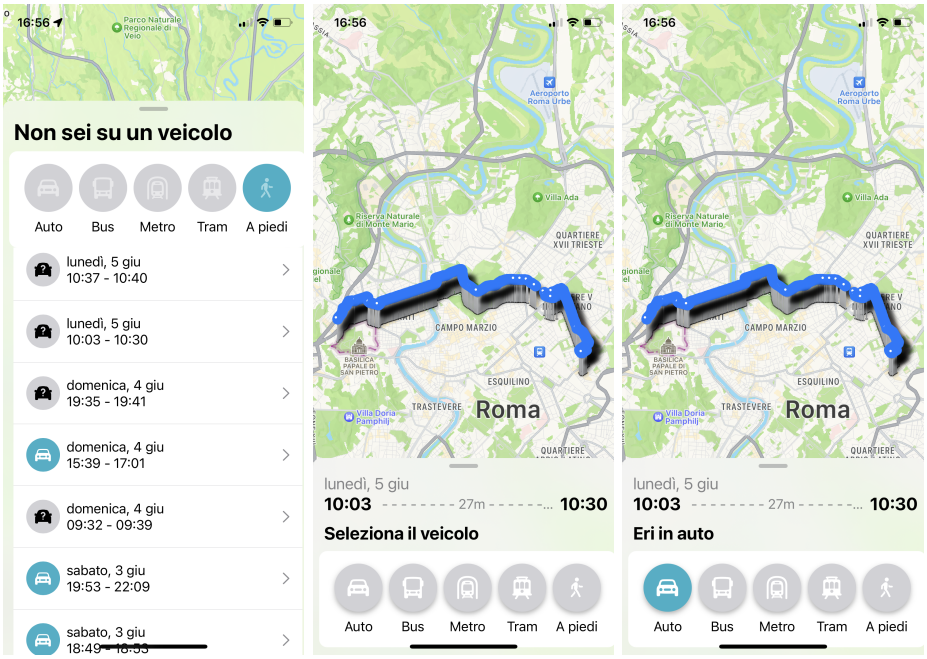


Figure 2.7. Example screens from the data collection app to recognize the user's means of transport. In this version, the app does not yet implement a recognition model. Therefore, the labeling of each trip is done manually by the user. On the left: "You are not in a vehicle". The user can assign a vehicle type to each of their previous trips. Center: "Select the vehicle". Upon selecting a trip from the history, the trace of collected GPS locations is shown to the user, along with the day and time it took place, as a reminder. Right: "You were in a car". The user labeled the collected data by selecting the means of transport.

2.6.2.4 Model training and distribution

We train our AI models on dedicated servers. We train a prediction model on the data collected by the companion app; once the model reaches the target accuracy, it is distributed to the apps and can be run locally. As described in the next section, the released application will continue to gather data that can be used to update the model over time. In the future, we plan to implement personalized applications that learn the habits of individual users; in this case, each user will have a different model, which will be refined over time using only the feedback given by that user. Our approach is to design and develop models suitable to be run directly on the user's device. This approach minimizes reliance on external computing resources and prioritizes user privacy in the final application. Indeed, data privacy can be upheld by keeping the data stored on the user's device and only occasionally transmitting it to the server for future model updates. Moreover, it removes the delay that would introduce the communication towards an external resource, allowing for better use

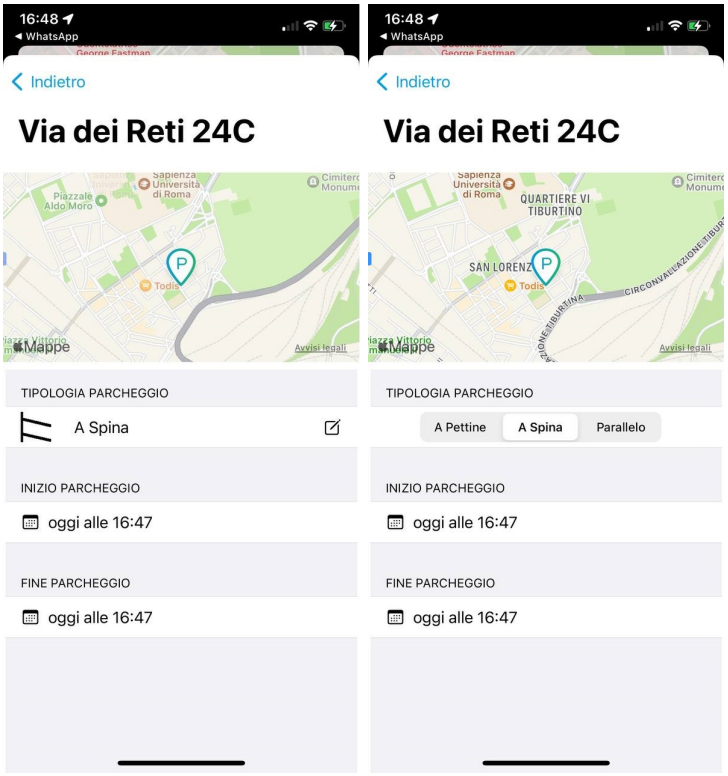


Figure 2.8. Example screen from the released parking app. On the left: “Type of parking". The app predicted the type of parking “angle". On the right: the user can correct the prediction by selecting a different type of parking, “perpendicular", “angle", or “parallel".

of models designed for critical contexts.

2.6.2.5 Release, feedback, and Human-AI collaboration

Once the model is distributed and operational, we prioritize the user’s involvement and empowerment by allowing them to offer feedback on the generated predictions. This feedback mechanism is crucial for collecting valuable insights and identifying potential issues or inaccuracies in the model’s output. By actively encouraging user feedback, we establish a collaborative environment where users play an active role in improving the system’s performance and enhancing their experience.

Furthermore, we acknowledge the significance of transparency and user control in AI-driven systems. To ensure users always remain in charge of the system and clearly understand its operations, we enable users to revert any changes made by the AI easily. Through such a collaborative approach, we address potential issues, improve the accuracy of future predictions, and cultivate a trustworthy and user-friendly design.

Regarding this matter, we collected valuable feedback directly from users during

the test phases of an app that gathered different parking-related features. Users expressed their appreciation when the underlying AI understood context and user actions. For example, they were pleased when the AI accurately recognized the location and time of parking events and the specific type of parking. In contrast, users did not respond favorably when prompted to input information manually. For example, our app initially asked for the type of parking each time a parking event was detected (the aim was to involve the user regardless of the model's guess). However, we learned that this approach needed to be better received. We removed the notification and allowed the model to make an independent guess. We then made the model classification visible to users, enabling them to correct it if necessary. Users positively received this modification, as it gave them more autonomy and control over the data input process without the burden of repetitive requests for interaction. This process of keeping users in charge of the system and making them understand what is happening by also giving the option to undo actions taken by the AI is a crucial step to allow them to recover, understand, and avoid errors [35].

2.6.3 Updating models

The model can be updated by re-training it over newly collected data using user feedback. This is useful in two cases: when the app is released, and thus the availability of data is limited, and when the app must keep track of each user's habits. In the first case, the model is re-trained over the previous data plus the new feedback, and this is repeated over time until the model's precision stops growing. On the other hand, if the application must keep track of individual habits, the model will be re-trained daily in a sliding window fashion, where older data is gradually no longer used to make predictions. In any case, the updated model is distributed to the apps, allowing them always to give the most recent and accurate predictions. To do so, we need to have in place a system to pre-process the collected data, run the model training, save it, and let the apps know that there is a new version of the model available and that they need to update their local version.

We implement the described system as follows:

1. Run a Python script to download the collected data and pre-process it to organize it for the training phase.
2. Train the new model using the new pre-processed data.
3. Save the newly trained model and notify the server that a new version is available.
4. Update the latest model version number on the server to ensure that consequent requests to check for the model version by the apps will trigger the model update task [9].

5. (Optional) Send a silent notification to the app to let it know a new model version is available. This can trigger the model update task directly without the user waiting for it when the app gets opened next time.

This process enables us to enhance the model's accuracy over time and ensure that the data classification aligns with the desired outcomes.

2.6.4 Discussion

This section proposes an engineering approach to integrate AI into mobile applications, specifically within the context of car parking. Our research demonstrates the potential of AI in enhancing user experiences and facilitating tasks within mobile applications. Exploiting this approach, which leverages context awareness, implicit interaction, and machine learning, we have designed an AI-powered application that provides personalized and practical assistance to drivers.

Our approach addresses the unique challenges of AI integration into mobile applications, such as identifying tasks that can benefit from AI, balancing user experience with AI automation, and gathering quality data for training while preserving the user's privacy. We have shown that with careful design and continuous model updating, it is possible to create an app that meets evolving user needs.

In the context of smart parking, we illustrate the potential that AI applications can provide by giving real-time advice, enhancing the driving experience, and mitigating safety risks associated with active smartphone use while driving.

The described approach is generalizable to different domains that can benefit the application of Artificial Intelligence to mobile User Interfaces to obtain implicit and context-aware interaction. We successfully applied it to another context, post-earthquake presence recognition, demonstrating its versatility and potential for broader application.

Chapter 3

Implicit Interaction supported through BLE

3.1 Introduction

As mobility systems become increasingly automated, interaction design shifts from supporting *drivers* to supporting *occupants*. In this transition, vehicles and mobility services are expected to adapt to passenger presence, social setting, and situational context, which requires sensing not only *where* someone is, but also *who is around them* and how that environment changes over time [202, 83]. Traditional context sensing in mobility typically relies on GPS, inertial sensors, cameras, or multi-sensor fusion. While effective, these approaches can introduce trade-offs in terms of power consumption, deployment effort, or privacy, and they are not always ideal for passenger-focused applications that require unobtrusive, lightweight inference [199, 161, 140].

This chapter explores an alternative enabler for implicit interaction: Bluetooth Low Energy (BLE) visibility. Modern smartphones continuously emit and detect BLE advertisements, often using randomized addresses and without requiring pairing or direct connections [145, 10]. By treating smartphones as proxies for their owners, aggregate properties of nearby BLE signals such as persistence, turnover, and device density can provide a subtle but useful cue of co-presence. Importantly, this can be done without tracking precise location and without aiming to identify individuals, which makes BLE-based sensing attractive for privacy-sensitive and resource-constrained settings [184, 1]. The core intuition is that a small set of consistently observed devices over repeated scans is indicative of shared proximity within a confined space (e.g., a private car or robotaxi), whereas many transient devices suggest higher-density environments (e.g., bus or metro), enabling implicit inference of both passenger presence and broader mobility context [89, 3].

The two papers in this chapter develop this idea along a coherent path. First, we study BLE beacon visibility as a general signal for human presence and density in everyday environments, framing it as a low-cost alternative to camera- or infrastructure-heavy solutions and clarifying its privacy and deployment implications [199, 7, 227]. Building on this foundation, we then investigate how BLE-based sensing can support context-aware user experiences in automated mobility by detecting co-travelers and estimating vehicle type from the stability and volume of nearby devices, thereby enabling passenger-aware, adaptive interaction strategies [202, 89]. Together, these contributions position BLE as a practical sensing primitive for implicit interaction: lightweight enough to run on-device, expressive enough to capture social context, and flexible enough to generalize from mobility to broader ubiquitous computing scenarios.

3.2 Related Works

Transportation Mode Detection (TMD) using smartphones has been a long-standing research topic at the intersection of transportation science, mobile and ubiquitous computing, and HCI. The dominant approach is to infer mobility context from location and inertial signals, sometimes enriched with additional sensor streams. In contrast, our work investigates Bluetooth Low Energy (BLE) *visibility* as an implicit cue for co-presence and occupancy, treating nearby devices as proxies for nearby people. This direction is comparatively underexplored in TMD, yet it is attractive for mobile interaction because it can be lightweight, privacy-conscious, and directly informative of social context, which is increasingly relevant for passenger-centered and automated mobility experiences [202, 83].

3.2.1 Inertial and motion-based TMD

A large body of work relies on Inertial Measurement Units (IMUs), typically accelerometers and gyroscopes, to classify transportation modes from motion dynamics. These methods can be effective across a range of movements, and recent contributions continue to improve robustness and handling of practical issues such as class imbalance and device orientation. For instance, Xu [219] addressed imbalance through focal-loss-based training, while Van et al. [206] explored rotation-invariant feature representations. Earlier benchmarking work by Fang et al. [78] highlights the breadth of IMU-based classifiers and evaluation practices.

A recurring limitation, however, is that IMU-only systems can struggle to distinguish modes with similar acceleration profiles (e.g., certain vehicle types), and they often require frequent sampling to maintain accuracy, which can increase battery consumption. Moreover, while commercial mobile operating systems provide coarse

activity recognition (e.g., walking vs. in-vehicle), these abstractions typically do not differentiate specific transportation modes and may not expose detailed signals or model behavior to developers, limiting transparency and interface-level control.

3.2.2 Location and trajectory-based approaches

GPS and trajectory analysis represent another major TMD family, leveraging spatiotemporal patterns, speed profiles, and route characteristics. While GPS can be informative, it performs poorly in tunnels, dense urban areas, and underground transit, and it can be energy-intensive when sampled continuously. Several works therefore enrich GPS with contextual features, road-network constraints, or additional sensor streams. Sowlati et al. [194] reported that GPS alone was insufficient and improved results by integrating contextual and spatial information. DE et al. [64] incorporated road networks and other signals for improved robustness, while Ma et al. [115] proposed a deep learning framework on GPS trajectories, at the cost of substantial labeled data and computational complexity.

Overall, the literature consistently shows that combining signals tends to improve TMD accuracy, but it also increases system complexity, energy consumption, and permission requirements, which can be problematic for lightweight mobile applications.

3.2.3 Sensor fusion and non-traditional modalities

Multi-sensor fusion is widely adopted to improve robustness in real-world conditions. For example, Fourez et al. [79] and Chen et al. [52] demonstrated that fusing GPS with accelerometers can yield more reliable classification. Beyond IMU and GPS, other modalities have been explored. Audio-based approaches can achieve high accuracy in vehicle contexts, as shown by Lee et al. [106, 107], but continuous audio capture raises privacy concerns and may be unreliable in noisy environments. Barometric pressure sensing has also been used as a low-power complementary signal [177], though it often provides limited discriminative power on its own.

Environmental proximity signals, particularly Bluetooth and WiFi, have attracted interest as an alternative or complementary source of context. Mohammed et al. [127] and Bjerre Nielsen et al. [43] found that proximity-related cues can improve detection, especially in crowded public transport where GPS becomes unreliable. Coroama et al. [57] combined Bluetooth and WiFi proximity with traditional sensors and supported background collection and manual correction, showing the promise of proximity-aware pipelines. However, much of this line of work relied on Classic Bluetooth, which has discovery limitations and can be less suitable for continuous, energy-efficient sensing. In addition, proximity-only signals were often reported as insufficient for high-precision real-time mode classification when used in isolation.

3.2.4 BLE visibility for co-presence and context-aware interaction

BLE introduces a different sensing opportunity: smartphones can broadcast and observe BLE advertisements without pairing, and modern designs commonly employ randomized addresses, enabling passive detection without direct identification of devices [145, 10]. Compared to many traditional sensing pipelines, BLE scanning can be relatively lightweight and can avoid collecting precise location. This makes it appealing for scenarios where power consumption, deployability, and privacy are central constraints [184, 1]. Instead of inferring context purely from motion or trajectories, BLE visibility patterns can capture *social density* and co-presence: how many devices are nearby, how stable they are across scan cycles, and how frequently they appear or disappear. These cues are particularly relevant in passenger-focused mobility, where knowing whether a user is alone, accompanied, or in a crowded vehicle can directly inform adaptive interface strategies [202, 83].

Our work connects this sensing primitive to implicit interaction in mobility systems and to the broader research thread on proactive, context-aware smartphone support. Prior contributions in the parking domain demonstrate how passive smartphone sensing can enable task assistance while reducing user burden [40], and AI-enhanced mobile applications have been proposed to leverage passive cues to support or automate user tasks [31]. From an HCI perspective, these systems raise questions about transparency, evaluation, and user control in implicit or proactive interactions [38]. In response, we frame BLE-based inference as a support for interface-aware decisions that remain legible and user-correctable, aligning with prior discussions on managing uncertainty and supporting correction in AI-assisted systems [35].

3.2.5 Summary and positioning

Existing TMD approaches face well-known trade-offs [102]. IMU- and GPS-based systems can be accurate but may be power-hungry and privacy-sensitive. Audio methods can be effective but more invasive. Multi-sensor fusion improves robustness at the cost of complexity, energy consumption, and broader permissions. Commercial OS activity recognition is convenient but coarse-grained and often lacks transparency and controllability.

Within this landscape, BLE visibility complements established methods by focusing on co-presence and device-flow cues rather than only motion or location. This makes it well suited for intelligent mobility applications and passenger-aware, adaptive interfaces, where understanding social context can be as important as recognizing movement. The novelty lies both in the sensing angle (BLE as a proxy for occupancy and density) and in its integration into an interface-oriented pipeline that prioritizes privacy, low overhead, and user-understandable outcomes.

3.3 Towards Context-Aware UX in Automated Mobility: BLE Based Passenger Detection via Smartphones

As vehicles become more automated, the dynamics between people and these systems are shifting, transforming passengers from drivers into occupants. Automated vehicles are increasingly expected to respond to passenger presence, preferences, and context, requiring a deeper understanding of who is in the vehicle and how that context changes over time [202]. This section explores how BLE sensing on smartphones can help automated vehicles infer the presence of other passengers. The core idea is simple: by detecting other nearby BLE-enabled devices carried by co-travelers, a smartphone can offer clues about whether a user is alone, on a shared ride, or in public transport. This low-level sensing helps build higher-level context, informing how a vehicle adapts its interface or behavior [89]. We propose a system that scans for nearby BLE signals and uses their consistency and volume to infer transport mode. A few `Stable Devices` (those appearing consistently for at least five scan cycles) likely indicate a private car, autonomous car, or robotaxi [3], while a larger number of appearing and disappearing devices suggests a bus or metro. By treating smartphones as proxies for people, the system estimates passenger density to support adaptive interaction features, accommodating users with multiple devices.

We also assess vehicle type by monitoring the frequency of BLE device appearance/disappearance during scans, and by counting `Stable Devices`. The average number of `Stable Devices` varies by vehicle type and traffic conditions, helping us evaluate the user's vehicle. Unlike traditional methods relying on GPS, accelerometers, or cameras, BLE scanning is lightweight. It does not track user location or record personal data [184], making it ideal for scenarios prioritizing power consumption, privacy, and unobtrusiveness, especially in passenger-focused applications [1]. The motivation is to enhance automated vehicle responsiveness and adaptability [83]. For example, a vehicle detecting a lone occupant might offer more verbal feedback, while recognizing others nearby could emphasize visual cues or limit distractions. BLE-based sensing introduces a level of awareness currently lacking in many systems. The remainder of this section is organized as follows: Section 3.3.1 presents our BLE-based detection approach. Section 3.3.2 details our classification model and evaluation. Section 3.3.3 describes the mobile app and data collection. Section 3.3.4 explores potential use cases in human-automated vehicles. Section 3.3.5 discusses implications for designing adaptive, passenger-aware vehicle systems.

3.3.1 Methodology and Data Collection

Smartphone ownership worldwide is on the rise, especially in Western countries, making it increasingly uncommon for individuals to be without a smartphone or related smart device [56]. We hypothesize that smartphones can effectively serve

as proxies for their users, allowing for reliable inference of individual presence. To this end, we propose a BLE-based method to detect and track nearby smartphones, facilitating the estimation of individuals traveling in proximity to a user. Our approach utilizes periodic BLE scanning to identify devices emitting beacon signals, presenting a scalable and efficient solution for mobility tracking. However, several challenges exist in ensuring accurate estimations. Not all detected BLE signals originate from smartphones, many individuals carry multiple BLE-enabled devices, such as smartwatches and fitness trackers, leading to potential overestimation. Additionally, stationary BLE-equipped devices like IoT equipment can introduce noise, complicating the differentiation between personal devices and environmental signals. To mitigate these issues, our method identifies `Stable Neighbors` that consistently appear across multiple scans, filtering out `Transient Devices` (those visible for only a few seconds) and enhancing estimation reliability. By applying machine learning techniques for signal classification and pattern recognition, our work aims to improve the understanding of user transportation in complex environments.

We can infer whether a user is traveling alone in a car or with others in a shared vehicle based on how many nearby smartphones are detectable through BLE. The basic idea is simple: BLE-enabled smartphones can detect other BLE devices that are advertising nearby. If someone is consistently surrounded by a few other phones, that might suggest a private car, taxi or a carpool. A large, fluctuating number of nearby devices might indicate a bus or metro. Other cues can be used to disambiguate certain situations, such as when a user is stuck in traffic on the motorway. In this scenario, the user travels many kilometers, surrounded by the same vehicles for an extended period. Likely, the surrounding vehicles have not changed during this time. Additionally, since the user was previously in a car, the high number of Bluetooth devices nearby makes it unlikely that the user is in a different mode of transportation. We implemented this logic in an iOS application that periodically scans for BLE signals and logs nearby devices' identifiers and signal strengths. The app requires minimal explicit interaction from the user. To reduce noise, we ignore `Transient Devices` and focus on those that appear consistently over time. This helps filter out people simply walking by or standing nearby for a moment. We also avoid pairing and authentication steps to keep the scanning process lightweight.

In an urban setting, we gathered data during actual commutes using buses, trams, metros, and private cars. A total of 30 users tested the app for sessions lasting between 15 minutes to over an hour, varying by route. We varied the time of day, location, and occupancy levels to capture different crowding conditions. Phones were positioned in typical user locations, like in hand, bag, or pocket, accurately representing BLE signal behavior in different usage contexts. Each scan captured the detected device's timestamp, Received Signal Strength Indicator (RSSI), and a hashed Universally Unique Identifier (UUID). We deliberately avoided storing

any user identifying information to preserve privacy. For each trip, we requested that users record the ground truth label (car, bus, or metro). We noted special conditions, such as high passenger density or unusually weak signals due to barriers or phone placement. We processed the data into sliding windows of fixed length (e.g., 30 seconds), computing features such as the number of stable devices, average RSSI, and the variance in detected signals. One key metric was the "presence count", which indicates how many devices appeared consistently across multiple scans in the same window. This is more stable than relying on single scan results, which can be noisy due to movement or radio interference. While the system does not explicitly track movement, location, or sound, it captures the co-presence of other devices in the same environment. This focus on passive proximity sensing fills a gap in the TMD space, offering a less invasive method that is easier to deploy than alternatives requiring GPS, audio, or visual input. The following section explains how we trained and validated a model using this dataset to predict the transportation mode.

3.3.2 Model and Evaluation

The central question we wanted to answer was simple: Can BLE data alone help us figure out what vehicle a person travels in? We developed a basic classification pipeline to explore this using features extracted from BLE scanning windows. We did not aim for cutting edge model performance; instead, we focused on feasibility, interpretability, and usefulness for real world deployment.

3.3.2.1 Feature Design and Labeling

The BLE scanning process captured data in discrete time windows. We extracted a few straightforward features, among others, for each window: the number of unique devices detected, the average duration those devices stayed visible, and the variation in their RSSI. These features were chosen because they reflect different travel conditions. For example, a private car ride typically involves one or two stable devices, often with stable signal strength. A public bus or tram usually results in a fluctuating set of devices entering and exiting the range. Walking, on the other hand, often produces scattered, inconsistent detections. There are exceptions and edge cases to these patterns. We labeled each window manually based on our travel diaries, using four classes: Car, Bus-Tram, Subway, and Walk. The dataset included over 100 labeled windows, which we split for training and evaluation.

3.3.2.2 Model Performance

We trained a Random Forest classifier with hyperparameters: max_depth of 15, min_samples_leaf of 1, min_samples_split of 10, and 80 estimators. Cross-

validation for the four classes were [0.743, 0.779, 0.673, 0.729, 0.711], yielding a mean score of 0.725 and indicating moderate performance with some variability. The results are summarized in Table 3.2. The model performed well on "Car" with high precision, recall, and F1-score (0.85). These trips tend to be cleanly separable because of the consistent presence of one or two nearby devices and low signal variability. Support indicates the number of true instances of each class in the test set. "Bus-Tram" had good precision (0.71) but much lower recall (0.50), leading to a moderate F1-score (0.59) and showed decent performance as well. We chose to aggregate bus and tram readings, as their data were pretty similar, and at the same time, they both had a low number of samples. "Subway" showed the weakest performance with a low F1-score of 0.27, indicating difficulty in identifying this class. Detecting subway rides reliably was more challenging, likely due to the loss of GSM signal in tunnels, which limited the number of recorded trips. Subway rides are quite different from other modes of transport, as they exhibit noticeable "spikes" when approaching different stops. While this behavior may appear similar to that of buses or trams, it is much more pronounced in the case of subways. Even though the number of examples is low, it is important to maintain subway rides as a separate category. Aggregating them with other categories would create unnecessary noise. Walking was also somewhat ambiguous, especially in low traffic areas with few nearby devices.

Class	**Precision**	**Recall**	**F1 Score**	**Support**
Bus-Tram	0.64	0.70	0.67	10
Car	0.94	0.81	0.87	77
Subway	0.20	0.50	0.29	8
Walk	0.67	0.57	0.62	14
Overall Accuracy			**0.74**	109

Table 3.1. Classification Report

Table 3.2. A table showing classification performance for four transport modes: Bus-Tram, Car, Subway, and Walk. Metrics include Precision, Recall, F1 Score, and Support. Car has the highest F1 score of 0.87 and the largest support (77). Subway shows poor performance with an F1 score of 0.29. The overall classification accuracy is 0.74, based on 109 samples.

3.3.2.3 Average Performance Metrics

The macro average (which treats each class equally) and the weighted average (which accounts for how standard each class is) are reported in Table 3.4. The weighted F1 score of 0.77 reflects solid performance, given the class imbalance in our dataset. Most of the windows came from car trips.

Average Type	Precision	Recall	F1 Score
Macro Average	0.61	0.64	0.61
Weighted Average	0.82	0.74	0.77

Table 3.3. Macro and Weighted Averages

Table 3.4. A Table which shows the macro and weighted average precision, recall, and F1 scores for the classification task.

3.3.2.4 Interface Integration and Feedback

The significance of this model lies in its accuracy and the positive impact it has on the user experience. As said previously, we utilized a data collection app to gather user trip data manually. This collected data was then analyzed, and a Random Forest classifier was trained. We integrated the output of this model into a new test application with a very simple interface, allowing users to receive real-time estimates of their current vehicle. This test app was disseminated to users, who were encouraged to make new trips again. If users disagreed with a prediction, they had the option to override it, and these corrections were logged to enhance future versions of the model. The primary aim of this test application was to verify whether the user collected data aligned with the selected vehicle type and to assess the consistency of our predictions. Overall, the results indicated that our predictions were largely accurate. The logs showed that most users agreed with the app's predictions. When errors occurred, they often involved edge cases, like riding an empty tram (mistaken for walking) or driving alongside a bus (mistaken for bus). These instances revealed where BLE data alone might struggle, suggesting that lightweight context features like time of day or zone classification could help resolve ambiguities without sacrificing privacy or efficiency.

3.3.2.5 Key Takeaways

This evaluation shows that BLE can contribute to solving the challenges of transportation mode detection. In fact, BLE offers a meaningful and low power signal that can complement traditional methods [2]. Its ability to reflect co-presence and density makes it exceptionally well suited for supporting vehicle passenger interaction in automated systems. By using BLE in this way, we take a step toward context aware, energy efficient, and privacy conscious transport mode detection. These qualities are particularly important for scalable deployment in consumer centric applications.

3.3.3 User Interface Design

Although BLE scanning does not require much user attention, we designed a companion mobile interface that gives users insight into what the system is doing. The aim was to make the sensing process understandable and controllable without overwhelming users with technical details. The app features a streamlined timeline view that tracks nearby devices detected during scanning. To enhance user understanding, we use familiar visual cues like colored bars and brief messages, avoiding overwhelming users with raw signal strength or technical details. When a trip begins, notifications are minimal and unobtrusive. The system intelligently monitors the user's status and, upon detecting a transition from Walking to Automotive, sends a timely notification to confirm if the user is in a means of transportation. This approach ensures the user receives relevant information at the right moment while keeping the interface clean and user friendly. Users can easily click the notification to select their mode of transportation and collect travel data. Figure 3.6 illustrates the app's main screen during live use. Users can view the current transportation status and switch between different modes of transport. This feature is especially helpful in cases where the system mistakenly identifies a situation. For example, if a user is walking but the system inaccurately detects automotive travel, users can correct it without needing to stop or restart the app. The goal is to keep users informed while minimizing the need for constant interaction.

Once a trip is completed, the app generates a visual summary, as shown in Figure 3.4. This screen displays the path taken and overlays a color coded timeline that reflects device density at each stage. Red segments indicate high density. Blue or green segments point to lower densities typical of car vehicles. This summary helps users understand why the system made a specific classification. We included a history view to encourage reflection and support future applications like trip logging or feedback (Figure 3.2). Each entry shows the result's transport mode, timestamp, and a transportation mode icon. This screen helps users revisit previous sessions and see how the system performs. We also gave careful attention to privacy and control. Users can pause scanning at any time using a prominent toggle. The app includes a short, readable explanation of what data is collected and what is not. For example, it emphasizes that no identifying information is stored, and BLE data is processed only to count devices, not to track individuals. This transparency is crucial in shared mobility contexts, where co-location can feel sensitive. While the current interface is minimal, it is a starting point for building richer human vehicle interaction features. In future iterations, the app could adapt its interface based on user patterns or support custom alerts when unusual patterns are detected. For example, a user might be notified if the system detects a sudden shift from a quiet private ride to a crowded public setting. Ultimately, the interface plays a dual role. It helps users understand and trust the passive sensing that powers the system,

and it creates an entry point for more dynamic and personalized experiences inside automated vehicles.

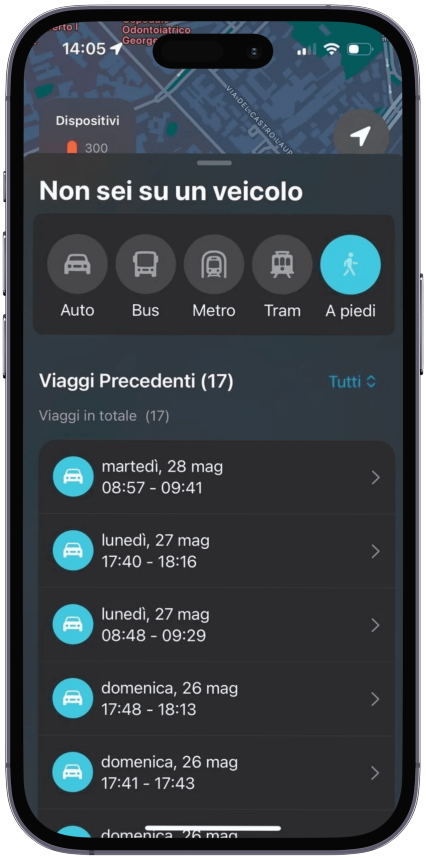


Figure 3.1. Trip history interface showing mode classification, timestamps, and quick visual summaries. Translation: *"Non sei su un veicolo"* = "You are not in a vehicle", *"Auto"* = "Car", *"Bus"* = "Bus", *"Metro"* = "Subway", *"Tram"* = "Tram", *"A piedi"* = "On foot", *"Viaggi Precedenti"* = "Previous trips", *"Tutti"* = "All". Dates and times (e.g., *"martedì, 28 mag"*) indicate the day (Tuesday, 28 May).

Figure 3.2. The user interface for viewing trip history, with a filter option to select different transportation modes

3.3.4 Human Automated Vehicle Use Cases

While this system was developed primarily for transportation mode detection, the exact underlying mechanism can be applied to support a range of use cases in human automated vehicle interaction. One of the most direct and practical applications is passenger detection. Automated vehicles need to understand their environment, not just outside the vehicle but also inside. Detecting how many passengers are present and whether someone is alone or part of a group can inform how the vehicle adapts its behavior, interfaces, and services. Using BLE, our system can provide a rough but proper estimate of how many people are nearby based on the number

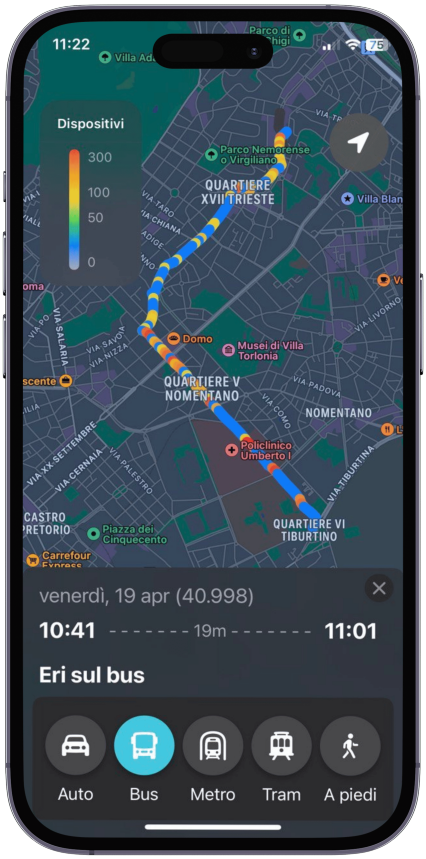


Figure 3.3. Completed trip view with density-based heatmap reflecting BLE activity along the route. Translation: *"Dispositivi"* = "Devices", *"venerdì, 19 apr"* = "Friday, 19 April", *"Eri sul bus"* = "You were on the bus", *"Auto"* = "Car", *"Bus"* = "Bus", *"Metro"* = "Subway", *"Tram"* = "Tram", *"A piedi"* = "On foot".

Figure 3.4. Completed trip view with a density-based heatmap displaying BLE activity along the route.

and stability of visible devices. If a user is in an automated shuttle or shared car, the presence of multiple BLE devices suggests a group context. The system can reasonably infer that the user is likely alone if no other devices are visible.

This type of information could be valuable in several ways:

- **Adjusting in-vehicle interfaces**: A vehicle might offer different interaction options depending on whether the user is alone (e.g., voice assistant defaults to private mode) or in a group (e.g., larger shared screen, reduced verbal feedback).

- **Enabling safety checks**: In ride pooling or last mile shuttle services, BLE-based detection could help confirm that all expected passengers have boarded or disembarked without relying solely on cameras or additional hardware.

- **Supporting adaptive behavior**: The system could modulate cabin lighting, volume, or seat configuration based on inferred occupancy level, improving comfort and reducing the need for manual adjustments.

As BLE detection is not exact, it does not provide identities or precise counts. It often gives a lightweight signal that is enough for adaptive systems to make better choices. Importantly, it does so in a privacy preserving and non-intrusive way, which is especially relevant for HAV environments where user trust and consent are critical.

3.3.5 Discussion

This work offers a novel, lightweight approach to understanding the user's travel environment through BLE sensing. While much of the existing literature has focused on detecting transport modes for broader mobility or urban analytics purposes [27] [167] [139], our focus is slightly different: how BLE-based proximity sensing can support user-aware features inside automated vehicles and shared autonomous transport contexts. A core challenge in automated vehicles is inferring who is present, how many people are on board, and under what conditions they travel [112]. This information can support adjustments to in-vehicle interfaces, safety mechanisms, and service personalization. Our system does not track users explicitly but instead captures patterns in the presence of nearby devices. These patterns, the number of nearby devices, stability over time, and variation can act as rough signals of whether a user is alone, in a carpool, or in a crowded metro or bus. The model we trained was effective at identifying car trips and reliably detecting shared transportation modes such as buses or trams. However, classifying metro trips proved to be more challenging due difficulties in our data collection in tunnels and other enclosed areas. Despite this, in many instances, other devices provided sufficient signal strength to support high level system adaptations. From an HAV perspective, this opens up several interaction possibilities. For instance, the vehicle might suppress or simplify voice interactions when it infers that others surround the user or automatically activate privacy modes when it senses a single occupant [55]. BLE-based presence could also assist in validating whether all expected passengers are present in shuttle contexts without requiring cameras or biometric data.

One significant contribution of the system is its focus on user-centered feedback. In the interface we developed, users are informed about the current classification and have the ability to make corrections. This feature helps to ensure that the system's behavior is both explainable and adjustable, qualities that are becoming increasingly important as vehicle interfaces evolve towards greater autonomy and adaptability. While there are limitations to BLE-based sensing, it remains a valuable tool for understanding passenger presence. This technology may not provide high precision for counting passengers or identifying individuals. Factors such as hardware

discrepancies, placement of the smartphone, and temporary interference can lead to signal variability. Nevertheless, these challenges are manageable in scenarios where a general awareness of passenger presence is sufficient to enhance responsiveness and improve overall efficiency. This work shows how BLE sensing could play a helpful role in the broader design of context aware automated mobility systems, especially those that aim to be more responsive to the presence and needs of passengers. BLE-based proximity detection offers a energy efficient [193], cost effective and privacy conscious way to enhance environmental awareness, which can facilitate interactions between humans and automated vehicles. BLE sensing stands out from other sensors as it focuses on detecting the presence of nearby devices without revealing specific locations, unlike GPS or Wi-Fi. It does not track your personal movements like motion sensors or capture audio like a microphone, allowing it to infer context without directly gathering personal information. Moreover, BLE devices avoid tracking by periodically broadcasting a random, temporary address instead of a fixed one, preventing observers from linking their presence over time to build a movement history [12]. By determining whether a user is alone or in a shared space and providing feedback tailored to the user, the system opens up possibilities for more adaptive and human-aware automated mobility solutions. Future research could build on this by incorporating additional lightweight cues and exploring how users react to interface changes influenced by BLE-based sensing. In addition, we intend to conduct further user testing to assess the capability of the proposed system and discern different means of transportation.

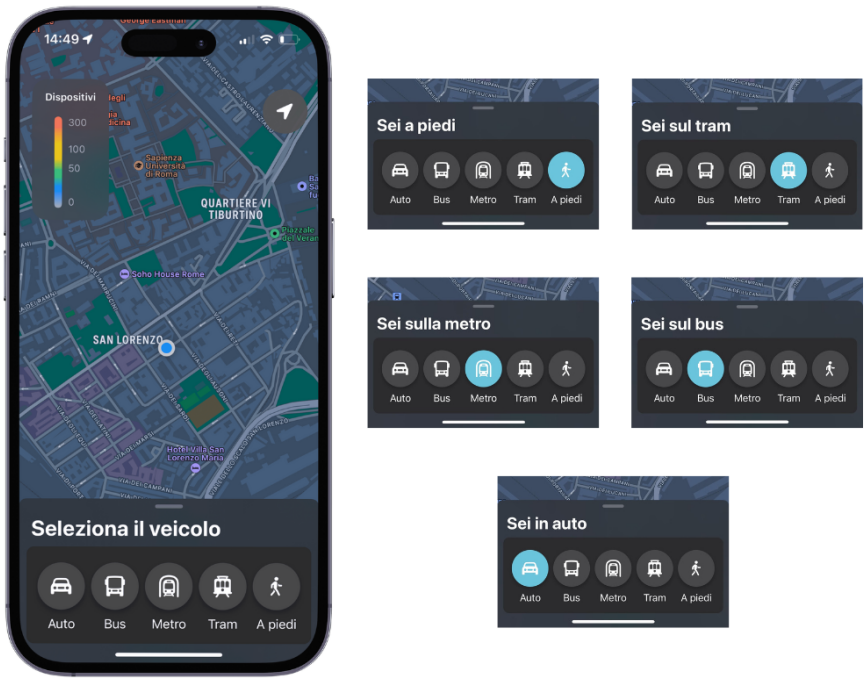


Figure 3.5. Main view of the app showing current detection status and manual override controls. Translation: *"Seleziona il veicolo"* = "Select the vehicle", *"Dispositivi"* = "Devices", *"Sei a piedi"* = "You are on foot", *"Sei sul tram"* = "You are on the tram", *"Sei sulla metro"* = "You are on the subway", *"Sei sul bus"* = "You are on the bus", *"Sei in auto"* = "You are in a car", *"Auto"* = "Car", *"Bus"* = "Bus", *"Metro"* = "Subway", *"Tram"* = "Tram", *"A piedi"* = "On foot".

Figure 3.6. Main view of the application showing current vehicle selected for transport mode detection.

3.4 Detecting Human Presence via Smartphone BLE Beaconing: Preliminary Investigations

Detecting human presence in everyday environments is a crucial capability for mobile and ubiquitous systems, allowing for context-aware experiences without intrusive sensing methods [199, 7]. Most presence detection technologies rely on cameras, GPS traces, WiFi association logs, or complex sensor fusion techniques, each of which has trade-offs, such as high power consumption, significant upfront implementation costs due to sensor deployments, and privacy concerns [161, 207, 140, 227].

BLE beaconing offers an alternative approach: smartphones periodically transmit advertisements over BLE using randomized addresses, and detecting these signals is possible without the need for direct connection [145, 10]. This creates a subtle signal of co-presence that can be detected without the need for pairing or identifying individual devices. Given that, smartphone ownership is on the rise, with nearly everyone now owning a smartphone [56]. When smartphones are considered as proxies for their owners, the aggregated characteristics of these signals, such as the number of devices that remain visible over a period of time, can provide insights into human presence and density at specific locations and times.

Our previous research [63] demonstrated that Bluetooth Low Energy (BLE) visibility patterns are stable enough to support classification tasks in mobility contexts. The persistence and turnover of nearby devices allowed for the differentiation of various modes of transportation. Similar studies that relied on smartphone beaconing technology, specifically Bluetooth and WiFi signals, were successful in detecting different transportation modes [43, 57]. This approach could be adapted and implemented across various contexts, unlocking new opportunities and innovative solutions [40, 39, 31]. Building on these insights, we explore whether this fundamental concept can be applied more broadly in other scenarios.

We recommend using the visibility of BLE beacon signals as a general indicator of nearby individuals. This application can be extended beyond transportation to address various other issues, including crowd awareness, indoor occupancy detection, accessibility, and adaptive environments [134]. Additionally, it can facilitate personalized services, optimize resource allocation, and enhance safety measures. Our goal is not to identify specific individuals or monitor their movements. Instead, we aim to explore whether simple stability measures, along with other contextual factors, can provide reliable signals of presence and density. This approach should respect privacy while being beneficial to both this system's users and urban planning and infrastructure developers. We aim to investigate the utilization of smartphone BLE beacon signals to detect people in various contexts.

3.4.1 Methodology

We view BLE advertisement signals from nearby devices (Smartphones) as a stream of observations that can be organized into time windows and summarized using interpretable features. A key concept in this context is `Stable Neighbors`, which refers to devices that remain visible across multiple consecutive observations within a given time window.

Stability plays two important roles. First, it helps filter out temporary detections caused by passersby signals and radio noise. Second, it provides a more reliable measure of co-presence than single-scan counts, which can vary due to movement or interference. To capture different aspects of the scene, we could use complementary summaries. These include the count of stable devices within a time window (indicating presence), the rate at which devices appear or disappear (turnover), and basic statistics of the Received Signal Strength Indicator (RSSI) [184]. These metrics help reflect environmental conditions without aiming for precise distance estimation or device tracking.

These features can be designed to be conservative, privacy-aware, and adaptable. Setting conservative thresholds, like needing multiple consecutive observations to deem a device stable, helps minimize false positives caused by transient sources. This includes stationary IoT devices nearby a user [4], which may not be relevant to the analysis when the user is in motion. Privacy awareness is achieved by focusing on aggregate properties, such as counts, persistence, and turnover, rather than on individual identities or trajectories. Improving adaptability involves adjusting window sizes and stability thresholds to match the specific dynamics of diverse environments, including public spaces during events, meeting rooms in offices, metro or bus occupancy, traffic management, and other presence-related applications. This approach enables the same set of features to effectively serve different contexts while maintaining an explainable and straightforward representation.

3.4.2 Preliminary Observations & Challenges

Our previous studies [63] indicate that stability-based summaries effectively track occupancy and flow across diverse environments. In this context, low stable neighbor counts with minimal turnover could be associated with quiet areas, such as small offices or study rooms. Moderate counts, accompanied by intermittent turnover, could correspond to corridors and lobbies where people pass through in waves. In contrast, higher and fluctuating counts, accompanied by significant turnover, might indicate crowded public spaces, platforms, or events [128]. Additionally, the variance in the RSSI seems to correlate more closely with environmental factors, such as reflection, attenuation, and device placement, rather than proximity alone. This observation reinforces the decision to rely primarily on persistence and turnover for

presence signals.

Tracking presence with multiple devices carried by a single person [108], such as phones, watches, and earbuds, presents several challenges. These devices can inflate presence counts. While maintaining stability can help reduce this issue, it cannot completely eliminate the ambiguity without additional cues.

Moreover, stationary or semi-mobile IoT devices that are irrelevant to the tracking process can introduce noise into the data. Implementing strict stability criteria and space-specific calibrations can minimize their impact, although some residual bias may still remain. Additionally, signal variability caused by obstacles, human bodies, and interference complicates the readings for each scan [129]. Therefore, aggregated measurements over time tend to be more reliable than instantaneous readings. Using BLE technology to achieve precise headcounts can be challenging. Instead, employing a tiered density estimate that classifies occupancy levels as low, medium, or high provides a more accurate representation and is often sufficient for adaptive systems.

Ethical considerations are crucial in this context. Presence signals should not be connected to individuals' identities, used for long-term tracking, or employed for covert inferences. Although Bluetooth Low Energy (BLE) protocols, along with the iOS and Android implementations of BLE technologies, have security measures in place to prevent these issues, there is still a possibility that malicious actors may attempt to exploit them. Therefore, deployments should prioritize transparency and provide users with options to opt out.

3.4.3 Applications & Future Directions

Treating smartphones as proxies for people and using BLE beacons as ambient presence signals open up various applications where collective awareness is more important than individual identity. In buildings, estimates of occupancy levels can assist with room booking, heating, ventilation, and air conditioning (HVAC) control, lighting management, and space planning, all without the need for cameras or badges. In public spaces and transport hubs, understanding crowd density and movement can enhance wayfinding, queue management, and overall operations, thereby improving comfort and safety for users [187, 82].

For accessibility, these technologies can help individuals who are blind or have low vision by providing navigation assistance related to crowd levels along their routes. They can suggest less crowded paths or times and offer wait estimates based on the presence of others. Moreover, adaptive interfaces on devices or in shared environments can adjust the type and frequency of notifications depending on whether people are present. This capability can minimize disruptions in busy settings while ensuring that notifications remain responsive when users are alone.

Evaluating these systems at scale will help assess detection latency, resilience to environmental changes, and practical utility in real-world deployments. However, including ethical safeguards, precise consent mechanisms, transparent data handling, strict data aggregation practices without identity resolution, and privacy impact assessments is necessary. By focusing on aggregate and stability-based signals rather than individual identification, BLE beacon signals from smartphones can support practical, low-friction awareness for pervasive systems while maintaining personal privacy.

Chapter 4

Indoor Localization Using Large Language Models

4.1 Introduction

While GNSS has become the standard solution for outdoor localization, its performance significantly degrades indoors, and there is no universal solution for indoor positioning. Alternative methods, such as NFC tags, Bluetooth beacons, WiFi signals, and sensor-based indoor mapping, have been proposed; however, due to their reliance on additional infrastructure or extensive mapping, they are not widely adopted. Users who get lost inside buildings often lack readily available solutions to find their location. This work investigates a new approach to localizing users by employing vision-capable Large Language Models (LLMs) that exploit evacuation maps. These maps are typically mandated by law and include floor layouts, as well as useful reference points such as doors, emergency exits, elevators, and the locations of various fire safety equipment. According to our approach, users describe their surroundings to the LLM (e.g., "*I see a fire extinguisher near an emergency exit*"). The LLM then analyzes the map to find areas consistent with the user's descriptions and pinpoints the user's position accordingly. Although the LLM does not provide precise localization, it offers users a clear visualization of their position within the building, which can also serve as a basis for navigation. This approach was investigated as follows. After a preliminary experimentation phase to evaluate different vision-capable LLMs, two user studies were conducted with the chosen LLM: one at the University of Oulu (**UO**) in Finland (N = 17) and another at Sapienza University of Rome (**Sapienza**) in Italy (N = 20). Since this work aims to assess the feasibility of an out-of-the-box indoor localization approach, conducting experiments in two universities across different countries enabled the evaluation of its robustness across diverse settings, including differences in building layouts, evacuation map formats, and participant backgrounds. Our studies show that 25

users out of 37 did not follow the LLM's guidance and described their surroundings in diverse and unpredictable ways, failing to provide descriptions of relevant elements, which resulted in poor localization and a reported low level of trust. In the 12 cases where users successfully provided relevant elements, localization was successful, and users demonstrated that they understood their position. Users report no frustration during the interaction, and no perceived difficulty in describing their surroundings.

4.2 Related Works

To the best of our knowledge, no prior work has used vision-capable LLMs in combination with evacuation maps for user localization. Existing indoor localization methods span a wide range of techniques [76]: some rely on additional infrastructure such as Bluetooth beacons, on-site cameras, or RFID systems, while others depend on sensors available in personal devices, such as smartphone IMUs, LiDAR, or RGB cameras, and many require extensive pre-mapping of the environment, for example by recording magnetic field signatures or performing 3D scans. In contrast, this work investigates whether vision-capable LLMs can provide a ready-to-use alternative for indoor localization, contributing to the ongoing investigation of their spatial understanding and reasoning [210]. A notable line of research explores their use in robot navigation, where robots are guided through indoor or outdoor environments using natural language instructions while relying on vision-enabled LLMs to interpret the surrounding scene and generate appropriate movement commands [186, 93, 178]. Regarding the use of evacuation maps, Peter et al. [164] proposed combining them with IMU sensors to aid user navigation; their work, and subsequent studies inspired by it, shifted more toward indoor mapping than navigation.

4.3 Methodology

4.3.1 Study Setup

Participants were recruited from the respective university communities. Each participant completed the session individually. At the beginning of each session, the purpose and procedure of the experiment were briefly explained, without providing details on how the LLM performed localization. Participants were then asked to complete a short background questionnaire collecting demographic information and details about their experience with localization and LLMs. Once finished, they were accompanied to the selected location where the study was conducted and provided with a smartphone with access to the LLM, using the standard ChatGPT interface. The interface displayed the following message:
"*I'm ready to help you find your location on this floor using your description! Please*

tell me about your surroundings, focusing on emergency equipment—what do you see? For example, you can mention nearby doors, staircases, elevators, fire extinguishers, or anything else noticeable"
Participants were asked to interact with the LLM in English, and could choose whether to communicate via text or speech. All interactions were recorded, and any verbal comments made to the researcher during the session were also noted. At the end of each session, the LLM produced an image of the evacuation map displaying a red "X" marker indicating its estimated location of the participant. Participants were then given a final questionnaire to assess their impressions of the outcome.

4.3.2 Study Locations and Evacuation Maps

Both studies were conducted on a floor of the building where students get lost easily, choosing a specific location based on the number of identifiable clues present on the evacuation map. To properly test the interaction between users and the LLM, the location was selected so that it was not so easily identifiable as to be trivial (e.g., in front of the only elevator on the floor), but not so difficult as to make localization impossible (e.g., in the corner of an empty room). The following section on the study outcomes discusses the choice in further detail. Fig. 4.1 and Fig. 4.2 show the two evacuation maps, which display the floor layout, including rooms and hallways, as well as emergency labels such as exit routes, elevators, first aid kits, fire extinguishers, fire hoses, and fire alarm buttons. However, they do not provide detailed room information; for example, bathrooms are not indicated, nor are office numbers or room names. In the UO map, additional yellow labels indicate the emergency exits, each identified by "E" followed by the exit number, as well as the various wings of the building. In the figures, we highlighted the positions where each study took place.

4.3.3 LLM Selection and Prompt

This work required an LLM capable of interpreting images and inferring the user's position from their description. At the time of the experiments (May–June 2025), several vision-capable LLMs were available. Qwen2.5-VL, LLaVA 1.6, Llama 3.2-Vision, GPT-o4-mini, and GPT-o4-mini-high were considered for the study, as they offer vision capabilities while maintaining relatively fast response times. The most suitable model was identified during a preliminary experimentation phase by evaluating how accurately each candidate matched textual descriptions to specific locations on an evacuation map and by assessing its ability to recognize standard emergency-equipment symbols correctly. GPT-o4-mini and GPT-o4-mini-high were the best candidates for the studies, as they demonstrated significantly higher precision compared to the other models. The OpenAI platform's support for Python execution

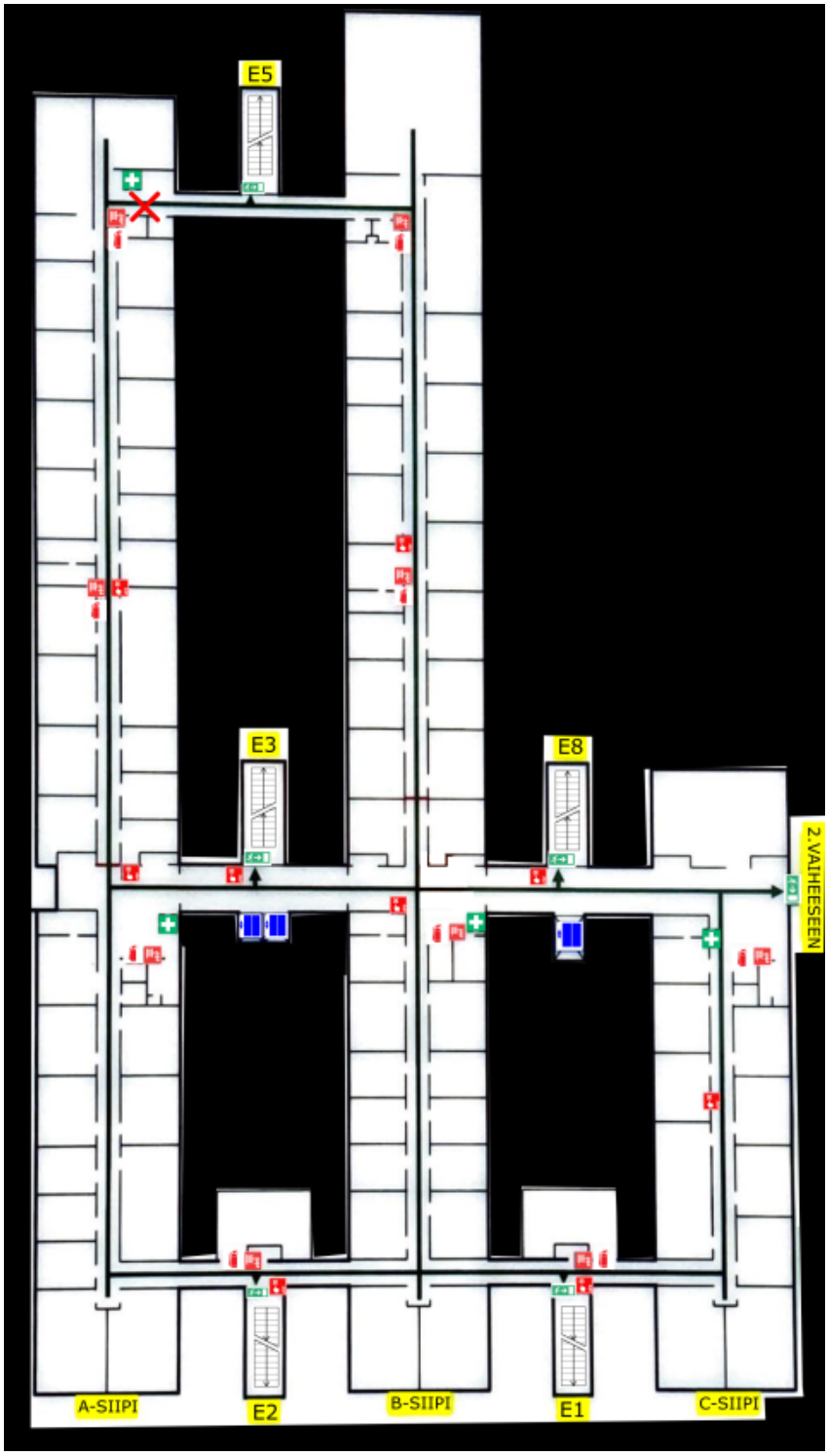


Figure 4.1. Evacuation map of the chosen floor in UO. The red X marks the study location.

was also a decisive factor, as it allows the model to directly modify the floor plan and display the user's estimated position without requiring a separate visualization

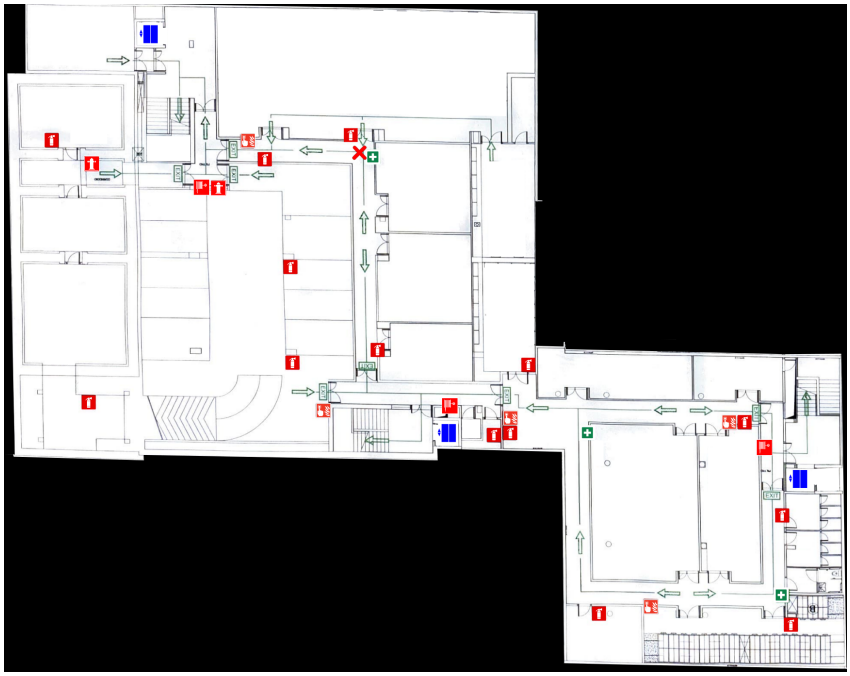


Figure 4.2. Evacuation map of the chosen floor in Sapienza. The red X marks the study location.

framework. Eventually, GPT-o4-mini-high was selected for the study, as it possesses superior reasoning capabilities compared to GPT-o4-mini, producing better results when presented with longer descriptions. A concise prompt was designed to provide minimal context on the task to the LLM and guide it in placing the red marker on the map. Our preliminary study showed that overly long and detailed prompt text would lead the model to ignore certain parts of it. The prompt was the same for both studies: *"This GPT helps users who are lost on a specific floor, using an uploaded evacuation map to determine their location. Users describe their surroundings in natural language. GPT will ask questions relevant to safety equipment, and will interpret these descriptions to deduce their likely position on the floor layout. It will not reference the image directly in conversation with the user, but will use it to compute and visualize the inferred location. Once the user's location is determined, it will compute the corresponding x,y coordinates on the map and overlay a red cross at that spot using matplotlib. The resulting image will be displayed to the user. The GPT must write in english."* .

An example description of the UO location that is correctly recognised by the LLM using this prompt is: "*I'm in a corner, there's a first aid kit nearby, a fire extinguisher, and a door leading to stairs in front of me*". For Sapienza, a corresponding correct example is: "*There's a door in front of me, with a fire extinguisher and a first aid kit*". These examples show that it is not necessary to specify the exact position of each feature in the description.

4.3.4 Data Collected throughout the study

At the beginning of the study, each participant completed a pre-test questionnaire that included demographic questions, such as age range (18–24, 25–31, 32–45, 45+), gender, highest level of education, and preferred language of communication with LLMs, and questions regarding their use of localization services and LLMs, rated on a 5-point Likert scale (1 = low/negative, 5 = high/positive), including: "*How often do you use location-based services (e.g., Google Maps or similar) in indoor settings?*", "*How often do you use localization services (e.g., Tuudo, MazeMap, or similar) to navigate the university?*", and "*How often do you use conversational agents (ChatGPT, Claude, Gemini, etc.)?*". After the test, participants completed a final questionnaire whose first question was: "*Based on the map and the conversation, do you think the LLM was able to identify your position?*" with possible answers **Yes**, **No**, and **Not Sure**. Participants who responded **No** or **Not Sure** were verbally asked to explain the reason for their doubts, and to indicate on the map the location where they believed they were actually situated. The remaining questions, rated on a 5-point Likert scale (1 = low/negative, 5 = high/positive) such that higher scores always indicate better outcomes, included the following items: "*How much do you trust its answer?*", "*Regardless of its accuracy, did you find that your location was presented clearly?*", "*How easy was it for you to describe your surroundings to the agent?*", and "*Did you find the conversation with the agent frustrating?*".

4.4 Study Outcomes

4.4.1 OU Study

The selected location is on a floor primarily occupied by offices, which students typically visit to meet with professors and discuss exam-related matters. The exact spot is in the upper-left corner of this floor, near a first aid kit, a fire extinguisher, and a fire hose. Three similar locations can be found on the map, as the same arrangement of equipment appears in the middle part at each intersection.
A total of N = 17 individuals took part in this study. Most participants identified as male (82%) and were between 25 and 31 years old (mean age = 28 years, estimated from ranges). All participants chose English as their preferred language when interacting with LLMs, and most held a master's degree (52%), followed by those with a bachelor's degree (35%) and high school diploma (11%). Participants reported being somewhat familiar with the chosen floor (mean = 3.29 out of 5; SD = 0.91). They reported using indoor localization services when moving inside the university (mean = 3.23, SD = 1.03). They also report frequent use of LLMs in everyday tasks (mean = 4.11, SD = 0.92), as well as localization and navigation services in contexts outside the university (mean = 4.41, SD = 0.71), indicating a general familiarity

with technologies relevant to those used in this study. GPT correctly localized 6 participants out of 17 based on their conversations, and it presented them with an evacuation map that displayed an accurate marker of their position. When asked whether they thought the LLM was able to identify their position, all six participants confirmed it. Among the remaining 11 participants with incorrect localization, all were confident that the location was wrong, except for one who reported being unsure. When shown the evacuation map, however, only 4 out of 11 were able to pinpoint their actual location on it correctly. Participants reported low trust in the LLM's prediction (mean = 2.70, SD = 0.91). They found it relatively easy to describe their surroundings through conversation (mean = 3.00, SD = 1.00), and did not find the process frustrating (mean = 3.80, std = 0.8). They also tended to agree that the map, regardless of whether the localization was correct, provided a clear way to represent their position on the floor (mean = 3.05, SD = 1.08). This is consistent with their understanding of the LLM's prediction, as participants were generally accurate in determining whether the proposed location was correct. However, the map alone did not appear sufficient for accurate self-localization, as 7 out of the 11 incorrectly localized participants were unable to pinpoint their actual position when shown the evacuation map. The duration of the conversation, measured from the start until just before generating the resulting image, was on average 44 seconds (SD = 77s), with the quickest at 5s (user initial description: "*Near TS318, TS320 and exit E5*") and the longest at 5 minutes (user initial description: "*I am surrounded by green doors and a green roof*"). Image generation took an average of 85s (SD = 51s), bringing the total conversation time to an average of 121s (SD = 104s).

4.4.2 Sapienza Study

The selected location is on a basement floor, which hosts several laboratories that students visit to carry out specific tasks or take exams. The precise spot is in the upper-right section of the floor, in front of a door, with a fire extinguisher on the leftmost wall and a first aid kit on the rightmost wall. A similar area is present in the bottom-right corner of the map. A total of N = 20 students took part in this study. Most participants identified as male (70%) and were between 25 and 31 years old (mean age = 24 years, estimated from ranges). Most participants chose Italian as their preferred language when interacting with LLMs (70%), and the majority held a bachelor's degree (50%), followed by those with a high school diploma (35%), doctorate (10%), and master's degree (5%). Participants reported not being familiar with the chosen floor (mean = 2.95, SD = 1.14). They reported not using indoor localization services when moving inside the university (mean = 1.80, SD = 0.80). It should be noted that, unlike at the University of Oulu, there is no indoor navigation application available for use inside Sapienza's buildings. They also report frequent use of LLMs in everyday tasks (mean = 3.70, SD = 1.03), as

well as localization and navigation services in contexts outside the university (mean = 3.85, SD = 1.20), indicating some familiarity with the technologies involved in this study. GPT correctly localized 6 of the 20 participants based on their conversations, and it presented them with an evacuation map that displayed an accurate marker of their position. When asked whether they thought the LLM managed to identify their position, all six confirmed it. Among the remaining 14 with an incorrect localization, 8 were confident that the localization was wrong, and 6 of them were able to pinpoint their actual position on the evacuation map. Participants tended not to trust the LLM's prediction (mean = 2.85, SD = 0.87). They felt that describing their surroundings through conversation was acceptable (mean = 3.20, SD = 1.43), and did not experience frustration during the process (mean = 4.45, SD = 0.75). They also tended to agree that the evacuation map provided a clear way to represent their position on the floor (mean = 3.30, SD = 1.20), despite a high number of participants reporting being unsure about the presented localization. Nevertheless, the map alone was not sufficient for self-localization, as 8 out of 14 incorrectly localized participants were unable to pinpoint their position when shown the evacuation map. The conversation duration, measured from start to before before generating the resulting image, was on average 20 seconds (SD = 12s), with the quickest at 5s (User initial description: "*I am at underground floor, in a corner with 3 doors in front of me and one fire extinguisher at the end of the corner, I see a yellow door in front of me and another corridor with a yellow door on my left. I have a green door behind me and I am near a health kit*"), and the slowest at 35s (User initial description: "*I'm underground and the walls are withe with green doors* "). Image generation took on average 37s (SD = 23s), resulting in a total conversation time of 56s on average (SD = 28s).

4.5 Discussion

The two studies highlight several important insights about the use of vision-capable LLMs for coarse indoor localization. When participants provided descriptions containing spatially discriminative and map-relevant cues, the LLM was often able to identify the location correctly, and the resulting evacuation-map visualization was generally clear enough to support orientation. At the same time, the main limitation did not appear to lie simply in whether users were "good" or "bad" at describing their surroundings. Rather, the results suggest a mismatch between what the system needed in order to localize reliably and what users could reasonably be expected to notice, select, and verbalize in the moment. In this sense, the issue is better understood as an interaction-design and cognitive-support problem than as a user failure.

Although participants generally did not report the task as frustrating, most

descriptions turned out to be too generic or insufficiently discriminative for accurate localization. This is likely due to a combination of factors: the initial instruction did not adequately communicate what kinds of environmental cues were most useful, the interaction relied on users identifying map-relevant details under uncertainty, and the conversational format placed a non-trivial cognitive burden on participants, who had to inspect the environment, infer which elements might matter, and translate them into a verbal description. The longest conversations in both studies were associated with highly generic initial descriptions, and these did not lead to successful localization. This reinforces the interpretation that the bottleneck was not user unwillingness, but the lack of sufficient scaffolding to help users provide actionable input.

No substantial differences emerged between the two experiments, despite variations in location, country, and culture. This suggests that the observed limitation is not strongly tied to a specific building or participant population, but may instead reflect a broader usability issue in the current interaction design. Users also reported generally low trust in the tool. Participants who were correctly localized tended to express higher trust on average, but the current sample is too limited to support strong conclusions about this relationship.

A key strength of the proposed approach is that it requires no additional infrastructure: an interface to an LLM and access to the building floor plans are sufficient. This makes the method appealing as a lightweight alternative to indoor localization solutions that depend on dedicated hardware or prior environmental instrumentation. However, the studies also make clear that a conversational interface alone is not enough. If reliable localization depends on users spontaneously identifying a small set of meaningful spatial cues, then the system should do more to support that process.

For this reason, a promising next step is to redesign the interaction so that the system reduces cognitive load and guides perception more explicitly. Rather than relying only on open-ended descriptions, future versions could provide a structured interface with selectable environmental features, progressive prompts, or checklists of potentially relevant cues, such as doors, staircases, safety equipment, corridor shape, or nearby intersections. Such scaffolding could help users focus on the information most useful for localization, reduce ambiguity in the input, and make the overall interaction more robust and trustworthy.

Chapter 5

Supporting User Input with LLMs

5.1 Introduction

A large share of everyday interaction with digital systems still revolves around structured data entry: filling forms, configuring services, populating databases, and translating domain documents into fields that downstream processes can validate and compute on. Despite decades of UI refinement, these activities remain tedious and error-prone because they force users to adapt their intent to rigid schemas and to repeatedly transcribe information from heterogeneous sources. This friction is especially visible in administrative and industrial workflows, where required data is often distributed across certificates, datasheets, reports, or emails, and where even small inconsistencies in formatting or terminology can lead to delays, mistakes, and frustration.

In this context, Large Language Models (LLMs) open a new design space for interaction engineering: rather than asking users to manually map their knowledge to a form, systems can accept natural language descriptions and existing documents as input, and use LLMs as semantic intermediaries to produce structured outputs aligned with predefined schemas. Crucially, this is not a purely automation-centric vision. Because LLMs are probabilistic and can be wrong in subtle ways, effective systems must adopt mixed-initiative interaction, where the model proposes candidate values and justifications, while users retain oversight through review, correction, and final approval. Mechanisms such as confidence-aware suggestions, editable outputs, and dialogue-based refinement help balance efficiency with trust, accountability, and data quality.

The two papers collected in this subsection contribute to this agenda from complementary angles. First, we present a domain-agnostic engineering blueprint

for transforming unstructured descriptions into structured input through LLM agents, emphasizing modularity, controllability, and integration within heterogeneous infrastructures via an MCP-compatible design [225]. Second, we instantiate and evaluate these ideas in a concrete web-based form-filling scenario, where users upload technical documents and collaborate with an LLM to extract and populate complex fields, iteratively validating and correcting the output in a human-in-the-loop workflow [5]. Together, these works position LLMs as interaction-level components that reduce the burden of structured data entry while preserving user control, supporting a practical path from intent and documents to reliable, schema-compliant inputs.

5.2 Related Works

Structured input generation is a fundamental activity across domains such as administration, healthcare, education, and industrial workflows, where users are repeatedly asked to translate intent and existing information into rigid schemas through form filling, record keeping, and database-style data entry. This translation is often slow and cognitively demanding, and small inconsistencies in formatting or terminology can propagate downstream and disrupt subsequent processing. Recent work argues that Large Language Models (LLMs) can reduce this burden by interpreting natural language, extracting context from documents, proposing field values, and flagging inconsistencies, thus improving efficiency while lowering cognitive load [53].

5.2.1 From rule-based assistance to LLM-driven structured input

Early data-entry assistance relied primarily on rule-based techniques (e.g., templates, regular expressions, keyword matching) that were effective only in narrow contexts and brittle under linguistic variation. Machine-learning approaches later introduced adaptive prediction and error correction, learning from user behavior and historical data [208]. While these methods improved personalization, they still struggled when inputs were heterogeneous, incomplete, or embedded in free text and semi-structured documents.

LLMs broaden this design space by enabling semantic interpretation and flexible schema alignment. Recent systems explore domain-agnostic form-filling support and structured value suggestion across varied interfaces [16]. Other lines of work propose hybrid pipelines that combine statistical models with rule-based heuristics to improve extraction and generation reliability [212]. Compared to traditional ML pipelines, LLMs are particularly suited to tasks that require mapping between human language and predefined data structures, and they naturally support conversational

interaction for clarification and refinement.

5.2.2 Human–AI interaction principles and mixed-initiative workflows

Because LLM outputs are probabilistic and may include plausible but incorrect values, the design of LLM-assisted input systems must follow established human–AI interaction principles that preserve user control and foster trust. Amershi et al. [6] emphasize making capabilities and limitations clear, timing assistance appropriately, and ensuring users can provide feedback and override automated behavior. A classic strategy is mixed-initiative interaction [90], where the system acts proactively when confident but defers to the user when uncertainty is high, and always supports review, correction, and rejection.

These concerns align with Human-Centered AI perspectives that frame AI as an augmentation technology rather than a replacement for human judgment. Shneiderman's HCAI framework highlights the importance of transparency, accountability, and user agency in AI-enabled systems [188]. In structured input generation, this translates into interface mechanisms such as editable suggestions, confidence-aware highlighting, traceable evidence from source documents, and explicit validation steps before submission.

5.2.3 Evidence from deployment-oriented studies and application domains

Beyond conceptual frameworks, several empirical studies motivate the use of LLM assistance in professional and high-stakes settings. Noy and Zhang [144] investigated the impact of ChatGPT on workplace writing tasks, reporting substantial reductions in completion time and improvements in output quality, especially among lower-performing participants. While not focused on form filling per se, these findings support the broader claim that LLMs can substitute for repetitive cognitive effort, shifting user attention toward higher-level refinement.

In accessibility-oriented work, Cuadra et al. [58] developed a multimodal assistant to help older adults and people with disabilities complete healthcare forms, highlighting the value of adaptive interaction in information-dense and consequential contexts. Hakimov et al. [85] advocated for modular LLM-assisted form completion architectures that decompose the end-to-end task into micro-activities (e.g., question grouping, information extraction, validation), improving both accuracy and user engagement through structured workflows. In other professional domains, Martin et al. [121] benchmarked LLMs on contract review tasks, illustrating the potential for large efficiency gains in high-volume, high-precision pipelines. Finally, Kessel et al. [100] explored LLM chatbot assistance in citizen-science digitization, finding

improved user experience and data quality while emphasizing the need for structured processes and error-mitigation strategies to manage model unpredictability.

5.2.4 Gaps and positioning of our work

Despite growing evidence of LLM utility for reducing data-entry effort, a gap remains between prototypes and deployable solutions. Many systems remain domain-specific, rely on brittle prompt engineering, or provide limited transparency about why a value was proposed and how it relates to the target schema. Moreover, relatively few approaches demonstrate how to integrate LLM-based structured input generation into heterogeneous infrastructures while preserving user oversight and enabling mixed-initiative control at scale.

Our work addresses these limitations in two steps. First, we introduced an engineering blueprint for LLM agents that transform natural language descriptions and heterogeneous documents (e.g., PDFs and spreadsheets) into schema-compliant structured inputs, explicitly coupling automation with human-in-the-loop verification [225]. Building on that foundation, we then investigated LLM-assisted form filling in a concrete, domain-specific scenario, emphasizing dialogue-based review and correction, and evaluating usability through user testing in a semantically complex setting [5]. Together, these contributions advance structured input generation from a model capability to an interaction-centric, integration-aware pipeline that prioritizes transparency, adaptability, and user control.

5.3 Engineering Large Language Model Agents for Transforming Unstructured Descriptions into Structured Input

The rapid advancement of Large Language Models (LLMs) is reshaping how users interact with digital systems. Beyond their widespread use in content generation, summarization, and retrieval, LLMs offer untapped potential in user interface engineering, particularly in how users input and structure information for downstream processes.

Structured data entry, such as form filling, database population, or system configuration, remains a common but tedious activity across domains ranging from administration to industrial workflows. These tasks often force users to adapt their mental models to rigid input formats, which can be time-consuming and error-prone. Even small inconsistencies in formatting or terminology may disrupt the process, reducing efficiency and increasing frustration. Allowing users to instead express their intent in natural language, and relying on LLMs to translate that intent into structured outputs, can substantially reduce cognitive load and improve usability. Moreover, many workflows involve documents that already contain the required information, such as certificates, datasheets, or resumes. Leveraging these documents through intelligent extraction eliminates redundant re-entry of data and ensures that existing resources are fully exploited.

The promise of LLMs in this space is twofold. First, they can serve as semantic intermediaries, capable of interpreting unstructured user inputs and aligning them with predefined schemas. Second, they enable mixed-initiative interaction, where automation handles straightforward cases while human oversight ensures correctness in ambiguous ones. This balance is crucial: while automation improves speed, user validation preserves trust, accountability, and accuracy. Confidence scores, editable outputs, and transparent reasoning are all mechanisms that support such a human-in-the-loop workflow.

While prior work has explored LLM-assisted form filling, schema alignment, and proactive assistance, many existing systems remain limited to narrow domains or require extensive bespoke engineering to integrate with real infrastructures. A gap remains in demonstrating how LLM agents can be embedded in heterogeneous environments while maintaining transparency and user control.

This work addresses that gap by introducing a domain-agnostic, MCP-compatible architecture that shows how LLMs can operate across diverse datasets and document formats, map them to arbitrary schemas, and support confidence-based mixed-initiative interaction. The novelty lies in the integration strategy, modular engineering design, and emphasis on controllability rather than on proposing a new model.

In summary, our contributions are:

- a modular and domain-agnostic pipeline for LLM-driven structured input generation,
- an integration strategy based on the Model-Context Protocol enabling deployment within existing infrastructures,
- design principles for confidence-based mixed-initiative interaction grounded in HCAI literature.

By building on prior work in implicit interaction, usability evaluation, and context-aware interfaces, we propose a reusable blueprint for embedding LLM-driven structured input generation into interactive systems.

5.3.1 Methodology

Building on our studies in need-finding, usability evaluation [41, 29], and implicit interaction interfaces [39, 33, 35, 38], this research examines how Large Language Models (LLMs) can support structured input generation in real-world workflows. Section 3 presents the methodological foundations of the system. We expand on the design rationale behind the AI agent, the integration strategy using the Model-Context Protocol (MCP), and the mechanisms adopted to improve robustness. We also outline the evaluation dimensions that guide the ongoing assessment of the approach.

Our approach combines LLM-based semantic reasoning with modular integration strategies and user-centered evaluation. In doing so, it enables structured input generation to be applied across domains with minimal reconfiguration, while preserving transparency and ensuring that users remain in control of the final outputs.

5.3.1.1 AI Agent Design

The AI agent is designed as a schema-aware extraction component that interprets natural language descriptions and document content and maps them onto the fields defined in the provided schema. The agent combines semantic interpretation, document analysis, and schema alignment in one reasoning process. Figure 5.1 illustrates how user descriptions and files flow into the agent, which processes them in one pass and produces the structured output that mirrors the schema. The figure is referenced early because it illustrates the conceptual flow guiding all subsequent design decisions.

5.3.1.2 Integration with Existing Infrastructures

To support deployment in heterogeneous systems, we rely on the Model-Context Protocol (MCP) [8]. MCP provides a lightweight specification for connecting LLM agents to external tools, files, and application contexts. In our implementation, MCP enables the agent to request form schemas, retrieve documents, and invoke preprocessing modules without embedding system-specific code in the prompt. This separation between model logic and system capabilities makes the approach portable and adaptable to new domains.

5.3.1.3 Improving Accuracy and Handling Ambiguity

To improve robustness, we adopt an iterative refinement loop inspired by mixed-initiative principles. The initial extraction is performed using schema-guided prompting. When ambiguities or competing interpretations arise, the agent evaluates multiple candidates and selects the value best aligned with the schema, assigning an appropriate confidence score. Users may edit the output, and their corrections can be reintegrated into later iterations to progressively adjust the prompt configuration and reduce recurring errors. This refinement loop supports transparency and adaptability without requiring complex retraining procedures.

5.3.1.4 User-Centered Evaluation

This section outlines the evaluation dimensions guiding the ongoing assessment of the system. While a full user study is part of future work, we adopted established HCI methods for evaluating cognitive load, usability, and trust [6, 188]. These dimensions inform the design of the system and structure the feasibility checks conducted within the MICS project.

Evaluation focuses on three dimensions:

- **Cognitive load:** to be assessed through task completion time, error rates, and subjective workload measures.
- **User experience:** to be measured via usability questionnaires and qualitative feedback.
- **Trust and oversight:** analyzed through interactions with confidence scores, editable outputs, and validation mechanisms.

These dimensions guide the design of the system and inform future empirical studies, ensuring alignment with human-centered principles such as transparency, control, and adaptability.

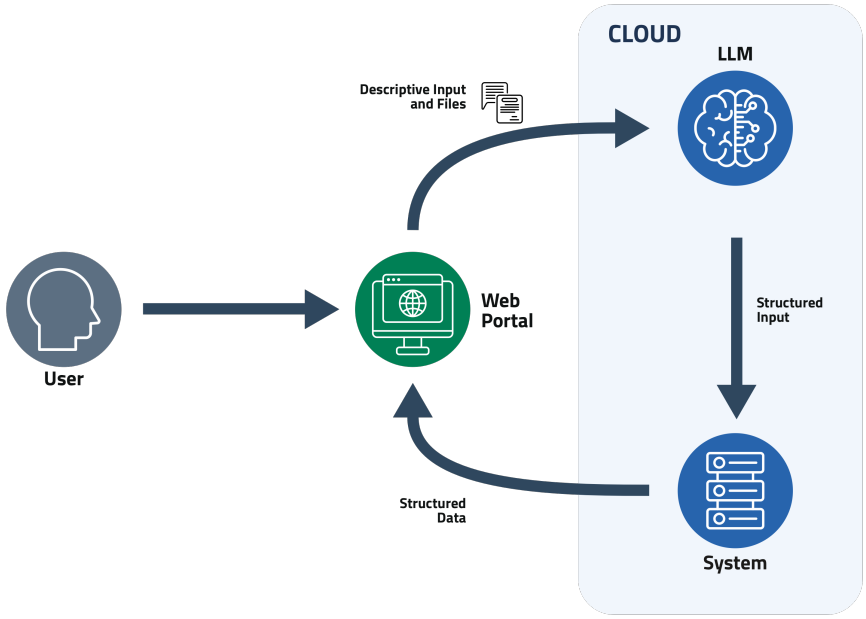


Figure 5.1. The LLM receives both natural language descriptions and uploaded documents to generate structured output for form completion.

5.3.2 Engineering and Modular Pipeline

5.3.2.1 Pipeline Overview

The system is implemented as a modular pipeline that connects form extraction, document processing, LLM reasoning, and structured output generation. The architecture prioritizes transparency, portability, and the ability to test each component independently. Figure 5.2 shows the four main stages:

1. **Form Retrieval:** A script fetches a form's HTML structure given a form ID and a web page URL, and converts it into a structured `form_fields` JSON schema 5.3.2.2. This schema defines the fields to be completed, their types, and any validation patterns. From this representation, the system automatically generates a textual description that informs the user about what information is expected and suggests which types of documents (e.g., invoices, resumes, certificates) could support the extraction of the required data.

2. **Input Collection:** The user provides a natural language description of the desired input and may optionally upload relevant documents. To guide this step, the system produces a contextual prompt that helps the user formulate an effective description and select appropriate files. This mixed-initiative approach

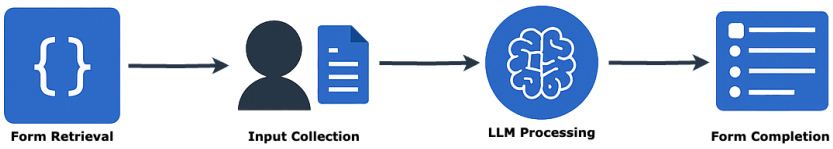


Figure 5.2. System Workflow pipeline

reduces cognitive load by making expectations explicit and by suggesting what kind of information is most useful.

3. **LLM Processing:** The agent receives as input the `form_fields`, the user's description, and the uploaded documents. These are processed using the `gpt-4o-mini` model equipped with the `File Search` tool. The LLM performs semantic matching between schema fields and the extracted information, resolves ambiguities when multiple candidate values are present, and integrates information across different sources to produce coherent outputs.

4. **Form Completion:** The agent outputs a structured JSON object containing the field `name`, the extracted `value`, and an associated confidence score ranging from 0 to 5. This JSON is then injected into the original form to complete it automatically. Confidence values guide subsequent user interaction: high-confidence predictions can be accepted directly, while medium- or low-confidence predictions are flagged for review, enabling a transparent human-in-the-loop validation process.

This modular design provides two advantages. First, each stage of the pipeline can be tested and refined independently, facilitating iterative development and targeted evaluation. Second, the architecture supports extensibility, allowing the system to be adapted to different domains simply by changing the schema and prompt configuration rather than the underlying code. By treating the LLM as a context-aware reasoning component and surrounding it with system modules for data collection, interpretation, and integration, we create a cohesive environment where users can interact naturally while benefiting from structured and verifiable outputs.

By structuring the workflow in this way, the system achieves a balance between automation and oversight. It reduces the effort required for repetitive data entry tasks while ensuring that users remain in control of the final outputs. The modularity of the pipeline also makes it portable to different domains, as only the schema and associated prompts need to be adapted to new tasks, without redesigning the entire architecture.

5.3.2.2 Agent Design and Prompt Logic

At the core of our system lies a specialized LLM agent whose behavior is shaped by a carefully crafted system prompt. This prompt enables the agent to perform three key tasks effectively. First, it interprets the user's intent based on a natural language description, extracting the essential information needed to complete the task. Second, it analyzes the contents of any accompanying documents, such as PDFs or spreadsheets, to identify and extract relevant data that may support or clarify the user's request. Finally, the agent combines these sources to generate a structured output, filling in the appropriate form fields with the most accurate values and assigning a confidence score to each entry.

To ensure reproducibility, we provide a concise overview of the prompt structure used to configure the specialized LLM agent. The full prompt is not included due to its length and implementation-specific formatting, but the core components are stable and can be readily recreated.

The system prompt is structured around three main responsibilities:

1. **Task framing:** defines the agent's role as a schema-alignment assistant responsible for interpreting natural language descriptions and uploaded documents in order to populate the provided `form_fields`.

2. **Schema interpretation:** instructs the model on how to interpret field names, types, validation patterns, and domain hints contained in the `json_schema`. The schema itself defines the expected output structure; the prompt does not specify explicit output formatting rules beyond instructing the model to follow what is defined in the schema.

3. **Confidence scoring logic:** describes how the model should internally assess certainty, how to identify ambiguous or partially supported extractions, and how to assign confidence values from 0 to 5. The schema includes a `confidence` field, which drives the expected format of the model's output.

A short illustrative excerpt is shown below:

> "You receive a JSON schema describing the fields of a form, including the structure of the expected output. Use the schema to extract relevant information from the user's description and documents. Populate the fields defined in the schema and assign a confidence score from 0 to 5 based on the reliability of the extracted information."

The expected `json_schema` input format is:

Listing 5.1: JSON Input Schema

```
{"form_fields": [ {"name": "<input name>", "type": "<text|
    number>", "pattern": "<regex>"} ], "description": "<natural
     language description>" }
```

The output is a list of `form_fields` entries with corresponding `value` and `confidence` fields (0–5). The confidence score measures the agent's certainty about each extracted value and serves as a mechanism for guiding user interaction. Values with a confidence score of 5 are considered highly reliable and are used directly to populate the form without requiring user intervention. Scores of 3 or 4 indicate moderate confidence; these values are still used but are explicitly flagged for the user to review and confirm before submission. Entries with scores below three are discarded and not used in the form, prompting the system to either leave the field blank or request additional input from the user.

Given the description: *Hi! I'm Mario Rossi, you can find me at mario.rossi@example.com or call me at +3955511449683.* The agent will return the following structured output:

Listing 5.2: JSON Output Example

```
{"form_fields": [
    {"name": "firstName", "value": "Mario", "confidence": 5},
    {"name": "lastName", "value": "Rossi", "confidence": 5},
    {"name": "email", "value": "mario.rossi@example.com", "
       confidence": 5},
    {"name": "phone", "value": "+3955511449683", "confidence":
        5}
]}
```

Once generated, the structured output from the AI agent serves as a direct data source for the target system, such as a web form interface, enabling an automatic population of its fields. By aligning the agent's output format with the system's expected input schema, the filled values can be programmatically injected into the form, completing the data entry process with minimal user intervention.

5.3.2.3 Implementation and Feasibility Tests

The prototype has been implemented in Python, integrating the `gpt-4o-mini` model with the File Search tool to process unstructured documents. Each stage of the pipeline, form retrieval, input collection, LLM processing, and form, completionwas designed as an independent module, allowing isolated testing and refinement.

The early evaluation conducted within the Made in Italy Circolare e Sostenibile (MICS) project [125] serves as a feasibility assessment of the proposed pipeline. The goal of this evaluation is not to measure usability or user experience, but to

verify that the system can (i) parse heterogeneous domain documents, (ii) extract schema-relevant values, and (iii) assign confidence scores that correlate with expert judgment. The pilot dataset includes technical datasheets, short textual descriptions, and packaging specifications provided by project partners. For each document set, two domain experts annotated the correct values for each schema field, which served as a reference for assessing extraction correctness.

Preliminary observations indicate that the system can correctly populate a substantial portion of fields in the target schema. High-confidence predictions (≥ 4) frequently aligned with expert annotations, while low-confidence predictions typically corresponded to ambiguous phrasing, incomplete documents, or conflicting sources. Although not intended as a full evaluation, these observations confirm the technical feasibility of the approach and highlight where additional preprocessing and prompt refinement are required. A broader and more controlled evaluation will follow as part of the next phase of the project.

5.3.3 Discussion

This paper presented a modular pipeline for transforming unstructured natural language descriptions and heterogeneous documents into structured inputs aligned with predefined schemas. By combining form introspection, document processing, and schema-guided LLM reasoning, the approach demonstrates how intelligent agents can reduce the burden of manual data entry while preserving human oversight through editable outputs and confidence scores.

The preliminary evaluation within the MICS project provided encouraging early evidence of feasibility, particularly in handling domain-specific descriptions and technical documents. These findings suggest that LLM-based structured input generation can support real-world workflows that rely on diverse data sources and rigid schemas.

Several directions remain open for future work. Scaling the evaluation to more diverse datasets and broader project partners will be key to assessing robustness and generalizability. Additional preprocessing-such as OCR for scanned PDFs, table extraction, or domain-specific parsers—could improve handling of noisy inputs. Iterative refinement strategies that incorporate user corrections into subsequent predictions also represent a promising avenue for improving reliability and transparency.

The approach nonetheless has limitations. Confidence scores, while useful for mixed-initiative interaction, are not statistically calibrated and may lead to over- or under-trust. Extraction performance decreases with low-quality or irregular documents, and terminology inconsistencies still introduce ambiguity. Finally, the current evaluation remains preliminary and does not yet include controlled user studies, which will be essential to assess workload reduction, trust, and usability in

depth.

Overall, this work provides a practical foundation for integrating LLM-driven structured input generation into existing infrastructures. By outlining both the engineering approach and its current constraints, it aims to support future research on adaptive, transparent, and domain-agnostic systems for intelligent data entry.

5.4 Automating Form Completion with Large Language Models

Form completion is an everyday but often frustrating task, particularly when required data are dispersed across different documents. Despite their prevalence, digital forms offer limited innovation to reduce user workload or prevent transcription errors. Recent advances in artificial intelligence, notably LLMs, have opened new possibilities for automating context understanding, content extraction, and form field suggestion, all with conversational feedback. Our work builds on prior research [225] in LLM-assisted data entry by focusing on a real-world scenario from the MICS (Made in Italy Circolare e Sostenibile) project [125]. In MICS, waste producers must submit detailed, technical information—often scattered in technical datasheets and certifications—into a web platform to promote the reuse of industrial waste. Even for experts, this is challenging and time-consuming, making it a strong candidate for AI assistance.

5.4.1 System Overview

Our prototype enables users to upload technical documents and leverages GPT-4o-mini [146] to automatically extract and suggest form field values. Users remain in control, with the ability to review, correct, or refine the system's suggestions using a conversational interface. Each suggested value includes a confidence score to help users prioritize their review. The system's architecture combines two main components: a specialized LLM agent ("materialGPT"), guided by a custom prompt following state-of-the-art guidelines [14, 46]. While full conversational integration is still in progress, the system currently supports both document-driven pre-filling and separate interactive chat for clarifications.

5.4.2 Case Study and Evaluation

We evaluated our approach with fifteen participants unfamiliar with materials engineering, reflecting realistic users tasked with data entry. Each participant completed the MICS form for two materials, one manually, one with LLM assistance, using real datasheets and certifications. Seven key fields were filled, ranging from free text (material description) to selection fields (material type), checkboxes (certification presence), and percentages (recycled content). The evaluation combined quantitative and qualitative methods. Participants worked in three groups: manual completion, LLM-assisted completion, and LLM-assisted with conversational capabilities. All participants verbalized their thoughts during the task and completed the System Usability Scale (SUS) questionnaire [45] afterward.

5.4.3 Results

LLM assistance reduced average form completion time by over 40%, with some users seeing reductions over 50%. Notably, users, even without domain knowledge, navigated the form more confidently and made fewer errors when using AI suggestions. While the LLM sometimes generated incorrect or inconsistent values (especially in descriptive fields), users were generally able to spot and correct obvious mistakes. Compared to manual completion, the total error rate was lower with AI support. Usability scores were high: 93% of participants indicated they would frequently use the AI-assisted system, and the average SUS score was 86.3. Participants appreciated not having to search for information themselves and felt their role shift from data entry to reviewer, especially when conversational interaction was available.

5.4.4 Discussion

This preliminary study is limited by its small sample size, the non-expert user group, and the use of only two types of materials with well-structured documents. The current prototype does not yet fully integrate real-time LLM conversation in the form interface, nor does it expose confidence levels to users. Finally, the persistence of occasional LLM "hallucinations" [96] means human oversight remains criticalb[35]. Future work will focus on tighter integration of conversational AI into the form interface, broader document and user type coverage, and mechanisms to highlight uncertain or potentially unreliable auto-filled values. We envision this approach evolving into a robust, domain-aware assistant capable of supporting complex, high-stakes workflows in digital administration and industry.

Part II

LLMs for interaction engineering and design evaluation

Overview

This part presents the main research stream of this thesis and examines how large language models can support the engineering and evaluation of interactive systems. Rather than treating LLMs as standalone chat interfaces, this part frames them as components embedded within existing design and assessment workflows [190]. In this role, models generate intermediate artifacts such as critiques, rationales, structured notes, candidate requirements, or reusable context that designers can inspect, validate, and iteratively refine. The overarching goal is to combine selective automation with designer agency, so LLMs accelerate design work while preserving human control, accountability, and traceability of decisions [91].

Chapter 6

Designing for Fallible Automation

6.1 Introduction

> *Emanuele has a car that automatically opens when he takes the driver's door handle with the keys in his pocket. However, one winter day, pulling the handle, he notices it is ice-covered, and the car is not opening. So he had to look for the keys in his jacket pockets and manually open the car.*

Intelligent systems can make a real difference in our lives when they work, but they do not always work.

In this paper, we analyze the different contexts in which one chooses to integrate artificial intelligence into an interface and the implications of this choice in managing user interaction. As a driving example, we will use an application to help users deal with car-related tasks. This application attempts to minimize explicit interaction, so our focus is particularly oriented toward implicit interactions. However, many of the design issues identified appear to have a broader scope. Where possible we will choose the simplest examples that exhibit a particular phenomenon; some are not very "intelligent" but share one or more of the three C's of AI-based systems: Complexity, Uncertainty, or Coadaptation.

The paper will focus on explicating different types of errors, recognizing an error state, and managing and resolving these inevitable situations. We aim to understand how to design more *robust* human-AI systems so that these initial errors do not lead to more catastrophic failures. However, first, we will motivate the general area and case study.

6.1.1 To err is AI

To err is human – as researchers and professionals in human-computer interaction, part of our expertise and culture is understanding our users' glories and fallibilities. We do not expect our users to perform like automatons; we know there will be lapses of concentration, misinterpretation of data, limited experience, and physical slips.

Effective human-computer interaction design creates systems that work and that are robust despite these occasional human lapses or mistakes.

We design systems with features that scaffold the users' memory, for example, through recognition rather than recall; features that highlight potential slips, for example, spelling checkers; features that reduce the impact of errors, for example, undo; and features that help the users detect and repair problems, for example ensuring rapid feedback on the effects of actions. All in all, these design elements prevent user errors from becoming system failures.

Note the contrast of the human with an automaton: the machine that performs flawlessly time after time, processing billions of calculations, printing millions of payslips; infallible albeit limited and unimaginative. Of course, hardware can fail, especially for very large-scale computation. Much of the complexity of cloud-computing infrastructure and algorithms, for example, MapReduce, is about making the overall computer system behave as if it is flawless [65]. The expectation is that the overall system should behave ... like an automaton. Recall that HAL, the AI in Kubrick's 2001, becomes homicidal precisely because it had made a mistake and was trying to cover that up.

Of course, we know AI systems are not like this. They are trained on limited data and often use limited sensor data. They are not simply following pre-determined rules but attempting to interpret the environment, particularly the behaviors and intentions of users. The results of an AI system are richer and more nuanced than an "automaton" but, consequently, not flawless.

Effective human-AI interaction design creates robust systems that work, despite these occasional AI lapses or mistakes.

6.1.1.1 When it is a good idea?

Using AI-based systems with implicit interaction can be a great idea when the AI system can perform complex tasks that are difficult or time-consuming for users to perform manually. The user experience can be substantially improved by, for instance, chatbots or virtual assistants that can comprehend natural language and offer individualized advice or answers.

Additionally, implicit interaction can be useful in situations where the user is not able or willing to provide explicit input, such as in the case of a driving user.

6.1.1.2 When it is not a good idea?

Implicit interaction may not be the ideal strategy in some circumstances. For instance, specific human input may be required if the system demands high accuracy or precision to guarantee the desired result. Implicit interaction may also be viewed as obtrusive or confusing when the user needs total control over the system.

6.1.2 Driving examples

The examples used in this chapter are drawn mainly from car-related tasks, and in particular from smart-parking scenarios [39, 33, 23, 152]. These examples are retained not because the argument is limited to mobility, but because they provide a concrete and safety-relevant setting in which the consequences of fallible automation are easy to observe. In such contexts, errors are not merely abstract prediction failures: they directly affect whether users can complete tasks, understand system behavior, and recover from breakdowns. For this reason, mobility examples provide a useful lens for discussing broader design issues that also apply to other AI-based and proactive systems.

Implicit interaction in the context of smart parking can provide several benefits, including reduced demands on driver attention, shorter parking-search time, improved use of parking spaces, and potentially reduced traffic congestion [21]. A running example throughout this chapter is an intelligent parking application that infers whether the user is driving, where the car has been parked, and when parking-related support should be activated. For instance, the system may share the parked-car location with other authorized users, such as family members using the same vehicle. To extend the example toward more critical situations, one may also imagine the same system supporting related automated actions, such as unlocking the car when the owner approaches. Another example of implicit support is automatic parking-fee payment: when a vehicle enters and leaves a parking space, the system may infer the parking session and complete the payment without requiring explicit user input.

Although these examples originate in sensing-based mobility applications, the same design logic extends to other AI systems discussed in this thesis. In LLM-based assistants, for instance, a system may infer when a reminder, suggestion, or contextual action is appropriate on the basis of remembered context or partially implicit signals. In both cases, the central issue is the same: once the system begins acting on inferred rather than fully explicit input, errors in recognition, timing, or interpretation can quickly propagate into user-facing failures.

As a recurring example, consider a user, Emanuele, who relies on an AI-based app to track the parking location of his car and share this information with his son. In the ideal case, the app generates an event when it detects a change in activity with high certainty. For example, if the system infers that the user has been driving

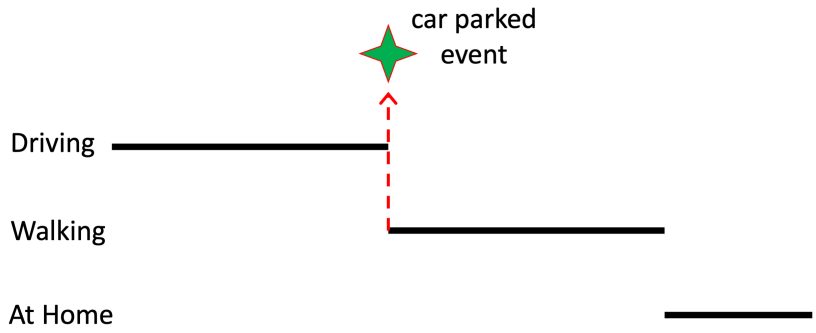


Figure 6.1. Ideal case – perfect sensing and an event is generated at the moment the user's activity changes

and then starts walking, a "car parked" event is generated (Figure 6.1).

Problems arise when the raw sensing input is ambiguous. In these cases, there may be a period of uncertainty instead of a direct transition from driving to walking. If this uncertainty is brief, the lower-level sensing layer may still generate the "car parked" event at an acceptable moment. If the uncertainty persists, however, there may be no clear trigger point, so the event may not be generated at all, or it may be generated too late, causing a substantial error in the estimated parking location.

In this example, Emanuele parks at night, and his son uses the app the following morning to find the car. The previous evening, however, the sensors on Emanuele's phone remained uncertain about whether he had stopped driving. As a result, the system either failed to generate the "car parked" event or generated it in a significantly incorrect location. Emanuele may still remember where he parked, but his son has no reliable information to recover the car's location. The importance of this example is not limited to parking: it illustrates a general property of fallible automation, namely that an upstream inference error can remain invisible at first and only later surface as a practical breakdown for the user.

6.1.3 Chapter Outline

This chapter is organized as follows. Section 6.2 provides a literature overview of previous human and system error and repair research. In section 6.3, we define and categorize errors that can occur in human-AI systems. In section 6.4, we discuss methods for detecting errors and the roles and responsibilities of the different actors in the detection and repair process. Section 6.5 explores the challenges of sensing and detecting failures, including cases where sensor data is inconsistent or ambiguous. In section 6.6, we discuss design strategies for addressing the challenges identified in the previous sections, including improving early detection and communication, ensuring human-AI system reliability, and clarifying responsibility for detecting and repairing errors. Conclusions and ongoing work are reported in section 6.7.

6.2 Related Works

Errors in human-AI systems have become a critical topic in recent years as AI systems are increasingly integrated into various aspects of our lives. Despite the significant progress made in AI research, AI systems are still prone to errors that can have serious consequences, particularly in safety-critical applications such as autonomous vehicles, medical diagnosis, and financial decision-making. This has led to a growing body of research focused on understanding the nature of errors in human-AI systems and developing strategies to minimize their occurrence and impact.

6.2.1 Levels of automation

Before AI was born, several automatic systems were built. The work by Parasuraman et al. [156] proposes a model that describes the different types and levels of human interaction with automation. The model is based on three types of interaction: direct, supervisory, and indirect. Direct interaction involves the human controlling the automation directly, while supervisory interaction involves the human monitoring and intervening in the automation's actions. Indirect interaction involves automation making decisions, and taking actions on behalf of the human.

The model also defines four levels of interaction, ranging from the lowest level, where the human is only observing the automation, to the highest level, where the automation can perform tasks autonomously without human intervention. When considering automobile automation, the Society of Automotive Engineers has made further distinctions and defined five levels [172, 173], which have become influential and have been adopted or adapted by many national and international standards:

However, Shneiderman argues that the one-dimensional view of automation implied by these levels of automation is too simplistic. Instead of a single dimension between human control and computer automation, he suggests considering a two-dimensional framework with higher and lower levels of human control compatible with higher and lower levels of levels of automation [189]. Crucially he considers the point at which *both* human control and computer automation are high, working together, as sweet spot for "reliable, safe and trustworthy" AI systems.

6.2.2 Intelligent interactions

One of the iconic early uses of AI in user interaction was EAGER (Extraction, Analysis, and Generalization Environment for Repetitive tasks), which allowed users to program repetitive tasks by example [59]. EAGER is based on the idea that users can demonstrate how to perform a task once, then the system will automatically extract the relevant information and generalize it to perform the task in other

instances. The paper describes the design and implementation of EAGER, including the algorithms used to extract and generalize examples, and presents several case studies demonstrating the system's effectiveness. The authors argue that EAGER has several advantages over traditional programming methods, including ease of use and increased productivity. Since EAGER there has been continuous work within the intelligent user interfaces community, albeit until recent years more limited uptake in deployed systems.

In a paper from 2017 [171], Human Information Interaction (HII) refers to the process of humans interacting with information to achieve a specific goal. This can involve searching for information, processing it, and using it to make decisions. HII is a complex process that can be influenced by a wide range of factors, including individual differences, the nature of the task, and the characteristics of the information itself. Crucially HII and AI have increasingly overlapped in areas of big data analysis and systems using big data to generate models for intelligent interactions.

6.2.3 Errors in human-AI systems

We now proceed to review the existing literature on errors in human-AI systems, with a particular focus on the causes of errors, the types of errors that can occur, and the approaches that have been proposed for mitigating errors in these systems.

Errors can occur in human-AI systems when there is a mismatch between the expectations and capabilities of humans and AI. Errors can arise for various reasons, such as data bias, lack of transparency in decision-making, or miscommunication between humans and AI systems. For example, AI systems may make errors in image recognition tasks when they encounter images that are different from the ones they were trained on or when used in contexts that were not anticipated during their development. Human users may also make errors when interacting with AI systems, such as misinterpreting the system's output or failing to provide the system with the necessary inputs.

To minimize errors in human-AI systems, it is important to design AI systems that are transparent, explainable, and robust to variations in data and context. It is also important to ensure that humans are properly trained to interact with AI systems and understand the limitations and capabilities of these systems. Ongoing monitoring and evaluation of AI systems can help identify and address errors as they arise and improve their overall performance and reliability.

There has also been research regarding Second Language Learning [72] and Error Remediation in those systems using Artificial Intelligence. AI systems can be trained to analyze learner performance and identify patterns of errors that are common among learners. This analysis can be used to develop targeted interventions to help learners improve their language skills. For example, an AI system may identify that

a group of learners is struggling with a particular grammar rule and provide them with additional exercises or feedback to help them master that rule.

To be effective, AI systems used for second language learning must be able to identify errors accurately and provide appropriate feedback to learners. This requires the system to be trained on large learner performance data datasets and recognize subtle variations in language use that may indicate errors.

Errors can occur in human-AI systems used for second language learning if the system is not adequately calibrated or cannot recognize the full range of errors that learners may make. For example, an AI system may fail to recognize errors unique to certain dialects or resulting from interference from a learner's first language.

It is important to continually evaluate the AI system's performance and make adjustments as needed to minimize errors in human-AI systems used for second language learning. This may involve retraining the AI system on new data or adjusting the AI system's algorithms to recognize certain errors better than others. Additionally, it is important to provide learners with opportunities to interact with human teachers or tutors who can provide additional feedback and support to help them overcome errors and improve their language skills.

6.2.4 Human and system error and repair

Preventing and dealing with user errors has always been a central part of the HCI literature. For example, two of Nielsen's heuristics are "Error prevention" and "Help users recognize, diagnose, and recover from errors" [137]. The first concerns scaffolding, such as using fixed sets of options rather than free typing or 'recognition rather than recall' [191], whereas the second concerns what happens after an error occurs.

The latter is closer to the main focus of this paper, and the importance of being to tell *that* something has gone wrong is central in Norman's influential seven-stage model [141]. Three stages are about the user working out what to do and doing it, but three are about assessing the outcomes of their action on the system, that is *feedback*. The model makes distinctions about problems at different levels, most importantly between slips and mistakes. The former, a slip, is when the user's intended action is correct, but there is a problem in executing it, for example, pressing the wrong key. The second, a mistake, is when the users' fundamental model of the system state or behavior is incorrect. So they formulate an incorrect action, for example, that they think ctrl-U means "undo".

Crucially, the ability to perform this assessment and evaluation depends on the system giving timely and informative feedback. Not surprisingly, this is a key part of classic user interface design, for example, "Visibility of system status" in Nielsen's heuristics [137] and "Offer informative feedback" in Shneiderman's Eight Golden

Rules [191]. These do not prevent things from going wrong but mean that errors are noticed and thus can be fixed. Shneiderman's "permit easy reversal of actions" [191] is precisely addressing this ability to recover.

Human communication is rarely problem free; we mishear and misunderstand one another and yet manage. This is in part due to processes of *repair*, where we realize problems have occurred and deal with them, but most often in ways that do not interrupt the overall flow of the conversation. Frohlich used conversational analysis of human-human repair to inform the design of human-machine dialogues [80].

In conversation and other aspects of life, timeliness is critical for repair; it is usually far easier to correct errors as soon as they happen than later when there may be further knock-on effects of the error. Some years ago, Stephen Brewster noted an expert slip with on-screen buttons. The expert user would occasionally not properly press the button. Still, precisely because they were experts, they did not pay attention to the semantic feedback and only noticed too late that the error had occurred. Adding appropriate sound did not prevent these errors from occurring, but it did mean that they were immediately noticed and hence could be repaired [44].

The middle stage of Norman's seven-stage model is system execution, which, as noted previously, is often assumed to be flawless, at least in execution, if not in design. Some design approaches to "recognize, diagnose, and recover from errors" apply equally well to AI and user errors. However, there may be additional problems as the AI errors may not be immediately apparent to the users.

Seamful design [51] addresses this by embracing the deficiencies in sensing and system behavior and bringing them to the surface as an explicit part of the interaction. This is often used in an entertainment context, for example, the early work using gaps in WiFi as part of gameplay. However, it also has more prosaic applications, for example, the fact that a mobile phone shows signal-strength bars allows various ameliorative actions, such as standing closer to a window.

Appropriate intelligence [71] suggests that limitations in AI performance should be managed by embedding the intelligent algorithms within interactive contexts that make the errors in the AI less damaging, for example, using easy-to-accept suggestions rather than full automation. Formal techniques for the design of sensor-rich IoT systems [70], expand on this by modeling both human activity and sensor limitations and then attempting to match the certainty of sensing and the varying consequences of errors in different situations to create rules and set thresholds, which are highly likely to be correct in the most critical situations, whilst giving 'best efforts' where it is less critical.

6.3 Define errors and failures

Accuracy and other metrics are crucial in designing ML/DL models, but models usually have less than 100% accuracy. Indeed, AI can make a wrong assumption about the context or user behavior due to bad sensing or deductions.

This section will look at different kinds of AI errors that may occur, and different dimensions in which to classify them. In fact, an error in the strict sense of the term is to be considered a malfunction of the system with respect to the design expectations, but within a complex system, errors can be defined from several points of view. In this section, we distinguish between observable and unobservable errors (6.3.1) and look at the differences between errors of omission and commission (6.3.2). Finally, we provide our own classification based on users' perception of errors and their changing expectations and behaviors (6.3.3).

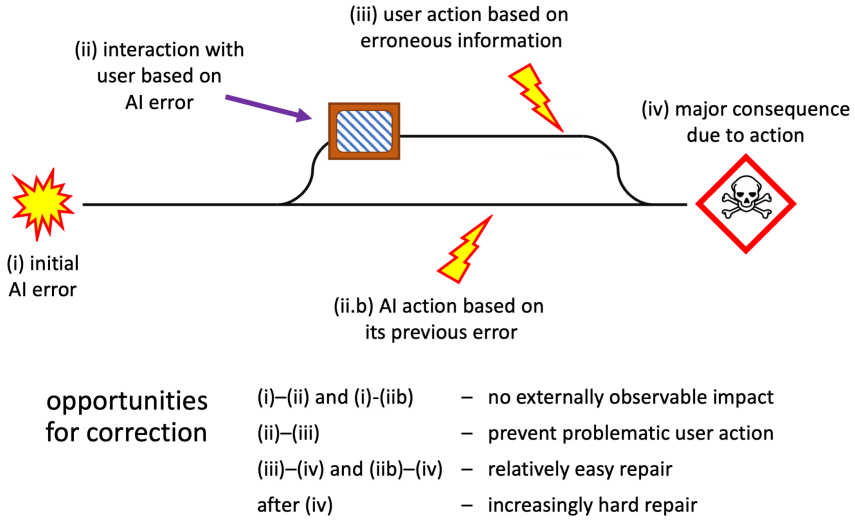


Figure 6.2. Error Timeline

6.3.1 Observable errors

When the AI gets something wrong, that mistake may not cause a problem for the user, let alone a system failure. Based on the deductions and calculations of the AI, subsequent actions are usually programmed for the system's functioning. However, not all of them directly affect the user and their system experience. Based on this consideration, we distinguish between observable and unobservable errors on the user's part. Figure 6.2 schematizes the possible flow of actions that derive from a system's decisions made based on an AI error.

The system's functions can be divided into background and foreground.

Background actions happen when the system seems to be doing nothing, such

as when the smartphone is in the user's pocket. These functions can be activated because they are programmed, but the context and the environment can also trigger them. On the other hand, when the system does something and exhibits observable behavior, such as updating the user interface, these results fall under the category of foreground actions.

When an AI makes a computation, such as a context assessment, whether the result is right or wrong, background and foreground actions can be triggered. If the AI makes a mistake and the consequent action is in the foreground, this will result in an error observable to the user. As suggested in the upper path of Figure 6.2, an observable error can also induce the user to propagate the error with one or more new actions. Similarly, a background action triggered by an error condition is also an error, but in this case, it is not user-observable and will not initially generate additional error conditions.

Please note that an unobservable AI error can become observable later, impacting the consequences, as discussed in the following section 6.4. Moreover, in this phase, we distinguish between the possibility for the user to detect and intercept the error, and not between detected and undetected errors.

Also, note that some AI errors could generate erroneous actions within the same system that are observable in some cases and for some users but unobservable for others. Consider the example presented in 6.1.2. Assume that Emanuele parks his car, but the AI wrongly register its location due to some sensing errors or uncertainty. This error is a background error for Emanuele, who can proceed and get home without perceiving any errors in his user experience. But later, Emanuele's son needs to get the car, so he walks to the wrong car location to discover it is not there. In this second case, the action resulting from the AI error becomes observable and actively damaging.

This dimension of distinction, observable or unobservable, is intuitive but fundamental for introducing an error's criticality level. Closely linked to the observability of an error is the possibility of intercepting and repairing the error.

6.3.2 AI errors of commission and omission

For human errors, one often distinguishes errors of commission: things done wrongly, and errors of omission: things that are not done when they should have been. Both can have deleterious consequences, but typically errors of commission are regarded as more severe or blameworthy. Similarly, an AI can act in error by doing something wrong or doing nothing (when supposed to do something). For example, the AI inaccurately estimates Emanuele's car parking location and wrongly modifies the UI to inform his son about it (commission error). On the other hand, the AI may incorrectly assume that Emanuele is still traveling, not informing his son

that the car is free to use (omission error). Table 6.1 describes the four possibilities of an AI taking actions that could be either correct or wrong based on the design expectations.

	Expect something	Expect nothing
Does something	correct/wrong	wrong
Does nothing	wrong	correct

Table 6.1. AI actions based on design expectations

Note we are referring here to design expectations. Ideally, the user will understand the system; hence, design and user expectations are the same. Still, in practice, user expectations could be misplaced: for example, if the user believes the system can do something it cannot.

Note, too, that this will interact with whether the AI errors are observable. Typically, wrong actions (commission error) will be observable, but erroneous inaction (omission error) may not be noticed, even if the user expects the outcome. For example, the user may expect the system to identify free parking spots vacated by other app users; however, if the system fails to notice these parking spots, the user would hardly tell if there are no parking spots or if the AI did not notice them.

6.3.3 Users perception of errors, changing behavior and expectations

It is interesting to highlight the many degrees of perception the user may have toward AI errors before discussing their likelihood of occurrence and the value of error recognition in preventing more severe failures. Below, we discuss the concepts of detection, perception, and understanding.

- Detection is the ability of a human to notice when an AI system makes an error. This may involve recognizing a discrepancy between the expected and actual output of the system or identifying patterns or trends that suggest a problem. For example, Emanuele notices that the app registered their last parking spot in the wrong location.

- Perception is the ability of a human to interpret and make sense of the error detected in the AI system. This may involve having an idea of the context and predicting the consequences of the error. For example, Emanuele predicts that if the app shows the parked car in the wrong location, it may fail to track when the car's location changes again.

- Understanding is the ability of a human to comprehend the underlying causes and factors that may have contributed to the occurrence of the error in the AI

system. This may involve knowledge of the technical aspects of the system, as well as the broader social, ethical, and legal implications of the error. Understanding an AI error may also involve identifying potential solutions or strategies for preventing similar errors from occurring in the future. For example, suppose Emanuele understands that the AI system fails to correctly locate the parked car when there is a poor connection, like in the underground parking lot of his home. In that case, he may manually register the car's position every time he gets home.

Based on that and what was introduced in the previous sections, we propose an additional error classification based on user perception, understanding, and changing expectations and behavior. We, therefore, distinguish errors into three types:

- Type I – The AI does something that causes problems in 'normal' (pre-intervention) behavior, but the user still does not understand the system enough. This type of error is unobservable until very late, when consequences may be costly. For example, the car unlocks as Emanuele passes it (maybe simple proximity-based switching), but he is rushing to a shop, then a thief notices, opens the door and steals something. Emanuele did not understand that proximity would unlock the car, so he could not detect the error. Hence it leads to failure.

- Type II - The user has begun to build a perception of the AI and has expectations about what it will do, but it does something different. For example, Emanuele reaches for the car door handle, expecting it to be unlocked, but it is still locked, and he breaks his fingernails. The user has a level of understanding of the system, but it is not enough to prevent failures.

- Type III – The user's fundamental model or knowledge of the world has changed due to a smart app/environment, and they are less able to do something—for example, not being able to find their way in a well-known city without a navigation app because they have become used to simply following directions. In Type III, the user's expectation is that the AI does not make errors, but there is no AI. In this case, the user can have a complete understanding of the system or not, but the accent is on the complete trust that the user has in the system.

In the next section (6.4), we will consider how an AI initial error may or may not lead to critical consequences, i.e., whether *AI error* leads to *system failure*. A failure impacts the human-AI system and possibly the user's life, and a non-negligible cost may be required to repair it. Typically, a failure comes from the user's or AI's wrong actions misdirected by an initial AI error that cannot be undone.

6.4 Error detection and repair to prevent failures

We have outlined a wide range of different types of AI errors. We saw in Section 6.2.4, when discussing human error, that early detection and repair are at the heart of preventing minor slips from becoming major problems. The same is true when we look at AI errors. No matter the course of the error, the earlier problems are found, the more likely they are to be fixable.

6.4.1 Importance of detection

Detection, by the human or the AI, is the ability to notice when an AI system makes an error. This may involve recognizing a discrepancy between the expected and actual output of the system or identifying patterns or trends that suggest a problem. As anticipated, the importance of detecting an *error* lies in the possibility of repairing it and taking an alternative action. Otherwise, an undetected error is likely to become a *system failure* - a significant error that impacts the user experience and may require a non-negligible cost to be repaired. Referring to the above example in section 6.1.2, suppose Emanuele's son can detect the error. In that case, the failure may be prevented: for example, if the AI notified him when the parking location was not detected, or its estimation is not accurate enough.

On the other hand, the repair is impossible if the error is undetected. Eventually, the user will acknowledge it, but it will have already become a failure. In our example, Emanuele's son would walk to the wrong parking location to discover that the car was not there.

Of course, the repairing action itself may be costly, and the timing of the detection is relevant: an early detected error is usually reparable with lower costs than a late detected one. For example, if Emanuele's son somehow realizes that an error occurred and is in a hurry to get the car, he can wake up his father to know where the car is actually parked.

Effective repair makes an AI system robust, and detection is a necessary condition for repair.

6.4.2 Detection and Repair - who does what?

We have seen that errors or inaccuracies are often inevitable; detecting them and repairing any consequences before they become severe is essential.

In a robust human-AI system, two main aspects must be addressed: who *detects* the problem and who will *repair* it. Indeed, both humans and AI can weigh in to detect and repair errors; it is left to the designer to choose if the agent detecting the error is the same as the one that repairs it or not.

When the user is in charge of detecting and repairing the error, a robust design should aid human detection by explicitly displaying the system state. In our example, the AI may notify Emanuele about the registered parking location, so that he could check it and fix it before going to sleep.

If the AI should detect and repair this error, it may take into account additional information gathered after the event generation. For example, if the AI is wrongly assuming that Emanuele is still driving, but then detects that he is actually connected to his home WiFi - hence detecting the error, it can autonomously make a new estimation about the car's parking location.

When the agent detecting the error is not the same as the one that needs to repair, there must be some communication between the two.

In our example, two possible design solutions are:

- Human detects, AI repairs – Emanuele notices that the car parking position on the app is wrong and give this negative feedback to the AI, which will include it in a new estimation of the car parking location.
- AI detects, Human repairs – The AI might see that its estimation accuracy is under a certain threshold, and alert Emanuele, who selects the correct car parking position on a map.

Understanding these different options allows us to consider different potential paths to error detection and recovery; for example, the user detects – user tells the AI – AI fixes; or AI detects – AI tells user – user fixes. There needs to be AI-state visualization or various forms of user interaction within these places. We will return to several of these when we discuss design implications in Section 6.6. The AI system is intelligent and changes potential user interactions with it. Several of these interactions are similar to the corresponding cases of detection and repair when the primary error was due to a human mistake or misunderstanding (often itself due to a design error).

Note that, in this section, we have lightly made the assumption that AI can detect errors. In reality, designing and developing the AI detection process can be much more complicated than human detection, which can also be based on perception and understanding. For this reason, we have dedicated the following section to the particular case of how AI can detect errors.

6.5 Sensing errors

It is reasonable that a computer system that is or isn't equipped with AI can detect certain forms of inconsistency or problems with user input as it can take in

different factors; for example, a spelling checker may not know what the user meant to write but can tell that the word typed is not in the dictionary. In contrast, if the computer system can detect its own error, why couldn't it simply do it right before making the error? Typically the answer to this lies in the changing information available to the system both from the user and the environment. In this section, we look at how this can be achieved by combining knowledge of behavior and the varying degrees of certainty of sensor data and inferences.

6.5.1 Times and information

In most AI inference/estimation tasks, we are considering two main time points:

- the time when the inference/estimation is made
- the time the inference/estimation is about

Figure 6.3 shows the time of estimation on the horizontal axis and the time it is about on the vertical axis. Points along the diagonal represent point estimates. When the time of inference is the same as the time it is about, the AI system is using its sensing data to make an inference about the current state of the world. For example, if the AI detects that Emanuele has got in his car, it may turn on the radio.

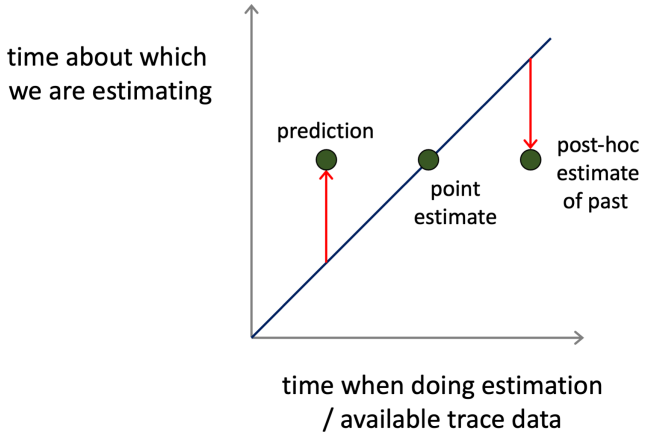


Figure 6.3. Two times: time of estimation vs. time about which we are estimating

In contrast, points above the line represent situations when the thing being estimated is in the future; that is a prediction. In the parking context, this may be when Emanuele walks out of his office and heads toward his car. The system may predict that he will drive away and therefore alert another driver that the parking space will become available.

Finally, points below the time represent post-hoc estimates about the past. These do not immediately sound useful, but consider the example above. Even if the system

does not detect the correct moment when Emanuele parks his car at night, the important thing is that the correct location is given to his son in the morning.

Crucially, more information may be available when the system makes inferences later. In general, the later an estimation is made, the greater its accuracy. This can happen for a prediction; for example, if Emanuele walks towards his car and turns into a shop, the prediction of him freeing his parking spot will change. It can also happen retrospectively, if, for example, the car parking estimation can correct itself overnight before Emanuele's son looks for the car.

In the former case, it is clear that new information is available (the user walking into the shop). Still, it is unclear what additional information would be available for post-hoc correction of the vehicle position. Sometimes, there can be delays in sensor information or other data becoming available; this is common in some application areas; for example, if some sensors are not attached to permanent networks, or there is a need for raw sensor data to be processed, such as the parking application.

For other applications, no new sensing about the time of interest is available after the event. Happily, knowledge of future states can be combined with models of human activity to improve post-hoc estimates.

6.5.2 Post-hoc estimation – traces and inconsistency

In some cases, we can create state models of normal behavior with valid traces, such as:

> ... driving <car parked> walking ...

These might come from ad hoc modeling or machine learning inference from user traces.

This model can be used for prediction, especially in cases with additional probabilistic knowledge. For example, if Emanuele is known to be at home (based on WiFi reception) at 2 am, they may be assumed to be sleeping and unlikely to drive again before the morning. However, these models can also be used to 'play backward', inferring past states, especially when sensing data is inaccurate or incomplete.

If the system comes into a certain state, it may be possible to identify strong inconsistencies. For example, if the state was 'driving' followed by an uncertain state and then definite 'walking', then the system knows that a parking event should have happened in between, even though it is unclear where. That is, the system has detected a certain failure, albeit potentially a considerable time after the event.

This does not help correct any past actions such as, but it can help later. For example, this could be conveyed via a 'best guess' estimate, such as the last point when the system was deemed to be driving, or by one that explicitly conveys

uncertainty, such as showing the set of recorded positions between the last definite driving location and the first definite walking one.

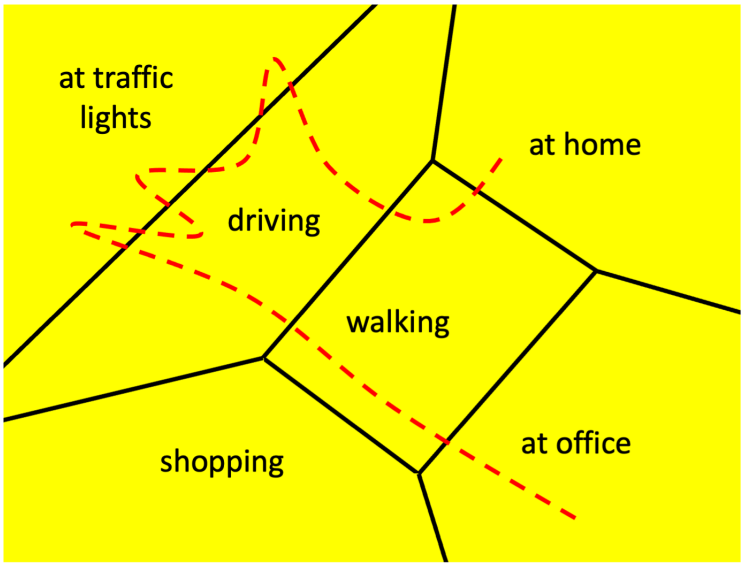


Figure 6.4. Unambiguous sensing

6.5.3 Levels of ambiguity

Figure 6.4 shows the ideal situation where sensing is unambiguous. The overall area represents the set of all possible sensing values. For each state of the real world, sets of unambiguous sensor readings correspond to the state, and each trace of user activity moves ambiguously through a consistent path of states. The dashed line shows an example trace, where Emanuele is at his office one evening, walks to the car, drives, periodically stops at traffic lights, and then walks home.

However, sensing is rarely as perfect or accurate as this. In practice, there are levels of ambiguity as shown in Figure 6.5:

certain This is a state in which the AI is confident about detecting the context. Considering smart parking, we can be fairly certain that Emanuele is driving if there are engine-like vibrations, the GPS shows rapid movement, and there is relatively little body movement. Alternatively, if there is minimal engine noise, slow GPS movement (1–3 km/h), and periodic (approximately 1 Hz) body movement, the AI can safely conclude that Emanuele is walking. Every hint that can help the system stay in or return to this state, is welcome.

uncertain This is when there are some possible unusual behaviors and the system is not completely confident but within the range of possible variation of the previous certain state. The default behavior assumes the system is still within

the previous state. An example would be if Emanuele is walking and stops momentarily to look in a shop window or chat with a friend – for a short while, there is no GPS movement, low body movement, and no engine noise.

incoherence Here the AI is unsure as the sensor readings do not match the typical variation of any usual behavior. An example of this will be if there is both engine noise and periodic leg movement. If these states are transient, they would be filtered out, but if they persist for any length of time, they suggest a state that is totally unknown to the system; maybe Emanuele is dancing in the car seat while waiting in a long road queue. This may simply be recorded internally as an unknown state, but if any critical action is to be taken, this may be a time to warn the user explicitly before it's too late.

For obvious reasons, we informally refer to these as the egg yolk, egg white, and frying pan, respectively.

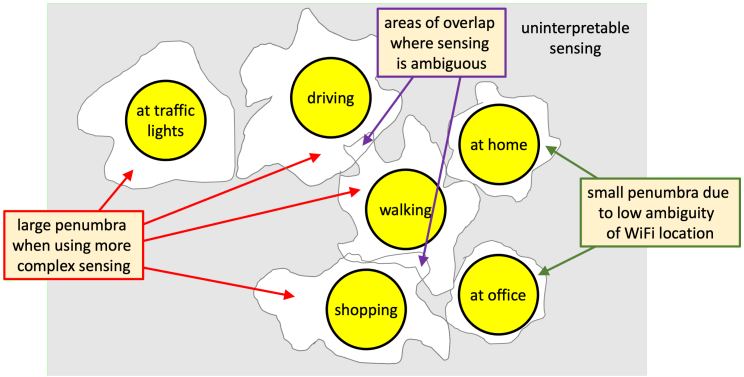


Figure 6.5. Ambiguous and incoherent sensing

Note, we may also have continuous levels of certainty rather than three fixed levels, but using distinct regions can help us think about detection and intervention strategies. Also, we can think of these instantaneously, but they have a temporal nature; for example, their 'stationary in a car' sensor readings that last a long time might be regarded as unusual or incoherent, even if short stationary periods are common.

6.5.4 Using traces and ambiguity for post-hoc detection

We return now to the parking at night example in section 6.1.2 and see how the knowledge of common traces and ambiguity levels can help the AI system to better deal with errors and offer the potential for AI or human repair. Figure 6.6 shows the scenario with the period of uncertainty and the normal variation levels (the egg white region) for the driving and walking states.

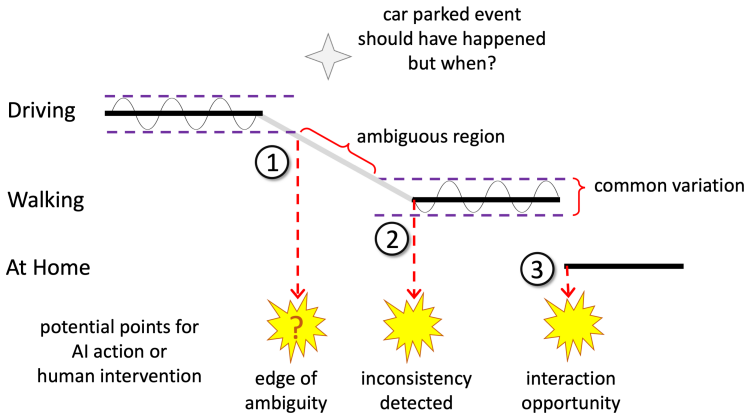


Figure 6.6. Difficult case

In the ideal situation (figure 6.1), the parking event was generated when the state changed from driving to walking, but in this case, there is no clear point of change.

In Figure 6.6, point (1) is the first time the sensor readings fall outside the common variation of driving and, thus, the first point at which the AI system's model of the world is incoherent. AI's information is not enough to generate a parking event, and potentially it is missing to detect a parking location; the consequences of this error are quite important in our example. This could be a point at which the AI system requests an explicit confirmation. However, we normally wish to avoid this in implicit interaction.

Point (2) is the first time the system can inform Emanuele that something has gone wrong and that there is an inconsistency in its past sensing. There has been a transition from driving to walking, but no parking event has been generated. This could again be a chance for subtle and ignorable user interaction for clarification. Alternatively, the AI system could take corrective action itself as suggested in Section 6.5.2, but using the knowledge of normal levels of variation to make estimates more accurate. As noted, up until point (1), it is fairly safe to assume that Emanuele was still driving. Similarly, the time at which Emanuele can be confidently inferred to be walking can be pushed back to the first point the sensors enter its range of variation. The period of uncertainty and hence the geographic inaccuracy has been reduced.

Finally, at point (3), Emanuele is at home. Even though this is somewhat removed from the point of ambiguity, it is the first time he is in a context that makes explicit interaction more acceptable. Of course, he may have forgotten the parking location by this time, but it is better to clarify this at point (3) than when in a hurry in the morning.

6.5.5 Dealing with sensor-layer failure

In a safety-critical situation, we might want to report every low-level sensor failure and perhaps get human confirmation or overrides. Typically we do not want this for most applications. Furthermore, the knowledge of the right level of warning/request for user intervention needs application knowledge. Architecturally there needs to be rich enough API between layers to enable higher levels of application software to decide which lower-level failures matter and how to weigh up the severity of the problem (for the user) with the certainty that there is a problem.

In the case of an incoherent trace, we have a high certainty that a problem has occurred, but it may be some time after the actual important event. In the car parking example, Emanuele might get a prompt from his parking app when it is certain he is walking, or if not before, when he is at home (as sensed by GPS, WiFi signal). When asked at this point, he is more likely to remember than the next morning. Also, repairing at that point might avoid wasting time for his son, who may be in a hurry the next morning.

The ambiguity region is a little more complicated. The ambiguity means we are uncertain of the state, and it is possible that interaction at this point would actually cause an interruption, maybe while Emanuele is close maneuvering while parking. However, this is also the ideal point to request user information as it is closer to the real change point.

6.6 Design implications

Understanding this understanding of AI errors leads to a high-level design strategy. In general, it poses questions to ask, given a potential error or inaccuracy with the system; more detailed heuristics will depend on particular situations.

6.6.1 Helping early detection

As we have seen, early detection makes effective repair more likely. We have two design options:

- help the user – Provide appropriate visualization, audio, or other feedback or status to the user so that they are more likely to detect a problem.
- help the AI – Create mechanisms so that the AI system can detect its errors. This will typically happen after some time when more information becomes available.

Sometimes one or other design options may be easier to implement or more effective. However, note they are not mutually exclusive, and both can be employed.

For example, in the case of the parking app unlocking the car as the user approaches, this could immediately produce a small notification sound. However, if this is missed, the system might still notice if the user has walked away again.

6.6.2 Helping communicate

In the case when detection and repair are performed by different agents, we need to seek appropriate communication options:

- user to the AI – When the user has detected a problem, but the AI needs to correct it, we need to find easy ways to let the system know. Ideally, this can be designed to make use of the fact that the detection will be due to a recent notification or status change so that it can be highly contextual and not require extensive interaction.
- AI to the user – When the AI has detected its own error but needs to inform a user, this needs to balance the need to inform the user as soon as possible to enable repair before problems get worse while being subtle enough to avoid distracting the user, or being tedious.

6.6.3 System Reliability - when accuracy may become a problem

Another aspect that designers should be concerned with is the level of reliability the user expects from the AI. This expectation is usually accumulated by the active experience of the system, as after a certain period of use, the user will begin to accumulate knowledge of the system, experiencing situations in which the AI will work more or less accurately. Of course, one of the goals is for the user to trust the AI - and we certainly want to avoid them discovering situations of unreliability in critical contexts. In our car example, it would be better for the user not to discover that the car unlocks itself as they walk past it, only when the thief has already stolen it. A possible design solution would be an AI that advertises its real level of accuracy, to avoid deluding the user, but manifesting a certain level of uncertainty can be highly detrimental to building user trust, leading them to abandon the system quickly. In short, the designer has to deal with the honest reliability/user trust trade-off. In any case, we can distinguish two situations:

6.6.3.1 Low reliability

When an AI has relatively low reliability, the users expect this unreliability.

Here the rules of appropriate intelligence come in an ‘active’ way. The unreliable AI should not actively do something hard to repair – for example, interrupting

important work or rewriting text without asking. However, the user monitors things, so situation errors that go undetected are rare – the human ensures detection. In the car door example, if the system opens the door 70% of the time, but the other 30% fails to detect, the driver may gently try the door handle before pulling hard.

6.6.3.2 High reliability

Sometimes the worst problems occur if the system is very reliable, perhaps correct 99% of the time, or 99.9% of the time, but still occasionally gets things wrong.

In these cases, the human comes to expect the AI to behave correctly and therefore is unprepared for failure. For example, if the AI detects paid parking areas with 99.9% accuracy and pays automatically, the driver will get used to not checking if the payment occurred and may be fined when the AI fails.

6.6.3.3 Responsibility for detection

In the low-reliability situation, the human effectively takes responsibility for monitoring the system's behavior (without necessarily even being aware that is what they are doing).

In a high-reliability situation, the human will be unlikely to notice, and therefore we need to design strategies that help alert the user to unexpected situations.

For example, suppose the user is walking very quickly along the pavement towards their car and don't appear to be slowing; the parking detection system might assume they will not enter the car and so not unlock it (precautionary principle for avoiding theft). However, if the user stops suddenly (perhaps was just in a hurry or almost missed noticing which of the line of cars was theirs) and reaches for the car door, we will likely get the broken finger-nail situation! To avoid this, the car app could detect that while its action is NOT to unlock, it is a potentially ambiguous situation and so do something to warn the user, perhaps vibrate their phone (a sort of 'hello' from the car as they walk past) or make the car handle glow red.

6.7 Discussion

This chapter has highlighted the importance of considering errors and failures in human-AI systems, and the challenges involved in detecting, communicating, and repairing them. A central point is that AI systems are not infallible: they can fail through both commission and omission, and these failures may have significant consequences for users, especially when the system acts proactively or operates on the basis of inferred rather than explicit input. Moreover, the meaning of an error is

not fixed. Users' perceptions can change over time, and repeated exposure to errors may reshape both expectations and behavior.

To prevent local errors from becoming larger failures, it is essential to support early detection and clear repair mechanisms. This requires effective collaboration between human and AI, as well as communication and feedback structures that allow users to notice when something may be wrong, understand what has happened, and determine how to recover. In this sense, error handling is not only a technical problem, but also an interaction-design problem.

Research on human-AI interaction suggests that detecting errors in AI systems is inherently difficult, and that no single solution is sufficient across all contexts. For this reason, systems benefit from a range of complementary strategies, including post-hoc estimation, the use of traces and ambiguity, and the combination of multiple sensing or evidence layers. These strategies do not eliminate fallibility, but they can make failures more manageable and reduce the risk that incorrect inferences silently propagate into user-facing breakdowns.

Designers of human-AI systems must therefore account for error detection and repair from the outset, building systems that are resilient to failure and able to support changing user expectations and practices over time. In mobility contexts, this may involve combining multiple sensor sources or designing better recovery mechanisms when inferred context is uncertain. In other AI-based systems, including LLM-driven assistants, the same principle applies even when the underlying signals are different: what matters is whether the system can make uncertainty visible, support correction, and avoid presenting unreliable output as settled fact. This broader perspective is one reason why the discussion of fallible automation remains relevant beyond the sensing-based examples used in this chapter.

Finally, it is essential to understand how errors and failures affect user trust and confidence. Once trust is damaged, rebuilding it can be difficult, especially when users do not understand why the system failed or how to regain control. Studying how users perceive AI errors can therefore inform not only repair strategies, but also the design of systems that remain intelligible, accountable, and worthy of calibrated trust even when automation is imperfect.

Chapter 7

Enhancing Interface Design with AI

7.1 Introduction

In the rapidly evolving field of Human-Computer Interaction (HCI), integrating Artificial Intelligence (AI) offers groundbreaking opportunities for enhancing user experience and interface design. This paper presents CWGPT, a conversational AI tool based on ChatGPT-4, designed to assist in the usability evaluation of web interfaces, providing users with an expert usability evaluation inspired by the well-known Cognitive Walkthrough method.

We address two research questions:

- **RQ1** Can ChatGPT-4 be effectively leveraged to build a tool to evaluate web interface usability?
- **RQ2** If so, is this tool beneficial for novices in interface design?

Regarding RQ1, we executed Cognitive Walkthroughs (CWs) on ten selected web applications and compared the outcomes with the analyses conducted by CWGPT. The web apps were selected from the exam projects submitted by the students of a Web and Software Architecture course, who were requested to develop a social web app for sharing photos. The results indicated the potential of ChatGPT-4 in effectively conducting CWs. To explore the potential benefits of CWGPT for novices approaching interface design, we conducted an exploratory user study involving five author-students. Acknowledging the involvement of only five users, we recognize that our research does not yield significant quantitative results but rather qualitative insights, and we argue it serves as a foundation for further investigation into the promising role of AI-based tools in interface design and validation processes.

7.2 Related Works

Our work is inspired by Cognitive Walkthrough, a review process in which a group of experts evaluate a design aspect in the context of one or more specific tasks. In general, the input to a walkthrough session includes an interface (whether a description, working prototype, or series of screenshots), a task scenario, assumptions on the user base and context of use, and a sequence of actions that should be performed to complete the designated task [166]. As the process requires experts and detailed reviews, it is time-consuming and expensive. To the best of our knowledge, no other works currently exploit Large Language Models (LLM) to perform usability evaluation in the spirit of cognitive walkthroughs.

The use of LLM to imitate human interactions with interfaces is an emerging research field. In particular, in automatic Graphical User Interface (GUI) testing, researchers are experimenting by using LLM to receive realistic feedback from application testing. They let the model use the application by annotating its comments and the line of reasoning it follows [113, 220]. The use of LLM in UI layout generation is noteworthy, where the model aims to help identify target users and their needs and construct basic interfaces for the tasks [28, 19]. In particular, York [221] presented a work that experiments with the use of LLM to help novices and student designers develop ideas, designs, and code to begin their projects. Moreover, Schimdt [181] discussed the transformative impact of Generative Artificial Intelligence (GenAI) and Large Language Models (LLMs), emphasizing their capacity to expedite multiple stages within the development lifecycle of interactive systems.

Regarding the use of ChatGPT, recent works adopted it to simulate how a user would perform a task given the application [113, 215, 220]. GPT agents [147] have also been used to receive feedback on different aspects of usability [220]. These works test Android mobile applications, as they do not use the actual graphical interface as input but rather a natural language description of the application based on Android's hierarchy files.

More in detail, Liu et Al. [113] used ChatGPT-3 to discover tasks automatically, performing activity coverage tests and bug detection, finally showing significant improvements in these three tasks compared to the baseline models [111, 110]. Wen et Al. [215] used GPT to test the completion rate of the authors' designed tasks. The work highlights the limitations of using natural language descriptors of GUIs, as GPT failed to complete tasks involving unnamed elements, such as checkmark buttons and search boxes. Finally, Yoon et Al. [220] presented a GUI testing tool that uses different GPT agents to perform different tasks: planner, actor, observer, and reflector.

Ultimately, these works are limited by the natural language descriptor of the GUI, which, depending on the one employed, makes comparisons between tools

unfair, as some elements might not be described satisfactorily. On the other hand, researchers responded positively to the comments left by GPTs, marking them as helpful in understanding real users' problems when navigating through interfaces. However, the focus of their work is on the capacity of LLM to perform realistic tests.

7.3 CWGPT

Leveraging the builder feature of ChatGPT-4, we developed CWGPT, a specialized conversational agent for facilitating CW-inspired evaluations in web interface design. This agent aims to integrate the analytical capabilities of AI with usability evaluation methodologies. CWGPT is published on *https://chat.openai.com/g/g-Hl2FKGIN3-cwgpt*ChatGPT and is available through ChatGPT-4. We decided to focus on web interfaces because of their widespread usage, ease of capture through screenshots, and the challenges they pose for novice designers, making them ideal for usability evaluation. Additionally, our access to students from the Web and Software Architecture course provided a readily available source of real-world web interfaces for testing, perfectly aligning with our target user base.

It is important to note that our tool does not autonomously navigate an interface by clicking connectors between views. Instead, it engages users interactively, requesting them to perform actions (e.g., clicking a button) to proceed. This approach enables the tool to explore the interface in an informed manner while actively involving users in the evaluation process.

For the development of CWGPT, we utilized OpenAI's GPT builder, a specialized tool crafted for constructing customized GPT-based chat interfaces. This builder functions as a conversational agent itself, enabling customization through natural language interactions. Throughout our development process, we iteratively refined our prompts to achieve the desired behavior for CWGPT. Our experience highlighted the importance of precise prompt engineering, a skill underscored in current literature [14]. Ultimately, we found that linking each new instruction to the preceding one in the prompt yielded the most effective solution to meet our diverse range of requirements.

7.3.1 CWGPT evaluation process

CWGPT initiates each session by requesting a specific task for evaluation from the users, along with a screenshot of as starting interface. Relying on ChatGPT-4's advanced language processing abilities, CWGPT is not confined to predefined prompts, enabling it to process various user inputs, such as requests for assistance, clarification, or supplementary information.

In the first interactive exploration phase, CWGPT hypothesizes the most probable

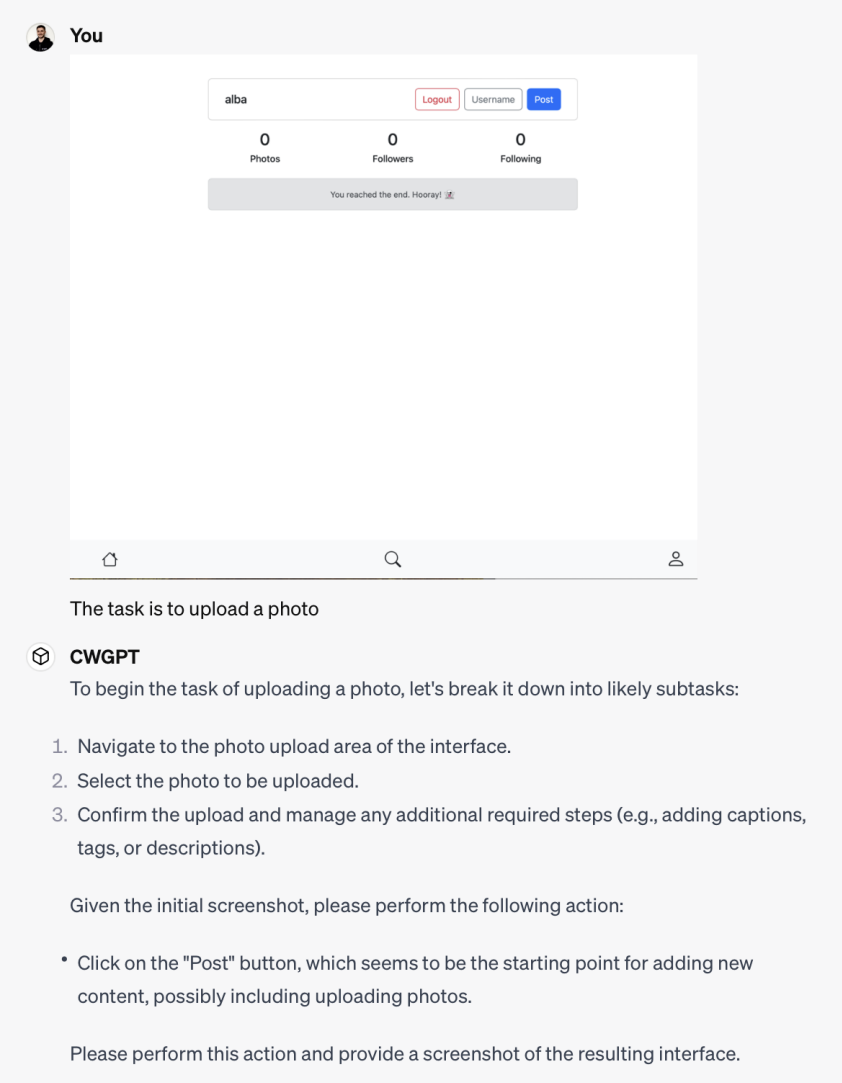


Figure 7.1. Example of interaction with CWGPT.

subtasks and corresponding actions that lead to the completion of the target task (Fig. 7.1). Users are prompted to execute specific actions for each identified subtask and then provide a screenshot of the interface post-action.

After the execution of all the subtasks, CWGPT starts the evaluation phase. Applying the core principles of the CW method as outlined by Wharton et al. [216], it assesses each action in terms of its effectiveness in helping the user achieve their goals, the visibility and accessibility of the action, the user's ability to associate the action with the desired outcome, and recognition of progress towards task completion.

Upon concluding both phases, CWGPT assembles a comprehensive evaluation of the interface, including insights into the usability strengths and weaknesses of the interface, and practical with suggestions for enhancements.

7.4 Exploratory Study

7.4.1 Comparison of human and CWGPT evaluations (RQ1)

7.4.1.1 Methodology

We selected ten web applications displaying noticeable usability issues from projects developed during a Web and Software Architecture course. Notably, this course primarily focuses on development, and interface design is outside the scope of the course. All the selected interfaces have been included as supplementary material. The chosen task for evaluation across all interfaces was the standard action of "Upload a photo". Our objectives in this phase were twofold: firstly, to assess the capability of CWGPT in determining the sequence of actions required for task completion, and secondly, to evaluate whether CWGPT identified similar, fewer, or additional usability issues compared to human-conducted CWs.

We ourselves conducted standard CWs for each interface. For each action in completing the task, we answered the traditional CW's four questions, with one of the four answers: 1) Yes, 2) Likely yes, 3) Likely no, 4) No, and with brief comments for non-affirmative answers. In the following, we refer to this phase as the experts- or human-led CWs. Concurrently, we employed CWGPT to evaluate the same set of interfaces. The style of the obtained CWGPT answers is similar, although more detailed and verbose.

7.4.1.2 Results

CWGPT consistently demonstrated its capability to determine task sequences independently, highlighting its effectiveness in usability evaluation. In one case, it considered the task completed while missing the last action of the sequence ("Click on the *close* button of the dialog"), terminating the session early.

Regarding the detection of usability issues, to quantify the agreement between the experts' CWs and the results obtained by CWGPT we collectively compared the answers to the CWs' questions.

The ten web interfaces analyzed required a total of 32 actions to perform the task "Upload a photo" (average of 3.2 actions per interface). For each action, a CW requires answering 4 questions, totaling 128 questions. We compared the answers from the human experts with those generated by CWGPT. If both the answers are either "Yes" or "Likely yes", or when they are both "No" or "Likely no", we count it as an agreement. Alternatively, we count it as a disagreement. Out of the 128 answers, as reported in Table 7.1, 116 showed agreement, 8 exhibited disagreement, and 4 could not be compared because, on one occasion, CWGPT did not recognize an action and thus did not answer the questions.

Total actions	32
Total questions	128
Average actions to complete the task	3.2
Comparable questions	124
Total agreement	116 (93.55%)
Total disagreement	8 (6.45%)

Table 7.1. Overall agreements between experts and CWGPT answers

In summary, the results of this experiment phase indicate a clear trend of agreement between CWGPT and human experts. Regarding the quality of evaluation assessments, CWGPT consistently provided more detailed evaluations than the succinct "Yes" or "Likely" responses given by the experts. Additionally, each assessment conducted by CWGPT concluded with a comprehensive summary and practical suggestions for enhancing the interface. Notably, in all 10 instances, we agreed with CWGPT's summaries and suggestions, further highlighting the value of its assessments. As a negative note, we report that, among the reasons for disagreement, CWGPT did not pay attention to issues that were evident to human evaluators, or categorized them as "minor improvements". This tendency was particularly noticeable in interfaces heavily reliant on visual design, such as color schemes, layout, or visual hierarchy.

7.4.2 User study of CWGPT (RQ2)

7.4.2.1 Methodology

In the second phase of our study, we assessed the practical value and acceptance of CWGPT in aiding designers with web interface usability evaluations. Five student authors of selected interfaces from the first phase participated in the experiment. Interface selection was based on author availability rather than issue severity. Each student, aged 20-25, evaluated the "Upload a photo" task using CWGPT on their authored interface. Among them, four were male and one was female. Although four out of 5 had some HCI background, they required refreshers on CW specifics. All participants were familiar with ChatGPT.

Each test session, led by an interviewer and an observer, followed a structured procedure:

- **Initial Interview:** We asked the student their age, HCI background, and their familiarity with CWs and ChatGPT.
- **Task Execution:** The student independently performed the designated task, noting any observed mistakes or issues with the interface.

- **CWGPT Think Aloud:** The student utilized CWGPT to conduct a Cognitive Walkthrough (CW) of their interface on the task "Upload a photo". Introduced briefly to CWGPT, they were encouraged to use it freely, similar to any other ChatGPT-based chat, possibly without any intervention by the interviewer. This session was conducted with the *think-aloud method*.

- **Interface Changes Interview:** Upon completion, the student was asked if they felt their interface needed improvements and agreed with the CWGPT evaluation. If so, the student was asked to describe which interface elements needed improvement.

- **Feedback and SUS:** To conclude, the student was invited to share perspectives on the utility of CWGPT and its overall experience. Subsequently, they filled out a standard System Usability Scale (SUS) questionnaire [45].

7.4.2.2 Results

In the initial phase of the test sessions (*Task Execution*), participants generally exhibited a positive outlook, believing their interfaces did not exhibit significant problems, though they acknowledged general room for improvement. Two participants shared that the task was relatively straightforward and did not reveal any immediate, critical issues.

Then, participants were asked to use CWGPT to perform a CW evaluation on their interfaces, focusing on the "Upload a photo" task (*CWGPT Think Aloud).* While they managed the usage of CWGPT with relative ease, there were some notable observations.

Participants initially displayed some uncertainty in initiating the interaction with CWGPT. Two individuals needed help with how to commence and were suggested by the interviewer to request CWGPT's assistance directly. In contrast, the remaining participants attempted to initiate the process by sharing only part of the required prompt, either sending only the screenshot or solely the task. However, they were promptly corrected by CWGPT, facilitating their understanding of the necessary steps.

Of particular interest, one participant with no prior background in Human-Computer Interaction (HCI) encountered challenges in grasping the essence of the evaluation process, initially struggling with the concept of a "task". Nevertheless, this participant persisted in the conversation led by CWGPT and completed the session. Interestingly, upon reviewing the generated CW and overall evaluation, this student better understood the tool's purpose and utility.

During interactions, CWGPT sometimes showed uncertainty about the appropriate action to take, resulting in unnecessary steps. Participants were surprised but

realized these variations were due to unclear design elements needing improvement.

In one specific case, CWGPT instructed the participant to perform an action related to an element that did not exist within the interface. This prompted uncertainty from the student regarding whether they could correct the instruction. However, it was noted that the student was not entirely surprised by this occurrence, understanding that AI can occasionally generate unexpected outputs or "hallucinations". After completing the CWGPT session (*Interface Changes Interview*), all students unanimously recognized the necessity for modifications and improvements to their interfaces. They comprehensively grasped the most pertinent usability issues identified during the evaluation process. Notably, one student expressed a minor disagreement with a suggestion received, deeming it "excessive" and questioning whether a human user would perceive it as a genuine problem. This suggestion related to adding labels to icons was initially ambiguous for the tool but validated as standard by the experts. In another instance, a student was so impressed with the tool's capabilities that they inquired about its capacity to analyze interfaces of different types and utilize it for a personal project.

Finally, participants were invited to share their general feedback and insights regarding their experience with CWGPT (*Feedback and System Usability Scale (SUS)*). Encouragingly, all students expressed satisfaction and contentment with the results achieved through CWGPT. They uniformly stated that the tool was easy to use and adept at identifying compelling issues. Notably, no significant negative feedback emerged during this phase, underlining the tool's overall positive reception.

The results trends of the SUS questionnaire, employed as a standard benchmark, were predominantly positive, particularly about questions assessing the system's complexity and usability. Due to the limited number of participants, we do not present the standard formulas for interpreting the results. The complete questionnaire and responses have been included as supplementary material.

7.5 Discussion

Our exploratory study highlights promising aspects of CWGPT's usability in web interface evaluations. Comparing CWs by experts with CWGPT evaluations, we demonstrated its ability to discern task completion sequences and explore interfaces through screenshots, compensating for its inability to navigate directly. Usability assessments involving students further emphasized its user-friendliness and effectiveness in identifying issues, reinforced by positive feedback. However, several limitations require a discussion.

One concern is CWGPT's effectiveness in handling visual and design aspects, like color schemes and layout arrangement. For example, it did not recognize small

buttons in remote corners as issues, unlike human evaluators. Moreover, providing CWGPT with entire page screenshots, including content below the fold, might lead to erroneous identifications of concealed controls, hindering its ability to detect usability issues.

Another concern is dynamic interfaces, which we did not directly assess but could pose challenges for CWGPT, especially beyond static screenshots. Moreover, the interfaces we tested were straightforward, leaving room for more thorough evaluations with more complex yet static interfaces.

Finally, like any AI-based tool, CWGPT may encounter errors due to AI unpredictability, such as "hallucinations" or conversation deviations. Even prompts crafted differently than expected might yield varying responses, potentially deviating from expected conversation flows. Therefore, it is crucial to acknowledge the inherent susceptibility to errors in AI [35] and adjust expectations accordingly when using the tool.

In summary, these limitations are important factors to consider when using CWGPT in usability evaluations. While our study provides promising insights, future research should explore these aspects in different settings to understand their implications comprehensively. CWGPT might replace experts in some usability testing aspects, especially where systematic evaluation using CW principles is required and expert availability is limited. Ultimately, the aim is to enhance usability awareness and practices among novices.

The proposed conversation agent offers a user-friendly approach to usability assessment, making it particularly advantageous for students and novice designers seeking to enhance their interface design skills. With an exploratory study, we addressed the validity of CWGPT evaluations (RQ1) and the practical utility and acceptance of CWGPT in assisting designers (RQ2).

Chapter 8

Large Language Models Adoption in Need Finding

8.1 Introduction

In the realm of User-Centered Design (UCD), the quest for innovative tools and methodologies to enhance the design process is ongoing. The UCD approach, rooted in placing the user at the forefront of design considerations, requires a deep understanding of user needs, preferences, and contexts. As such, the design community continually explores new avenues to enrich the UCD process, making it more efficient, inclusive, and effective. One of the most promising frontiers in this exploration is integrating artificial intelligence (AI), mainly through the use of Large Language Models (LLMs).

LLMs, such as the popular GPT-4, have demonstrated remarkable capabilities in generating human-like text based on vast datasets. Their potential to simulate aspects of human interaction and understanding presents a unique opportunity for augmenting various phases of the UCD process. This includes tasks such as defining the main objectives of a design project, identifying key stakeholders, creating detailed personas, and even simulating focus group discussions [182]. However, applying LLMs within the UCD framework is not without challenges, especially when it comes to accurately capturing the full spectrum of human emotions, cultural contexts, and nuanced preferences.

This exploratory study seeks to explore the viability and effectiveness of LLMs as assisting tools in the early stages of the UCD process, specifically focusing on the need-finding phase. We advocate that even though LLMs are suitable to substitute the designer during the need-finding process, if adapted to assist them rather than replace them the outcomes are more adherent to reality and hence preferable. By providing example prompts and discussing the results of three practical case studies

of mobile development (a university app, a fintech app, and an earthquake-warning app) - we provide an exploratory study on how to be assisted in the need-finding phase by LLMs. Our focus is not to advocate for replacing human-centric processes with AI but rather to understand how LLMs can complement existing methodologies, where they excel, and where they fall short.

8.2 Related Works

The integration of artificial intelligence, particularly Large Language Models (LLMs) like ChatGPT, into Human-Computer Interaction (HCI) processes is gaining attention, spanning early design stages to evaluation [181].

Among others, the recent work of Schmidt et al. [182] highlights the increasing potential of using LLMs in the Human-Centered Design (HCD) process, especially for enhancing design iteration efficiency. This area of exploration is strictly related to our investigation into the utility of ChatGPT during the initial phases of User-Centered Design (UCD).

Further research, such as the studies by Ekvall et al. [75], York [221], and Atlas [14], delves into ChatGPT's application in UX design and education. These studies provide insights into how ChatGPT can aid ideation, prototyping, and feedback, complementing the design process and education.

Tabone et al. [197] show how ChatGPT can effectively analyze text data, indicating its potential to enhance HCI research methods through detailed summaries and analyses.

In terms of evaluation, Duan et al. [73], and Bisante et al. [29] explore LLMs' role in automating feedback for UI design and facilitating evaluations of web interfaces, respectively. Their work points to ways that LLMs can streamline the evaluation process.

However, despite these positive developments, integrating LLMs into design processes still brings challenges, including ensuring AI-generated output's relevance and accuracy and ethical considerations [95]. The impact of AI on user interactions and interface management also raises important questions [35].

This growing interest in LLMs within HCI underscores the technology's potential to impact design processes significantly while also highlighting areas that need careful consideration. Our study contributes to this conversation by examining how LLMs can support UCD's early stages, aiming to shed light on both the advantages and challenges of AI in design.

8.3 Methodology

In this work, we investigate the effectiveness of LLMs as assistive tools in the early stages of the UCD process, with a specific focus on need-finding. Our guiding question is the following: can the outcomes produced by direct prompting, where AI partially substitutes the designer, be improved through an LLM-assisted approach that instead supports and extends the designer's work?

LLMs can be directly prompted to delineate user needs for a given design idea, for example *"What are the user needs for a fintech mobile app?"*, potentially bypassing traditional UCD phases, especially those involving hard-to-reach users. As an alternative, we explore whether LLMs can be used to assist more conventional need-finding activities rather than replace them.

Based on our experience in the field, we first identified a simplified need-finding process and then integrated it with LLM-based assistance. The resulting LLM-assisted approach is reported below:

1. **Definition of Target User Profile:** We prompt the LLM with general application details to outline the likely target user profile.
2. **Preliminary Definition of User Needs:** The LLM is asked to draft an initial set of user needs.
3. **Persona Creation:** The LLM generates three or more detailed personas intended to reflect plausible variation within the target user group.
4. **Simulated Interviews:** The LLM is used to simulate interviews with the generated personas. The questions can be written by the designers, by the model, or through a hybrid process.
5. **Refinement of User Needs:** The LLM revises the initial list of needs based on the simulated interview material.

This workflow should be interpreted as an exploratory design aid rather than as a substitute for empirical user research. In particular, steps 3–5 introduce an important methodological limitation: the same model family is used both to generate the personas and to simulate their interview responses. As a consequence, the process risks producing a synthetic "echo chamber," in which later outputs reinforce assumptions already introduced in earlier prompts rather than surfacing genuinely new perspectives. Relatedly, generated personas may reproduce stereotypical or overly generic representations of users, especially when the design domain involves social, cultural, or demographic variation that is weakly specified in the prompt. For this reason, we treat the resulting personas, interviews, and refined needs as

provisional artifacts for ideation and reflection, not as a replacement for real users or as validated evidence of actual needs.

We deployed ChatGPT across three design domains to compare direct prompts with the assisted workflow. We used ChatGPT version 4, updated in December 2023.

Experiments were divided into two distinct chat sessions for each design domain in order to avoid context overlap. In one session, ChatGPT was directly prompted to list user needs from a specified design concept. In the other, we followed the assisted need-finding steps. Both sessions were initiated with the same starting prompt, *"I want to design a mobile application...[design idea]"*, to ensure consistency. The adopted design ideas were:

- *"...to support university students in managing university activities (e.g., book exams, find classrooms)"*
- *"...to support bank customers in performing mobile banking operations (e.g., send/receive payments, invest, track expenses)"*
- *"...to receive warnings about upcoming earthquakes"*

These domains were selected based on our prior experience in developing similar applications, which allowed us to inspect the outputs with respect to their depth, plausibility, applicability to realistic contexts, and potential to stimulate design reflection. At the same time, this choice also introduces a limitation: the evaluation relies on researcher judgment rather than on external ground truth or direct validation with representative users. Accordingly, the goal of the study is not to claim that the LLM-generated outputs are accurate representations of real users, but rather to examine whether they can function as useful intermediate artifacts in early-stage design.

8.4 An example of Direct prompts vs. LLM Assisted Need-finding

The same prompts were used across the three design domains. In this section, we report and comment on the experiment conducted with the first design idea.

In one session, we directly prompted ChatGPT with the following request:

"I want to design a mobile application to support university students in managing university activities (e.g., book exams, find classrooms). Define the user needs for such an application."

ChatGPT replied with a list of 10 user needs for a hypothetical university management app, each accompanied by a corresponding feature suggestion. The list was broad and relevant, offering a plausible starting point for the need-finding phase. Although there was some overlap between items, for example "finding classroom locations" appearing both under "academic schedule management" and "classroom locator," the response still provided a coherent first approximation of the design space.

We then began a new session and applied the LLM-assisted need-finding workflow. Below we report the prompts and a summary of the corresponding outputs.

1. **Prompt**: *"I want to design a mobile application to support university students in managing university activities (e.g., book exams, find classrooms). Define the user profile target for such an application."*

   ChatGPT replied with a structured five-point list. It described demographic characteristics, a set of expected needs, technological preferences, behavioral traits, and some additional considerations such as accessibility and privacy. We judged the profile plausible for the target context. At the same time, the answer already anticipated several of the needs later produced in subsequent prompts, which suggests that later stages of the workflow partly elaborate assumptions already introduced at this first step.

2. **Prompt**: *"Define expected user needs for such an application."*

   The response included seven main points, each containing two or three more specific needs, for a total of sixteen needs. A direct numerical comparison with the previous direct-prompt session is not straightforward, because the granularity of the answers differed. Some needs that had previously been grouped together were now listed separately, while other new needs appeared, such as financial and administrative support or campus news and events. Overall, this step broadened the design space, but it remained closely aligned with the assumptions already introduced in the target-user profile.

3. **Prompt**: *"Create three personas."*

   ChatGPT generated three personas, each described in terms of name, age, academic year, major, level of technological familiarity, and primary device. It also specified goals, challenges, and app-related needs for each persona. The resulting personas were coherent and easy to read, but they were also strongly consistent with the needs already listed in the previous step. This is useful from a workflow perspective, because it gives designers concrete artifacts to reason with, but it also highlights a methodological risk: the personas do not introduce an independent source of evidence, and instead largely restate and reorganize what the model had already produced.

4. **Prompt**: *"Simulate an interview with Persona n"*

   For each of the three personas, ChatGPT generated a simulated interview following a similar structure. The model used the previously generated persona description to personalize both the questions and the answers. Each interview included opening and closing turns, questions about the persona's challenges, and follow-up prompts connecting those challenges to potential application features. This produced readable and internally consistent interview transcripts. However, because the same LLM had already generated both the personas and the underlying needs, the interviews should not be interpreted as independent validation. Rather, they function as a reflective device that can help designers inspect the implications of earlier assumptions. In other words, this step is best understood as structured elaboration, not as evidence about how real students would actually respond.

5. **Prompt**: *"Based on the interview transcripts, how would you integrate or modify the expected user needs? Provide a complete list of user needs indicating whenever there is a variation from the original list."*

   In the final response, ChatGPT reiterated the seven main points and sixteen needs identified earlier, indicating for each whether it remained unchanged or required enhancement after the simulated interviews. Four additional needs were introduced. However, many of the reported "enhancements" were minor reformulations rather than substantial revisions. For example, the need "Exam Booking System," originally described as *"A feature to book exams, view exam schedules, and receive reminders as exam dates approach"*, was updated to *"Enhanced to include not just booking and tracking exams but also integrating study reminders based on exam dates"*, which does not fundamentally alter the original requirement. This again suggests that the assisted workflow tends to refine and expand previous outputs more than it surfaces genuinely novel or contradictory insights.

This example shows that the LLM-assisted workflow can generate a richer chain of intermediate artifacts than a single direct prompt, including target profiles, personas, interviews, and revised needs. These artifacts may help designers think through a problem more systematically. At the same time, the example also illustrates the main methodological caution of the approach: because each step builds on outputs produced by the same model, the process can amplify internal consistency without necessarily increasing real-world validity.

8.5 Engineering

8.5.1 Integrating Different Engineering Approaches

Integrating Large Language Models (LLMs) into user-centered design (UCD) processes presents a unique challenge: merging diverse engineering methodologies, such as traditional software engineering (e.g., waterfall, agile), with AI-specific approaches like iterative model tuning and data-driven development. We employ several strategies to effectively reconcile these processes to enhance system performance and user experience.

In AI systems, especially LLMs, the iterative feedback loops are crucial [31]. These systems thrive on continuous feedback to improve. By incorporating AI tuning within the Agile-UCD process, we facilitate an environment where the AI model is not static but evolves through iterative development cycles. This method retains the flexibility of agile development and introduces the structured rigor of AI model training. This synergy ensures the AI components are finely tuned to meet user needs and system performance metrics.

The architecture of these systems is often decoupled to allow for modular updates and integration. For instance, in a mobile app development scenario where LLMs assist with user need identification, the AI module can be developed and updated independently from the core application. This decoupling allows for smoother integration of LLM functionalities and facilitates more accessible updates and maintenance without disrupting the system. However, updating the model may also impact the overall interaction.

8.5.2 Addressing Non-Deterministic Aspects of AI

Non-determinism in AI, particularly in LLMs, poses significant challenges for usability, utility, and reliability in UCD processes, as these systems can produce different outputs even when provided with the same input. Addressing this unpredictability is crucial for developing consistent and reliable user-centered designs.

Real-world scenario testing is a primary strategy for mitigating the effects of non-determinism. By testing LLM-generated outputs, such as user personas, against actual user profiles in mobile app development, designers can gain insights into the practical implications of AI unpredictability on the design process. These insights help refine the AI models to better align with user expectations and real-world usability.

Version control and traceability are essential for managing the non-deterministic nature of LLMs. By meticulously tracking prompt versions and the changes they undergo, designers can gather valuable insights into how different prompts influence

LLM behavior. This practice is essential in Agile-UCD processes, where understanding the consistency and reliability of AI-generated outputs is crucial for making informed design decisions. For example, versioning different iterations of an LLM prompt allows teams to compare outcomes, helping to determine if newer prompts generate more accurate or realistic results when measured against real-world data. This systematic approach enhances the quality of AI outputs and supports continuous improvement in the design process.

8.5.3 Ensuring Humans Remain in the Loop

A critical challenge when integrating Large Language Models (LLMs) into user-centered design (UCD) processes is ensuring that humans remain actively involved in decision-making. Although LLMs can automate certain aspects of the design process, there is always the risk that these models generate erroneous or inappropriate content. It is, therefore, essential to implement mechanisms that preserve human oversight and validation throughout the design lifecycle.

Here are key strategies to ensure that humans remain in control:

- **Human-in-the-Loop (HITL) Systems**: One of the most effective ways to maintain human oversight is through HITL mechanisms. In this approach, LLM-generated content, such as user personas or design suggestions, must be validated by human designers at critical junctures. We advise implementing this via checkpoints where humans review, approve, or modify AI outputs, ensuring no fully automated decision is made without human validation.
- **Interactive AI Tools**: LLMs should function as assistants to designers rather than replacements. While LLMs can offer design suggestions or generate insights, the final decision-making authority rests with the human designer. For example, when evaluating user personas, the designer determines which personas are realistic, should be considered, and should be discarded. This approach balances AI-driven efficiency with human judgment, ensuring that the designer makes critical and creative decisions.
- **Transparency and Explainability**: AI models should provide transparent reasoning behind their outputs. For instance, if an LLM generates a design suggestion or user persona, it should explain how it reached that conclusion. This transparency enables designers to understand the AI's rationale and intervene when needed, avoiding over-reliance on potentially flawed AI outputs.
- **Fallback Mechanisms**: When an LLM generates outputs that are flagged as incorrect, unreliable, or potentially harmful, fallback mechanisms should be employed to maintain design quality. The system can prompt the designer to manually refine the output or revert to a more straightforward,

human-controlled process. This ensures that AI-generated content does not compromise the integrity of the design. For instance, when dealing with sensitive data, the system could automatically alert users not to input sensitive information or instruct the LLM to exclude such data from its output. This proactive approach safeguards the user and the design process from unintended risks.

By implementing these strategies, human designers can retain control over the Agile-UCD process while benefiting from the efficiency and insights provided by LLMs.

8.5.4 Guaranteeing Positive Aspects for Human Users

A core objective of user-centered design is to ensure that systems provide positive emotional experiences for users. When integrating Large Language Models (LLMs) into the design process, it becomes essential to evaluate how AI-generated outputs affect these emotional experiences by enhancing or detracting from them. Ensuring that AI contributes positively to the user's emotional engagement requires thoughtful implementation and careful consideration of personalization and user control.

One way LLMs can enhance a system's hedonic aspects is through personalized experiences. AI-driven systems can create more engaging and emotionally satisfying interactions by tailoring outputs to align with individual user preferences. For example, as discussed in the case studies, LLM-generated personas and user needs can be personalized based on demographic information or direct user feedback. This allows the system to adapt to each user's unique characteristics, making the interaction more meaningful and enjoyable. Personalization improves the system's utility and deepens the emotional connection users feel toward the design, fostering a sense of being understood and catered to.

However, integrating non-deterministic AI systems, like LLMs, can frustrate users when outputs are inconsistent or inaccurate. To reduce this risk and ensure a positive user experience, it is essential to maintain transparency about how the AI operates. Users should be made aware of the AI's decision-making processes, mainly when results are unexpected or unclear. Allowing users to adjust or refine AI-generated content is another key strategy to mitigate frustration. For instance, allowing users to modify or clarify LLM-generated personas or design suggestions gives them more control over the interaction. This increased user agency ensures the system better aligns with their expectations, reducing frustration and fostering a more seamless and enjoyable experience.

8.5.5 Assessing Risk vs. Value in Specific Domains

Incorporating Large Language Models into user-centered design processes necessitates carefully evaluating risks and benefits, which vary considerably across different domains. Our study's need-finding flow—which leverages LLMs to assist in defining user needs and generating user personas—highlights the importance of this assessment.

In high-stakes areas like healthcare and finance, where errors can have severe consequences, rigorous risk assessments and strict validation protocols are essential. For instance, in the earthquake-warning app case study 8.3, utilizing LLMs to identify user needs and generate personas carries significant risks if the outputs need to be more accurate or complete. Misidentifying user needs in such a critical application could lead to features that fail to provide timely warnings, potentially endangering lives. Therefore, integrating LLMs in these domains requires stringent human oversight in the need-finding flow. Designers must meticulously review AI-generated user needs and personas to ensure they are accurate, relevant, and comply with safety and regulatory standards. This includes cross-referencing AI outputs with expert knowledge and real-world data to mitigate the risks associated with erroneous or misleading information.

In contrast, lower-risk domains such as education or entertainment present fewer severe consequences from potential AI inaccuracies, and the benefits of using LLMs can outweigh the risks. As demonstrated in the university app case study 8.3, LLMs can streamline the need-finding process by efficiently generating comprehensive user needs and diverse personas. This accelerates the design timeline and provides deeper insights into user expectations and behaviors. In this context, the AI-assisted generation of user personas allowed designers to explore a broader range of user types and scenarios than might be feasible manually, enhancing the overall design quality with minimal risk.

By tailoring the integration of LLMs in the need-finding flow and user-persona generation to the specific risk profile of each domain, designers can maximize the benefits while minimizing potential drawbacks. This balanced approach allows for the effective incorporation of AI into UCD processes across various fields.

8.6 Discussion

Our exploratory study suggests that the LLM-assisted method can provide a richer and more articulated set of intermediate artifacts than direct prompting alone. In practice, this may help designers structure the early design space more explicitly and reflect on possible user needs in greater detail. In this sense, the value of the approach lies less in producing a single "better" answer and more in supporting

a broader and more traceable exploration of candidate needs, profiles, and design directions. At the same time, these advantages should be interpreted cautiously, because greater detail does not necessarily imply greater validity.

A first limitation concerns persona generation and simulated interviews. We intentionally restricted the analysis to three personas in order to keep the workflow manageable. While the resulting personas were plausible and captured some variation in needs, they cannot be considered representative of the entire target group, and they are especially unlikely to cover edge cases or less typical users. More importantly, because the same LLM is used to generate the personas and then simulate interviews with them, the process risks reinforcing the assumptions already embedded in the earlier prompts. This creates a synthetic echo chamber in which later outputs may appear consistent not because they are well grounded in real users, but because they are internally aligned with the model's own previous generations.

A related issue is that, although the AI-generated interviews were coherent and contextually relevant, they lacked the ambiguity, contradiction, and emotional depth that often characterize real user interviews. This limitation is not only about realism in a general sense, but also about method. Real need-finding often benefits from unexpected remarks, hesitation, conflicting motivations, and situated experiences that challenge the designer's assumptions. Simulated interviews, by contrast, tend to remain cleaner, more uniform, and more easily aligned with the framing already established by the prompt. As a consequence, they are better understood as instruments for structured reflection than as substitutes for empirical engagement with users.

Another significant challenge is the detection of inaccuracies, overgeneralizations, or hallucinated content in the outputs produced by LLMs. Here, the involvement of designers remains essential. Compared with a direct prompting strategy, the assisted workflow can make inaccuracies easier to spot because the reasoning process is partially externalized through intermediate artifacts such as profiles, personas, and revised needs. However, this does not remove the need for domain expertise. Human judgment is still required to assess whether the generated content is plausible, whether it reflects meaningful distinctions, and whether it risks introducing stereotypes or unsupported assumptions.

One possible extension would be to instruct the LLM to generate a broader range of personas and conduct additional interviews automatically. Such an extension could increase coverage, but it would not by itself solve the core methodological issue. Future experiments should therefore evaluate not only the variety of generated personas, but also the extent to which the resulting insights remain repetitive, stereotyped, or detached from real user perspectives. A particularly important direction is to compare LLM-generated personas and interview outcomes with data collected from actual users, in order to understand where the method is useful as a

scaffold and where it diverges from empirical reality.

The results across the three mobile app domains indicate both promise and clear limits. LLMs can support the need-finding phase of UCD by accelerating the production of useful intermediate materials and by helping designers inspect a problem from multiple angles. However, these outputs remain synthetic artifacts. Their practical value depends on keeping designers in the loop and on grounding the resulting insights, whenever possible, through comparison with real users and real contexts.

Chapter 9

Heuristic Evaluation of Implicit Interactions in Proactive Systems

9.1 Introduction

With the growing interest in AI-based technologies and their increasing integration into everyday systems, interaction is increasingly shifting from explicit command-driven use toward more seamless and context-sensitive forms of support [200]. As users become more accustomed to smart, context-aware systems, expectations are rising for technology to anticipate and respond to needs with minimal direct input [124]. Proactive agents, often powered by AI and grounded in implicit interaction, interpret user actions or contextual cues without requiring explicit commands, delivering support automatically and, ideally, in ways that feel both useful and natural [66].

This shift raises new challenges not only for system design, but also for usability evaluation. Established HCI evaluation methods were largely developed for interfaces in which users act explicitly and system responses are visible, immediate, and directly attributable to those actions. By contrast, proactive and implicit systems often operate in the background, rely on inferred context, and produce effects whose causes may not be immediately apparent to users. As a result, existing usability frameworks, including Nielsen's heuristics [137] and broader HCI notions such as learnability, flexibility, and robustness, may not transfer directly to this class of systems.

Although the motivating examples in this chapter are rooted in mobility and sensing-based interaction, the issues discussed here are broader. The same evaluation challenges also arise in newer AI systems, including LLM-based proactive assistants,

where support may be triggered by inferred user intent, remembered context, or system-initiated suggestions rather than by a clear explicit request. In this sense, implicit interaction provides a useful conceptual lens for examining a wider family of proactive systems, including those discussed elsewhere in this thesis in relation to LLM-supported interaction and design evaluation. This chapter therefore contributes to Part II by clarifying how established usability principles may need to be reinterpreted when system behavior is proactive, partially autonomous, and driven by inference.

This chapter addresses the challenges of evaluating implicit interactions in proactive systems, recognizing the need for criteria and interpretive guidance that align with the designer's goal of creating natural and seamless experiences without obscuring system behavior or reducing user control. A central focus is the applicability of Nielsen's evaluation heuristics, and more specifically how this framework can be adapted to evaluate proactive systems in which interaction is shaped by implicit input and inferred context.

The remainder of this chapter is organized as follows. Section 9.2 provides an overview of relevant literature. Section 9.3 discusses the user's spectrum of intention in implicit interaction with proactive agents, with a focus on the relationship between implicit input and its resulting effects. Section 9.4 examines the challenges of applying Nielsen's heuristics in this setting. Finally, Section 9.5 discusses the implications of these observations and draws conclusions.

9.2 Related Works

Proactive systems are intelligent technologies designed to autonomously anticipate user needs and act without explicit commands, relying on contextual data, historical behavior, and reasoning mechanisms [205, 200, 84]. A defining feature of these systems is implicit interaction, where actions are triggered by subtle cues such as sensor data or user activity, rather than direct input [124, 205].

Schmidt's early work [180] laid the foundation for implicit interaction by exploring how systems could infer user intentions without explicit input, emphasizing context-awareness and passive sensing. Building on this, Schmidt and Herrmann [183] introduced the concept of intervention interfaces, allowing users to retain control in highly automated systems, a principle closely related to balancing autonomy and user involvement in implicit interaction design.

Context-awareness has been extensively studied in HCI, with Cockton and Gram [54] highlighting the evolution of usability to accommodate non-traditional input in technologies like smart homes. Similarly, Dix's spectrum of intention [69] categorizes user interactions from explicit to incidental, providing a framework for

understanding low-intention interactions and their seamless integration into users' primary tasks. Serim and Jacucci [185] extended this discussion, identifying key types of implicit interaction and emphasizing the need for systems to avoid making users feel manipulated or disempowered.

Proactive systems leveraging implicit interaction have been explored across various domains. For instance, conversational agents utilize cues such as conversational history and user preferences to tailor responses [67, 66]. Similarly, sensing through smartphone inertial sensors has proven effective for understanding diverse contexts; for example, in earlier studies we addressed earthquake detection [151] and the identification of driving-related behaviors [23, 39]. However, challenges remain in areas such as transparency, usability, and trust, as users often find it difficult to comprehend or anticipate system behavior [143].

Recent work has focused on evaluation frameworks for implicit systems. Serim and Jacucci [185] emphasized the importance of distinguishing between helpful and intrusive interactions in ubiquitous computing environments. Mueller et al. [133] and Bennett et al. [26] both explored how systems can preserve user autonomy while seamlessly integrating with natural behaviors. Additionally, Bisante et al. [35] highlighted the importance of error detection and repair in AI-based implicit systems to prevent interaction breakdowns.

Although established usability frameworks like Nielsen's heuristics [137] provide a foundation for evaluating explicit interactions, their application to implicit systems remains underexplored. Amershi et al. [6] and Helms and Brown [88] provide relevant insights for designing human-AI interactions and sensing-based systems, but further adaptation is required to address the unique challenges of implicit interaction.

Building on these works, this paper discusses how to apply Nielsen's heuristics to evaluate implicit interactions in proactive systems, advancing evaluation practices for intelligent, context-aware systems.

9.3 Implicit interaction with Proactive Agents

Proactive agents are characterized by their ability to anticipate user intentions, relying on contextual information, acting autonomously, and operating without requiring explicit user interaction. Instead, these systems use implicit signals, such as sensor data, historical patterns, and contextual information, to make decisions. In this work, we focus on the challenges of evaluating the implicit interaction part [205, 200, 84].

As noted by Serim and Jacucci [185], the concept of implicit interaction has been defined in various ways in the literature. Our interpretation of implicit interaction builds on Dix's spectrum of intention [69] [70], which frames implicit interaction on

a continuum of sensor-based interactions, categorized by the degree of user intention behind the use of sensed inputs.

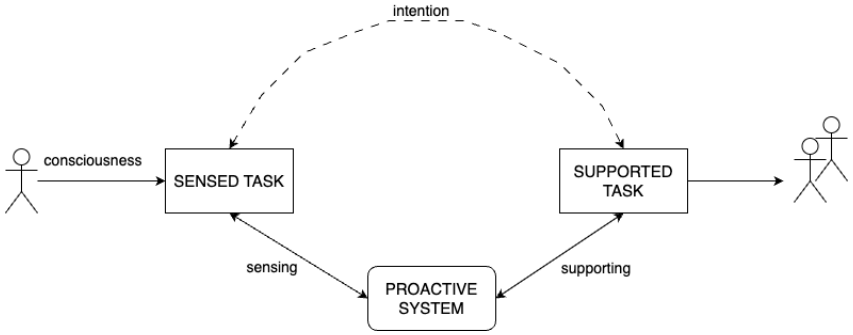


Figure 9.1. Human-system interaction for sensing and supporting tasks

At the high-intention end there are very explicit interactions, such as gesture-based systems, flicking a tablet to turn a page or annotation systems [17]. These are sometimes called 'implicit' as they recruit tacit knowledge and may become part of practiced use.

The opposite, low-intention end, (termed 'incidental' interactions) are situations where the user being sensed simply carries on their activities, with no explicit intention that the sensed data may be used by a proactive system (although they may know if they stop to think about it). It is this low-intention end of the spectrum that is the primary focus of this paper.

In these low-intention situations, it is possible to distinguish two user tasks, the sensed and supported tasks (called primary and secondary tasks in early literature):

- Sensed (primary) task: a task that the user performs regardless of the proactive system. The system senses what the user is doing to build knowledge to help in other tasks. For example, you watch a streamed film because you want to watch it, but the system builds a model of your own preferences, to help make suggestions for you or other users in the next choices.

- Supported (secondary) task: thanks to data sensed during the implicit interaction of the sensed tasks, the proactive system supports a 'future' user's task. As shown in fig. 9.2, the use of sensing data may happen in the same interaction episode (pink circle) or in subsequent episodes (orange/green circles). But also, several sensed tasks can help support a single or several supported tasks, done over time and/or done by different users. A supported task could be considered as non-proactive when the system gathers data during the sensed task and uses it to assist the user in a way that requires explicit input or triggers from the user. In this case, the system does not autonomously act but instead provides support based on user-initiated actions. In this work,

we focus only on supported tasks that are proactive and based on implicit interaction.

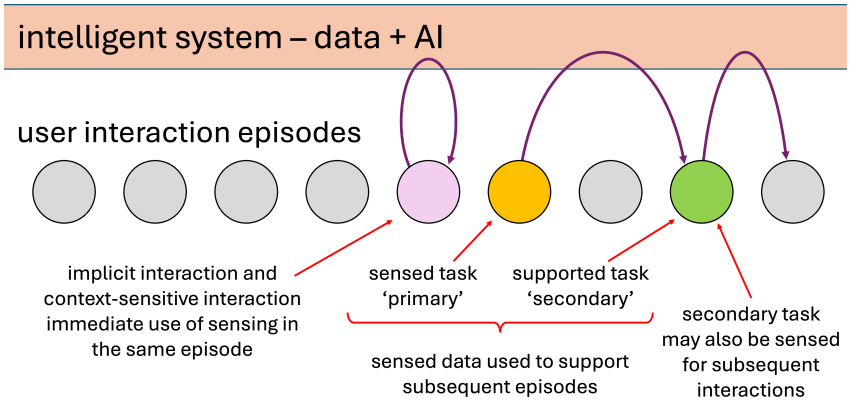


Figure 9.2. Sensed and supported tasks

9.4 Applicability of Nielsen's Heuristics

Nielsen's heuristics, traditionally applied to explicit, user-driven interfaces, are often more challenging to apply to proactive systems. In these systems, the interaction is shaped by context and user behavior rather than direct commands, and as a result, some heuristics may be less relevant or require reinterpretation.

In this section, we discuss which are the challenges of applying each of Nielsen's usability heuristics to proactive interfaces, referencing the spectrum of intentionality of the user and our focus within implicit interaction.

9.4.1 Visibility of System Status

Visibility of system status helps users understand cause-and-effect relationships in interaction and predict the system's behavior. However, maintaining the visibility of system status is often challenging with proactive systems, as implicit data gathering during the sensed task inherently obscures the system's status, posing also potential privacy issues.

Users typically engage in the sensed task naturally, with the proactive system's sensing occurring in the background. This lack of explicit feedback may lead users to focus more on the outcomes of supported tasks than on the system status itself. Moreover, when the supported task is proactive, users may find it particularly difficult to understand why specific actions are being proposed.

For example, when a light automatically switches on, its status is immediately visible, offering clear feedback. In fact, in instances where the sensed and supported

tasks overlap (the 'pink circle' scenario), the visibility of system status becomes more direct. However, in cases like air conditioning, where the effect (temperature change) is more gradual or less perceptible, the system status may be harder to detect.

In any case, if users can perceive the system's status - whether through feedback or other cues - it can help them form a mental model of how the system works. Thus, while the visibility of system status does not typically affect the sensed task directly (e.g., the user may not be consciously aware of background sensing), it can influence how the user engages with supported tasks.

9.4.2 Match Between the System and the Real World

This heuristic emphasizes that systems should use familiar concepts, language, and conventions to make interactions intuitive. In proactive systems, this applies especially on designing of the sensing process, hence the sensed tasks, to reflect real-world behaviors and expectations.

For example, users might expect lights to turn on based on their presence in a room, as this aligns with a real-world understanding of occupancy. However, if the system instead requires specific movement to activate the lights, it violates this heuristic by introducing an artificial constraint that does not match the user's mental model of presence and response. This mismatch can lead to confusion and frustration, as users may struggle to understand or predict the system's behavior.

The core challenge lies in designing proactive systems that not only interpret user behavior accurately but also translate this understanding into actions that resonate with the user's perception of how the real world operates.

9.4.3 User Control and Freedom

Users should ideally retain a degree of control over systems, including the ability to undo or cancel actions. However, in proactive systems, users often relinquish some level of control, as direct intervention would shift the interaction from implicit to explicit. Moreover, in many real-world implicit interaction scenarios, undoing implicit actions can be difficult or even impossible. For example, in systems like automatic toll collection at toll booths, once the user's vehicle passes through the toll gate, there is no way to undo the transaction or change the decision to take that particular route.

In proactive systems based on implicit interactions, control can only be exercised if users are aware of the system's operations and have a mental model of how the sensing mechanism works and how it supports secondary tasks. For instance, if users understand how a smart home system senses their movement to turn on lights, they

may intentionally modify their behavior to manipulate the outcome. However, when they do this, the interaction moves from incidental to intentional—situating the user higher on the spectrum of intention.

It is important to recognize that users generally appreciate having control, and this can enhance the usability of a system. Even though implicit interactions reduce direct user control, the option to influence the system—once users develop awareness—can provide them with a sense of agency. This is not necessarily negative from a usability perspective. However, in the case of implicit interactions designed to operate in the incidental or low-intention spectrum, control is not inherently applicable. With respect to sensed and supported tasks, this heuristic mainly applies to the relationship between the sensed and supported tasks.

9.4.4 Consistency and Standards

When considering consistency within proactive systems, we propose an interpretation that relates to maintaining uniform behavior across similar elements within the same environment. For example, if a building has multiple automatic doors, all the doors should function in a consistent manner to meet user expectations.

Inconsistent behavior across similar systems — such as one door opening automatically and another requiring pressing a button — can disrupt the seamless experience that implicit interactions aim to provide. This heuristic applies mainly to the design of the relationship between sensed and supported tasks.

Regarding the concept of 'standards', we argue that it aligns with what feels most natural to the user. More specifically, there are two levels of standards to consider:

- **User expectations based on prior experience:** Users often anticipate certain behaviors from systems based on what they have encountered before. For example, users might not expect elevators to arrive automatically without needing to press a button, but they do expect automatic doors to open in modern environments, especially when there is no visible handle. These implicit standards are based on context and familiarity with similar interactions.

- **Naturalness as the standard:** The most natural interaction often becomes the "standard" for users. For example, if users were to encounter an elevator designed to operate automatically, they would naturally expect a simple action—such as approaching it—to trigger its functionality. In contrast, requiring an exaggerated or unnatural action to call the elevator would feel cumbersome and counterintuitive. The closer a system aligns with users' natural expectations, the smoother and more seamless the interaction becomes. Understandably, what is considered natural can vary among users and may

depend on the complexity of the system or task. While acknowledging these variations, we propose this general definition as a starting point for discussing naturalness in interaction design.

9.4.5 Error Prevention

The HCI literature distinguishes between two types of errors: *slips*, which occur when users intend to perform the correct action but inadvertently perform the wrong one, and *mistakes*, which arise when users form incorrect intentions based on faulty mental models [169, 141].

In low-intention, implicit interaction systems, we argue that the risk of slips does not apply as the interaction is natural and not reliant on conscious input. However, mistakes can still happen, especially when users have an inaccurate understanding of how the system works. For example, a user may believe that their mere presence in a room will trigger the lights to turn on, when in fact, the system requires movement to activate the lights.

To prevent errors, particularly mistakes, Nielsen suggests several design considerations that remain relevant in proactive systems. Below, we discuss how each of these aspects applies to implicit interactions:

- **Minimize memory burdens:** Proactive systems should avoid requiring the user to recall prior actions or states. Requiring memory recall shifts the interaction from implicit to explicit, undermining the system's seamless nature. The design should ensure that the system reacts naturally to the current context without placing cognitive demands on the user. This aspect applies to sensed tasks.
- **Support a simple mental model:** A user's mental model of the system should be as simple and accurate as possible. In a proactive system, if users misunderstand how their actions trigger responses, they may form incorrect expectations and make mistakes. For instance, when the user mistakenly assumes that being inside a room activates the lights when the system actually senses motion detection. Clear system feedback or subtle cues can help form more accurate mental models. This aspect involves both sensed and supported tasks.
- **Warning users:** This should never be necessary for the sensed task, as the goal is to simply observe the user. If there is a need to warn users about problems, for example "please do not sit still for too long", the system has failed. It is the designer's responsibility to ensure that these problems do not occur. This said, no system is perfect, and there may be times when breaking the illusion of invisibility may be necessary. A good example of this is in

fall-detection systems: at the point a fall has been detected, most will offer ways to warn the users that their posture (maybe simply having a yoga session on the floor) is going to be treated as an incident before the ambulance is called. However, the general rule is that these are rare exceptions.

Moreover, proactive, intelligent systems today are often powered by AI and are typically trained on limited datasets, relying on constrained sensor input to interpret the environment, including users' behaviors and intentions. This inherent limitation makes such systems prone to system errors, further underscoring the critical need for designers to prioritize robust error-handling mechanisms. Effective error handling should take into account not only user errors, but also is particularly crucial to prevent system errors from escalating into full system failures, ensuring the reliability and usability of these systems [35].

9.4.6 Recognition Rather than Recall

This heuristic does not usually apply to implicit proactive systems. If users need to recall specific actions or information, the interaction becomes explicit. In a low-intention implicit system, users should not need to remember prior inputs for the system to function correctly. In practice, there can be some middle ground, for example, if people modify their walking speed as they approach an automatic door, which is effectively a form of (implicit) recall, but the underlying assumption is that the basic implicit interaction functions well enough based on the users normal (non AI-assisted) behaviour and so does not require explicit means to provide recognition. Thus, when applicable, this heuristic applies to sensed tasks.

9.4.7 Flexibility and Efficiency of Use

The heuristic of flexibility and efficiency of use suggests that systems should allow users with different skill levels or preferences to interact with the system in ways that best suit their needs.

In implicit interactions, this flexibility can manifest through personalization across two dimensions:

- **Interaction personalization:** Different implicit interactions can trigger the same outcome based on the user. For instance, a light might turn on when one user sits down at a table, but for another user, it activates when they enter the room. This dimension applies to sensed tasks.
- **Content personalization:** Different outcomes can result from the same interaction depending on the user. For example, one user's presence may

trigger only specific lights to turn on, while another user's presence activates all the lights. This dimension applies to supported tasks.

Customization in the traditional sense is less relevant in proactive systems, as it requires explicit interaction. Likewise, accelerators (shortcuts) are not applicable since implicit interactions are already designed to be the most efficient, natural responses.

In contrast, adaptations are common in proactive systems, for example, if the user frequently turns the lights to a dimmed setting after entering the room, the systems can adapt to default to that setting. However, the challenge is that normal adaptation should also be based on implicit cues and not on explicit user settings.

9.4.8 Aesthetic and Minimalist Design

Although aesthetics may not be the first consideration in implicit interactions, this heuristic still holds relevance. The design of proactive systems should prioritize simplicity and elegance, both in the selection of the sensed task and the support of the secondary.

9.4.9 Help Users Recognize, Diagnose, and Recover from Errors

This heuristic is less applicable in proactive systems. While feedback on the execution of a supported task can help users recognize when something has gone wrong (e.g., a door failing to open or a light not turning on), diagnosing and resolving such errors often requires explicit user intervention through traditional interfaces. This can be a challenge in systems relying on implicit interactions, which inherently reduce the user's direct control.

A common approach to designing for error recovery involves supporting undo functionalities. This can be achieved by providing contextual feedback that helps users understand and correct unintended actions. However, as discussed under *User Control and Freedom*, undoing actions in physical implicit interactions poses greater challenges compared to traditional digital systems. Furthermore, enabling undo functionality in these contexts often requires shifting the user toward a more deliberate, high-intention mode of interaction.

Consequently, this heuristic primarily applies to the "recognize" aspect. Any system failure should be clearly communicated to the user through explicit feedback mechanisms, ensuring the user can identify the issue and take corrective action. Challenges related to error detection and recovery in AI-based user interfaces have been explored in detail in earlier work [35].

9.4.10 Help and Documentation

This heuristic tends to be less relevant for proactive systems, especially in low-intention interactions where the system monitors users' natural behaviors and responds automatically. In these cases, if documentation is needed to explain how to interact with the system, it may indicate that the interaction is no longer truly intuitive. Ideally, proactive systems should be designed to operate in the background, with users engaging effortlessly and without the need for external guidance.

However, many natural user interfaces (NUIs), while leveraging 'natural' interactions, often feel intuitive only after users have learned how to engage with them. For example, gesture-based systems may seem natural, but they often require initial learning or practice, where documentation or onboarding can be helpful.

9.5 Discussion

The increasing prevalence of implicit and proactive interactions calls for more robust design and evaluation criteria to ensure usability and effectiveness. Traditional usability principles remain relevant, but their interpretation becomes less straightforward when interaction is no longer centered on explicit commands and directly observable cause–effect relationships.

In this chapter, implicit interaction was framed by distinguishing between the sensed task, namely the activity performed by the user and monitored by the system, and the supported task, namely the activity that the system attempts to assist on the basis of the sensed data. This distinction is useful because it clarifies that usability in proactive systems does not depend only on whether the final support is helpful, but also on whether the sensing process and the relationship between sensed and supported activity remain compatible with the user's expectations, autonomy, and ongoing practice.

Our analysis highlights the importance of carefully selecting the sensed task and designing the sensing mechanism in ways that preserve core HCI principles such as consistency, error prevention, and alignment with the real world. Since the sensed task is normally something the user would perform regardless of the system, it is important that the design does not pressure users to distort or artificially adapt that activity merely to make the system work. When users must change their ordinary behavior to accommodate the sensing logic, this should be treated as a design warning sign rather than as a minor inconvenience.

The supported task is equally important. Proactive support should not be judged only by whether it is technically triggered, but by whether it improves the user's ability to carry out the relevant activity. This brings into play principles such as feedback, predictability, error prevention, personalization, and support for an

understandable mental model. If the support is poorly timed, difficult to interpret, or easier to ignore than to use, then the implicit interaction may fail even if the sensing component works as intended.

The relationship between sensed and supported tasks is therefore central. It affects system-status visibility, user control and freedom, and the broader intelligibility of the interaction. In some situations, users may not want the supported task to occur even if the sensed task is correctly recognized. In other cases, users may deliberately perform or exaggerate a sensed action in order to obtain the corresponding support. Both situations show that proactive behavior is not neutral: it shapes expectations and can alter how users understand both the system and their own actions.

For this reason, the chapter's broader implication extends beyond sensing-based mobility examples. Similar evaluation challenges arise in other proactive AI systems, including LLM-based assistants that trigger suggestions, reminders, or contextual actions on the basis of inferred intent, remembered interaction history, or partially implicit signals. In these systems too, the question is not only whether the output is useful, but whether the user can understand why it appeared, whether it arrives at the right moment, and whether it supports rather than undermines agency. This is one of the main reasons why the discussion of implicit interaction remains relevant within Part II of this thesis.

Ultimately, when proactive support fails to help the supported task, the value of the implicit interaction is substantially reduced. If the support is mistimed, misleading, intrusive, or harder to recover from than explicit interaction would have been, then the system reveals a more fundamental design flaw. In this sense, evaluating proactive systems requires more than checking whether automation is present. It requires examining whether the automation fits naturally within the user's activity, whether it preserves autonomy, and whether it can be understood and corrected when it goes wrong.

As implicit interactions continue to evolve and become more prevalent, these considerations will become increasingly important. Designers and evaluators should therefore reflect not only on the technical feasibility of proactive support, but also on how such support is perceived, interpreted, and managed by users in practice. Only under these conditions can proactive systems become genuinely natural and effective rather than merely automatic.

Chapter 10

Transforming Interactive Systems with Large Language Models

10.1 Introduction

The emergence of powerful LLMs such as GPT-4 has introduced new possibilities for augmenting key stages of the design process. These models can generate text, simulate users, extract structure from unstructured inputs, and reason about interaction flows—capabilities that lend themselves well to tasks such as user modeling, interface evaluation, and early-stage ideation. My research investigates how to make these capabilities practically useful to designers, focusing on how LLMs can generate adaptive personas, simulate user interactions, and provide structured feedback on prototypes.

Over the past two years, I have explored LLM-based tools that assist in evaluating interactive systems through virtual Cognitive Walkthroughs and AI-driven simulations. These tools aim to identify usability issues early in the development cycle, before costly user testing takes place [41, 29]. They are not meant to replace traditional evaluation methods, but to complement them—particularly in the early design stages, where access to users may be limited and iteration speed is critical.

My earlier research focused on implicit interaction in mobile and in-vehicle interfaces [39, 34, 35, 38]. This work dealt with how systems can adapt to user behavior without relying on explicit input, and it shaped my interest in building interactive systems that are context-aware, adaptive, and user-sensitive. That foundation now informs my current exploration of how LLMs can serve as reasoning agents that support both the designer and the user during interface development.

This research combines prompt engineering, iterative prototyping, and hybrid validation strategies—where AI-generated feedback is compared against real-user data and expert evaluation. The aim is to ensure that the results are not only scalable and efficient, but also interpretable and reliable in real-world scenarios.

In the final phase of my PhD, I plan to refine these tools and study how they can be embedded into existing design workflows. The goal is to define a practical, generalizable framework for using LLMs in interface design and usability evaluation—one that maintains the core values of human-centered design while leveraging the strengths of generative models.

10.2 Related Works

User-Centered Design (UCD) has traditionally relied on qualitative methods like interviews, focus groups, and direct observation to gather insights into user needs and behaviors [142, 81]. These methods, while effective, are often time-consuming and hard to scale. Recent advances in Large Language Models (LLMs) offer new ways to support early-stage design tasks, including ideation, need-finding, and usability evaluation [182, 47].

LLMs such as GPT-4 have shown strong capabilities in generating content, analyzing text, and simulating human-like interactions. This has led to applications in several design-related areas: persona generation [175, 41], design ideation support [41, 28, 19, 221, 181], automated usability evaluations, and user simulation for testing interfaces [29]. Despite these capabilities, LLMs still struggle with implicit needs, cultural context, and emotional nuance, making it necessary to combine them with more traditional human-centered methods [29].

A key focus in early design is need-finding, which helps define user requirements and shape design directions. Classic approaches such as ethnographies and structured interviews [142, 81] are thorough but often difficult to apply at scale. LLMs offer an alternative by helping synthesize user needs from large datasets and generate potential problem spaces [175, 182].

In previous work [41], we explored how ChatGPT-based tools could support need-finding across three case studies. The results showed that LLMs can effectively augment design workflows by surfacing structured insights and analyzing contextual information. However, their output should be interpreted as assistive rather than definitive, since they lack real-world experience and may overlook subtle user expectations.

LLMs have also been used to assist with UI layout generation and prototyping. For instance, they can help define target users, generate design variants, and even produce initial interface code [28, 221, 181]. These tools can be especially helpful

for novice designers or in contexts where fast iteration is needed.

On the evaluation side, usability testing remains essential for validating whether interfaces meet user needs. Cognitive Walkthroughs (CWs) are one common method, where experts step through tasks from the user's perspective [166]. While useful, CWs require time and expertise. Commercial platforms now offer remote testing tools like session recordings, A/B tests, and heatmaps [204, 114], but for many designers—especially those with limited experience—the sheer number of tools and data can be overwhelming [104].

To make early-stage evaluations more accessible, our recent work introduced a ChatGPT-based tool that conducts CW-inspired assessments [29]. These evaluations help identify usability issues early in the process, without requiring real users. However, they still have clear limitations: LLMs struggle with visual feedback, interaction timing, and non-verbal cues, which suggests that hybrid approaches combining AI and human input are likely to be more effective.

Lastly, LLMs have also been used for automated GUI testing, where they interact with interfaces and produce structured feedback. While traditional tools focus on task completion or bug reports, LLM-based agents bring a more interpretive layer by explaining their behavior and reasoning [113]. This area shows promise but still requires refinement to better approximate real-world user behavior.

10.3 Methodology

This research adopts an iterative prototyping approach, combining empirical user testing with expert evaluation to explore how Large Language Models (LLMs) can support interface design and usability assessment. The goal is to develop a practical and interpretable framework for integrating LLMs into early-stage design workflows, particularly when access to users is limited or fast iteration is required.

The first phase focused on prompt design. I created structured prompts to guide LLMs in generating realistic and diverse user personas, which were then used to simulate interviews and focus group responses. These simulations provided early insights into user expectations and design assumptions, helping surface potential problems in low-fidelity prototypes.

Beyond ideation, the same personas were involved in simulated usability walkthroughs. Here, the model reasoned through interaction flows from different user perspectives, flagging issues such as poor labeling, unclear paths, or mismatched expectations. These walkthroughs followed the principles of Cognitive Walkthroughs but adapted the process for LLM-based generation. An example of this prompt structure is shown in Figure 10.1, where the user specifies a task and uploads a screenshot, prompting the model to iteratively suggest actions and ask for follow-up

Figure 10.1. Example of a prompt with CWGPT to evaluate an interface [29]

inputs in a structured loop.

To evaluate the reliability of this approach, I adopted a hybrid validation strategy. LLM-generated insights were compared with real user data and assessed by HCI experts to understand how well they aligned with actual behavior and expectations. This helped reveal both the value and the limitations of using LLMs in early evaluation stages.

Results from this work are reported across several studies. In [29], we introduced a ChatGPT-4-based tool for conducting automated walkthroughs, showing its ability to surface subtle usability issues often missed in early heuristics. In [41], we applied LLMs to need-finding tasks across three design contexts, demonstrating their usefulness in generating structured, relevant insights to support ideation. Finally, in [35], we examined how AI-driven interfaces handle user and system errors, evaluating strategies for real-time recovery and trust maintenance.

This methodology combines the speed and scale of generative models with the grounding of human-centered design. By treating LLMs as assistive tools within a validated pipeline, the approach supports more efficient, informed, and user-aware design decisions.

10.4 Ongoing Work

The current phase of this research builds on previous work on need-finding and AI-assisted Cognitive Walkthroughs [38, 29], with a focus on improving how LLMs can support the early stages of interface design and evaluation. The main goal is to refine the use of LLMs in simulating plausible user behavior and generating structured feedback that designers can inspect, question, and act on.

One area of focus is improving the quality and specificity of AI-generated personas and interaction scenarios. This involves designing better prompts and incorporating contextual information from specific domains to produce more nuanced simulations. At the same time, these generated artifacts should be treated cautiously: as discussed in previous chapters, synthetic personas and scenarios can become internally consistent without necessarily being well grounded in real user experience. For this reason, the objective is not to replace empirical user understanding, but to create intermediate materials that can help structure early design exploration and surface candidate issues for further validation.

These simulations are then used in LLM-based walkthrough-style evaluations aimed at identifying potential usability problems, such as unclear instructions, navigation friction, or task breakdowns. Their practical value lies mainly in early-stage iteration, where fast feedback may be useful even before more formal evaluation with users becomes feasible.

At the same time, this research is developing hybrid evaluation workflows that combine AI-generated insights with expert review and, where possible, user-based validation. This layered approach addresses known limitations of LLMs, including their difficulty in interpreting visual layouts, emotional tone, and subtle contextual cues, and helps ensure that evaluations remain grounded in a more realistic understanding of user needs and interface use.

Despite encouraging early results, several limitations remain. LLM-generated feedback, while often useful, lacks the variability, unpredictability, and situated depth of real user responses. These tools also struggle with complex or dynamic interfaces, especially when multimodal elements are involved. In particular, when interfaces are generated or adapted dynamically, the evaluation problem becomes more challenging: the same flexibility that supports personalization may also reduce consistency across interactions, making it harder for users to form stable mental models and spatial memories of the interface. Another limitation is the tendency of LLMs to rely heavily on general usability heuristics, which may cause them to miss domain-specific design challenges or overproduce familiar categories of critique.

To address these issues, future work will increase the role of human validation within the evaluation pipeline and explore stronger ways of conditioning LLMs on multimodal and domain-specific input. Another direction is embedding these tools

directly into existing design environments, so that designers can run lightweight evaluations while iterating on wireframes, flows, and interface components. Here too, the challenge will be to support rapid iteration without encouraging overreliance on synthetic feedback at the expense of real user evidence.

This work moves toward building scalable and accessible evaluation methods that use LLMs not as replacements for human-centered techniques, but as augmentations to them. The long-term goal is to support both expert designers and less experienced practitioners in creating more usable and user-aware interactive systems, while preserving grounding, interpretability, and consistency.

10.5 Research Objectives, Contributions, and Dissertation Goals

This research contributes to the field of Human-Computer Interaction (HCI) by exploring how Large Language Models (LLMs) can support the design and evaluation of interactive systems. The main objective is to develop an LLM-assisted evaluation framework that helps designers simulate user interactions and detect usability issues early in the development process. This builds on previous work on AI-guided need-finding, virtual walkthroughs, and error recovery [29, 35, 41, 38].

The goal of this framework is not to replace human evaluation, but to offer a scalable method for generating useful feedback at the prototype stage—especially in situations where traditional methods are too slow, costly, or resource-intensive. By combining prompt-driven simulations, hybrid validation strategies, and expert review, the framework aims to deliver actionable insights that improve interface quality and usability.

So far, the work has resulted in multiple functional prototypes and peer-reviewed publications, showing the potential of LLMs in supporting both ideation and evaluation tasks. Current efforts are focused on extending empirical testing, improving the reliability of model-generated insights, and embedding these tools into design environments for more seamless use. Collaborations with external partners will help evaluate the framework in applied contexts.

The broader aim of the dissertation is to define a practical methodology for integrating LLMs into real-world design workflows—one that respects core HCI values such as transparency, user agency, and interpretability. By bridging AI-driven automation with human-centered design practices, this work hopes to contribute to a more adaptive and accessible future for interface development.

10.6 Discussion

This research explores how Large Language Models can support the design and evaluation of interactive systems, with a focus on two main directions: using LLMs to simulate user behavior and assist usability evaluation, and supporting early-stage design activities such as persona generation and need-finding.

The findings discussed throughout this chapter suggest that LLM-assisted methods can improve the speed and breadth of early interface evaluation, especially during prototyping, when direct access to users may be limited. In these settings, LLM-based tools can help designers surface potential issues, generate alternative interpretations of an interaction flow, and iterate more quickly on tentative design ideas.

At the same time, the work also highlights clear limitations. LLMs can miss domain-specific nuances, struggle with multimodal inputs, and generate feedback that appears plausible but may not remain valid under real-world conditions. For this reason, the most promising use of these systems is not as a replacement for human evaluation, but as part of a hybrid workflow in which AI-generated outputs are reviewed and grounded through expert judgment and user-based validation.

An additional challenge concerns the role of LLMs in systems that do not merely analyze interfaces, but also help generate or dynamically adapt them. When interface structure, labels, or navigation paths are generated on the fly, the resulting flexibility may improve personalization and short-term task support, but it can also introduce costs for users. In particular, dynamically generated interfaces may weaken the stability of the spatial and structural cues that users rely on to build familiarity over time. If the placement of controls, the wording of options, or the organization of content changes too often, users may find it harder to form a reliable mental model of the system, remember where functions are located, and reuse prior experience efficiently. This suggests that adaptive and LLM-mediated interfaces should be designed with particular care, preserving enough consistency to support orientation and learning even when personalization is introduced.

The long-term goal of this research is therefore not simply to make interface design and evaluation faster, but to develop reliable, interpretable, and accessible tools that support both experienced designers and newcomers while remaining aligned with the principles of human-centered design. In this perspective, the contribution of LLMs is strongest when they expand the range of design alternatives and evaluative perspectives available to humans, while the final responsibility for grounding, consistency, and usability remains with the designer.

Chapter 11

Memory Augmented Generation

11.1 Introduction

Large Language Models (LLMs) have achieved remarkable progress in generating fluent and contextually appropriate text. Yet most current systems operate within the boundaries of a single prompt or, at best, a limited conversational window. This constraint makes it difficult for agents to sustain meaningful, personalized, and context-rich interactions over time. By contrast, human communication depends on memory [179]: we recall past discussions, adapt to personal preferences, and build on shared experiences. Without memory, LLMs risk remaining reactive tools that reset with every session rather than evolving into trusted collaborators.

Recent work has begun to explore memory-augmented systems, but many approaches still treat memory as either a flat retrieval store or an extended context buffer. Such designs improve access to past information but do not capture the diversity of memory functions that shape human dialogue. Human cognition distinguishes between short-term recall, long-term factual knowledge, episodic experiences, and context awareness, each playing a distinct role in communication [123]. Bringing this perspective into LLM design can help move beyond longer contexts toward richer and more adaptive interactions.

To address this need, we propose the Mixed Memory-Augmented Generation (MMAG) pattern. MMAG organizes memory into five interacting types: conversational history, long-term user traits, episodic and event-linked knowledge, sensory and context-aware signals, and short-term working memory. This taxonomy draws inspiration from cognitive psychology while mapping directly to technical components such as vector databases, structured stores, and event triggers. By coordinating across layers, agents can recall relevant past exchanges, adapt tone and recommendations, surface timely reminders, and remain sensitive to environmental context without overwhelming users.

We implement and study MMAG in the Heero conversational agent[1], a language learning platform where continuity and personalization are crucial for engagement. Even with a partial deployment that combines conversational history with encrypted long-term bios, Heero demonstrates tangible improvements in user retention and conversation length. These results highlight the practical value of modular memory design and motivate further extensions.

The rest of this paper introduces the MMAG taxonomy, discusses coordination and implementation strategies, and reports on user-facing outcomes. We close with a discussion of opportunities and risks, as well as open challenges in balancing technical performance, user autonomy, and ethical responsibility when building memory-rich agents.

11.2 Related Works

The integration of memory into large language models has become a growing area of interest as researchers and practitioners seek to extend the capabilities of LLMs beyond short-lived prompts. Early efforts focused on retrieval-augmented generation (RAG) methods, which dynamically pull relevant documents from external sources. More recent work has shifted toward deeper and more structured forms of memory that aim to simulate aspects of human cognition, including long-term memory, episodic recall, and contextual awareness.

MemGPT [148] is a representative example of this trend. It treats memory as a system-level construct, organizing context into long-term and working memory and introducing scheduling mechanisms that mimic operating system processes. This helps the model maintain relevant information across conversations without bloating the context window. Similarly, Retentive Networks [116] offer architectural improvements to improve long-range coherence, introducing structured memory representations directly into the model's design. These systems show that expanding memory capabilities isn't just about scaling up context size but rethinking how memory is organized and accessed.

Another line of work has focused on hybrid approaches that combine storage and retrieval strategies. MemoryBank [226], for instance, proposes a two-tiered memory system that uses dense retrieval for semantic matching and sparse memory for temporal relevance. This framework supports long-term personalization and adapts to user interaction patterns over time. On the applied side, Medium articles such as Nigam's guide to memory-augmented RAG [138] and Kim's tutorial on modern LLM memory techniques [103] walk through practical strategies for managing memory in real-world systems, including the use of time-based triggers and structured memory

[1] `https://heero.me/`

stores.

As memory-augmented systems proliferate, researchers have begun to ask how well these systems actually remember. Tan et al. [198] explore this question directly, analyzing how LLMs retain and forget information across sessions. Similarly, Maharana et al. [116] propose evaluation methods for long-term memory in dialogue agents, focusing on the alignment between remembered content and user expectations. LongMemEval [217] introduces a comprehensive benchmarking suite that enables developers to evaluate memory retrieval, relevance, and latency in long-term interactive settings under a unified framework.

Surveys and conceptual papers have also helped shape the field. Wu et al. [218] present "From Human Memory to AI Memory: A Survey on Memory Mechanisms in the Era of LLMs," a thorough taxonomy of memory components in LLM systems, while Paul's overview [162] highlights the emerging consensus on the need for modular, human-aligned memory structures. Both emphasize the importance of distinguishing between different types of memory, such as conversational, biographical, and contextual, and recognizing their distinct roles in interaction.

Finally, applied perspectives from industry, such as the IBM Research blog on memory-augmented LLMs [94], help bridge the gap between research and deployment. These sources focus on practical concerns like latency, privacy, and system design, factors that are essential when building memory-rich systems meant to operate in open-ended, real-world settings.

Our work builds on this foundation by proposing a unified pattern, that we call Mixed Memory-Augmented Generation (MMAG), that organizes memory across several distinct but interacting components. Unlike approaches that treat memory as a single block or retrieval mechanism, MMAG introduces a layered system that reflects the diversity of memory types found in human interaction, including biographical data, time-sensitive cues, and situational context.

11.3 Memory Taxonomy

Memory in human cognition is not monolithic; it consists of multiple interdependent systems with distinct functions. Cognitive psychology differentiates between episodic memory, semantic memory, working memory, and procedural memory, among others. Inspired by this framework, we organize memory for LLM-based agents into five categories: conversational, long-term user, episodic and event-linked, sensory and context-aware, and short-term working memory. Each type captures a different dimension of interaction, and together they form a taxonomy that supports richer, more human-aligned behavior. This taxonomy both mirrors human memory systems and maps onto practical technical components such as vector databases,

key–value stores, and scheduling modules.

11.3.1 Conversational Memory

Conversational memory represents the thread-level context that allows an agent to sustain coherence across ongoing dialogues. Unlike single-turn interactions, human conversations depend heavily on continuity: we expect our interlocutors to remember prior exchanges, resolve references, and maintain shared understanding over time [48]. For LLM systems, this memory can be realized through mechanisms such as dialogue threading, summarization of past turns, and reference resolution techniques. A well-structured conversational memory enables the agent to reintroduce prior topics seamlessly, manage interruptions, and maintain a consistent persona. Technically, this may involve sliding context windows combined with compressed conversation histories or retrieval from structured dialogue logs.

11.3.2 Long-Term User Memory

Long-term user memory functions as a form of biographical storage, encoding information about a person's preferences, background, and traits. This aligns with human semantic memory, which captures factual knowledge that is not tied to specific events [98]. In practice, such memory allows an agent to adjust tone, personalize recommendations, or adapt interaction style based on accumulated knowledge of the user. For example, remembering dietary preferences when suggesting recipes, or recalling that a user prefers concise explanations. Implementations must address privacy concerns, ensuring that sensitive personal information is stored securely, possibly encrypted, and retrieved with explicit user consent. Techniques such as federated learning and selective forgetting can support personalization while protecting user autonomy.

11.3.3 Episodic and Event-Linked Memories

Episodic memory in humans refers to the recollection of specific events situated in time and place [120]. For LLM agents, this translates into storing and using information tied to particular events or routines.

Time-Linked Event Memory

Time-linked event memory captures knowledge of scheduled or anticipated events that are relevant to the user. This includes upcoming meetings, anniversaries, or travel plans. When properly managed, this memory enables timely and proactive interactions, such as reminding the user that a colleague's visit is approaching or

surfacing relevant context before a scheduled task. Technically, it requires integration with scheduling systems, timestamped memory storage, and trigger-based retrieval mechanisms.

Routine and Habitual Cues

Beyond discrete events, agents can benefit from recognizing routines and habitual behaviors. Detecting recurring patterns, such as a user regularly discussing cooking on weekends, enables generative suggestions anchored in habits. This memory layer parallels human procedural memory, which encodes repeated activities and skills. Implementation can leverage pattern detection algorithms over interaction logs, generating nudges or lightweight reminders without overwhelming the user. The challenge lies in offering support that feels natural and useful rather than intrusive [35].

11.3.4 Sensory and Context-Aware Memory

Humans rely on sensory memory to situate conversations within the surrounding environment [160]. For LLM agents, sensory and context-aware memory corresponds to integrating signals such as location, weather, or time of day. For instance, the agent might acknowledge that commuting in the rain could make for a difficult day, or adapt its tone depending on whether the interaction occurs during work hours or leisure time. This dimension supports contextual sensitivity and grounding, but it must avoid over-personalization that may feel invasive. Design choices should balance proactive assistance with user comfort, aiming for behavior that is "helpful but not creepy."

11.3.5 Short-Term Working Memory

Working memory in psychology refers to the capacity to hold and manipulate information over short intervals in support of ongoing tasks [18]. For LLM agents, short-term working memory provides a transient workspace for task-specific goals and immediate conversational focus. Unlike long-term storage, this memory is ephemeral: once the task or dialogue session concludes, the content can be discarded. This distinction helps avoid clutter in the persistent memory layers while ensuring coherence during complex interactions, such as multi-step problem solving. Technically, it can be implemented as an in-session scratchpad or temporary buffer that feeds into reasoning and generation pipelines.

Memory Type	Cognitive Psychology Analogy	Technical Component
Conversational Memory	Discourse-level memory, reference resolution	Dialogue threading, summarization, context windows, vector retrieval
Long-Term User Memory	Semantic/biographical memory	Secure profile store, encrypted databases, federated learning, preference embedding
Time-Linked Event Memory	Episodic memory (situated in time)	Scheduling modules, timestamped storage, event-triggered retrieval
Routine and Habitual Cues	Procedural memory, habit formation	Pattern detection, habit recognition, lightweight nudges
Sensory and Context-Aware Memory	Sensory integration, situational awareness	Contextual signals (location, weather, time), multimodal inputs, adaptive prompting
Short-Term Working Memory	Working memory (task-specific recall)	In-session buffers, ephemeral scratchpads, temporary embeddings

Figure 11.1. Taxonomy of memory types in MMAG, showing the mapping between human cognitive psychology and technical components for LLM-based agents.

11.4 Coordination Across Memory Types

While each memory type contributes distinct capabilities, the real value of a mixed memory system lies in how these modules interact. In human cognition, memory systems rarely operate in isolation: episodic memories can draw on semantic facts, working memory relies on retrieval from long-term stores, and contextual cues shape what is recalled. A similar principle applies to LLM-based agents. Coordinating memory types is essential for ensuring that responses are not only accurate and personalized but also coherent across time and situation.

11.4.1 Interaction Between Memory Modules

The modules described in the taxonomy work best when orchestrated rather than siloed. Conversational memory provides immediate context for dialogue, but it may need supplementation from long-term user memory to resolve references about past preferences or facts. Episodic memories can trigger reminders or inject relevance into a conversation, while sensory context acts as a filter that shapes tone and timing. Short-term working memory acts as the integrative layer, temporarily holding information drawn from other modules to support reasoning within the current task.

11.4.2 Prioritization and Conflict Resolution

A central challenge is deciding which memory signals should take precedence when multiple sources are relevant. For example, a conversational thread might suggest one line of continuation, while sensory context (e.g., a change in location) points to another. In such cases, prioritization strategies are required. These may include:

- **Recency heuristics**: prioritizing the most recent and contextually salient memory.
- **User-centric weighting**: giving precedence to long-term user traits when they explicitly affect personalization.
- **Task-driven rules**: elevating working memory for ongoing problem-solving to prevent distraction.

Conflict resolution can also involve generating candidate responses under different memory constraints and selecting the one that best balances personalization, coherence, and utility.

Example Modular Architecture

A practical architecture for coordination involves modular memory management with retrieval triggers. Each memory type is encapsulated as a service with defined input–output interfaces. A central memory controller orchestrates these services, deciding when to query each memory store and how to merge the retrieved information. Retrieval can be proactive (event-driven, such as surfacing an upcoming meeting) or reactive (on-demand, such as recalling a past conversation when the user references it). Prioritization policies and conflict-resolution mechanisms are encoded in the controller, which balances responsiveness with user comfort and privacy.

This modular approach allows new memory types to be integrated without redesigning the entire system. For example, adding a multimodal sensory memory that incorporates visual signals would only require extending the controller's coordination logic. In this way, the architecture remains extensible, robust, and aligned with the complexity of human-like memory use.

11.5 Implementation Considerations

The memory taxonomy outlined earlier provides a conceptual framework, but practical deployment requires concrete implementation strategies. In our system, memory is managed through a modular interface that separates storage, retrieval,

and integration with the LLM. Our suggested implementation demonstrates how conversational and long-term user memory can be realized in practice, while leaving room for extension toward episodic, sensory, and working memory.

Memory Storage and Retrieval Architecture

The system defines a `Memory` interface with methods for persisting individual messages, storing message batches, and retrieving a conversation history formatted for OpenAI models. This ensures that conversational memory is both append-only (new turns are remembered in Firestore) and retrievable in chronological order. To respect model limits, a token-based pruning mechanism discards excess context once a threshold is reached (90k tokens in our implementation), approximating a form of working memory management. In addition, long-term user traits are injected as structured system messages via a dedicated function, which retrieves biographical data and traits from a persistent store and appends them to the prompt context.

Scalability and Performance Implications

Storing conversation histories in Firestore ensures persistence and cross-session continuity but introduces challenges as histories grow. To mitigate latency and context overload, the implementation relies on lightweight filtering: empty or malformed messages are skipped, and only the most relevant slice of conversation is retrieved for the model. This design allows the system to scale without degrading user experience. At the same time, the modular interface makes it straightforward to swap in more advanced backends (e.g., hybrid vector + sparse retrieval) if scaling requirements grow.

Prompt Engineering for Memory Referencing

A critical step lies in transforming stored memory into model-readable input. The implementation uses a conversion layer that maps user and assistant turns into OpenAI-compatible message roles. Biographical memory is prepended as a system message, ensuring it is factored into generation without cluttering the conversational history. This illustrates a general design principle: conversational memory should be injected as dialogue turns, while long-term knowledge is best represented as higher-priority system-level context. This separation reduces noise and allows prompt space to be allocated proportionally to different memory types.

Privacy, Security, and User Control

Long-term user memory introduces sensitive information, which makes privacy and security essential considerations. In our implementation, user bios are encrypted using envelope encryption and then compressed before being persisted in private S3 buckets, ensuring confidentiality and efficiency in storage. Personal traits, bios, and KPIs should be encrypted in storage, auditable by users, and erasable on demand. Beyond technical safeguards, user trust depends on granting individuals meaningful control over their data, including the ability to inspect what has been remembered, edit inaccuracies, or request selective forgetting. While the current implementation demonstrates feasibility, further work is needed to extend full user control and to integrate stronger privacy guarantees such as differential storage policies or federated personalization.

Extensibility Toward Full Taxonomy

Although the present implementation covers conversational and long-term user memory, the same architecture can be extended to episodic and sensory layers. Event-linked memories can be stored as timestamped Firestore entries and scheduled for proactive retrieval. Context-aware memory can be integrated through lightweight API calls (e.g., weather, location) feeding into system prompts. Because memory access is funneled through the `Memory` interface, these extensions require minimal changes to the orchestration layer, reinforcing modularity as a central design choice.

11.6 User Studies

11.6.1 Use Case

To ground the proposed architecture in practice, we discuss its deployment within the Heero app, a conversational agent designed to support language learning through personalized dialogue. Heero provides a useful testbed for mixed memory-augmented generation because sustained learning interactions benefit from both conversational continuity and personalization. In the current implementation, the backend integrates conversational memory, by retrieving recent dialogue turns, and long-term user memory, encoded through encrypted user bios, so that the agent can adapt tone, recall prior progress, and maintain a sense of familiarity across sessions.

For example, conversational memory allows Heero to resolve references to earlier exercises or questions within the same session, while long-term user memory enables it to recall a learner's goals, such as preparing for a trip or improving professional English, and adjust dialogue scenarios accordingly. In this way, the system can provide interactions that feel more consistent and supportive than short-lived,

context-free chatbots.

11.6.2 Evaluation Strategies

Evaluating memory usefulness and reliability requires a combination of user-centered and technical measures. At this stage, however, the evidence should be interpreted as preliminary and exploratory. The current results come from an early product deployment and are intended as proof-of-concept evidence that memory can affect engagement and perceived continuity, rather than as a definitive validation of MMAG as a general framework.

User-centric metrics We considered perceived helpfulness, non-intrusiveness, and emotional impact. Helpfulness captures whether memory improves the quality of interaction, for instance by supporting continuity and more personalized prompts. Non-intrusiveness reflects whether memory behaviors feel supportive without becoming invasive or uncomfortable. Emotional impact concerns whether users feel more motivated, comfortable, and engaged when memory is active.

In an exploratory A/B deployment on 200 users, of whom approximately 80% used the standard version of the app and 20% used the memory-enhanced version, we observed a 20% increase in retention over a four-week period and a 30% increase in average conversation duration in the memory-enhanced condition. These findings suggest that memory features may have made interactions more engaging and sustained. Users also qualitatively reported that the agent felt as if it "knew" them better, which is consistent with the intended effect of long-term and conversational memory. At the same time, these results should be interpreted cautiously: the groups were imbalanced, the evaluation was conducted in a real deployment context rather than a controlled experiment, and we did not perform formal statistical significance tests. For this reason, we treat these findings as encouraging early evidence rather than conclusive proof of effect.

Technical metrics From a system perspective, we considered retrieval accuracy, latency, and memory leakage. Retrieval accuracy concerns whether the correct memory is recalled when needed. Latency concerns the time overhead introduced by memory retrieval and prompt construction. Memory leakage refers to cases in which information persists beyond its intended scope or is surfaced incorrectly.

A key practical observation from the deployment was that introducing additional memory layers did not produce a noticeable degradation in conversational responsiveness. In the current architecture, certain operations, such as generating or updating biographical memory entries, are performed asynchronously and cached for reuse, keeping prompt construction relatively lightweight. Internal monitoring

and manual audits suggested that average response latency remained in the same operational range as before memory integration. Still, this should be regarded as an engineering observation rather than a formal benchmark, since latency was not evaluated through a controlled performance study.

Mixed evaluation Looking at user behavior alongside system performance highlights the trade-offs of memory design. For example, aggressive pruning strategies help keep latency low but may weaken continuity, while deeper personalization may improve engagement but also increase the risk of discomfort if the agent appears too intrusive. In Heero, the most promising balance appeared to come from lightweight conversational pruning combined with richer long-term user memory. This combination preserved responsiveness while still making the interaction feel more continuous and supportive. However, this conclusion is again preliminary and specific to the current deployment setting.

11.7 Discussion

Equipping conversational agents with richer memory opens important opportunities, but it also introduces technical, methodological, and ethical challenges. Memory allows agents to move beyond reactive single-turn responses toward more sustained interactions. In the Heero case, memory made it possible to recall user goals, reinforce progress, and adapt tone, and the early deployment data suggest that this may improve engagement and retention. Even so, the current evidence remains exploratory and should not be over-generalized beyond this use case.

A central tension lies in balancing proactivity with user autonomy. When an agent recalls past events or habits, it can provide timely reminders or more relevant prompts. The same behavior, however, may also feel intrusive if users do not expect it, do not understand why it happens, or believe that the system is inferring too much from limited evidence. Striking the right balance therefore requires careful design choices: memory should be transparent, users should remain able to control what is stored and surfaced, and proactive interventions should be framed as supportive suggestions rather than prescriptive actions.

Several open challenges remain. From a technical perspective, scaling memory systems without introducing latency, retrieval errors, or leakage is a continuing concern. From a social and interaction perspective, questions of trust, agency, and emotional dependence become especially important when these systems are deployed in sensitive contexts such as education, healthcare, or personal productivity. Privacy and fairness are equally central: agents should avoid reinforcing biases encoded in long-term memories, should respect cultural differences in expectations around personalization, and should provide users with ways to inspect, edit, and erase stored

information.

Future work will extend MMAG along several dimensions. First, multimodal memories could connect textual interaction with visual, auditory, or sensor-based context for richer grounding. Second, long-term memory mechanisms could evolve beyond relatively static bios toward more dynamic representations of goals, habits, and progress. As LLM context windows continue to expand, systems will also be able to manage longer and more diverse records, although this does not remove the need for prioritization and selective retrieval. A particularly important direction is increasing user agency over memory, for example through interfaces that allow people to view, edit, and selectively delete records.

MMAG offers both a taxonomy and a practical design pattern for equipping LLM-based agents with memory capabilities inspired by human cognition. The current Heero deployment provides promising early evidence that such memory can support more continuous and engaging interactions. However, the long-term value of memory-rich systems will depend not only on technical performance, but also on how well they balance personalization, transparency, and user autonomy.

Chapter 12

Conclusions

This thesis set out to explore a simple but stubborn problem: how can we use AI to make interactive systems better without making them fragile, opaque, or harder to trust? The work approached the problem from two complementary directions. On one side, AI becomes part of the interaction itself, through implicit sensing and inference that reduce the need for explicit user input. On the other side, AI becomes a tool for the people who design and evaluate interfaces, accelerating early-stage work through Large Language Models and structured, reusable evaluation workflows. The dissertation frames this as a dual contribution to interactive systems, with a shared emphasis on benefiting from automation while keeping both interaction and design processes robust when AI is uncertain or fallible.

12.1 What this thesis contributed

A first contribution of the thesis is a set of methods and systems that treat the smartphone as a practical platform for implicit interaction in mobility. In this context, the central design goal is not only detecting events and states, but doing so in a way that respects the realities of everyday use: users are moving, attention is limited, and sensing is noisy. This perspective leads naturally to interaction designs that assume ambiguity will happen and therefore must be handled through feedback, user-correctable outcomes, and conservative proactive behaviour. The mobility part of the thesis also broadened the sensing palette by investigating Bluetooth Low Energy visibility as a way to infer co-presence and social density, which can matter as much as motion when the goal is to support passenger-aware, context-adaptive interaction. In this framing, BLE is interesting not only as a sensing signal, but because it can support privacy, low overhead, and user-understandable outcomes, which are critical constraints in real deployments.

A second contribution is methodological: the thesis argues that LLMs are

most useful in HCI when they are placed inside workflows that make their output inspectable and comparable, rather than treated as authoritative answers. In the dissertation, LLMs are used to support early evaluation activities, such as generating walkthrough-style feedback and simulating task flows, while being explicitly validated through a hybrid strategy that combines model outputs with expert review and, when possible, user data. This is motivated by a pragmatic observation repeated across the thesis: LLM feedback can be fast and often helpful, but it can also be shallow, overly generic, or inconsistent, especially when interfaces are complex, dynamic, or multimodal. For this reason, the work positions LLMs as augmentations to human-centred techniques, not replacements, and aims to embed them into design environments so that lightweight evaluation becomes part of iterative prototyping rather than a separate, heavyweight phase.

A third contribution is applied and engineering-oriented: the thesis investigated how LLMs can support structured input generation and form completion in real settings where information is dispersed across heterogeneous documents. The work proposed a modular pipeline that combines form introspection, document processing, and schema-guided LLM reasoning, and it makes two interaction choices that are essential for trust: outputs remain editable, and confidence scores guide review effort. The thesis also makes clear that these systems only become viable when they are integration-aware and mixed-initiative by design, meaning that they fit into existing infrastructures and explicitly support human oversight at scale.

12.2 A unifying lesson: design for fallible automation

Across both parts, a consistent lesson emerged. The key challenge is not eliminating errors, because both sensing-driven inference and generative models will remain imperfect. The more productive objective is to design the interaction so that errors are contained: the system should communicate uncertainty, allow the user to correct outcomes quickly, and avoid irreversible consequences when confidence is low. The thesis explicitly frames this as a shared theme across sensing-driven interaction and LLM-based workflow support: automation can still be valuable even when it is fallible, as long as systems remain transparent, user-controlled, and equipped with recovery mechanisms.

This view also helps align two communities that are often discussed separately. In mobility sensing, the main risks relate to misclassification and context shifts. In LLM-supported design and evaluation, the main risks relate to plausibility without grounding, variability across prompts, and difficulty analysing visual and multimodal interaction. In both cases, the right response is similar: make the system legible, keep humans in the loop, and treat error handling as a first-class design problem rather than an afterthought.

12.3 Limitations

The thesis also has limitations that define where results should be interpreted conservatively.

First, some contributions are grounded in prototypes, early evaluations, or bounded datasets. This is appropriate for exploring feasibility and shaping methodology, but it limits claims about long-term adoption, learning effects, and behaviour in uncontrolled environments. The dissertation itself points toward the need for broader validation and deployment-oriented evaluation as a next step.

Second, both sensing pipelines and LLM pipelines are sensitive to context. BLE visibility patterns can vary with environment and device ecosystems, and LLM-based systems can degrade with noisy documents, inconsistent terminology, or interface changes. In the structured input pipeline, this is visible in how performance decreases with low-quality or irregular documents, and in how confidence scores, while useful, are not statistically calibrated and can lead to over-trust or under-trust.

Third, multimodality remains a practical bottleneck for LLM-based evaluation. The thesis notes that current tools struggle with complex or dynamic interfaces and with aspects that depend on visual layout, emotional tone, or subtle user signals. This reinforces the importance of hybrid validation and careful scoping of what the model is asked to do in an evaluation workflow.

12.4 Future work

Several directions follow naturally from the thesis results.

A first direction is to move from promising prototypes to sustained, deployment-oriented studies. In implicit interaction systems, this means longitudinal validation in everyday mobility, focusing on trust calibration, perceived control, and how people learn to correct or override automation. In LLM-assisted design tools, this means evaluating impact in real design teams, measuring not only speed but also downstream outcomes such as rework, defect escape, and the quality of design decisions.

A second direction is stronger uncertainty handling. For structured input generation, the pipeline can be improved by scaling evaluation to more diverse datasets and partners, adding preprocessing such as table extraction for scanned or noisy PDFs, and incorporating iterative refinement strategies that learn from user corrections. These additions are not only technical upgrades; they support a better interaction contract, where uncertainty is acknowledged and progressively reduced through mixed-initiative collaboration.

A third direction is tighter integration of LLM-based evaluation into the tools

designers already use. The thesis points toward embedding lightweight evaluation loops into design environments so that prompt-driven simulations, walkthrough-style checks, and hybrid validation become part of routine iteration. This would make evaluation more continuous and less episodic, while preserving the thesis principle that LLMs should augment, not replace, human-centred methods.

Finally, future work should deepen multimodal evaluation capabilities, not only by adopting more capable models, but by building better task formulations, benchmarking procedures, and datasets that allow consistent comparisons between AI-generated feedback and human findings.

12.5 Closing remarks

This thesis presented AI-augmented interaction as a spectrum, from low-level sensing that enables implicit behaviour detection to high-level language-driven workflows that support interaction engineering and design evaluation. The strongest takeaway is that usefulness comes from designing around fallibility. When systems surface uncertainty, support correction, and avoid over-automation, they can reduce user burden and speed up design work without sacrificing core HCI values such as transparency, user agency, and interpretability.

Bibliography

[1] Afaneh, M. Bluetooth low energy (ble): A complete guide. https://novelbits.io/bluetooth-low-energy-ble-complete-guide/ (2022). Accessed: 2024-10-23.

[2] Afaneh, M. Bluetooth low energy power consumption – how to achieve maximum battery life. https://novelbits.io/ble-power-consumption-optimization/ (2023). Accessed: 2025-07-17.

[3] Agency, E. E. Occupancy rates. https://www.eea.europa.eu/publications/ENVISSUENo12/page029.html (2020). Accessed: 2025-07-17.

[4] Ahmed, S. H. and Rani, S. A hybrid approach, smart street use case and future aspects for internet of things in smart cities. *Future Generation Computer Systems*, **79** (2018), 941.

[5] Aiuti, A., et al. Automating Form Completion with Large Language Models. In *Proceedings of the 16th Biannual Conference of the Italian SIGCHI Chapter*, CHItaly '25. Association for Computing Machinery, New York, NY, USA (2025). ISBN 9798400721021. Available from: https://doi.org/10.1145/3750069.3755963, doi:10.1145/3750069.3755963.

[6] Amershi, S., et al. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–13 (2019). doi:10.1145/3290605.3300233.

[7] Ansari, M. A. and Singh, D. K. Human detection techniques for real time surveillance: a comprehensive survey. *Multimedia tools and applications*, **80** (2021), 8759.

[8] Anthropic. Introducing the model context protocol (2024). Accessed: 2025-03-06. Available from: https://www.anthropic.com/news/model-context-protocol.

[9] Apple. Downloading and compiling a model on the user's device. Available from: https://developer.apple.com/documentation/coreml/downloading_and_compiling_a_model_on_the_user_s_device.

[10] Apple. Apple developer - bluetooth. https://developer.apple.com/bluetooth/ (2025). Accessed: 2025-10-14.

[11] Apple Maps. Apple Maps. https://apps.apple.com/us/app/apple-maps/id915056765 (2022). Accessed in February 2025.

[12] Argenox. Bluetooth low energy (ble) security & privacy: A 2025 guide. https://argenox.com/blog/bluetooth-low-energy-ble-security-privacy-a-2025-guide (2025). Accessed: 2025-07-17.

[13] Astarita, V., Festa, D., and Giofré, V. Mobile systems applied to traffic management and safety: a state of the art. *Procedia Computer Science*, **134** (2018), 407. doi:10.1016/j.procs.2018.07.191.

[14] Atlas, S. ChatGPT for Higher Education and Professional Development: A Guide to Conversational AI. https://digitalcommons.uri.edu/cba_facpubs/548 (2023). Accessed in February 2025.

[15] Autor, D. H. Why are there still so many jobs? the history and future of workplace automation. *Journal of economic perspectives*, **29** (2015), 3.

[16] Aveni, T. J., Fox, A., and Hartmann, B. Omnifill: Domain-agnostic form filling suggestions using multi-faceted context (2023). Available from: https://arxiv.org/abs/2310.17826, arXiv:2310.17826.

[17] Avola, D., Bottoni, P., Laureti, M., Levialdi, S., and Panizzi, E. Managing groups and group annotations in madcow. In *Proceedings of the 6th International Conference on Databases in Networked Information Systems*, DNIS'10, p. 194–209. Springer-Verlag, Berlin, Heidelberg (2010). ISBN 3642120377. Available from: https://doi.org/10.1007/978-3-642-12038-1_13, doi:10.1007/978-3-642-12038-1_13.

[18] Baddeley, A. Working memory. *Current Biology*, **20** (2010), R136. Available from: https://www.sciencedirect.com/science/article/pii/S0960982209021332, doi:https://doi.org/10.1016/j.cub.2009.12.014.

[19] Bale, A. S., Vada, Y. R., Oshiojum, B. E., Lakkineni, U. K., Rao, C., Venkatesh, K., and Rani, I. ChatGPT in Software Development: Methods and Cross-Domain Applications. *International Journal of Intelligent Systems and Applications in Engineering*, **11** (2023), 636.

[20] Bassetti, E., Berti, A., Bisante, A., Magnante, A., and Panizzi, E. Exploiting user behavior to predict parking availability through machine learning. *Smart Cities*, **5** (2022), 1243. Available from: https://www.mdpi.com/2624-6511/5/4/64, doi:10.3390/smartcities5040064.

[21] Bassetti, E., Berti, A., Bisante, A., Magnante, A., and Panizzi, E. Exploiting user behavior to predict parking availability through machine learning. *Smart Cities*, **5** (2022), 1243. Available from: https://www.mdpi.com/2624-6511/5/4/64, doi:10.3390/smartcities5040064.

[22] Bassetti, E., Cavallaccio, G., De Marsico, M., and Panizzi, E. Poster: Human presence detection after earthquakes: An ai-based implicit user interface on the smartphone. In *Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter*, CHItaly '23. Association for Computing Machinery, New York, NY, USA (2023). ISBN 9798400708060. Available from: https://doi.org/10.1145/3605390.3610832, doi:10.1145/3605390.3610832.

[23] Bassetti, E., Luciani, A., and Panizzi, E. Ml classification of car parking with implicit interaction on the driver's smartphone. In *Human-Computer Interaction – INTERACT 2021*, pp. 291–299. Springer International Publishing, Cham (2021). ISBN 978-3-030-85612-0. doi:10.1007/978-3-030-85613-7_21.

[24] Bassetti, E. and Panizzi, E. Earthquake detection at the edge: Iot crowdsensing network. *Information*, **13** (2022), 195.

[25] Ben-Elia, E., Benenson, I., Levy, N., Ashkenazi, S., and Evgeni, M. Serious game-based study of urban parking dynamics. In *XIII NECTAR International Conference* (2015). Available from: https://www.researchgate.net/publication/279971852_Serious_game-based_study_of_urban_parking_dynamics.

[26] Bennett, D., Metatla, O., Roudaut, A., and Mekler, E. How does HCI Understand Human Agency and Autonomy? In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–18. Association for Computing Machinery (2023). doi:10.1145/3544548.3580651.

[27] Biancat, J., Brighenti, C., and Brighenti, A. Review of transportation mode detection techniques. *EAI Endorsed Transactions on Ambient Systems*, **1** (2014), e7. Available from: https://www.proquest.com/scholarly-journals/review-transportation-mode-detection-techniques/docview/2305924145/se-2, doi:10.4108/amsys.1.4.e7.

[28] Bilgram, V. and Laarmann, F. Accelerating Innovation with Generative AI: AI-augmented Digital Prototyping and Innovation Methods. *IEEE Engineering Management Review*, **51** (2023), 18. doi:10.1109/EMR.2023.3272799.

[29] Bisante, A., Datla, V., Panizzi, E., Trasciatti, G., and Zeppieri, S. Enhancing Interface Design with AI: An Exploratory Study on a ChatGPT-4-Based Tool for Cognitive Walkthrough Inspired Evaluations. In *Proceedings*

of Advanced Visual Interfaces 2024, AVI '24. Association for Computing Machinery, New York, NY, USA (2024). ISBN 9798400717642. doi:10.1145/3656650.3656676.

[30] Bisante, A., Datla, V., Panizzi, E., Trasciatti, G., and Zeppieri, S. Enhancing Interface Design with AI: An Exploratory Study on a ChatGPT-4-Based Tool for Cognitive Walkthrough Inspired Evaluations. In *Proceedings of Advanced Visual Interfaces 2024*, AVI '24. Association for Computing Machinery, New York, NY, USA (2024). ISBN 9798400717642. doi:10.1145/3656650.3656676.

[31] Bisante, A., Datla, V. S. V., Trasciatti, G., Zeppieri, S., and Panizzi, E. An approach to leverage artificial intelligence for car-parking related mobile applications. In *International Symposium on Engineering Interactive Computer Systems* (edited by M. Harrison, C. Martinie, N. Micallef, P. Palanque, A. Schmidt, M. Winckler, E. Yigitbas, and L. Zaina), pp. 63–71. ACM, Swansea, Wales (2023). ISBN 978-3-031-59235-5. Available from: https://doi.org/10.1007/978-3-031-59235-5_7, doi:https://doi.org/10.1007/978-3-031-59235-5_7.

[32] Bisante, A., Datla, V. S. V., Trasciatti, G., Zeppieri, S., and Panizzi, E. An Approach to Leverage Artificial Intelligence for Car-Parking Related Mobile Applications. In *Engineering Interactive Computer Systems. EICS 2023 International Workshops and Doctoral Consortium* (edited by M. Harrison, C. Martinie, N. Micallef, P. Palanque, A. Schmidt, M. Winckler, E. Yigitbas, and L. Zaina), pp. 63–71. Springer Nature Switzerland, Cham (2024). ISBN 978-3-031-59235-5. Available from: https://doi.org/10.1007/978-3-031-59235-5_7.

[33] Bisante, A., Datla, V. S. V., Zeppieri, S., and Panizzi, E. Implicit interaction approach for car-related tasks on smartphone applications - a demo. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, AVI 2022, pp. 1–3. Association for Computing Machinery, New York, NY, USA (2022). ISBN 9781450397193. Available from: https://doi.org/10.1145/3531073.3534465, doi:10.1145/3531073.3534465.

[34] Bisante, A., Datla, V. S. V., Zeppieri, S., and Panizzi, E. Implicit Interaction Approach for Car-related Tasks On Smartphone Applications - A Demo. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, pp. 1–3 (2022). doi:10.1145/3531073.3534465.

[35] Bisante, A., Dix, A., Panizzi, E., and Zeppieri, S. To Err is AI. In *Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter*, CHItaly '23. Association for Computing Machinery, New York, NY, USA

(2023). ISBN 9798400708060. Available from: https://doi.org/10.1145/3605390.3605414, doi:10.1145/3605390.3605414.

[36] Bisante, A., Dix, A., Panizzi, E., and Zeppieri, S. To Err is AI. In *Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter*, CHItaly '23. Association for Computing Machinery, New York, NY, USA (2023). ISBN 9798400708060. doi:10.1145/3605390.3605414.

[37] Bisante, A., Dix, A., Panizzi, E., and Zeppieri, S. Implicit Interactions in Proactive Systems: Evaluation Challenges and Adaptations for Nielsen's Heuristics (2025).

[38] Bisante, A., Dix, A., Panizzi, E., and Zeppieri, S. Implicit interactions in proactive systems: Evaluation challenges and adaptations for nielsen's heuristics. In *Proceedings of the BEHAVE-AI Workshop at ACM IUI 2025*, pp. 161–169. CEUR-WS, Cagliari, Italy (2025). To appear.

[39] Bisante, A., Panizzi, E., and Zeppieri, S. Implicit interaction approach for car-related tasks on smartphone applications. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, AVI 2022, pp. 1–5. Association for Computing Machinery, New York, NY, USA (2022). ISBN 9781450397193. Available from: https://doi.org/10.1145/3531073.3531173, doi:10.1145/3531073.3531173.

[40] Bisante, A., Panizzi, E., and Zeppieri, S. Cruising-for-Parking Detection on the Smartphone Based on Implicit Interaction and Machine Learning. In *Proceedings of the 15th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pp. 93–102 (2023). doi:10.1145/3580585.3607162.

[41] Bisante, A., Zeppieri, S., Datla, V. S. V., Trasciatti, G., and Panizzi, E. Assessing large language models adoption in need finding: an exploratory study. In *Engineering Interactive Computer Systems. EICS 2024 International Workshops* (edited by L. Zaina, J. C. Campos, D. Spano, K. Luyten, P. Palanque, G. van der Veer, A. Ebert, S. R. Humayoun, and V. Memmesheimer), pp. 16–26. Springer Nature Switzerland, Cham (2025). ISBN 978-3-031-91760-8.

[42] Bisante, A., Zeppieri, S., Venkata Srikanth Varma, D., Trasciatti, G., and Panizzi, E. Assessing Large Language Models Adoption in Need Finding: an Exploratory Study (2024).

[43] Bjerre-Nielsen, A., Møller, J., et al. Inferring transportation mode from smartphone sensors: Evaluating the potential of wi-fi and bluetooth. In *PLOS ONE*, vol. 15, p. e0234003. Public Library of Science San Francisco, CA USA

(2020). Available from: https://doi.org/10.1371/journal.pone.0234003, doi:10.1371/journal.pone.0234003.

[44] Brewster, S. A., Wright, P. C., Dix, A. J., and Edwards, A. D. The sonic enhancement of graphical buttons. In *Human—Computer Interaction: Interact'95*, pp. 43–48. Springer (1995). doi:10.1007/978-1-5041-2896-4_7.

[45] Brooke, J. SUS: A 'Quick and Dirty' Usability Scale. In *Usability Evaluation in Industry*, pp. 189–194. Taylor & Francis (1996). doi:10.1201/9781498710411-35.

[46] Bsharat, S. M., Myrzakhan, A., and Shen, Z. Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4 (2024). Available from: https://arxiv.org/abs/2312.16171, arXiv:2312.16171.

[47] Bubeck, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4 (2023). Available from: https://arxiv.org/abs/2303.12712, arXiv:2303.12712.

[48] Campos, J., Kennedy, J., and Lehman, J. F. Challenges in exploiting conversational memory in human-agent interaction. In *Adaptive Agents and Multi-Agent Systems* (2018). Available from: https://api.semanticscholar.org/CorpusID:51875146.

[49] Can, B., Zaheer, A., Gabriele, P., Davide, M., Muftah, O., Eoin, O., Stephan, O., and Muhammad, K. Smart parking systems: Reviewing the literature, architecture and ways forward. *Smart Cities*, **4** (2021), 623. doi:10.3390/smartcities4020032.

[50] Carnelli, P. E., Yeh, J., Sooriyabandara, M., and Khan, A. ParkUs: A Novel Vehicle Parking Detection System. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 4650–4656 (2017). doi:10.1609/aaai.v31i2.19090.

[51] Chalmers, M., MacColl, I., and Bell, M. Seamful design: showing the seams in wearable computing. In *2003 IEE Eurowearable*, pp. 11–16 (2003). doi:10.1049/ic:20030140.

[52] Chen, R. and Wang, S. Enhancing transportation mode detection using multi-scale sensor fusion and spatial-topological attention. In *Proceedings of UbiComp 2023 Adjunct*, pp. 534–539. acm, Cancun, Quintana Roo, Mexico (2023). doi:https://doi.org/10.1145/3594739.3610751.

[53] Chiarello, F., Giordano, V., Spada, I., Barandoni, S., and Fantoni, G. Future applications of generative large language models: A data-driven case study on chatgpt. *Technovation*, **133** (2024), 103002. Available from: https://

www.sciencedirect.com/science/article/pii/S016649722400052X, doi:https://doi.org/10.1016/j.technovation.2024.103002.

[54] Cockton, G. and Gram, C. *Design Principles for Interactive Software*, vol. 1 of *IFIP Advances in Information and Communication Technology*. Springer (1996). doi:10.1007/978-0-387-34912-1.

[55] Commision, E. Building automated vehicles that are in tune with your emotions. https://projects.research-and-innovation.ec.europa.eu/en/projects/success-stories/all/building-automated-vehicles-are-tune-your-emotions (2023). Accessed: 2025-07-17.

[56] Company, P. R. Europeans without smartphones. https://www.principalrelocation.com/european-smartphones/ (2025). Accessed: 2025-07-18.

[57] Coroamă, V. C., Türk, C., and Mattern, F. Exploring the usefulness of bluetooth and wifi proximity for transportation mode recognition. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pp. 37–40. acm, London (2019). Available from: https://doi.org/10.1145/3341162.3343847, doi:10.1145/3341162.3343847.

[58] Cuadra, A., Breuch, J., Estrada, S., Ihim, D., Hung, I., Askaryar, D., Hassanien, M., Fessele, K. L., and Landay, J. A. Digital forms for all: A holistic multimodal large language model agent for health data entry. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **8** (2024). Available from: https://doi.org/10.1145/3659624, doi:10.1145/3659624.

[59] Cypher, A. Eager: Programming Repetitive Tasks By Example. *Proceedings of CHI'91*, (1991), 33. doi:10.1145/108844.108850.

[60] Dalla Chiara, G. and Goodchild, A. Do commercial vehicles cruise for parking? empirical evidence from seattle. *Transport Policy*, **97** (2020). doi:10.1016/j.tranpol.2020.06.013.

[61] Datla, V. S. V., Aiuti, A., Bisante, A., Trasciatti, G., Zeppieri, S., and Panizzi, E. Detecting Human Presence via Smartphone BLE Beaconing: Preliminary Investigations. In *Proceedings of the 24th International Conference on Mobile and Ubiquitous Multimedia*, MUM '25, pp. 483–485. Association for Computing Machinery, New York, NY, USA (2025). ISBN 9798400720154. Available from: https://doi.org/10.1145/3771882.3773958, doi:10.1145/3771882.3773958.

[62] Datla, V. S. V., Zeppieri, S., Aiuti, A., Bisante, A., Trasciatti, G., and Panizzi, E. Towards Context-Aware UX in Automated Mobility: BLE Based Passenger Detection via Smartphones. In *Proceedings of the 16th Biannual Conference of the Italian SIGCHI Chapter*, CHItaly '25. Association for Computing Machinery, New York, NY, USA (2025). ISBN 9798400721021. Available from: https://doi.org/10.1145/3750069.3750132, doi:10.1145/3750069.3750132.

[63] Datla, V. S. V., Zeppieri, S., Aiuti, A., Bisante, A., Trasciatti, G., and Panizzi, E. Towards context-aware ux in automated mobility: Ble based passenger detection via smartphones. In *Proceedings of the 16th Biannual Conference of the Italian SIGCHI Chapter (CHItaly2025)*, CHItaly '25. Association for Computing Machinery, New York, NY, USA (2025). ISBN 9798400721021. Available from: https://doi.org/10.1145/3750069.3750132, doi:10.1145/3750069.3750132.

[64] de Cossío, F. G. and Sabiron, G. A robust approach for transportation mode detection using smartphone-based gps sensors and road network information. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 959–965. IEEE, Bilbao, Bizkaia, Spain (2023). doi:10.1109/ITSC57777.2023.10422698.

[65] Dean, J. and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, **51** (2008), 107–113. doi:10.1145/1327452.1327492.

[66] Deng, Y., Lei, W., Lam, W., and Chua, T.-S. A survey on proactive dialogue systems: problems, methods, and prospects. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23 (2023). ISBN 978-1-956792-03-4. Available from: https://doi.org/10.24963/ijcai.2023/738, doi:10.24963/ijcai.2023/738.

[67] Deng, Y., Liao, L., Zheng, Z., Yang, G. H., and Chua, T.-S. Towards human-centered proactive conversational agents. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, p. 807–818. Association for Computing Machinery, New York, NY, USA (2024). ISBN 9798400704314. Available from: https://doi.org/10.1145/3626772.3657843, doi:10.1145/3626772.3657843.

[68] der Waerden, P., Timmermans, H., and Haberkorn, P. Studying Car Drivers' Parking Search Behavior using GPS Trip Loggers. In *Proceedings of the 11th International Conference on Design & Decision Support Systems, Eindhoven, the Netherlands.* (2012).

[69] Dix, A. Beyond intention - pushing boundaries with incidental interaction. In *Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing, Glasgow University, Vol. 9* (2002). Conference on Building Bridges: Interdisciplinary Context-Sensitive Computing ; Conference date: 01-01-1900.

[70] Dix, A. Activity Modelling for Low-Intention Interaction. In *The Handbook of Formal Methods in Human-Computer Interaction*, pp. 183–210. Springer (2017). doi:10.1007/978-3-319-51838-1_7.

[71] Dix, A., Beale, R., and Wood, A. Architectures to make simple visualisations using simple systems. In *Proceedings of the working conference on Advanced visual interfaces*, pp. 51–60 (2000). doi:10.1145/345513.345250.

[72] Dodigovic, M. Artificial Intelligence and Second Language Learning: An Efficient Approach to Error Remediation. *Language Awareness - LANG AWARE*, **16** (2007), 99. doi:10.2167/la416.0.

[73] Duan, P., Warner, J., and Hartmann, B. Towards Generating UI Design Feedback with LLMs. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23 Adjunct. Association for Computing Machinery, New York, NY, USA (2023). ISBN 9798400700965. doi:10.1145/3586182.3615810.

[74] EasyPark. Easypark. https://easyparkitalia.it/it (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=net.easypark.android.

[75] Ekvall, H. and Winnberg, P. Integrating ChatGPT into the UX Design Process: Ideation and Prototyping with LLMs (2023). Accessed in February 2025. Available from: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1777830&dswid=6989.

[76] El-Sheimy, N. and Li, Y. Indoor navigation: state of the art and future trends. *Satellite Navigation*, **2** (2021), 1. Available from: https://link.springer.com/article/10.1186/s43020-021-00041-3, doi:10.1186/s43020-021-00041-3.

[77] Fahim, A., Hasan, M., and Chowdhury, M. A. Smart parking systems: comprehensive review based on various aspects. *Heliyon*, **7** (2021). doi:10.1016/j.heliyon.2021.e07050.

[78] Fang, S.-H., Liao, H.-H., Fei, Y.-X., Chen, K.-H., Huang, J.-W., Lu, Y.-D., and Tsao, Y. Transportation modes classification using sensors on smartphones. *Sensors*, **16** (2016), 1324. Available from: https://www.mdpi.com/1424-8220/16/8/1324, doi:10.3390/s16081324.

[79] Fourez, T., Vidal, B., and Beghin, P. Transport mode detection on gps and accelerometer data: a temporality based workflow. In *Proceedings of ACM SIGSPATIAL*, vol. unknown, p. unknown (2023). Available from: https://hal.science/hal-04210285/.

[80] Frohlich, D., Drew, P., and Monk, A. Management of repair in human-computer interaction. *Human–Computer Interaction*, **9** (1994), 385. doi:10.1080/07370024.1994.9667211.

[81] Garrett, J. J. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders Publishing, USA, 2nd edn. (2010). ISBN 9780321683687.

[82] Girolami, M., Mavilia, F., and Delmastro, F. Sensing social interactions through ble beacons and commercial mobile devices. *Pervasive and Mobile Computing*, **67** (2020), 101198.

[83] Grobelna, I., Mailland, D., and Horwat, M. Design of automotive hmi: New challenges in enhancing user experience, safety, and security. *Applied Sciences*, **15** (2025), 5572. doi:https://doi.org/10.3390/app15105572.

[84] Grosinger, J. On proactive human-ai systems. In *International Workshop on Artificial Intelligence and Cognition* (2022). Available from: https://api.semanticscholar.org/CorpusID:258845464.

[85] Hakimov, S., Weiser, Y., and Schlangen, D. Evaluating modular dialogue system for form filling using large language models. In *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat (SCI-CHAT 2024)* (edited by Y. Graham, Q. Liu, G. Lampouras, I. Iacobacci, S. Madden, H. Khalid, and R. Qureshi), pp. 36–52. Association for Computational Linguistics, St. Julians, Malta (2024). Available from: https://aclanthology.org/2024.scichat-1.4/.

[86] Hampshire, R., Jordon, D., Akinbola, O., Richardson, K., Weinberger, R., Millard-Ball, A., and Karlin-Resnik, J. An Analysis of Parking Search Behavior using Video from Naturalistic Driving. *Transportation Research Record Journal of the Transportation Research Board*, (2017). doi:10.3141/2543-18.

[87] Hampshire, Robert and Shoup, Donald. What share of traffic is cruising for parking? *Journal of Transport Economics and Policy*, **52** (2018).

[88] Helms, K. and Brown, B. Design methods to investigate user experiences of implicit interactions. In *Proceedings of the 2016 ACM International Conference on Designing Interactive Systems*, pp. 829–840 (2016).

[89] Horvath, Z., Jenak, I., and Brachmann, F. Battery consumption of smartphone sensors. *Journal of Reliable Intelligent Environments*, **3** (2017), 131. doi:10.1109/SITIS.2015.10.

[90] Horvitz, E. Principles and challenges of mixed-initiative user interfaces. *Communications of the ACM*, **42** (1999), 74. doi:10.1145/330868.330878.

[91] Horvitz, E. Principles and Challenges of Mixed-Initiative User Interfaces. *Communications of the ACM*, **42** (1999), 74. doi:10.1145/330868.330878.

[92] Hossen, M. I., Michael, G. K. O., Connie, T., Lau, S. H., and Hossain, F. Smartphone-based context flow recognition for outdoor parking system with machine learning approaches. *ELECTRONICS*, **8** (2019), 784. Available from: https://login.ezproxy.uniroma1.it/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000482063200095&site=eds-live&scope=site, doi:10.3390/electronics8070784.

[93] Huang, C., Mees, O., Zeng, A., and Burgard, W. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10608–10615. IEEE (2023). Available from: https://dblp.org/rec/conf/icra/HuangMZB23, doi:10.1109/ICRA48891.2023.10160969.

[94] IBM Research. Memory-augmented llms: Unlocking new capabilities. https://research.ibm.com/blog/memory-augmented-LLMs (2024). Accessed: 2025-06-16.

[95] Illia, L., Colleoni, E., and Zyglidopoulos, S. Ethical implications of text generation in the age of artificial intelligence. *Business Ethics, the Environment & Responsibility*, **32** (2023), 201. doi:10.1111/beer.12479.

[96] Ji, Z., Lee, N., Fries, J., and Yu, T. Survey of hallucination in natural language generation. *ACM Computing Surveys*, (2023).

[97] Jones, M., Khan, A., Kulkarni, P., Carnelli, P., and Sooriyabandara, M. Parkus 2.0: Automated cruise detection for parking availability inference. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 2017, p. 242–251. Association for Computing Machinery, New York, NY, USA (2017). ISBN 9781450353687. Available from: https://doi.org/10.1145/3144457.3144495, doi:10.1145/3144457.3144495.

[98] Jones, M. N. and Avery, J. E. Semantic Memory. *Psychology*, (2019). Available from: https://api.semanticscholar.org/CorpusID:14184578.

[99] JustPark. Justpark. https://play.google.com/store/apps/details?id=com.justpark.jp (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=com.justpark.jp.

[100] Kessel, A.-L., Sahri, S., Groppe, S., Groppe, J., Khorashadizadeh, H., Pignal, M., Perez Pimparé, E., and Vignes-Lebbe, R. Impact of chatbots on user experience and data quality on citizen science platforms. *Computers*, **14** (2025). Available from: https://www.mdpi.com/2073-431X/14/1/21, doi:10.3390/computers14010021.

[101] Khan, I. and Khusro, S. Towards the design of context-aware adaptive user interfaces to minimize drivers' distractions. *Mob. Inf. Syst.*, **2020** (2020), 8858886:1. doi:10.1155/2020/8858886.

[102] Khan, I., Khusro, S., Ali, S., and Ahmad, J. Sensors are power hungry: An investigation of smartphone sensors impact on battery power from lifelogging perspective. *Bahria University Journal of Information & Communication Technology*, **9** (2016), 8. Available from: https://www.researchgate.net/publication/317494054_Sensors_are_Power_Hungry_An_Investigation_of_Smartphone_Sensors_Impact_on_Battery_Power_from_Lifelogging_Perspective.

[103] Kim, I. Advanced modern llm (part 1) — long-term memory-augmented large language modeling. https://medium.com/@ikim1994914/advanced-modern-llm-part-1-long-term-memory-augmented-large-language-modeling-3 (2024). Accessed: 2025-06-16.

[104] Kluivert, S. Usability: Where software testing tools fall short. https://medium.com/@dme_43393/usability-where-software-testing-tools-fall-short-6f56cbb9bf9f (2021). Accessed in February 2025.

[105] Lee, C., Park, S., Yang, T., and Lee, S.-H. Smart parking with fine-grained localization and user status sensing based on edge computing. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–5 (2019). doi:10.1109/VTCFall.2019.8891560.

[106] Lee, S., Lee, J., and Lee, K. Vehiclesense: A reliable sound-based transportation mode recognition system for smartphones. In *2017 IEEE 18th international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*, pp. 1–9. IEEE, Piscataway, New Jersey (2017). doi:10.1109/WoWMoM.2017.7974318.

[107] Lee, S., Lee, J., and Lee, K. Deepvehiclesense: An energy-efficient transportation mode recognition leveraging staged deep learning over sound

samples. *IEEE Transactions on Mobile Computing*, **22** (2022), 3270. doi:10.1109/TMC.2022.3141392.

[108] Lester, J., Hannaford, B., and Borriello, G. "are you with me?"–using accelerometers to determine if two devices are carried by the same person. In *International Conference on Pervasive Computing*, pp. 33–50. Springer (2004).

[109] Levy, N., Martens, K., and Benenson, I. Exploring cruising using agent-based and analytical models of parking. *Transportmetrica*, **9** (2012). doi:10.1080/18128602.2012.664575.

[110] Li, Y., Yang, Z., Guo, Y., and Chen, X. Droidbot: a lightweight ui-guided test input generator for android. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pp. 23–26. IEEE, Buenos Aires (2017). doi:10.1109/ICSE-C.2017.8.

[111] Li, Y., Yang, Z., Guo, Y., and Chen, X. Humanoid: A Deep Learning-based Approach to Automated Black-box Android App Testing (2020). arXiv:1901.02633.

[112] Lin, H.-Y., Kao, S.-F., and Wang, C.-C. A passenger detection and action recognition system for public transport vehicles. *Journal of Intelligent & Robotic Systems*, **110** (2024), 155. doi:https://doi.org/10.1007/s10846-024-02194-0.

[113] Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Che, X., Wang, D., and Wang, Q. Chatting with GPT-3 for Zero-Shot Human-Like Mobile Automated GUI Testing (2023). arXiv:2305.09434.

[114] Lookback. Lookback. https://www.lookback.com/ (2024).

[115] Ma, Y., Guan, X., Cao, J., and Wu, H. A multi-stage fusion network for transportation mode identification with varied scale representation of gps trajectories. *Transportation Research Part C: Emerging Technologies*, **150** (2023), 104088. doi:https://doi.org/10.1016/j.trc.2023.104088.

[116] Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. Evaluating very long-term conversational memory of llm agents (2024). Available from: https://arxiv.org/abs/2402.17753, arXiv:2402.17753.

[117] Mahfooz Ul Haque, H., Zulfiqar, H., Khan, S., and Haque, M. *Context-Aware Parking Systems in Urban Areas: A Survey and Early Experiments: 7th EAI International Conference, ICCASA 2018, and 4th EAI International Conference, ICTCC 2018, Viet Tri City, Vietnam, November 22–23, 2018, Proceedings*, pp. 25–35. Springer International Publishing, Cham (2019). ISBN 978-3-030-06151-7. doi:10.1007/978-3-030-06152-4_3.

[118] Mannini, L., Cipriani, E., Crisalli, U., Gemma, A., and Vaccaro, G. On-Street Parking Search Time Estimation Using FCD Data. *Transportation Research Procedia*, **27** (2017), 929. doi:10.1016/j.trpro.2017.12.149.

[119] Maps, G. Google maps. https://play.google.com/store/apps/details?id=com.google.android.apps.maps (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=com.google.android.apps.maps.

[120] Marsh, E. J. and Roediger, H. L. Episodic and Autobiographical Memory (2003). Available from: https://api.semanticscholar.org/CorpusID:150580462.

[121] Martin, L., Whitehouse, N., Yiu, S., Catterson, L., and Perera, R. Better call gpt, comparing large language models against lawyers. *ArXiv*, **abs/2401.16212** (2024). Available from: https://api.semanticscholar.org/CorpusID:267312546.

[122] Martinie, C. and Palanque, P. Task models based gameful design as a mean to increase engagement with automation. In *Workshop on Engaging with Automation (AutomationXP 2022) co-located CHI 2022*, no. Session 1: Engaging with Automation: Common Challenges in 3154. CEUR-WS. org (2022).

[123] Meara, P., Sperber, D., and Wilson, D. Relevance: Communication and cognition. *Modern Language Review*, **84** (1989), 894. Available from: https://api.semanticscholar.org/CorpusID:143399773.

[124] Meurisch, C., Mihale-Wilson, C. A., Hawlitschek, A., Giger, F., Müller, F., Hinz, O., and Mühlhäuser, M. Exploring User Expectations of Proactive AI Systems. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, **4** (2020). Available from: https://doi.org/10.1145/3432193, doi:10.1145/3432193.

[125] MICS Consortium. Made in italy circolare e sostenibile (mics). https://www.mics.tech/en/home (2025). Accessed May 2025.

[126] Millard-Ball, A., Hampshire, R., and Weinberger, R. Parking behaviour: The curious lack of cruising for parking in san francisco. *Land Use Policy*, **91** (2020), 103918. doi:10.1016/j.landusepol.2019.03.031.

[127] Mohammed, M. A., Lakhan, A., Abdulkareem, K. H., Zebari, D. A., Nedoma, J., Martnek, R., Kadry, S., and Garcia-Zapirain, B. Homomorphic federated learning schemes enabled pedestrian and vehicle detection system. *Internet of Things*, **23** (2023), 100903. doi:https://doi.org/10.1016/j.iot.2023.100903.

[128] Mokayed, H., Quan, T. Z., Alkhaled, L., and Sivakumar, V. Real-time human detection and counting system using deep learning computer vision techniques. In *Artificial Intelligence and Applications*, vol. 1, pp. 205–213 (2023). doi:https://doi.org/10.47852/bonviewAIA2202391.

[129] Montanari, A., Nawaz, S., Mascolo, C., and Sailer, K. A study of bluetooth low energy performance for human proximity detection in the workplace. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 90–99. IEEE (2017). doi:10.1109/PERCOM.2017.7917855.

[130] Montini, L. Extraction of transportation information from combined position and accelerometer tracks (2016). Available from: https://www.research-collection.ethz.ch/handle/20.500.11850/120639.

[131] Montini, L., Horni, A., Rieser, N., Kay, S., and Axhausen, K. Searching for parking in gps data. In *Proceedings of the 13th International Conference on Travel Behaviour Research* (2012). Available from: https://www.research-collection.ethz.ch/handle/20.500.11850/49659.

[132] Moreno, A. M., Seffah, A., Capilla, R., and Sanchez-Segura, M.-I. Hci practices for building usable software. *Computer*, **46** (2013), 100.

[133] Mueller, F., Semertzidis, N., Andres, J., Marshall, J., Benford, S., Li, X., Matjeka, L., and Mehta, Y. Toward Understanding the Design of Intertwined Human–Computer Integrations. *ACM Transactions on Computer-Human Interaction*, **30** (2023), 1. doi:10.1145/3590766.

[134] Münch, M., Huffstadt, K., and Schleif, F.-M. Towards a device-free passive presence detection system with bluetooth low energy beacons. In *ESANN* (2019).

[135] my parked car Automatically locate car, F. Find my parked car. https://play.google.com/store/apps/details?id=com.ofirmiron.findmycarandroidwear (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=com.ofirmiron.findmycarandroidwear.

[136] Nawaz, S., Efstratiou, C., and Mascolo, C. Parksense: a smartphone based sensing system for on-street parking. In *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pp. 75–86 (2013). doi:10.1145/2500423.2500438.

[137] Nielsen, J. Heuristic Evaluation. In *Usability Inspection Methods* (edited by J. Nielsen and R. L. Mack), chap. 2, pp. 25–62. John Wiley & Sons, New York (1994). doi:10.5555/189200.189209.

[138] Nigam, G. A complete guide to implementing memory-augmented rag. https://medium.com/aingineer/a-complete-guide-to-implementing-memory-augmented-rag-c3582a8dc74f (2024). Accessed: 2025-07-11.

[139] Nikolic, M. and Bierlaire, M. Review of transportation mode detection approaches based on smartphone data. In *17th Swiss Transport Research Conference*, CONF, p. unknown. unknwon, Monte Verità / Ascona (2017). Available from: http://infoscience.epfl.ch/record/229181.

[140] Nikouei, S. Y., Chen, Y., Song, S., Xu, R., Choi, B.-Y., and Faughnan, T. R. Real-time human detection as an edge service enabled by a lightweight cnn. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pp. 125–129. IEEE (2018). doi:10.1109/EDGE.2018.00025.

[141] Norman, D. A. *The Psychology of Everyday Things*. Basic Books (1988). ISBN 978-0-465-06710-7. Originally published in 1988; revised and expanded edition published in 2013 as *The Design of Everyday Things*.

[142] Norman, D. A. *The Design of Everyday Things*. Basic Books, Inc., USA (2002). ISBN 9780465067107.

[143] Nothdurft, F., Ultes, S., and Minker, W. Finding appropriate interaction strategies for proactive dialogue systems—an open quest (2015). Available from: https://api.semanticscholar.org/CorpusID:41174018.

[144] Noy, S. and Zhang, W. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, **381** (2023), 187. Available from: https://www.science.org/doi/abs/10.1126/science.adh2586, arXiv:https://www.science.org/doi/pdf/10.1126/science.adh2586, doi:10.1126/science.adh2586.

[145] Ong, C. Y. The ultimate guide to android bluetooth low energy. https://punchthrough.com/android-ble-guide/ (2024). Accessed: 2024-10-23.

[146] OpenAI. Hello gpt-4o (2024). Accessed May 2025. Available from: https://openai.com/index/hello-gpt-4o.

[147] OpenAI. Introducing GPTs. https://openai.com/blog/introducing-gpts (2024). Accessed in February 2025.

[148] Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. Memgpt: Towards llms as operating systems (2024). Available from: https://arxiv.org/abs/2310.08560, arXiv:2310.08560.

[149] Paidi, V., Fleyeh, H., Håkansson, J., and Nyberg, R. Smart parking sensors, technologies and applications for open parking lots: A review. *IET Intelligent Transport Systems*, **12** (2018). doi:10.1049/iet-its.2017.0406.

[150] Paidi, V., Fleyeh, H., Håkansson, J., and Nyberg, R. Tracking vehicle cruising in an open parking lot using deep learning and kalman filter. *Journal of Advanced Transportation*, **2021** (2021), 1. doi:10.1155/2021/1812647.

[151] Panizzi, E. The seismocloud app: Your smartphone as a seismometer. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, p. 336–337. Association for Computing Machinery, New York, NY, USA (2016). ISBN 9781450341318. Available from: https://doi.org/10.1145/2909132.2926070, doi:10.1145/2909132.2926070.

[152] Panizzi, E. and Bisante, A. Private or public parking type classifier on the driver's smartphone. *Procedia Computer Science*, **198** (2022), 231. 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare. Available from: https://www.sciencedirect.com/science/article/pii/S1877050921024728, doi:https://doi.org/10.1016/j.procs.2021.12.233.

[153] Panizzi, E., Bisante, A., Zeppieri, S., Aiuti, A., Trasciatti, G., and Datla, V. S. V. Nitsy Societa' a Responsabilita' Limitata. https://iris.uniroma1.it/handle/11573/1757140 (2025).

[154] Panizzi, E. and Calvitti, D. A framework to enhance the user experience of car mobile applications. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '18, p. 245–252. Association for Computing Machinery, New York, NY, USA (2018). ISBN 9781450359412. Available from: https://doi.org/10.1145/3236112.3236146, doi:10.1145/3236112.3236146.

[155] Panizzi, E., Trasciatti, G., Bisante, A., Zeppieri, S., Datla, V. S. V., Aiuti, A., and Antonelli, L. Metodo e dispositivo mobile per la localizzazione su mappa (2025). Available from: https://iris.uniroma1.it/handle/11573/1749756.

[156] Parasuraman, R., Sheridan, T., and Wickens, C. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, **30** (2000), 286. doi:10.1109/3468.844354.

[157] Parking, P. Parkopedia parking. https://play.google.com/store/apps/details?id=com.parkopedia (2022). Accessed in February 2025.

Available from: https://play.google.com/store/apps/details?id=com.parkopedia.

[158] Parking, P. Passport parking. https://play.google.com/store/apps/details?id=com.passportparking.mobile (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=com.passportparking.mobile.

[159] Parkster. Parkster. https://play.google.com/store/apps/details?id=se.parkster.client.android (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=se.parkster.client.android.

[160] Pastore, M. T., Zhou, Y., and Yost, W. A. Cross-Modal and Cognitive Processes in Sound Localization (2020). Available from: https://api.semanticscholar.org/CorpusID:226421664.

[161] Paul, M., Haque, S. M., and Chakraborty, S. Human detection in surveillance videos and its applications-a review. *EURASIP Journal on Advances in Signal Processing*, **2013** (2013), 1. doi:https://doi.org/10.1186/1687-6180-2013-176.

[162] Paul, R. State of memory-augmented language models. https://www.rohan-paul.com/p/state-of-memory-augmented-language (2024). Accessed: 2025-06-16.

[163] PayByPhone. Paybyphone. https://play.google.com/store/apps/details?id=com.paybyphone (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=com.paybyphone.

[164] Peter, M., Haala, N., Schenk, M., and Otto, T. Indoor navigation and modeling using photographed evacuation plans and MEMS IMU. In *Geospatial Data and Geovisualization: Environment, Security and Society: Special Joint Symposium of ISPRS Technical Commission IV and AutoCarto 2010 in Conjunction with ASPRS-CaGIS 2010 Fall Specialty Conference*. Orlando, FL, USA (2010). Conference contribution.

[165] Polak, J. and Axhausen, K. Parking search behaviour: A review of current research and future prospects. (1990).

[166] Polson, P. G., Lewis, C., Rieman, J., and Wharton, C. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, **36** (1992), 741. Available from: https://www.sciencedirect.com/science/article/pii/002073739290039N, doi:10.1016/0020-7373(92)90039-N.

[167] Prelipcean, A. C., Gidófalvi, G., and Susilo, Y. O. Transportation mode detection - an in-depth review of applicability and reliability. *Transport reviews*, **37** (2017), 442. Available from: https://doi.org/10.1080/01441647.2016.1246489, doi:10.1080/01441647.2016.1246489.

[168] Qin, H., Pang, Q., Yu, B., and Wang, Z. Analysis on cruising process for on-street parking using an spectral clustering method. *IET Intelligent Transport Systems*, **14** (2020). doi:10.1049/iet-its.2020.0459.

[169] Reason, J. *Human error*. Cambridge university press (1990).

[170] RingGo. Ringgo. https://play.google.com/store/apps/details?id=co.uk.ringgo.android (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=co.uk.ringgo.android.

[171] Russell, S., Moskowitz, I. S., and Raglin, A. *Human Information Interaction, Artificial Intelligence, and Errors*, pp. 71–101. Springer International Publishing, Cham (2017). ISBN 978-3-319-59719-5. Available from: https://doi.org/10.1007/978-3-319-59719-5_4, doi:10.1007/978-3-319-59719-5_4.

[172] SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE international*, **4970** (2018), 1.

[173] SAE International. SAE Levels of Driving Automation™ Refined for Clarity and International Audience. SAE Blog, posted: Monday, May 3, 2021, https://www.sae.org/blog/sae-j3016-update (2021). Accesse in February 2025.

[174] Sahoo, B., Shahjad, and Tanwar, P. Real-time smart parking: Challenges and solution using machine learning and iot. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, (2021), 451. doi:10.32628/CSEIT217295.

[175] Salminen, J., Kwak, H., Santos, J. a. M., Jung, S.-G., An, J., and Jansen, B. J. Persona perception scale: Developing and validating an instrument for human-like representations of data. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, p. 1–6. Association for Computing Machinery, New York, NY, USA (2018). ISBN 9781450356213. Available from: https://doi.org/10.1145/3170427.3188461, doi:10.1145/3170427.3188461.

[176] Salpietro, R., Bedogni, L., Di Felice, M., and Bononi, L. Park here! a smart parking system based on smartphones' embedded sensors and short range

communication technologies. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 18–23 (2015). doi:10.1109/WF-IoT.2015.7389020.

[177] Sankaran, K., Zhu, M., Guo, X. F., Ananda, A. L., Chan, M. C., and Peh, L.-S. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pp. 191–205. acm, Memphis, TN (2014). doi:https://doi.org/10.1145/2668332.2668343.

[178] Sathyamoorthy, A. J., Weerakoon, K., Elnoor, M., Zore, A., Ichter, B., Xia, F., Tan, J., Yu, W., and Manocha, D. CoNVOI: Context-aware navigation using vision language models in outdoor and indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13837–13844. IEEE (2024). Available from: https://dblp.org/rec/conf/iros/SathyamoorthyWE24, doi:10.1109/IROS58592.2024.10802716.

[179] Schank, R. C. and Abelson, R. P. Knowledge and Memory: The Real Story (1995). Available from: https://api.semanticscholar.org/CorpusID:142804819.

[180] Schmidt, A. Implicit Human Computer Interaction Through Context. *Personal Technologies*, **4** (1999), 191. doi:10.1007/BF01324126.

[181] Schmidt, A. Speeding Up the Engineering of Interactive Systems with Generative AI. In *Companion Proceedings of the 2023 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pp. 7–8. EICS 23, Swansea United Kingdom (2023). doi:https://doi.org/10.1145/3596454.3597176.

[182] Schmidt, A., Elagroudy, P., Draxler, F., Kreuter, F., and Welsch, R. Simulating the Human in HCD with ChatGPT: Redesigning Interaction Design with AI. *Interactions*, **31** (2024), 24–31. Available from: https://doi.org/10.1145/3637436.

[183] Schmidt, A. and Herrmann, T. Intervention user interfaces: a new interaction paradigm for automated systems. *Interactions*, **24** (2017), 40. doi:10.1145/3121357.

[184] Semiconductor, N. The complete guide to bluetooth low energy. https://response.nordicsemi.com/the-complete-guide-to-bluetooth-low-energy (2024). Accessed: 2024-10-23.

[185] Serim, B. and Jacucci, G. *Explicating "Implicit Interaction": An Examination of the Concept and Challenges for Research*, p. 1–16. CHI

'19. Association for Computing Machinery, New York, NY, USA (2019). ISBN 9781450359702. Available from: https://doi.org/10.1145/3290605.3300647, doi:10.1145/3290605.3300647.

[186] Shah, D., Osiński, B., Ichter, B., and Levine, S. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proceedings of The 6th Conference on Robot Learning (CoRL)*, vol. 205 of *Proceedings of Machine Learning Research*, pp. 492–504 (2023). Available from: https://proceedings.mlr.press/v205/.

[187] Shahbazian, R. and Trubitsyna, I. Human sensing by using radio frequency signals: A survey on occupancy and activity detection. *IEEE Access*, **11** (2023), 40878.

[188] Shneiderman, B. Human-centered ai: A new framework for understanding and designing ai systems. In *Human-Centered AI*, pp. 1–19. Oxford University Press (2020). doi:10.1093/oxfordhb/9780190882887.013.3.

[189] Shneiderman, B. *Human-centered AI*. Oxford University Press (2022).

[190] Shneiderman, B. *Human-centered AI*. Oxford University Press (2022).

[191] Shneiderman, B., Plaisant, C., Cohen, M. S., Jacobs, S., Elmqvist, N., and Diakopoulos, N. *Designing the user interface: strategies for effective human-computer interaction*. Pearson (2016).

[192] Shoup, D. C. Cruising for parking". *Transport Policy*, **13** (2006), 479. Parking. Available from: http://www.sciencedirect.com/science/article/pii/S0967070X06000448, doi:https://doi.org/10.1016/j.tranpol.2006.05.005.

[193] Software, F. L. Bluetooth le: ios vs. android. https://firstlinesoftware.com/blog/bluetooth-low-energy/ (2016). Accessed: 2025-07-17.

[194] Sowlati, S., Ali Abbaspour, R., and Chehreghan, A. An approach to assess the role of features in detection of transportation modes. *Transportation*, **unknown** (2024), 1. doi:https://doi.org/10.1007/s11116-024-10492-7.

[195] Stenneth, L., Wolfson, O., Xu, B., and Yu, P. S. Phonepark: Street parking using mobile phones. In *2012 IEEE 13th International Conference on Mobile Data Management*, pp. 278–279 (2012). doi:10.1109/MDM.2012.76.

[196] Stenneth, L., Wolfson, O., Yu, P., and Xu, B. Transportation mode detection using mobile phones and gis information. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pp. 54–63 (2011). doi:10.1145/2093973.2093982.

[197] Tabone, W. and De Winter, J. Using ChatGPT for human-computer interaction research: A primer. *Manuscript submitted for publication*, (2023). doi:https://doi.org/10.1098/rsos.231053.

[198] Tan, H., Zhang, Z., Ma, C., Chen, X., Dai, Q., and Dong, Z. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. In *Annual Meeting of the Association for Computational Linguistics* (2025). Available from: https://api.semanticscholar.org/CorpusID:280011015.

[199] Teixeira, T., Dublon, G., and Savvides, A. A survey of human-sensing: Methods for detecting presence, count, location, track, and identity. *ACM Computing Surveys*, **5** (2010), 59.

[200] Tennenhouse, D. Proactive computing. *Commun. ACM*, **43** (2000), 43–50. Available from: https://doi.org/10.1145/332833.332837, doi:10.1145/332833.332837.

[201] Trasciatti, G., Zeppieri, S., Datla, D. V. S. V., Chambers, A., and Panizzi, E. Investigating out-of-the-box indoor localization using large language models and evacuation maps. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)* (2026). Accepted at PerCom 2026, to be published.

[202] Tsiktsiris, D., Lalas, A., Dasygenis, M., and Votis, K. A complete in-cabin monitoring framework for autonomous vehicles in public transportation. *IET Intelligent Transport Systems*, **19** (2025), e12612. doi:https://doi.org/10.1049/itr2.12612.

[203] Uğur, N. G. and Turan, A. H. Mobile applications acceptance: a theoretical model proposal and empirical test. *International Journal of E-Adoption (IJEA)*, **11** (2019), 13.

[204] Usertesting. UserTesting human insight platform. https://www.usertesting.com/ (2024). Accessed in February 2025.

[205] van Berkel, N., Skov, M. B., and Kjeldskov, J. Human-ai interaction: intermittent, continuous, and proactive. *Interactions*, **28** (2021), 67–71. Available from: https://doi.org/10.1145/3486941, doi:10.1145/3486941.

[206] Van Der Donckt, J., Van Der Donckt, J., and Van Hoecke, S. Magnitude and rotation invariant detection of transportation modes with missing data modalities. *arXiv preprint arXiv:2407.11048*, **unkown** (2024), 597. doi:https://doi.org/10.48550/arXiv.2407.11048.

[207] Vanhaeverbeke, J., Deprost, E., Verstockt, S., and Van Hoecke, S. Cross-room co 2-based presence detection for occupancy profiling. *IEEE Access*, (2025). doi:10.1109/ACCESS.2025.3588031.

[208] Villena-Román, J., Collada-Pérez, S., Lana-Serrano, S., and González, J. C. Hybrid Approach Combining Machine Learning and a Rule-Based Expert System for Text Categorization. In *The Florida AI Research Society* (2011). Available from: https://api.semanticscholar.org/CorpusID:15122710.

[209] Waerden, P., Timmermans, H., and Hove, L. Gps data and car drivers' parking search behavior in the city of turnhout, belgium. *Lecture Notes in Geoinformation and Cartography*, **214** (2015), 247. doi:10.1007/978-3-319-11463-7_18.

[210] Wang, J., Ming, Y., Shi, Z., Vineet, V., Wang, X., Li, Y., and Joshi, N. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. *Advances in Neural Information Processing Systems*, **37** (2024), 75392. Available from: https://arxiv.org/abs/2406.14852.

[211] Waze. Waze. https://play.google.com/store/apps/details?id=com.waze (2022). Accessed in February 2025. Available from: https://play.google.com/store/apps/details?id=com.waze.

[212] Wei, W., Zhou, B., and Leontidis, G. A hybrid natural language generation system integrating rules and deep learning algorithms (2020). Available from: https://arxiv.org/abs/2006.09213, arXiv:2006.09213.

[213] Weinberger, R., Millard-Ball, A., and Hampshire, R. Parking-Cruising Caused Congestion. *SSRN Electronic Journal*, (2016). Available from: AvailableatSSRN:https://ssrn.com/abstract=2906528orhttp://dx.doi.org/10.2139/ssrn.2906528, doi:10.2139/ssrn.2906528.

[214] Weinberger, R., Millard-Ball, A., and Hampshire, R. Parking search caused congestion: Where's all the fuss? *Transportation Research Part C: Emerging Technologies*, **120** (2020), 102781. doi:10.1016/j.trc.2020.102781.

[215] Wen, H., Wang, H., Liu, J., and Li, Y. DroidBot-GPT: GPT-powered UI Automation for Android (2024). arXiv:2304.07061.

[216] Wharton, C., Rieman, J., Lewis, C., and Polson, P. *The Cognitive Walkthrough Method: A Practitioner's Guide*, p. 105–140. John Wiley & Sons, Inc., USA (1994). ISBN 0471018775.

[217] Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *ArXiv*, **abs/2410.10813** (2024). Available from: https://api.semanticscholar.org/CorpusID:273345961.

[218] Wu, Y., Liang, S., Zhang, C., Wang, Y., Zhang, Y., Guo, H., Tang, R., and Liu, Y. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*, (2025). Available from: https://arxiv.org/abs/2504.15965, doi:10.48550/arXiv.2504.15965.

[219] Xu, X. Automatic classification of transportation modes using smartphone sensors: addressing imbalanced data and enhancing training with focal loss and artificial bee colony algorithm. *Journal of Optics*, **unknown** (2024), 1. doi:https://doi.org/10.1007/s12596-024-01703-6.

[220] Yoon, J., Feldt, R., and Yoo, S. Autonomous Large Language Model Agents Enabling Intent-Driven Mobile GUI Testing (2023). arXiv:2311.08649.

[221] York, E. Evaluating ChatGPT: Generative AI in UX Design and Web Development Pedagogy. In *Proceedings of the 41st ACM International Conference on Design of Communication*, pp. 197–201. 41st ACM International Conference on Design of Communication, Orlando FL USA (2023). doi:https://doi.org/10.1145/3615335.3623035.

[222] Zeppieri, S. MMAG: Mixed Memory-Augmented Generation for Large Language Models Applications (2025). Available from: https://arxiv.org/abs/2512.01710.

[223] Zeppieri, S. Transforming Interactive Systems with Large Language Models: Accelerating Interface Design and Evaluation. In *Proceedings of the CHItaly 2025 Doctoral Consortium*, CEUR Workshop Proceedings (2025).

[224] Zeppieri, S., Aiuti, A., Bisante, A., Datla, V. S. V., Trasciatti, G., and Panizzi, E. Engineering Large Language Model Agents for Transforming Unstructured Descriptions into Structured Input. In *Proceedings of the EISEAIT 2025 Workshop (co-located with EICS 2025)*, Lecture Notes in Computer Science (LNCS). Springer (2025).

[225] Zeppieri, S., Aiuti, A., Bisante, A., Datla, V. S. V., Trasciatti, G., and Panizzi, E. Engineering large language model agents for transforming unstructured descriptions into structured input. In *Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '25)*, Lecture Notes in Computer Science (LNCS). Springer (2025). To appear. Available from: https://eics.acm.org/2025/.

[226] Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y. Memorybank: Enhancing large language models with long-term memory (2023). Available from: https://arxiv.org/abs/2305.10250, arXiv:2305.10250.

[227] Zhou, R., Lu, X., Zhao, P., and Chen, J. Device-free presence detection and localization with svm and csi fingerprinting. *IEEE Sensors Journal*, **17** (2017), 7990. doi:10.1109/JSEN.2017.2762428.

[228] Zulfiqar, H., Ul Haque, H. M., Tariq, F., and Khan, R. M. A survey on smart parking systems in urban cities. *CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE*, (2021). doi:10.1002/cpe.6511.