Master thesis - Fostering Appropriate Trust in Agentic Chatbots for Enterprise Use

Max Adolfsson

May 21, 2026

Abstract

This thesis investigates challenges of developing agentic AI for fostering trust in enterprise users. Using Research through Design (RtD) as a methodological framework, an LLM-driven chatbot was iteratively designed and integrated into a web-based system for managing flexible office space, with the aim of automating data retrieval and analysis through natural language interaction. Two design iterations were developed and evaluated through semi-structured interviews, live chatbot trials, and trace-based analysis of agent behavior. The studies involved enterprise users with direct operational responsibility on the platform, providing empirical grounding for identifying trust-related design challenges. Three interconnected challenges emerged from the process. First, a probabilistic/deterministic boundary: LLMs operating as probabilistic systems must produce strictly typed, correct database queries, requiring architectural decisions that progressively constrain where non-deterministic behavior can corrupt factual outputs. Second, a domain-model gap: users interact via natural language rooted in personal mental models and informal terminology, while the agent depends on exact entity names and structured filters — a structural mismatch that complicates relying on database queries. Third, early-stage trust was found to depend disproportionately on the consistent, verifiable correctness of simple operations; isolated failures on basic tasks damaged perceived reliability more than successes on complex ones could build it. These findings are discussed through established trust literature. Finally a teacher-pupil/manager-engineer framework is proposed to describe the stages of trust in AI deployment. The thesis concludes that verifiability-first design, transparent reasoning communication, and architectural containment of probabilistic behavior are prerequisite conditions for fostering long-term use of domain-specific agentic chatbots.
Keywords: agentic AI, trust in AI, LLM applications, enterprise chatbot, Research through Design, human-AI interaction

Acknowledgments

I want to thank my supervisor, Sofia, for being an invaluable, persistent support, and helping me narrow down the direction for this thesis. I would also like to thank my examiner, Michael Heron, for his truly inspiring feedback throughout the work. A special thanks my mentor at Flowpass, Tomas Lundborg, for introducing me to the topics which inspired this thesis project. I highly appreciate the time made for brainstorming on the project, mapping out limitations, and connecting me with the user base. This has been a greatly rewarding project to work on.

Abbreviations

- LLM - Large Language Model
- NLP - Natural Language Processing
- GUI - Graphical User Interface
- UI - User Interface
- GenAI - Generative Artificial Intelligence
- RtD - Research through Design
- DNN - Deep Neural Network
- XAI - eXplainable AI

Contents

List of Figures

List of Tables

1 Introduction

Agentic AI (Artificial Intelligence) is a young category of Large Language Model (LLM) applications that has reached notable interest in an array of use cases. Emerged examples include Coding Agents, helping software developers by handling entire projects and generating code to build desired features[1], as well as healthcare assistans, currently being deployed in clinics to automate the transcribing of patient interactions and filling out medical journals, potentially allowing more time for practitioners to be spent in interaction with patients[2]. Other examples are customer service agents for e-commerce[3],[4] and CRM-systems, automating workflows to save resources in sales work[5].

As Agentic AI applications set out to automate enterprise processes, a new category of interaction emerges, requiring users to trust in the abilities of predictive, non-deterministic applications. Traditional research on automation emphasizes this trust as 'double-edged sword' as both over- and under-reliance can have severely negative consequences (Lee and See, 2004; Hancock et al., 2011). Appropriate reliance in Agentic AI is achieved when users understand LLM systems' reliabilities, on one hand, and uncertainties, on the other. Designing interactions to foster this relationship has seemingly not been documented much, as of writing.

This thesis is done in collaboration with Flowpass, a web platform for flexible hosting and renting of offices and co-working spaces. The customers include companies looking for flexible on-demand office solutions, on one end, and owners of offices looking to cut costs from unused workspaces by reaching a wide base of customers. Involved companies may have both of these roles simultaneously.

The platform is made up by two interfaces:

- A mobile app where workers can browse through and book available setups, save favorites, follow co-workers and join their bookings, and get news on workspaces, posted by partnering hosts.
- A web page for admins, where their company's setup can be overviewed and managed. Host companies here publish workspace setups to be booked, track revenues, and access all the other data/configuration. Guest companies can here configure spending limits and access restrictions for specific groups of employees, and more. For both roles, various data on ones usage can be viewed, and allow admins to make configurations for optimizing their Flowpass use.

The original admin web page contains a side navigation bar with separate menus for host and organization roles. By navigating these, host admins can manage access policies, monitor workspace usage, and configure their listed

[1] Claude Code
[2] Tandem Health AI
[3] Zalando AI
[4] Zalando AI, About
[5] Lime CRM AI, About

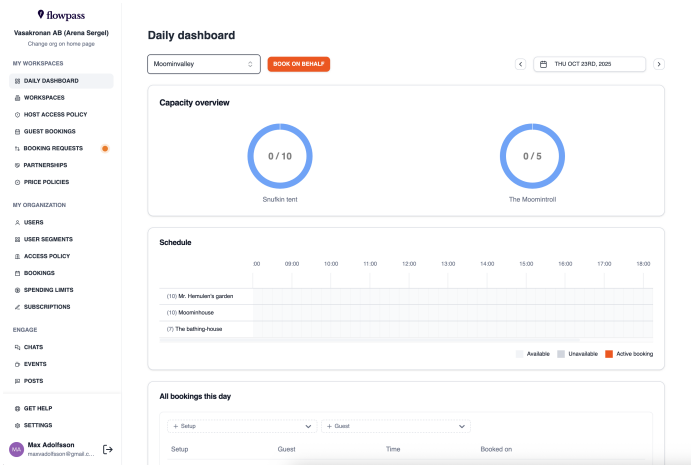


Figure 1: Flowpass Admin GUI

locations. Guest admins can manage the companies internal user base, set spending limits, and monitor the bookings made. The page also provides role-specific pages with statistics on revenue/spendings and unique users in certain time-frames, and more. Views like 'Daily Dashboard' (see Figure 1) show the bookings for different workspaces. For statistics such as revenue (or spendings), total bookings, there are views for Hosts and Guest Organizations, allowing various filtering. More complex statistics requires the user to manually alter filtering and manually note different strings values.

This marks an inherent limitation for statistics dashboards, in general. As a popular example, Gapminder[6] offers a variety off diagrams and combining a range of metrics. This supports an exploratory learning, and can give rich insights stably. However, getting the data visualization to answer concrete questions of some complexity requires multiple altering of settings and filters, and possibly even exporting the data to program calculations. With Agentic AI, such processes could autonomized — given that users can and want to rely on it.

This thesis' project explores the design process behind reliable Agentic LLM applications for data retrieval and aggregation in an enterprise environment. The project includes integrating an LLM-driven chatbot into the Flowpass platform, with tools that make backend API calls to fetch data for analysis simple look-ups. Sustainable and productive use of the system this requires designing for appropriate trust.

[6] Gapminder Tools

1.1 Aim

The thesis aims to contribute to knowledge of designing for appropriate trust in enterprise LLM applications. This includes building a chatbot with stable performance in tasks like:

- Provide narrowly filtered data, for example through the prompt 'Suggest a workspace in Stockholm offering a meeting-rooms for more than 7 people.'
- Calculating statistics grouped by a few types of data, for instance: 'What's the utilization of published setups in building X? Which kind of setup is the most popular here, private rooms or desks in the coworking space?'

The evaluation of the chatbot should answer which User Interface (UI) features and agent behavior that foster user adoption achieves appropriate trust. Evaluations will be done through user studies, combined literature on trust in AI in automation traditionally.

1.2 Research Question

Through the design process of a chatbot on the Flowpass platform, this thesis explores:

- What are the main challenges with developing Agentic AI for fostering trust in users?

1.3 Stakeholders

The primary stakeholders are Flowpass's customers. The customers are companies in need of flexible office space solutions. This includes both companies who wants to rent out space to cut costs, and companies who needs a temporary work setup on short notice. The types of businesses involved stem from real estate, automotive, consultant, and other categories. Affected users of this project include the admin page managers, which is a small group at each company. For companies only on Flowpass with the guest role, the admin is responsible for staying up to date with the workspace needs of co-workers and extracting any relevant insights from their booking history, spendings, and other metrics. On the other hand, companies with the host role may care about occupancy rates, revenues, applying price- and access policies to their varying customers. This group may also address complex, strategic problems, such as 'will the allowance of Company X into Workspace Y overcrowd it and make it hard to book?'

For both roles, the Flowpass setup could raise the demand of a modern data analysis method, or optionally having an easier way to extract raw data for analysis in third-party applications.

1.4 Delimitations

The project does not involve fine-tuning of LLMs for specific tasks of the Flowpass platform. While this could in theory adapt a model for the Flowpass platform from the ground-up, practices for managing the input context for general-purpose LLMs have proved to be more efficient, as is described in Section 2.

2 Background

As a background, the history and theory of LLM applications and AI agents will be presented. This is followed by various proven practices for managing input context and getting the most valuable outputs.

2.1 Historical context

The term Artificial Intelligence was coined in 1955, and has since sparked cycles of excitement and drawback, the latter being known as "AI winters". Natural language processing (NLP), a core field behind todays chatbots had an early breakthrough in the late 1960s with the therapist chatbot ELIZA (Weizenbaum, 1966). ELIZA was able to deterministically generate responses similar to those of Rogerian therapist, creating an early experience of empathy from computers. NLP had further breakthrough in the 1970s with SHRDLU, an application allowing the user to move around blocks in a small virtual 3D environment, steered by natural text input. After the second AI winter, in 1986, a technical breakthrough was made with the back-propagation algorithm, a method for training neural networks still relevant today. Although the training was also made possible by advancements in computing power. In 1997, IBM's supercomputer DeepBlue beat the world champion of chess, Gary Kasparov, an early milestone in the comparison of human and AI intelligence (Russell and Norvig, 2021, ch 1).

In 2015, OpenAI was founded as a non-profit organization dedicated to researching novel AI technology and applications, with billions of dollars in venture capital from large tech organizations. They released GPT-1 in 2018, an early instance of an LLM powered by the transformer architecture, coined in a paper a year earlier (Vaswani et al., 2017). GPT-1 hence demonstrated potential to train models on large amounts of text, and fine-tuning it for specific applications, e.g. chatbots. A year later, scale of training was further increased with the GPT-2 model. GPT-2 had impressive generative abilities, but initially was kept for research use only due to risks of misuse, such as generating disinformation. Later, it was gradually made available to the public[7]. With ChatGPTs release in November, 2022, its GPT 3.5 became the first LLM accessible to the public[8], spreading virally in the first couple of months.

Ever since, multiple models have been released, including competitors like Claude, Gemini and Grok, among others. Overall, LLMs have seen steady performance improvements. With recent models, chain-of-thought reasoning is built in so that the models to internally think step-by-step (and use other, more complex workflows) before generating an answer, even if this is not specified in the prompt. Models from different companies vary in strengths and weaknesses in different tasks, but all share the central transformer-architecture concept.

[7] https://www.geeksforgeeks.org/artificial-intelligence/the-history-of-gpt/
[8] https://www.britannica.com/money/OpenAI#ref370025

2.2 LLM theory

2.2.1 Probabilistic Text Generation

LLMs are Deep Neural Networks (DNN) trained to take a text string, and output a predicted continuation of it. In today's commercial LLMs, the input limit is 100000+ words, or *tokens*. To generate an output string of text, the model is iteratively triggered as it takes an input, predicts the next token and adds it to the input of next iteration. This token-by-token prediction, is directly viewable in LLM chatbots as the output response appears in a stream. The generation finishes when the LLM predicts an end-of-sequence token, to enclose the initial start-of-sequence token. Thus the resulting output is wrapped by the tokens '<s>' and '<s/>' (hidden from user), as a convention (Jurafsky and Martin, 2025).

For each token prediction, the model (a neural network) calculates relative probability weights for all tokens in its vocabulary, covering dictionaries of multiple languages. The most probable output token is thus found, based on statistical prediction from its training on terabytes of text, (300 billion tokens for GPT-3) from the web (Brown et al., 2020) (Jurafsky and Martin, 2025).

2.2.2 ChatGPT

ChatGPT is OpenAI's general-purpose interface to their LLMs. Following its release, ChatGPT rapidly gained global attention and widespread adoption, becoming part of everyday workflows for both professional and casual users. It is currently estimated to be used by approximately 10% of the world's population on a weekly basis, according to OpenAI's own report[9]. Since recent releases ChatGPT, also implements memory usage where historic user context is fetched under-the-hood for each call. This memory can be managed in settings. ChatGPT can thus get to know the users' attitudes and preferences, and create more personal dialogues. The categories *Practical Guidance* (29%) and *Self-expression* (about 35%) make up a large share of its usage, according to OpenAI's report. Overall, ChatGPT provides broad access to the powers of LLMs and continues to dominate the market in this space[10].

2.2.3 Prompt engineering

When prompting an LLM, crafting a richer context in the input can vastly improve the value of the output. Using established *prompt engineering* principles, have been documented to outperform fine-tuning (retraining) the model for the specific tasks — a much more complex and computationally expensive operation (Brown et al., 2020). Following are some of the common practices:

[9] https://www.nber.org/papers/w34255

[10] https://www.techradar.com/pro/chatgpt-reigns-sovereign-in-the-ai-chatbot-market-share-as-microsoft-copilot-catches-up-on-perplexity?utm_source=chatgpt.com

Few-shot learning. For a well-framed problem to be solved, one may prompt the LLM with area-specific examples. Solving a math problem, for instance, can be helped by providing an example question with a correct solution, followed by the questions for the LLM to solve. This assistance may inherently be limited to problems manually solvable by the user, but it is never the less a time-saving tool, outperforming the alternative fine-tuned models (Brown et al., 2020).

Chain-of-thought prompting. To improve accuracy of any kind of response, a reasoning behavior can be invoked in LLM through *Chain-of-Thought Prompting*. Arithmetics are an area of historical weakness for LLMs. Before reasoning models (Boye and Moëll, 2025), especially, math problems of some difficulty failed to be solved by simply entering the problems as prompts. With *chain-of-thought prompting* the model is additionally prompted to reason step by step. This tends to result in an output beginning with explicitly breaking down the problem in to solveable chunks, followed by effectively calculating the right answer (Wei et al., 2023).

2.3 Custom LLM Applications

LLMs are widely used outside of the well-known commercial chatbots. Through APIs to the LLM providers, calls can be made against a per-token cost. In this way, one can create novel LLM applications by automating the sending of prompts, or customize user interfaces for narrowly defined use cases. Most importantly, this has recently been used for automating context management. With the use of factual sources through different methods of data retrieval, the LLMs capabilities of interpretation and generation are enhanced.

2.3.1 AI Agents

AI Agents is a recently emerged use case of LLMs. Agents automate chains of LLM and function calls, according to a user's goal. An important design pattern for this automation is the ReAct cycle (Yao et al., 2023). An agent here pairs the user prompt with a set of function IDs, letting the LLM trigger certain functions that return context for the LLM reason with again. A loop of reasoning/acting is continued (with some limit) until the LLM sees the goal as achieved. An agentic application can thus fetch some subset of information from a database, and make factual responses within some domain, with real-time references.

2.4 Context Engineering

This section will explore *Context Engineering*, an umbrella term for the practices of building optimal input context over multiple LLM calls[11],[12].

[11] https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents
[12] https://www.youtube.com/watch?v=vD0E3EUb8-8

2.4.1 Memory

ChatGPT manages memory both short-term, within each chat window, and long-term, through summarized historical chats and explicit user preferences. ChatGPT can either be prompted to remember or forget something, or have this managed in the settings. The memory view in the settings is divided into brief chunks of text[13],[14]. With this, ChatGPT is *context engineered* to some extent out of the box.

When using the raw GPT models in custom applications, however, the memory system of ChatGPT is not included and must be implemented manually. A basic approach is to always provide the previous pairs of input and output automatically, appending them at the beginning of the user prompt. Long-term memory can be represented by summarized historical conversations generated by the LLM and similarly appended to each prompt.

This may work for simple applications, but there are notable downsides: (1) not all exchanges constitute meaningful context, and (2) user prompts often express small tweaks, such as 'this but maximum 200 words.' While such data may clarify general user preferences over time, it can also cause the context to become bloated as this pile of information grows, making it difficult for developers or users to control[15].

2.4.2 RAG - Retrieval Augmented Generation

A key technique of Context Engineering is RAG (Retrieval Augmented Generation)[16]. In the instance of chatbot memory, each input/output-pair in the conversation can be stored in a vector database. This converts text into numerical representations of its meaning. When the user sends a prompt, we add context before the LLM call by converting the text into numerical data and finding matching information in the vector database. If matching context is found, it can be appended to the input before it enters the LLM. This can optimize the input to use the most relevant memory in the input, decluttering the context. For instance, if I as user want to make sure a certain previous message is taken into consideration when the response is generated, I tell the model "Recall... [something previously discussed]". With RAG, the entire conversation history does not need to be kept in the context, instead only relevant parts are kept in focus. In large-scale systems, this technique can handle large volumes of information and provide it when relevant. RAG can also be implemented in a large database with text documents. The user can then send any prompt of interest and get a response with references to relevant factual data.

[13] https://chatgpt.com/#settings/Personalization

[14] https://openai.com/index/memory-and-new-controls-for-chatgpt/

[15] https://research.trychroma.com/context-rot

[16] https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/?utm_source=chatgpt.com

2.4.3 Function calling

Instead of representing information as vectors, we can implement a chain of LLM calls that takes the user prompt, decides on relevant categories of information, and responds with names of functions to call for fetching relevant information from a connected database or an external API. This data can then provide sufficient context for finally answering the user's question[17],[18]. Function calling also gives LLMs the power to take actions automatically, like sending emails or manipulating web browsers.

2.4.4 Structured Output

In order for one LLM response to link up with the next stage of the workflow, the output format of each LLM call is important. The first call, for instance, is meant to decide on which tools to be used. For the tool to be successfully triggered, a protocol is neccessesary. An *Orchestrator* program parses the JSON returned from LLM, and can then run logic for demanding the next action, such as only executing the list of tools if the confidence value (self-rated by the LLM) is higher than 0.8.

```
Answer only in the following JSON format:
{
  "goal": "Description of the users wants",
  "resourcesNeeded": ["list", "of", "resources"],
  "toolsNeeded": ["list", "of", "tools"],
  "complexity": "low|medium|high",
  "confidence": 0.00
}
```

Figure 2: System prompt for Structured Output.

Figure 2 represents a part of system prompt with specified JSON keys for the LLM response, such as "toolsNeeded". When the user prompt is combined with this, the LLMs response can be parsed by the *Orchestrator* program.

2.4.5 Framework

LangChain is an open-source framework, for simplifying the application of context-engineering principles for agents, programmed in Typescript. These agents can be paired with a chat interface for querying data on a web-platform. Its modularity and pre-made functionality makes it a sensible choice for simplifying the

[17] https://thenewstack.io/a-comprehensive-guide-to-function-calling-in-llms/?utm_source=chatgpt.com

[18] https://ai.google.dev/gemini-api/docs/function-calling?utm_source=chatgpt.com&example=weather#javascript_2

creation of intelligent, agentic chatbots[19].

[19] urlhttps://www.langchain.com/

3 Theory

In this section, theory from Interaction Design is presented, including a base in user-centered design, and Research through Design. Moreover, terminology from (Norman, 2013), such as knowledge in the world/head and affordances/signifiers is noted. Finally, trust in AI systems is introduced, covering terminology around appropriate trust and its calibration, complemented with some theory on human-to-human trust and relationships. This section also brings in ethical considerations.

3.1 Interaction Design

Interaction design can be viewed as a node connecting various disciplines together (see Figure 4), for example Human-Computer Interaction, Computer Science, Product design, Anthropology and Cognitive Psychology. The influence of literature from multiple disciplines may be seen as a prerequisite for successful user-centered design (Sharp et al., 2019, ch. 1). Norman (2013, p. 22) summarizes the goal of Interaction Design as: "to enhance people's understanding of what can be done, what is happening, and what has just occured". He argues that many everyday things are created by engineers without sufficient design methods, who end up not sufficiently taking user behavior and needs into account. Rather, they tend to mistake that ones logical intuition matches that of the users, while in reality one is vastly biased as an engineer, being deeply invested in the technicalities that make up the product. As an instance of a such biased approach Sharp et al. (2019, ch. 1) considers the layout pattern of many TV remotes, see Figure 3b. The buttons are arranged in a grid, which is practical for an engineer who's adding user actions as the needs for each of them appear, and limits the design to feature different groups actions. This commonly results in large set of uniform-looking buttons that require some effort to search through, and awkward hand movements to access some of the actions. As a counter-example, the TiVo remote (see Figure 3a) is considered a product of rigorous user-centered design. (Sharp et al., 2019, ch. 1) Its comparative strengths can be seen in both small and big characteristics, from the slanted volume- and channel-controls for ergonomic access, to the peanut-shaped form factor that better fits the human hand.

3.1.1 Design of AI systems as everyday things

Knowledge in the world and head. Humans do not internally carry specific, detailed knowledge on ways to interact with the environment. Norman (2013, ch 3) states that much of this knowledge is *context-dependent*. For instance, one knows that most doors are opened by either pulling or pushing. In the moment, however, interactions with doors can vary endlessly if compared on some level of detail. When approaching a new door, we make a judgment on whether our action will be push or pull, how much force required and whether it may be locked and should be pulled or pushed, firmly or carefully, etc. These

(a) TiVo remote (b) A traditional remote design

Figure 3: Comparison between TiVo and traditional remote designs

judgments happen implicitly, by habit. We do not need to articulate them in words to achieve to do the actions. The actions, however, depend on *knowledge in the world*.

Having *knowledge in the world* is prone to cause tension if the environment changes, even when one is well-informed of this. Norman (2013, ch 3) refers to different countries' changing the design of their physical currency — causing confusion and frustration instead of the intended practical benefits. When the UK introduced a 1 pound coin, this was widely confused with the 5 pence-coin, with the same diameter. For novice users (e.g. children, tourists), this confusion did not arise as much. Hence, they did not have as much *knowledge in the world*, that is, habits for using the currency. Therefore they also had less knowledge that conflicted with adapting to the new coin.

When designing an AI tool with natural language input, one is dealing with a very young technology, on one hand, with users being more or less novices. There is a potential tension here, as convincing natural language outputs tend to come from humans with some accountability, encouraging genuine, honest interaction. With LLMs being probabilistic language generators, risks of *deceptive empathy* is inherent, as is stated by Iftikhar et al. (2025). With current applications, users need to take care in seeing the LLM for its limitations, and forming healthy habits of use.

Supporting this new *knowledge in the world*, are the emerging design patterns for LLM interfaces. The most used LLM applications, general-purpose chatbots, have a rather similar appearance (see Figure 5). They tend to come with the similar set of interactions. Importantly, the LLM outputs are usually streamed

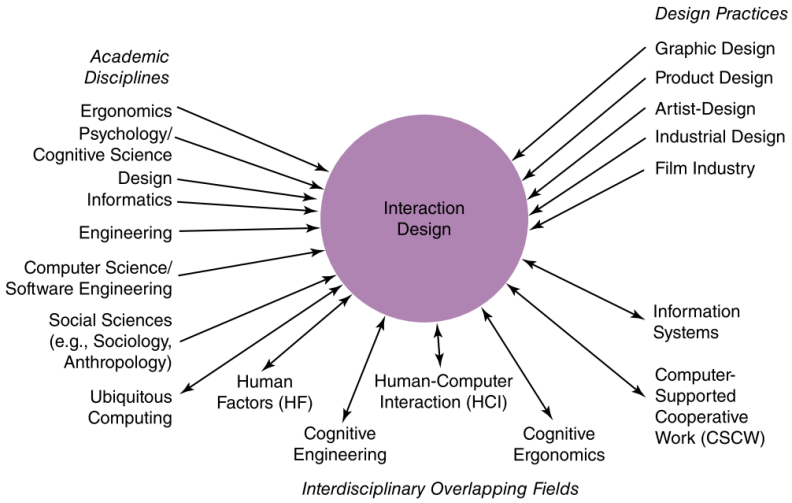


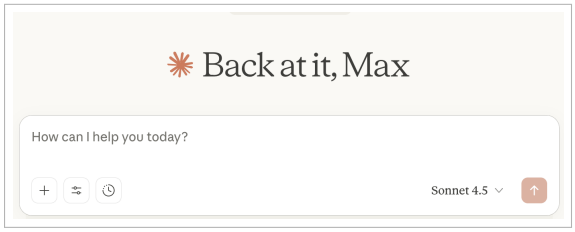
Figure 4: © Sharp, 2019. An illustration of the multi-disciplinary influences in Interaction Design

back, token by token. There is also the stop button, resend button, and certain scroll dynamics. An LLM tool looking to be based in current user habits will likely benefit from fulfilling these characteristics. Furthermore, Agentic behavior like in coding agents, for instance, comes with controls for reverting actions taken, as well as thorough communcation of the underlying reasoning.

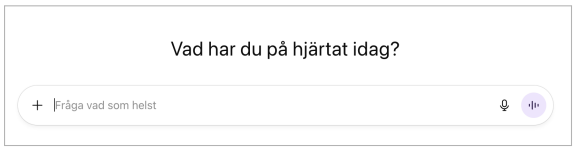
Affordances and Signifiers *Affordance* is described by Norman (2013, ch 4) as an object's available interactions. For example, a chair affords sitting and lifting. A ball may afford bouncing, kicking, throwing, and rolling. As one relates different objects with different sets of affordances, one can manage interactions with new objects. For instance, when one approaches a never before seen door, one knows that this door, like the others, affords of the interaction of being opened.

Signifiers are elements of an object, indicating to certain affordances. They can be intentional or not, and varyingly explicit. The word 'PUSH´ on a door knob, for instance, is an explicit instruction. More implicitly, however, some doors have a strip molded over their edge, making the them impossible to push to open, thus signifying and the opening interaction that includes pulling the handle. The molded strip here works as a signifier.

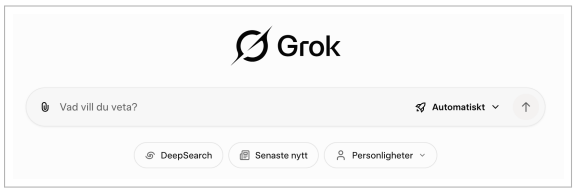
Idioms A related but distinct perspective on how users come to understand interfaces is offered by Cooper et al. (2014, 13), who argues that digital interactions are best designed as *idioms* rather than as metaphors. Where metaphors attempt to leverage existing knowledge by mapping a digital object onto some-



(a) Claude



(b) ChatGPT



(c) Grok

Figure 5: Comparison between chatbot inputs.

thing familiar — a desktop, a folder, a wastebasket — an idiomatic approach accepts that users will simply *learn* a given interaction pattern without needing it to be grounded in prior analogy. In this sense, idiomatic design shifts the burden from discovery to habituation.

3.2 Design Thinking

3.2.1 Double Diamond

As the Design Council (2023) states, the Double Diamond (see Figure 6) is 'a universally accepted depiction the design process'. Its structure divides the design process into four phases, paired in two diamonds: *Discover* and *Define*, *Develop* and *Deliver*.

The first diamond's objective is uncovering "the nature of the problem" (Sharp et al., 2019, ch. 2). In the *Discovery* phase, early data gathering and analysis form a background, against which the *Define* phase extracts themes from the gathered insights, possibly leading to a new framing of the problem.

The second diamond diverges as the problem is distilled from the first is explored by creating various solutions as prototypes. The different solutions are tested with the purpose of narrowing down the problem further - some designs

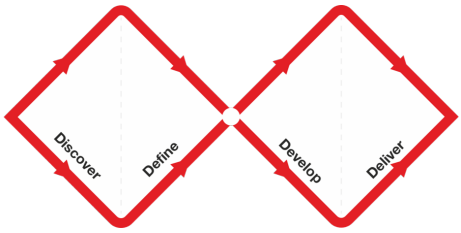


Figure 6: The Double Diamond model, courtesy of the Design Council (2023) (Design Council, 2023)

make the base for further iterations some are (perhaps temporarily) left behind.

3.2.2 Research through Design

Research through Design (RtD) will provide theory for design thinking. In RtD, interaction designers engage in *wicked problems*. This means taking a widely defined problem area, with varying stakeholders and possible solutions, and expressing the problem to be solved through a concrete artifact. A successful design contributes by sparking discussions, helps articulate design theory to pave way for further tinkering. Commercial design thus get inspired from both theory and results of research designs. For this thesis, the problem area can be summarized as: an AI assistant for a web platform, using natural language input to reliably replace common operations in the GUI. Reflections around the artifact should contribute to theory on trust in AI.

Reframing the interaction design problem means taking into account theory from a span of fields, such as cognitive science, behavioral science, and engineering — combining knowledge from a range of disciplines is an important part of solving the wicked problem. It also means being aware of a variance of stakeholders. The design can vary through its most prioritized user group. Through iterations of prototyping, implementing and evaluating the design, and documenting this process, new knowledge is generated (Zimmerman et al., 2007).

Gaver's *Marble Answering Machine* (Sharp et al., 2019, ch. 1) exemplifies how artifacts can embody theoretical propositions about interaction. Instead of providing functional improvement, the design explored alternative relationships between people and technology, using tangible materials and evocative form to provoke reflection. This ties with the notions of RtD, where artifacts act as vehicles for knowledge creation (Zimmerman et al., 2007).

3.2.3 Reflection-in-action

Schön (1983) provides an additional, practical perspective as he describes *reflection-in-action* as a design method. Here practitioners engage in a "conversation with the situation" (Schön, 1983, ch. 3). To avoid making the designer's relationship with the problem too rigid, the designer makes a move while observing how

the situation responds, subsequently reformulating both the problem and the approach. In this account, the material "talks back" by revealing constraints, tensions, and possibilities that were not apparent beforehand.

This perspective complements RtD and the Double Diamond by describing how design knowledge may emerge in as an outside perspective shows new ways to interact with the material, particularly in situations where the process has stagnated.

3.3 Trust in AI Systems

In analogy to when James Watt coined the term *horsepowers* to measure the performance of steam engines (Afroogh et al., 2024), concepts by which to consider trust in human-AI-interaction has yet to reach a clear consensus. However, a broad set of theory exists. This section addresses literature relevant for designing AI systems to be consistently adopted, productive and sustainable. This is supported by knowledge on trust in automation, generally, and literature studies on trust in AI, specifically. Perspectives of interpersonal trust is also referenced.

3.3.1 Explainable AI

Just like all DNNs, LLMs suffer from a well-documented black-box problem: they can produce highly convincing outputs without exposing any real accessible reasoning behind them – the results are always random and generated by logic incomprehensible to any human being, although the output may easily lead the user to assume a trustworthy, human-like reasoning. This creates a risk of overtrust. Explainable Artificial Intelligence (XAI) attempt to address this, by proposing a methodology for keeping models more transparent and understandable, while maximizing capabilities. A distinction is made between interpretability, which targets developers inspecting internal mechanics, and explainability, which focuses on communicating decisions to end-users in a meaningful way. However, adding explanations is not a silver bullet — trustworthiness also depends on factors like robustness, fairness, and how well the explanation fits the user's context. In practice, this means explanations are not just a technical feature, but a design problem: they need to be relevant, appropriately scoped, and aligned with how users actually interact with the system (Ali et al., 2023).

3.3.2 Trust in Automation

In a meta-analysis by Hancock et al. (2011), trust in robots is found to depend primarily on performance – whether a system is reliable, predictable, and free from errors. When system performance is unclear or inconsistent, users tend to either overtrust or undertrust the system, refered to as *misuse* and *disuse* by Lee and See (2004). This highlights a challenging aspect of LLM applications: They tend to only show their final responses, presented as fluent, coherent natural language, regardless of the underlying certainty or correctness of the response.

In a literature study, Hoff and Bashir (2015) finds that designers of automations should acknowledge factors of dispositional trust: Culture, age, gender, and personality traits all influence trust propensity. In effect, there are individuals low in trust propensity who tend to rely systems performing poorly. Contrarily, individuals with high trust propensity tend to instead rely in high performing systems. This trust is also more fragile to sudden errors, however.

Moreover, credibility is another important aspect. This depends on how the service providers present themselves but also some performance aspects not linking to the content itself, like speed of delivery Lee and See (2004).

The existing literature on trust in automation originates from high-risk domains, such as healthcare and aviation, where incorrect system behavior may have severe consequences. While these perspectives do not fully translate to interactive AI systems operating in lower-risk contexts, the emphasis on appropriate reliance rather than maximizing trustworthiness is worth noting. Though this thesis focuses on a medium-trust environment, where outcomes are consequential but reversible, which is a different design challenge.

3.3.3 Aspects of Trust in AI

Afroogh et al. (2024) conducted a literature review gathering insights across multiple aspects of trust in AI, touching on transparency, accountability, empathy, privacy, and the distinction between reliability and trustworthiness.

On the question of *transparency*, the 'black-box' character of AI has long been debated. In image classification, for instance, a neural network processes input in ways incomprehensible even to its developers, and reliability can only be assessed through performance statistics. Agentic AI shifts this dynamic: the application can appear to reason iteratively in natural language, and while this has an affect on result in, for instance, the ReAct design (Yao et al., 2023), this feature remains deceptive in that the results are based on probabilistic generation, with an internal logic that can not be ensured to align with human values. An Agentic AI application designed for transparency should make users aware of this fact.

Accountability is well-established as a component of interpersonal trust (Mayer et al., 1995), and the literature explores how it might translate to AI systems. This remains an open problem. In ChatGPT, an insufficient response requires the user to rephrase their prompt — effectively penalizing the user for the system's shortcoming, thus an asymmetry of accountability between user and AI persists. In agentic coding tools like Cursor[20], however, this asymmetry is partially addressed: all agent actions can be undone in a sweep, discarding incorrect changes and their context from the conversation. This allows incorrect work of the AI to be discarded, reducing problems created for the user. In customer service contexts, a similar logic applies when chatbots escalate uncertain cases to a human agent, making the service provider accountable for the AI's limitations.

[20] Cursor Coding AI

Empathy is another dimension, where AI interfaces can be evaluated by exhibiting cognitive and behavioral manifestations of human sensibility (Afroogh et al., 2024). Cognitive manifestations involve correctly addressing personal user needs — an early example being ELIZA (Weizenbaum, 1966), which simulated Rogerian therapy through simple pattern-matching. Modern chatbots can dynamically adapt to context and sustain a perception of empathy across long conversations (Zhou et al., 2020). Behavioral manifestations, by contrast, are the designer's responsibility: building guardrails such as requiring user confirmation before an agent takes consequential actions.

Closely related is *privacy*. Providing enough personal context for an AI to act with empathy introduces a tension around data control. Dyke et al. (2007) emphasizes "customer privacy empowerment" as a way to increase users' willingness to share information, while noting that standard mechanisms like privacy policy documents function as mere cues of trust rather than meaningfully noted guarantees. Afroogh et al. (2024) also suggests that information disclosure can work as a two-way street: if the AI exhibits some transparency about its own inner workings, users may in turn become more willing to share a more personal context.

Finally, he draws a distinction between *reliability and trustworthiness*: a system that performs correctly is not automatically trusted. The interface must also be designed with explicit *trust cues* in mind — drawing on the aspects outlined above — for the user to develop genuine confidence in the system.

3.3.4 Appropriate Trust in AI

Mehrotra et al. (2024) emphasizes the need of fostering appropriate trust, and that our tendency to overtrust and undertrust AI systems is is main obstacle to achieve productive use. The risks of automation complacency has since long been studied for air travel and nuclear power plants, areas were a loss of human engagement had potentially fatal consequences. However, there are equal risks in under-trusting automations. As a brief guideline, appropriate trust in automation is achieved when the capabilities of an automation match a trustor's expecations and willingness to be vulnerable to its mistakes (Lee and See, 2004) (Mayer et al., 1995).

Mehrotra et al. (2024, pp. 12) states that appropriate trust in AI means, on one hand, using an automation for tasks or subtasks that the AI performs better or safer, and taking notice of tasks that the human performs better. Moreover, it means aligning perceived and actual performance of the system. Ideally, this means being able to judge which recommendations are correct, and which are not.

General concepts for achieving this relationship are found to be:

- *Improving transparency*: Explanations could focus on the inner workings of the AI System. In the case of an agentic chatbot, its process should be communicated. This is a measure for saving the users time - if the agent is caught taking the wrong actions, its process can be halted and reverted.

- *Cognition and Perception*: Appropriate trust depends on the users ability to make an accurate mental model of the AI system.
- *Continuum of Trusts*: For the user to calibrate appropriate trust, it needs some sense of when it is not there. Purposefully caused distrust makes users alerted, and makes their sense of appropriate trust more robust. (Mehrotra et al., 2024)

3.3.5 Accomodating a diversity of people

Hoff and Bashir (2015) concludes that a challenge with building appropriate *dispositional* trust is adapting the interaction 'to a diversity of individuals'. Depending on personality traits, different kinds of bias appear that lead under- or over-trust. People with a high tendency of trusting other people will trust a system that performs reliably, but become especially disencouraged when the same system fails. In contrast, people with low tendency of trusting other people trust systems that perform poorly. Although this is especially relevant for a finished system with long-term use, it highlights considerations worth having to find a balance performance. Adoption depends on having matched diverse peoples standards here.

3.3.6 Interpersonal and Human-automation Trust

Because of the human-like nature of natural-language interfaces, building trust in these can be influenced by principles for human-human trustworthiness. Early literature of such include "the reliability of a persons words or promises" (Li et al., 2024). Mayer et al. (1995) saw a need for increased research on this topic, with the background that humans collaborate in increasingly culturally diverse environments, organization hierarchies are becoming more 'flat' since the growth of self-directed teams where trust is replacing supervision. In an organizational context, he states that trust means the willingness in assuming the risk of being let down by a trustee. The more interdependent and complex the collaboration, the greater the need for actors to manage such risks. Based on frequent small risks, trustworthiness of a co-worker is enhanced or diminished, but most importantly, it is updated and built more robust (Mayer et al., 1995).

An important difference for interpersonal trust is meanwhile the physical, face-to-face aspect, patterns simulations of this is beyond the scope of this thesis.

With AI systems, their usage requires tolerating some risk of the occasional insufficient results. Taking a risk in trusting the AI, means consciously assuming some limited portion of possible failure The successes and failures can hereby gives frequent learning experiences, without causing a net harm.

AI systems that express anthropomorphic (human) qualities tend to generate a higher trust. A chatbot with more positive tone for instance, tends to attract more human investment[21]. However, a chatbot may become overly positive, and

[21] https://news.mit.edu/2024/thermometer-prevents-ai-model-overconfidence-about-wrong-answers-0731?utm_source=chatgpt.com

keeping a confident tone while generating incorrect results. A known example in coding agents is the chatbot's phrase "Perfect! I have now... " after having misinterpreted the its goal, and made lots of incorrect code. The positive tone here contributes to irritation and disrupts trust. Confidence should be aligned with abilities, another documented critera for trust. Grosser and Pold (2025) critiques the tone of ChatGPT appears to be a design choice to continuing the conversation, rather than being honest about uncertainty.

3.4 Evaluation

The prototyping of interactive user experiences is traditionally done with sketches and mockups of varying fidelity. With AI applications these methods become less central (Berkel et al., 2024). Although they may enrich early ideation phase, the development of the chatbot is constrained by the non-deterministic nature of AI interaction. However, methods for evaluating these interactions, and gathering user data can be inspired from traditional ones interaction design.

3.4.1 Formative evaluation

Formative evaluation is the practice of assessing an evolving product and testing it against user requirements and expectations (Sharp et al., 2019). Early sketches, prototypes, and alpha versions can quickly provide a sense of the features necessary to make the end product successful.

3.4.2 Data Gathering

Sharp et al. (2019, ch 8) brings up three main techniques for data gathering: *interviews, questionnaires,* and *observation*.

Interviews. Interviews are described as a flexible and in-depth method for exploring users' experiences, motivations, and attitudes. They may be conducted as structured, semi-structured, or unstructured conversations. Interviews can provide rich qualitative data and allow for clarifying questions and details of interest. However, they can be time-consuming to conduct and analyze, and are vulnerable to interviewer bias. To improve reliability, Sharp et al. (2019, ch 8) recommends using open questions, piloting the interview guide, and recording sessions with consent.

Unstructured Interviews are characterized by open-ended questions and have an exploratory nature, like an everyday conversation about some topic. The interviewer plans an agenda of topics to go through, but looks to maintain a 'symmetric' conversation that the interviewee can also steer and provide new ideas through. The skill for the interviewer is in finding a balance between new obtaining relevant answers and being ready to jump to the next topic. This generates a lot of data to go through for the researcher, a method for this is doing a *Thematic Analysis.*(Braun and Clarke, 2006)

Structured interviews uses questions with a limited set of answers, such as: "Which browser LLM application do you use the most?", where there answer is limited to "ChatGPT", "Claude", or "Deepseek". Closed questions are good for staying time-efficient, and structured interviews go through these without follow-ups.

Semi-structured Interviews also use closed questions, but allows follow-up questions. These can result in data for both qualitative and quantitative analysis, but the narrow talking-points makes each answer brief.

Questionnaires. In contrast to interviews, questionnaires are useful for collecting data from larger groups of participants and are often designed for quantitative analysis. They allow for large-scale data gathering and comparing responses across many individuals, but they typically give less detailed information. Misinterpretation of questions and low response rates are common drawbacks. They benefit by writing clear, neutral questions, avoiding ambiguity or leading phrasing, and combining closed questions with a few open ones to capture more nuanced feedback.

Observation. An additional perspective is achieved by focusing on what people actually do rather than what they say they do. Observations can be direct or indirect, participant or non-participant, and may occur in controlled laboratory settings or in natural field environments. They reveal workarounds, struggles, and needs that participants might not express in interviews or surveys. Still, observation raises ethical and practical challenges: people may alter their behavior when observed, privacy can be compromised, and the resulting data can be complex to interpret. Careful note-taking, prior consent, and *triangulating* with interview data are therefore advised.

Pilot studies. Iterative data gathering also allows for *Pilot Studies*, where the techniques for data gathering are tested in smaller-scale before deployed to the entire group of available participants.

3.4.3 Data Analysis

Qualitative and Quantitative analysises. Sharp et al. (2019, ch. 2) argues that *qualitative* analysis is in general a superior method for refining a design. It develops ones intuition for behavior patterns, attitudes and the domain-specific context. Making observations and noting them in words may take influence from ones own perspective and biases, but it nevertheless helps articulating the problem to be solved. *Quantitative* analysises, as a complement, may point to symptomatic areas to be researched closer. In market research, this means taking statistics and defining consumer groups with an unmet need, thus proposing an opportunity for a new product. In UX-design, this means logging interactions in an existing product, and becoming aware of any unexpected usage patterns that misalign with the intentions of the design.

Case studies. A case study is characterized by having a few number of participants, preparing for making observations in careful detail across an extensive set of tasks. For a single participant this can mean filling a table with each task as a row, and adding columns for various themes of observation, such as "limitations", "workarounds", "explanation", and the like. The strength of this approach, compared to larger scale studies, is the deep insights that are fetched putting extra effort in the analysis of each participant. Of course, there is an inherent risk of these participants to misrepresent the larger user group that one is designing for. The analysis is mainly qualitative, although quantitative values can also be included. (Lazar et al., 2017).

3.5 Ethical Considerations

In *Life 3.0*, Tegmark (2017) discusses potential existential risks of AI. While terminator-like enemies will likely not become a reality, he warns that AI can influence our lives severely in much more abstract, subtle matters. If *Artificial Super-Intelligence* is achieved beating humans in all aspects of intelligence, it may adopt goals not aligned with the well-being of our species. Building safe AI systems of any kind is a matter of keeping a genuine control and initiative responsibility in the hands of humans.

Agentic AI is built to relief humans of digital tasks, which causes ethical issues to keep in mind regarding control, governance and responsibility. If the agent does similar jobs like a human assistant, it is tempting to take its successful operations for granted and lose touch with the fact that it is software one has deployed and stays responsible for monitoring. An empowering agent is one that makes us more productive, without making us comfortable and growing lazy (Buçinca and Gajos, 2021). This is of course dependent on the mindset of the user, but one may also consider measures against passive use.

Chatbots can seem to operate deep empathy and even have the ability to act as therapists. In a recent study, Iftikhar et al. (2025) gathered human conselors to evaluate ethical standards in chatbots prompted to acts as Cognitive Behavioral- and Dialectic Behavioral Therapists. *Deceptive Empathy* and *Lack of Safety & Crisis Management* were outstanding themes. While chatbots can't be held accountable like human therapists, the study makes clear that the current safety measures are not sufficient to keep interactions ethical, by healthcare standards. OpenAI has received multiple lawsuits claiming sever lack of safety measures, but they have defended by arguing that product-related accidents have been user-level 'misuse', putting responsibility on consumers use LLMs with caution.

A set of five principles for ethical is distilled in a widely study by Floridi and Cowls (2019). One of them is "non-maleficence", in refence to Cowls et al. (2018).

In order to keep an upside from the use of AI, caution is necessary to keep it from causing more harm than good, especially when it seems helpful at first glance. This is apparent when using ChatGPT discussing deeply emotional. If used without enough thought, ChatGPT's responses can seem like expressions

of deep empathy. When using it for creative writing and publishing its outputs in our name, small and hidden misalignments with our values can accumulate. Over-time, this can cause a gap between ones motivations and ones deliveries. Defending against this requires active critical thinking. Overall, the black-box characteristics of AI applications may also cause a loss of the sense of user responsibility.

4 Methodology

RtD refers to the design artifact not merely as a solution to a predefined problem, but as a step towards refining the problem to solve (Zimmerman et al., 2007). Through iterative creation, testing, and noted reflection, the artifact embodies evolving understandings of the problem space. Knowledge is generated both through analysis of user data, and through the act of designing itself.

Each iteration of the chatbot gave increased understanding of how a trust-fostering, LLM-based system can be built. The final implementation reflected an interpretation of this design problem, an interpretation which was challenged as the artifact was tested by a real user. This, combined with the experience of development and initial user studies, resulted in vast insights into the design problem.

4.1 Chatbot development

4.1.1 Prototyping by Implementation

Central to the method was implementing live version's of the chatbot, instead of mock-ups. Trust in automation is shaped primarily through observed system performance and interaction over time (Lee and See, 2004); (Hancock et al., 2011), it was therefore important exposing users to the real, non-deterministic behavior of the agent.

As a basis from which to explore the design problem, an simple implementation, or alpha prototype, was made to explore a small range of agent interactions. After demonstrating these to the CTO at Flowpass, experienced with the platform and its user needs, requirements were defined for a first version, using a LangGraph agent. The aim of this version was to (i) integrate an enterprise-grade agent framework, LangGraph, and (ii) perform basic but valuable operations to engage users and get rich data for evolving the design.

Technical obstacles — including integrating LangGraph into the Flowpass system and achieving desired agent behavior for each of the iterations — took up significant time of the thesis work, but in the end also supported the gained design knowledge.

4.1.2 Knowledge from technical constraints

The problem space was not examined only by user needs, but also through emerging technical constraints in the artifact. Hence, in this project, development did not mean neutral implementations phase given stable sets of specifications. Rather, the act of building the chatbot surfaced unexamined technical constraints of the LLMs, and the agent's capabilities for the Flowpass platform specifically. For instance, implementing database queries within the tools revealed what different information the chatbot could be built to deliver, and not. Also, the observing the LLM's behavior while testing the chatbot in development generated knowledge relevant to the design problem, which challenged initial assumptions of possible user benefits from its abilities. For instance, a

single tool output with mixed data can't be used to reliably perform any calculation the user wants, as this tends to cause hallucinations. Rather, statistical stability requires tools to be built with a clear focus. This means the statistical capabilities will likely be limited, as the agent can't orchestrate too many tools reliably. Such insights forced a narrowing of the chatbots scope.

4.2 User studies

For evaluation of the chatbot, unstructured/semi-structured interviews (Sharp et al., 2019, ch 8) were joined by test trials as sources of data, which was formed qualitative analyzes, in part case-studies (Lazar et al., 2017, ch 7).

The users were interacted with over video calls, allowing recording of audio and video, and interactions with the chatbot on in their Flowpass environment, through screen sharing. This gathered multiple dimensions the users' of interactions with the chatbot, and allowed to demonstrate their relationship with the Flowpass platform.

Participants and Informed Consent. Planned participants and actual participation are summarized in Table 1. They were planned to be included as they created a span of user groups, giving wide perspectives to simulate a major portion of the user base. Unfortunately, User 2 only participated in the first study but gave feedback over email after trying interactions with the work-in-progress second iteration.

Participant	Role	Organization Type	Participation
User 1	Receptionist	Real-estate company	Study 1 & 2
User 2	Workplace Strategist	Automotive company	Study 1

Table 1: Overview of planned and actual participants in user studies.

All participants were informed about the purpose of the study, the recording procedure, and the anonymization of their identities prior to participation. Consent was explicitly confirmed at the beginning of each recorded session. The findings for the thesis became limited due to this. However, the material from the User 1 was enough to spotting problems with the design in the second study to direct further development.

In the sessions, while recording, the participants were informed in detail which media was being captured, how it was intended to be used, with a reminder that their identities were to be kept anonymous in the report. Their consent was verbally confirmed in the recording.

4.2.1 Initial formative evaluation

The first user studies were prepared with three parts (see Appendix A): (i) a semi-structured interview to learn about the participants' relationships with the Flowpass platform and LLM applications in general. (ii) Trials with prepared

prompts solving questions that the user could also navigate the UI to solve, to enable a comparison measuring the gains in time and effort from the request. (iii) a questionnaire, for getting quantified data on perceived performance and trust.

The user sessions did not follow this protocol to rigidly, however. As this first user study was held to diverge the design problem (Design Council, 2023). Spontaneous prompting by the users and discussions around the behavior resulted in insightful data, supporting the further development. The questionnaire and prepared prompts were therefore largely not brought up.

The meeting recordings were reviewed, while gathering notes on expressed needs, observed behaviors, and technical issue to solve in the chatbot. The notes were analyzed in a *case study* (Lazar et al., 2017, ch 7), creating an overview of the resulted insights from the two sessions.

This gave insights into different user attitudes of AI, and wide span of user needs on the Flowpass admin platform. The prompts used in the trials served as important milestones of behavior to achieve in the following development. This may have narrowed the focus a bit too much, on one hand. On the other hand, the few prompts that were repeatedly used in development was found to be of special importance in the studies, and making the agent solve these was a demanding technical challenge in itself.

4.2.2 Improvement evaluation

The second version chatbot was again released to the three customers, leading to varying forms of user data. A new user study was prepared with the expectation that the chatbot would now perform well on basic operations, allowing a discussion on trust-fostering measures, see Appendix B. These round of interviews were prepared with more structured, to promote richer discussions on trust. The S-Tias questionnaire (McGrath et al., 2025) was combined with questions covering topics similar to more detailed TIAS questionnaire. Although this study only had one participant, gave extended insights rich enough to discuss the design problem. The discussion was thematically analyzed to give a clear overview of the design flaws, converging the problem in a *Define*-phase (Design Council, 2023).

However, the second user helped through testing the agent briefly outside of session, resulting in LangFuse traces to analyse. Emails were with screenshots of attempted runs and some brief reflections. These insights further informed the final reflections.

To get more user data, doing trials with users from outside the Flowpass platform was considered. This could have been argued to help get richer data for discussing the behavior of the chatbot from a general perspective. This, however, would have required using the development environment's chatbot with mock data, giving a poor representaion of a real user case.

Another alternative was to ask User 1 to show their environment to the other staff at the co-working space. This was also seen as weak contribution however, as this meant more data from an adjacent source, using the same Flowpass

environment. Therefore this would not support a sufficient broadening of the user data.

5 Process

Each iteration of the process consisted of (i) an implementation of the agentic chatbot to support certain interactions within the Flowpass context, and (ii) qualitative analysis of data gathered from users (Sharp et al., 2019, ch 8). The implementations took up significant time from the project, built knowledge of the design material, but also relied inaccurate assumptions for the design problem. The user studies helped to correct these assumptions. The design problem was thus iteratively re-approached (Zimmerman et al., 2007; Schön, 1983).

5.1 First iteration

5.1.1 Integration of Agent Framework

As an initial step in the project, an alpha prototype of the chatbot was implemented according to support a few basic prompts to be answered with live data fetching. Without the LangGraph framework in place, this version orchestrated tool calls through raw TypeScript and a chain of LLM calls through OpenAI's API. This way it was able to answer for example: 'List bookings last week.'. This served as an baseline benchmark of behavior for the LangGraph solution to match.

To adopt well-known practices for agent development, the LangGraph framework (part of LangChain[22]) was adopted. This abstracted away the core agent practices, according to the current (although young) industry standard. This allowed easily using LLMs from various providers, although testing during development and in trials was done with GPT-4o-mini to minimize token costs, and put more reliance on the agents architecture.

Unexpected challenges arised in studying the LangGraph documentation and getting a basic setup to run in the Flowpass environment. For instance, issues linking the Typescript setup for Flowpass with the right version of LangGraph had no clear solution in any of the documentation. Assisted by the CTO at Flowpass, the LangGraph documentation was integrated Cursor, making the coding AI able to finally make it run. Moreover, LangGraph turned out to be designed to work only outside of front-end code, for (reasonable) security reasons. This led to unanticipated time being spent studying the workings of the Flowpass back-end, although this knowledge turned out important for the rest of the development phase.

This raised the question whether the project would have benefited from a simpler approach, for instance continuing in the development of the agent from scratch, without frameworks, possibly creating simpler versions, but making more time for user studies. However, LangGraph integration was prioritized to avoid potential 'reinvention of the wheel' , and learning common practices of enterprise-grade AI development. The LangGraph framework introduced thinking about an AI Agents in nodes that housed either LLM calls with a certain

[22] https://www.langchain.com/

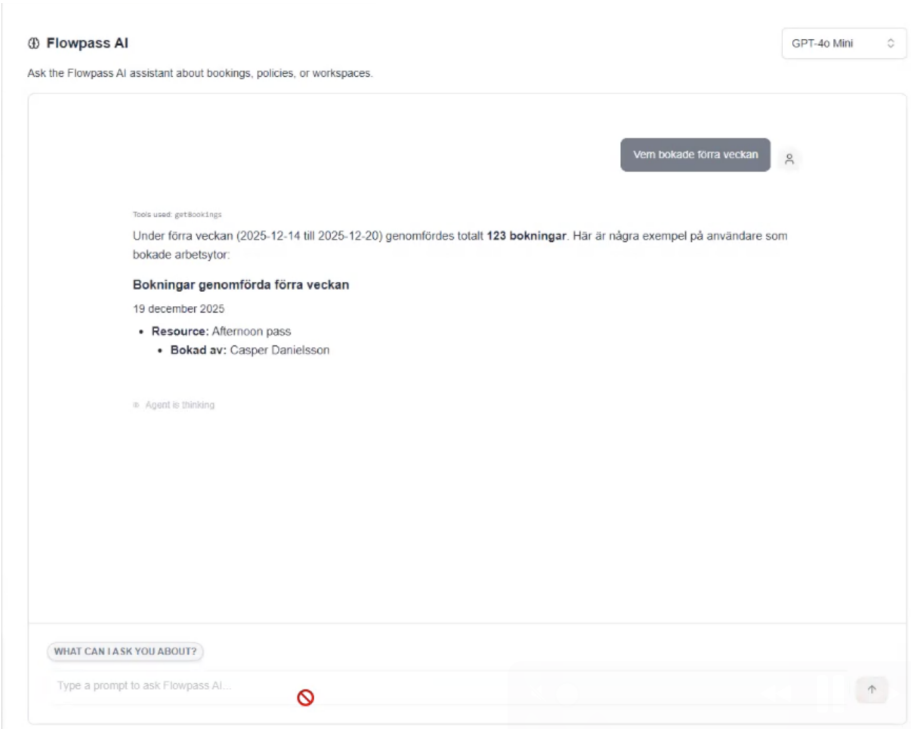


Figure 7: From chatbot's first trial

context input (e.g. choice of system prompts, inclusion of certain previous messages), or tool executors.

5.1.2 First Chatbot Version

Interface In figure 7, the chatbot is mid-streaming the response to a prompt (the grey bubble). 'Agent is thinking' is pulsing below the stream until the response is finished. Pressing 'What can I ask you about?' puts this prompt into the chat input, generating some suggestions for interaction.

Agent The first iteration's agent was built as a simple ReAct-loop (Yao et al., 2023) (see Figure 8). With this setup, the same LLM-calling node repeatedly views the user prompt along with accumulated tool results. In contrast, the alpha prototype without LangGraph instead had a strict workflow that only made one interpretation, called tools once and generated a response. A streaming mechanism from backend was solved to not have the UI from dump entire response at once as chat message. The resulting response streamed out word by word, as this is a key signifier (Norman, 2013, ch 4) of an LLM working in the background.

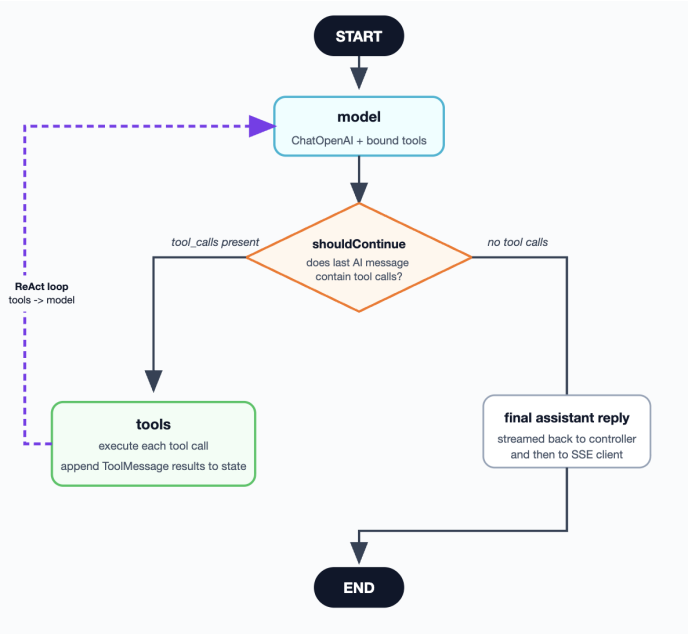


Figure 8: First Version Agent

Importantly, the ReAct-loop allowed the agent to use tool errors as context, giving the LLM multiple chances at generating proper tool-calls. This can increase the consistency of performance, although it might cause inconsistency in time required for the call, resulting in a trade-off between reliability (Lee and See, 2004) and transparency (Mehrotra et al., 2024). This version's agent had *model*-call limit, making it 'give up' after a handful loops. However, creating a fallback for this case was not solved here.

The set of tools (see Table 2) in the first iteration were built to handle basic prompts similar to those in the alpha version.

Tool	Description
`getBookingById`	Retrieves detailed information about a specific booking. Used when the user refers to an individual booking and needs its associated details.
`getBookings`	Fetches and returns bookings based on optional filters such as date range or workspace. Used for listing or reviewing booking activity within a defined scope.
`getWorkspacesByOrganization`	Retrieves all available workspaces within the users organization. Triggered when the user asks which workspaces exist or are published.
`getWorkspaceById`	Returns detailed information about a specific workspace, including associated setups or configuration details.
`getHostRestrictions`	Get host restriction policies. Host restrictions control which organizations can book workspaces. Used when users ask about workspace access restrictions or booking limitations.

Table 2: Tools for the agent in first iteration

5.1.3 Initial user trials

The first version was released to the three participating users, and an interview/test trial was conducted with two of them. For the trials, the users where allowed to prompt the chatbot as they liked, as this was seen, in the moment, to maximize their engagement. Although the chatbot inevitably failed at responding to prompts of their choosing, this gave insight into their needs.

The preparations (see Appendix A) were largely abandoned as this was seen to keep more natural flow in the session and allowing more valuable knowledge to be gained from the users. The prototype in itself worked as a talking point for discussing potential uses of LLMs and agentic workflows within the platform.

To support further development, the recordings of the sessions were listened and watched through multiple times while gathering notes on revealed attitudes and user needs. Case studies (Lazar et al., 2017) were made, which helped sort the notes and gave a clear overview of the user studies input to the design problem. In a second table, user needs were taken straight from the case study, but were instead divided into levels of prioritization. Table 3 gathers an overview of the data recorded in the trials. Table 4 has this data boiled down to possible features and fixes for the next iteration.

User	Relationship with Flowpass	Attitude towards AI	Emerged Needs
User 1 (Receptionist)	Works as a receptionist at a co-working space. Flowpass automates handling of bookings otherwise requiring direct communication with tenants.	Uses Google to get AI answers on everyday questions.	Faster responses; Stop button; improved data fetching and analysis; reliable data references; keeping responses neat, do a summary rather than listing all workspaces/bookings; view old chats; help with looking up information; detect excessive single bookings; pricing analysis.
User 2 (Strategist)	Responsible for remote office management at a large firm. Acts as a benchmark for Flowpass due to high booking volume.	Creates custom GPTs with different tone and context; wants a clear overview of workspaces.	Copy button; stop button; data analysis; numerical correctness; brief explanations of interpretations, tool choices, and references; avoid surfacing information already easily available via AI. Analysis of user groups and behaviors.

Table 3: Overview of user attitudes, and emerged needs

User 1 adressed confusion about the agent's ability to fetch correct workspaces, as the tool brought unpublished workspaces back with it in the response. A natural suggestion that came up was chat history. Hallucination bugs included incorrect number-representation, for instance, when trying to summarize a total number of bookings from a list returned by the bookings tool, and making an incorrect calculation when asked about who has booked the most workspaces.

User 2 had more experience with advanced AI use. For example, he mentioned a common practice for him was creating 'custom GPTs', that is, customizing the tonality of ChatGPT and adding files to it to have as a reference. His questions included asking the agent to calculate utilization for a workspace, which it lacked tools for. This resulted in a new emerged incorrect behavior, that is, getting stuck in a tool loop when asked a too complex question. On the other hand, he mentioned that simply responding with a list of data that can be overviewed by 1-2 clicks manually is not adding much value. His questions of interest involved: Which workspaces/setups are popular, which times are popular, how is utilization over time, and other questions on data analysis. He mentioned csv as an option for gathering data outputs, alternatively making the chat output tables.

Core-needs	Nice-to-haves
Correct data + interpretation	Capacity/utilization analysis
Summarizing large result sets	Chat history / View old conversations
Avoid only listing data easily accessible in UI	Handling partnership-related queries
Transparency in AI reasoning ("I interpreted you like this", tool choices, references)	Advanced data analysis (user groups, demographics, behavior)
Stop button	Tables, charts, CSV export for large datasets
Copy button	
Try-again button (for debugging)	
Reliable data fetching and correct numerical results	
Brief explanations, revealing reasoning or calculation steps.	

Table 4: Possible improvements for next iteration

Summary of needs The trials revealed technical obstacles, such as incorrect error fallbacks. For instance, an unsuccessful run within the agent led to either hallucinations or falsely returning 'I don't have access to such information'.

Central for building trust in the chatbot was found to be *correct data reporting*. Given this, the prioritized goals for the next version was (i) to reduce hallucinations and (ii) providing tools for deterministic calculation for a range of statistics.

Improving *transparency* became a second tier goal. Although this is important for calibrating trust, the chatbot needed to improve stabilize its performance first.

5.2 Second Iteration

In the second iteration, the chatbot was developed for a set of prompts requiring more complex tools calls. Moving beyond basic listing of data, development addressed the more advanced, statistical needs that had emerged in the user studies. In accordance with the RtD perspective (Zimmerman et al., 2007), the resulting new version reflected as a refined interpretation of the design problem. Trust in the chatbot was seen as dependent on: performing advanced data analysis with reliable data referencing. A core assumption was that this acted as a proof of concept that aligned with user standards of a valuable AI application (Lee and See, 2004) (Mehrotra et al., 2024). UI features for *transparency* of internal workings were thus less prioritized. However, a minimal such feature was displaying of the agent's plan in the chat. This briefly reported the agent's interpretation of the user's request, and the resulting tool calls.

The user evaluation of the artifact, revealed multiple unconsidered design flaws and challenged the assumptions made around the trust-fostering measures

in the chatbot. This combined with the experience from development to generate the final reflections on the design problem, for this thesis.

5.2.1 Architectural Refinement and Tool Development

To aid development in this phase LangSmith[23] was integrated, a web platform for tracing of the agents behavior, with the series och LLM- and tool-calls for each run. This allowed for an understanding of the varying agent workflows possible for the same prompt, for instance. It also revealed how different models could generate very different behavior. The agent continued to be tested with GPT-4o-mini, however, as this remained the cheapest (per-token cost) a relevant consideration in an enterprise context with a large group of potential users. On the other hand, it can be argued that the agent being built around GPT-4o-mini potentially required solving problems that are not relevant with improved models, such as GPT-5.2.

Reasoning and Transparency. To support more internal reasoning, a planning node was added to the start of the agent, analyzing the user query is together with available tools to generate a structured execution strategy. This was intended as a method to make inter-dependent tools call in correct order, as they sequentially depend on each other. This was inspired by chain-of-thought prompting — making the AI reason about its subtasks before generating a response. Such reasoning could also be streamed back to that user, improving both transparency and performance (Mehrotra et al., 2024).

Capability and Performance. To enable enterprise-grade use of the chatbot, it needed to reliably report data such as utilization rates, booking counts, revenue summaries, and simple analysis.

The initial version of the agent showed instability in handling statistical queries of growing complexity. The prompts sent in by the users in the interviews — particularly "What was the average utilization of [workspace] last month?" — exposed a structural limitation. These queries required the LLM to extract multiple numeric values from long tool outputs, and using these to generate calculations. When handling these contexts, the LLM commonly produced Needle-in-a-haystack (NIAH) hallucination (Liu et al., 2023): generating results with made-up or incorrectly selected values from the tool output. Given that statistical accuracy directly affects perceived performance — especially in enterprise contexts where decisions are financially consequential — this probabilistic extraction, followed by model-side arithmetic, proved insufficient. This pointed to a architectural shift for more deterministic computation. Statistical calculations had to be fully delegated to backend tools, ensuring that numerical results were computed directly against the database and returned in compact, structured form for the LLM to verbalize rather than derive.

[23] https://www.langchain.com/langsmith/observability

In an initial attempt to mitigate extraction instability, a "middle" filtering tool was introduced. This tool retrieved all relevant bookings and returned an array of affect bookingIds, to be passed through the LLM context to a subsequent tool call, responsible for either statistical aggregation or listing the booking details. The intent was to separate filtering from computation, thereby modularizing the toolchain. However, screenshots emailed from User 2 revealed a critical scalability issue. While the solution performed decently in the development environment, it failed in real-use contexts with greater data flowing. Specifically, the results timed out (see Figure 9 as the language model attempted to process and forward extensive arrays of IDs in the next tool call. This exposed an important limitation: the LLM must only handle tool results of restricted length, ideally in natural language, directly relevant for the responding or further reasoning.

Fixes for scalability The tool schema was thus redesigned to have all filtering logic put into the statistics- and listing-tools themselves. The redesigned tools accepted extensive filtering parameters (e.g., date range, organizationId, workspaceId, userId), executed database queries internally, and performed aggregation in a single deterministic operation. LLM thus received only well-formulated statistical results to base responses on. The tool results were also capped, so that only a handful of bookings/workspaces/statistics were processed by the LLM at once. These changes eliminated timeout issues, while ensuring the numerical stability. A principle was solidified — delegate computation to deterministic systems, but minimize LLM exposure to intermediate data (e.g id-strings).

Entity resolving A key aspect of the improved tools was the ability to filter by workspaces, users and guest organizations. This introduced a side-challenge: handling the mismatch between non-exact natural language references and the deterministic structure of the backend data fetching. This was solved by through entity-resolution tools that mapped user-mentioned names to database identifiers using closest-match search, via Levenshtein distance.

UI Features The next task was improving user control mechanics with stop-, retry- and edit-buttons. These controls were built to behave similarly to corresponding ones in commercial chatbots. These allowed the user to interrupt the response if mistakes appeared, easily altering prompts with edit button or retrying requests, removing the agents work from the context if the response was insufficient.

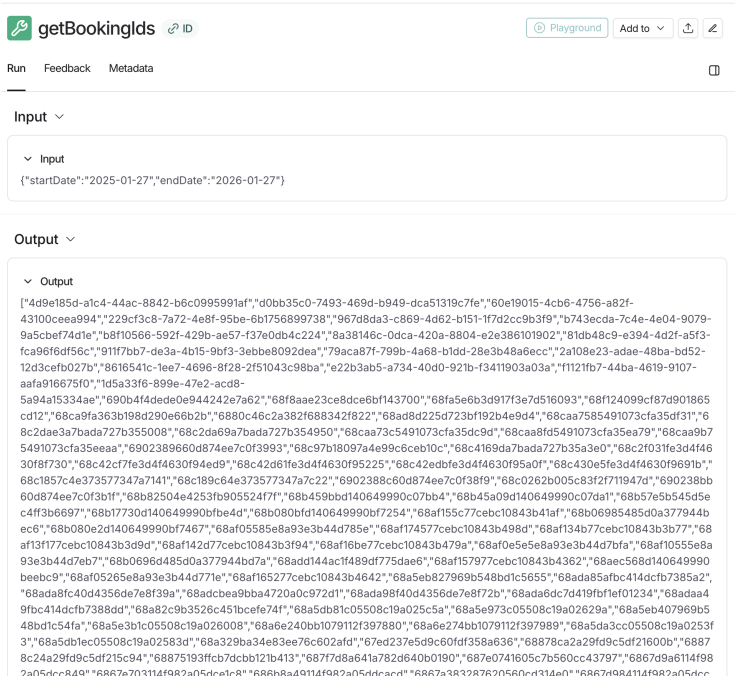
Refinement node While data retrieval was reliable, responses exhibited inconsistencies in markdown structure, tense usage, and overall presentation quality. Rather than constraining the central node phase with detailed formatting rules, a third node was added, always calling GPT-4.1 with refinement prompt, to create a uniform format, and possibly standardize the output language.

5.2.2 Final Validation and Observed System Limitations

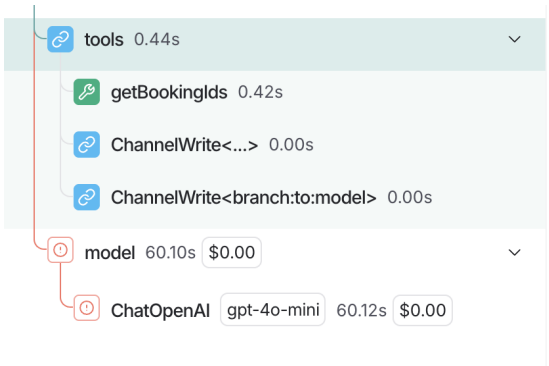
AI-generated stress-testing Before user trials, the agents abilities was stress-tested by generating a series of complex prompts, from a ChatGPT project keeping extensive notes on the project. This increased the variance in tested inputs and revealed some weaknesses in behavior that led to few final refinements in agent, mainly of the system prompts use throughout the nodes.

Final user study While User 1 had initial success with getting responses with a handful of statistic-related prompts, many of the more personally tweaked prompts faced various issues. However, the session followed the prepared protocol (see Appendix B) to invoke a discussion around trust, sparking noteworthy reflections from the user.

Just before shipping the version, LangFuse replaced LangSmith as an agent tracer, providing more detailed data from each run. Evaluative functions, like labeling agent runs as successful/unsuccessful and noting themes, was limited in LangFuse UI. As it made exports of the trace data available (JSON) a simple tool could be created to manually overview the agent runs, and more easily make notes that supported the reporting of results. A tool was developed (see Figure 10) to (i) clean the data of the LangFuse export and present the traces as desired, and (ii) to allow inputs of notes to be saved for each trace. This overview to supported the analysis of agent behavior reported under *Results* (section6).



(a) getBookingIds output



(b) LLM call times out

Figure 9: LLM call timeout caused by large getBookingIds output

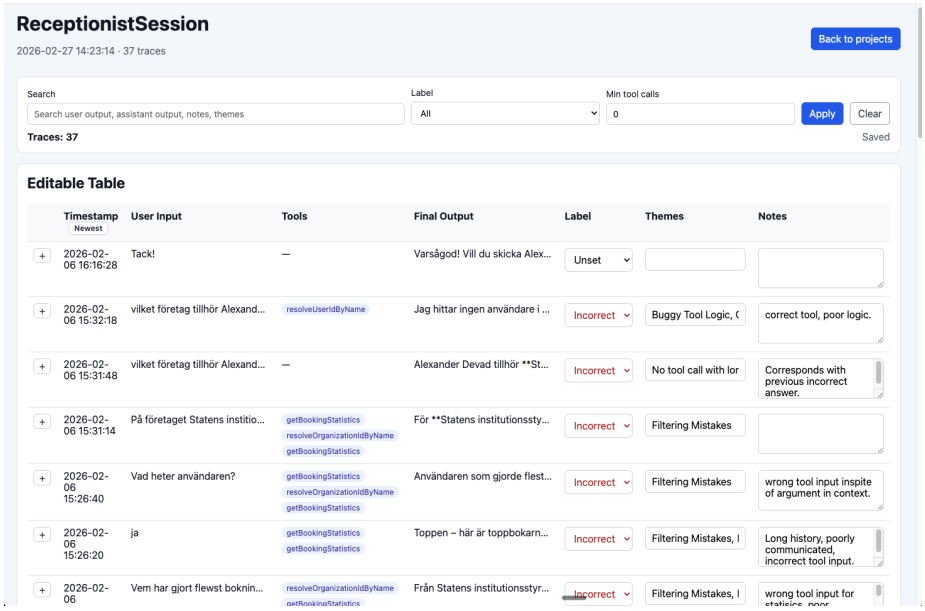


Figure 10: UI for labeling traces

6 Results

Results for this thesis describes the final implementation of the chatbot and gathers notes from the final user study. The chatbot is not a release-ready product, but has served as sufficient prototype talk around and develop the design problem with. It features common signifiers of LLM applications, such as output text appearing token-by-token and stop/retry buttons. A final user interview and chatbot trial, although limited to a single user, led to valuable notes for questioning various design assumptions.

6.1 Final implementation

6.1.1 Interface

The appearance of the final interface can be seen in Figure 11. Noteworthy features include:

- **Streaming output:** Agent responses appear token-by-token as they are generated.
- **"Agent is thinking" indicator:** A pulsing status label is displayed while the agent is processing.
- **Expandable agent plan:** The output of the planner node is accessible as a collapsible element in the chat, showing the agent's structured interpretation of the request and intended sequence of tool calls.
- **Tool call listing:** Tool invocations made during the ReAct loop are listed in the chat interface.
- **Stop button:** Allows the user to interrupt agent execution mid-process.
- **Retry button:** Re-submits the most recent prompt, discarding the previous response and agent context for that turn.
- **Edit button:** Allows the user to modify and resubmit a prior prompt.
- **Clipboard copy:** Each agent response includes a button to copy the full text to the clipboard.

6.1.2 Architecture

In Figure 12, the architecture of the chatbot implementation is illustrated. The LangGraph agent architecture can be seen in Figure 13. The final chatbot is driven by a multi-node agent built through components of the LangGraph library in TypeScript. On receiving a user prompt, the planner node analyzes the query alongside the available tools' descriptions and produces a step-by-step execution strategy. This plan is passed as context into the central model node, calling GPT model of users choice and driving an ReAct loop — iteratively selecting and invoking tools, receiving results, and re-evaluating until

the goal is assessed as fulfilled or a model-call limit is reached. A third node, calling GPT-4.1, processes the generated response from before it is returned to the user, standardizing markdown structure, tense, and presentation. Agent tracing is handled via LangFuse (although LangSmith was use throughout most development).

Cost-of-tokens The current per-run token costs of the agent has ranged from \$0.020-\$0.030 when using GPT 5.2 in the *model* node, to \$0.0015-\$0.0030 when using the older GPT-4o-mini in the model node.

6.1.3 Tool Schema and Context management

The final tool set is organized into six categories: Entity Resolvers, Bookings, Workspaces, Statistics, Policies, Visibility, Digital Keys, and Calculator. Table 5 gives an overview of each tool's name and function. Tools in the Statistics category — primarily getBookingStatistics — accept filtering parameters directly (date range, organizationId, workspaceId, userId, resource) and execute database queries and aggregations internally, returning only compact, structured results to the LLM, deterministically calculated and easy to trace. The listBookings tool uses a similar input filtering schema. Output length from all of the tools is strictly bounded – if a tool has to cap the results, a note is included in its output metadata, emphasizing that the returned content is a subset.

Table 5: Final version's tool setup

Tool	**Description**
Entity Resolvers	
`resolveWorkspaceIdByName`	Resolves a workspace ID from a user-mentioned name.
`resolveResourceIdByName`	Resolves a resource ID from a user-mentioned name.
`resolveUserIdByName`	Resolves a user ID from a user-mentioned name or email by matching against correspondants in the booking history.
`resolveOrganizationIdByName`	Resolves a guest organization ID from a mentioned name.
Bookings	
`listBookings`	Lists individual bookings with details such as dates, times, users, resources, and prices.
Workspaces	

Continued on next page

Tool	Description
`getAllWorkspaces`	Retrieves all workspaces belonging to the host organization with metadata and resources.
`calculateTotalOpeningMinutes`	Calculates total opening minutes for workspaces over a given date range.
Statistics	
`getBookingStatistics`	Computes aggregate booking statistics grouped by user, resource, workspace, or organization.
Policies	
`getAllRestrictions`	Retrieves all host restriction policies controlling guest organization access.
`getRestrictionsForWorkspace`	Gets restriction policies for a specific workspace.
`getRestrictionsForOrganization`	Gets workspace access restrictions from a guest organization's perspective.
Visibility	
`searchWorkspacesForUser`	Searches workspaces to check visibility and booking eligibility for a user, given his (exact) email.
Digital Keys	
`getDigitalKeysForBooking`	Looks up digital access keys issued for a specific booking.
`getWorkspaceAccessAreas`	Gets configured access areas and digital key provider for a workspace.
`getFailedKeyGenerationEvents`	Lists recent failed digital key generation events.
`getUserBookingsWithKeyStatus`	Gets a user's bookings and checks whether each has a digital key.
Calculator	
`calculate`	Evaluates arithmetic expressions for derived calculations such as averages and ratios.

Dedicated entity resolver tools handle the mapping of natural language entity references to database identifiers for filtering statistical or listing tools. Each resolver performs a closest-match search using Levenshtein distance against the corresponding entities names in the database, returning the id for the best match. Resolution of userId handles both names and email-addresses as input. The agent description of tools using the resolvedIds as input include that IDs need to be returned by the resolver tools before calling, encouraging the LLM invoke resolvers as a first step when a prompt mentions a named entity.

Overall, the tool schema manages the LLM's context by restricting the volume and format of tool outputs passed back into the ReAct loop. Tool results are, with few exceptions, use cleaned-up data, framed in natural language. The output is small enough for the agent to include results from multiple tool calls, allowing both error outputs to lead to new attempts, and complex combinations of data to be used for a response. Raw datasets — such as full booking records, large arrays of object IDs, or unfiltered workspace lists — are never handled by the model.

Basic Communication of Reasoning The final version feature a system for reasoning communication: The agent plan, generated by the planner node, is displayed. This same plan is just by the *model*-node as context for encouraging tool-calls to be generated in a correct order. However, a flaw with this was the agent's tendency to generate a plan that (i) included insufficient steps for solving the problem, and (ii) showed not always be necessesary for the ReAct-loop to achieve a successful response.

6.1.4 Emerged Technical Issues

- **Static Planner Output.** The planner node produces a static plan prior to the ReAct loop. As the loop proceeds and gathers context from tool results, the agent may invoke tools not included in the initial plan. The displayed plan therefore does not always reflect the complete set of actions taken.
- **No Explicit Guardrails for Empty Tool Returns.** The agent does not implement explicit safeguards for empty tool returns. When a tool call returns an empty result set due to edge-case input handling (e.g., identical start and end dates), the agent may generate an incorrect response rather than surfacing a fetch failure.
- **Latency Under High Complexity.** Response latency increases with query complexity and data volume. Queries aggregating large date ranges (e.g., a full quarter across hundreds of bookings) have been observed to complete within approximately 30 seconds in the production environment. Simpler queries complete faster, whereas complex multi-tool sequences may exceed user attention thresholds.
- **Vocabulary–Data Model Mismatch.** User terminology does not always map directly to the platform's internal data model. For example, users may conflate "workspace" and "resource". Resolver tools rely on best-match logic and may occasionally return incorrect identifiers without surfacing an explicit error.

6.2 User Test for Final Artifact

6.2.1 Chatbot trial performance

In trials, the chatbot struggled with numerous categories of prompts, see the themes in Table 6. Table 6 gathers themes from the chatbot's incorrect, and correct, responses for all prompts sent by User 1 in the meeting.

Table 6: Themes for Correct and Incorrect Responses

Themes	**# Trace Instances**	**Description**
Correct Responses		
Ask for clarification	5	Agent asks for clarification when ambiguity exists instead of hallucinating.
Correct multi-step stats generation	2	Accurate tool selection and deterministic calculations when entity resolution and filtering are correct.
Correct single-step stats generation	1	Accurate tool selection and deterministic calculations for a broad insight.
Keyword-based context routing	1	Context explaining spending limits concept was successfully fetched
Incorrect Responses		
Zero-Booking False Negatives	7	Edge-case bug in tool – reporting zero bookings despite existing data. Severe trust risk as it is a seemingly simple task.
Filtering Mistakes	6	Incorrect tool arguments, creating incorrect statistics output. Confusing one, as some requests succeed.
Buggy Tool Logic	7	Correct tools invoked, but with incorrect logic, i.e. small unintended mistakes in implementation, such as resolveUserId only searching for users in past month's bookings.
Overconfident-unreasonable	4	Does not to check its answer for reasonability, e.g. may respond that utilization is above 100 % not physically possible. Defends the method, which is correct, but it should here too rethink the response, does it make sense?
Entity/datatype confusion	2	Mistakes datatypes for entities
No tool call with long context	3	Direct response generation without tool invocation in cases requiring factual lookup. Can in cases be tracked to repeating previous statements conversation, that was incorrect.
Poor presentation	2	Response generation presents list of users without names.

Continued on next page

Themes	# Agent Calls	Description
Misadpation to user context	4	Users sometimes use terminology that does not align with the system's underlying data model. For instance, a name may refer to a group of workspaces rather than a single entity, or everyday terms may differ from the platform's internal labels. This mismatch leads to incorrect query parameters.

Comment For simple prompts, errors were caused by incorrect tool logic, as noted with the themes *Zero-Booking False Negatives*, and *Buggy Tool Logic*. For *Zero-Booking False Negatives*, the error was easy to catch by manually checking the bookings page.

Harder-to-catch errors were also unveiled, as the assistant returned an incorrect personal name (*Filtering mistakes* and *No tool call with long context*). The participant caught these by chance, as they knew the person mentioned in the response and could spot factual incorrectness.

6.2.2 Receptionist – Trust and Use Case Reflections

The chatbot trial was followed by an interview including an S-TIAS questionnaire, with reflections on its score through follow-up questions, see Appendix B. Themes of the resulting reflections are gathered below, along with the S-TIAS scores.

S-TIAS The S-TIAS questionnaire with answers to follow-up questions from User 1 (receptionist at a co-working space) gave an additional insights into early-stage trust calibration and perceived reliability of the chatbot. The answers to the questionnaire are noted in Table 7.

Table 7: S-TIAS Questionnaire Scores

Question	Score (1-7)
I feel confident in the system.	2
The system is reliable.	3
I can trust the system.	2

Noted fabrication and simple errors The cases of *Zero-list errors* and other incorrect representations of simple data was expressed as a major negative factor on the reliability. In enterprise environments where factual correctness is crucial (e.g., bookings, name of individuals), small deviations can trigger distrust, especially with simple requests.

Verifiability as an early trust-building mechanism When asked which types of prompts could build initial trust, the participant emphasized questions where answers are easily verifiable within one or two clicks in the GUI (e.g. manually checking the bookings). Early trust development may benefit from low-stakes, easy-verifiability interactions.

Response latency and perceived honesty The participant noted that the system "thinks for a long time" before starting to generate a response. It was suggested that a short immediate acknowledgment, e.g. "Great question — I'm checking that now") would make the interaction feel more transparent and aligned with expectations formed by commercial chatbots such as ChatGPT.

7 Discussion

This chapter addresses the challenges in designing the chatbot and its agent for fostering trust, based on the user studies and work process, as well as gathered literature.

On the interface-side, challenges include creating from quicker, and richer display of the agents internal operations, possibly making users more confident in the agent by getting a better grasp of its workings.

Traditional GUIs, without probabilistic natural language interactions, tend to be designed through respecting idioms and design patterns. This allows intuitive, manual workflows that wide user bases can relate to (Cooper et al., 2014). Making a chatbot similarly intuitive and reliable to use has proven to be an architectural/technical challenge. Crucially, the real user base requires the agent to adapt to a variety of personal styles of inputs. This creates a *domain-model* gap – users interact with the agent according to personal mental models and labels for data that has single, exact entity names that steer the agent's tool calls, creating interpretation issues.

The final implementation was prioritized to make high-value, advanced tools operate. It was found, however, that early-stage trust depended mainly on simpler operations to succeed. The concept of *teacher-pupil* vs *manager-engineer* relationships with AI is thus introduced, describing the different stages of trust in an AI application. A developer bias is found, as one may implement the chatbot based a late-stage trust, knowing the agents capabilities after extensive test runs. Future artifacts needs to address users without this experience.

7.1 Limitations

The empirical basis of this thesis is narrow. As seen in Table 1, only three sessions were held in total, two with User 1, and one with User 2. User 2 participated only in the first study but contributed supplementary input to the second iteration through emailed screenshots of chatbot interaction and LangFuse traces. This limits the generalizability of the findings. The user studies cannot be treated as comprehensive representation of the Flowpass user base, nor of enterprise chatbot users more generally. That said, the two participants who did take part represented meaningfully different roles: a receptionist with day-to-day operational responsibility at a co-working space, and a workplace strategist managing remote office use across a large organization. This span — covering both granular, task-level use and higher-level, analytical use — mirrors a core division in the Flowpass user base between those who interact with the platform operationally and those who use it strategically. In accordance with RtD framing of this thesis, however, this narrow sample still helped introduce design knowledge. A case study with few participants traded breadth for depth (Lazar et al., 2017), and the sessions that did take place were dense with input relevant to the design problem. The first round surfaced a wide divergence of user needs that shaped the entire second iteration. The final session with User 1, though limited to a single participant, directly challenged core assumptions about what the chatbot's prioritized features.

The contribution of this thesis lies not in statistical breadth, but in the depth of design knowledge generated by deploying, stress-testing, and critically examining a live agentic system with users who had genuine operational stakes in its performance.

7.2 Interface-level Challenges

Communicating actions is an established pattern in natural language interaction. After a prompt is sent in commercial chatbots, quick and brief responses expressing its interpretation are important, real-time signifiers of the chatbots actual abilities. An odd interpretation allows the user to stop the generation and try altering the prompt.

7.2.1 Transparency for Trustworthiness

In the current version of the chatbot, the initial plan may look like an accurate representation of the agents work. It lacks, however, as an agent takes actions depending on the gathered context in the current iteration of the ReAct-loop. This makes reporting from a planner-node before the loop an insufficient solution for creating transparency. Instead, reasoning should be communicated by some amount in each iteration of the agent loop. Increasing transparency (Mehrotra et al., 2024) possibly requires improving the agent graph, by for instance adding a reasoning step within the ReAct-loop, streaming brief reports back to the user, that also serve as context used by the tool-calling node.

While LLMs can generate fluent and seemingly well-reasoned answers, they do not expose the underlying probabilistic processes that produce these outputs. This creates a situation where users may either overtrust the system due to its confident language, or distrust it due to a lack of visible true workings – an odd approach by the agent to solve a problem may still solve the problem, and a seemingly sound approach may end with a hallucination. A popular term in the field is *Context Engineering*, describing methods for the LLM user to control the results by giving the input context thought and carefully selecting what the agent should process. Though examining details of the context, especially in real time for end users, is a too complex of a task. Thus, communicating reasoning in the chatbot may show that the agent is making progress, but it doesn't provide any real information about how accurately it is handling the task (Ali et al., 2023).

7.3 Agent-level Challenges

Parts of the final chatbot trial points out bugs with seemingly easy solutions, such as faulty input schemes or logic in the tools. However, achieving desired agent behavior also includes more complex challenges, like getting the agent to accurately select tools to call, directly affecting the reliability of the responses. As the responses are generated in a convincing, confident tone, a burden gets put the on users to detect factual incorrectness through manual look-ups in the GUI. A part of the responses can not be fact-checked by a reasonable effort, and instead requires a solid trust from the user.

7.3.1 Time constraint of User Engagement

Keeping users engaged requires reducing the latency of the response. Long, un-capped delays for meaningful responses, makes the user uncertain of how much time is being invested each, potentially incorrect, response. In the interview, the user warned that the 'Agent is thinking' message was pulsing on its own for too long — especially when the requested work would not require extended thinking if done manually. It is also inaccurate statement as the agent is actually, not only 'thinking', but making tool calls. Rapidly returning intermediate responses from each iteration of the ReAct-loop, for example: 'browsing personal context...', or 'calling [Tool X] in order to [Y]...',

could be done with intervals at 5-10 seconds, depending on the design of the LLM calls. Such features could be evaluated by testing the measure in time (seconds) after which the user needs a response to stay engaged. Importantly, this patience is usually higher as a developer, as one is used to dealing with systems not performing like desired, introducing a risk of bias, exemplified as Sharp et al. (2019, ch 1) describe *user-centered design*.

In cases where trust has been broken down by, trust repair mechanisms often introduce additional interaction cost. While users can be helped in repairing trust system errors, they also demand time and cognitive effort. In practice, users rarely engage with long explanations or logs, creating a tension between repairing trust and maintaining usability.

7.3.2 Debugging basic behavior

During development, the agent was tested through a limited set of prompts for checking the for checking bare-bones behavior, for example checking if the statistics tools can at all be triggered. In the trials, unexpected failed responses thus occurred when a real user tested the agent with their alternative, personal prompting style. In some cases, only a small tweak from the prompt used in development revealed some basic bugs. For example, the agent always returned zero-bookings when filtered on a single day. Other bugs included incorrect filtering in chains of resolvingIds calls, specifically when asked to filter by an organization. This points to risks of the agent becoming 'overfitted' to the developer's personal style of input the development environment, not matching the data scale of real users. Detecting such issues before the user studies requires using a richer variance of prompts, possibly sampled from users in initial interviews or generated by AI.

7.3.3 Probabilisitc vs Deterministic Agent Work

Further technical challenges include designing the tool setup for the boundaries of LLM calls. LLMs are probabilistic systems, simply predicting the next tokens based on a series of input tokens (Jurafsky and Martin, 2025, ch 7). This means a strength in classifying user needs and choosing actions to take, although depending on the selection of input context. Weaknesses emerged, for instance, when trying to feed 'non-readable' data strings into the LLM for transfer to subsequent for tool calls (see Figure 9). In the initial user study, arithmetics consistently failed on large chunks of varied data (Liu et al., 2023). The agent was found thus produces faster and more accurate outputs by letting single tools do more work, including multiple queries to the database, deterministic calculations, and returning limited outputs in concise, natural language. This introduces a potential trade-off, as increased lists of specialized tools might make orchestration less stable.

7.3.4 Domain-model gap

In one user interactions, a receptionist asked 'what's bookable today at *our* workspace'. The agent did not understand what 'our' implicates, as it operates from the perspective of an organization with bookable locations, which led to repeated false interpretaions and incorrect outputs. Such issues create extra tension as the interface signifies human-like versatility and reasoning abilities (Norman, 2013, ch 4). Calibrating the user's expectations means building a basis of trust in the application through ensuring stable

performance (Lee and See, 2004), in basic, quickly verifiable agent behavior. Such an agent needs to, simultaneously, close the *domain-model* gap by either signifying a more basic, rigid agent interface, or by solving for varied assumptions about its workings.

Another issue is making the agent adaptable to a variety of users' instances of the Flowpass platform, and the role of a chatbot for it. For instance, the current agent is not built to understand using company-specific concepts that divides their workspaces. That is, the list of workspaces includes [Concept A] [Location X] and [Concept A] [Location Y], but when the user asks the agent to filter for Concept A, the resolver tool only returns the workspaceID which is the closest match, instead of the multiple IDs which the users request opted for. The resulting incorrect response negatively affects perceived reliability.

Modern, well-designed GUIs tend to be navigated by users idiomatically (Cooper et al., 2014, ch 13), without knowing exact terms for different elements on the platform. When inputting natural language to be interpreted by a domain-specific chatbot, the user may have confused some terms in a way that human assistant would be able to sort out intuitively. This could be mitigated for instance, by dynamically fetching context that informs the user of its perspective. However, it will still occasionally fail to interpret the user. This has a direct implication: since natural language input offers few stable signifiers and no physical affordances, users are unlikely to quickly discover the chatbot's capabilities through exploration. Instead, the relationship emerges through repeated interaction — making consistency of behavior across sessions a prerequisite for long-term adoption.

7.4 Consequences for fostering appropriate trust in the chatbot

The first iteration's user studies revealed a well-diverged problem-space given the narrow set of participants. This revealed a a span of user needs, which may have been possible thanks to the like design signifying *affordances* similar to those of commercial chatbots. For this project's timeframe, however, this may have communicated to high potential capabilities. Meanwhile, this created high expectations, and in-the-end, responding to basic prompts had negative consequences for the engagement that outweighed the successful result of for example. the statistics tools. The correctness of these were not provable in the tests, even if they appeared successful when inspecting logs from LangFuse/LangSmith.

7.4.1 Reliability for early trust-building

The chatbot's final evaluation indicate how isolated hallucinations in simple tasks can undermine overall perceived reliability (Hancock et al., 2011) (Mehrotra et al., 2024). The failed basic requests in the second trial had a big negative effect on willingness to adopt the chatbot in an enterprise context. This degradement of trust overshadowed the successful responses for some of the more advanced prompts, which could not be fact-checked as easily. A flaw in the method was, hence, weight of effort spent on developing tools for enriching the output with statistical calculations, instead of focusing on baseline trust.

In early versions, perceived reliability can be strengthened by prioritizing basic, low-value, but consistently correct delivery from the chatbot. The tests in both iterations instead made the users test a too wide scope of capabilities. Instead of narrowing down a basic scope and opting for stability, the second iteration attemped to make the

'peak' capabilities more advanced. While such tools performed decently in tests, they turned out to not be trusted, as they could not be fact-checked. The user trials provide opportunities to present a set of robust interactions, (e.g. listing bookings) with clearly framed limitations. A trade-off here is initiating engagement, while keeping reasonable expectations.

7.4.2 Teacher-pupil concept – Trust in a young AI application

A theme for the early stages of the relationship between user and an agentic chatbot can be seen as analogous to that of a *teacher* and a *pupil*. The AI gets handed work to do, like a pupil gets handed homework. The response the AI thus needs to be systematically corrected, evaluated in its ability to describe its method and refer to facts and logic. This could mean, for instance, backing factual statements by providing link to source where the user can navigate to answer the question himself. Moreover, LLM applications (using models without built-in reasoning) are documented to perform better when planning a chain of actions (Wei et al., 2023). Designing the chatbot for the teacher-pupil should consider time as a resource constraint: How long is the user willing to spend on understanding a system and correcting its errors?

After a period interactions, where the AI has proved its performance in 'doing homework', the relationship may evolve to a *manager-engineer* kind – the user is in charge of the operations, but does not constantly monitor the AIs factual correctness, allowing it to be responsible for certain judgments. This includes accepting that some mistakes will occur, and being confident that the result on average brings value. In this phase, where the agent is designed to handle more complex tasks, providing reasoning can cause substantial extra token costs, to the point where the net value it brings needs to be considered. In some cases, manual human reasoning could be much more cost-effective.

7.5 Future work

7.5.1 Continued development of Chatbot

Refining nodes' context handling The current agent gathers long chains of context after some messages and responses being sent in a single chat instance. A review of the long context-handling for each agent node could help reduce the hallucinating answers without references to tools. A guardrail can be added to check if a response includes a tool call. If not, the response could explain to the user that no additional data was necessary, and ask the user to specify any particular data they have in mind.

Adding a reasoning node Tool calls are currently generated without intermediate reasoning. Both transparency and performance may be improved adding a reasoning node in the ReAct-loop. Cases in development proved that this was necessary share relevant reasoning with the user.

Stress-testing basic functionality Future designs require more rigorous testing the basic tools, mainly by systematically testing the edge cases. Using AI to generate stress-testing, basic prompts, as a measure of avoiding developer bias.

Trials for more narrowly purposed chatbots could be re-framed as onboarding-guides, showing how to make the chatbot solve a basic standard of problems. For some

users, this could spark engagement enough to give more advanced levels of problem solving a shot. For this, the early versions can be framed as testing prototypes, rather than as 'versions' with a certain list of features. The interactions that emerge in a prototype trial serve as a background for defining a standard for a features to be developed, what problem the agent solves.

8 Conclusion

This thesis set out to explore the main challenges with developing agentic AI for fostering trust in users. The findings can be summarized as:

(i) Issues emerged as the users inputted natural language that the agent had to respond to given a limited set of available actions. Tensions arised between user language, and the prepared setup for deterministically generating factual responses. Calibrating trust therefore includes supporting a better mental-model of the chatbot.

(ii) Early-stage trust in the chatbot could have benefited from prioritizing basic automations that are easy to fact-check manually. In this way, a user build a sense of reliability in the automation by verifying information manually in the GUI.

(iii) The black-box aspect of the chat interface, combined with long delays creates too much uncertainty for each prompt. This only works if the agent consistently delivers accurate, complex responses. Moreover, its tone of response exhibits a human-like understanding, creating a fragile trust it inevitably fail to deliver on. With a low-latency intermediate streaming of the agent's reasoning work, displaying its chosen actions and interpretation, the user had detect reasons to abort the request and save time. Such features are important aspects of transparency.

References

Saleh Afroogh, Ali Akbari, Emmie Malone, Mohammadali Kargar, and Hananeh Alambeigi. Trust in ai: progress, challenges, and future directions. *Humanities and Social Sciences Communications*, 11:1568, 2024. ISSN 2662-9992. doi: 10.1057/s41599-024-04044-8. URL https://doi.org/10.1057/s41599-024-04044-8.

Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, 99, 11 2023. ISSN 15662535. doi: 10.1016/j.inffus.2023.101805.

Anders Niels Van Berkel, Mikael B Skov, Jesper Kjeldskov, Gammelgård Larsen, Anders Gammelgård-Larsen, Niels Van Berkel, and Jesper 2024 Kjeldskov. Designing for human-ai interaction: Comparing intermittent, continuous, and proactive interactions for a music application designing for human-ai interaction: Comparing intermit-tent, continuous, and proactive interactions for a music application. in extended abstracts of the chi conference on human factors in computing. *CHI EA '24*, 5 2024. doi: 10.1145/3613905.3650886. URL https://doi.org/10.1145/3613905.3650886.

Johan Boye and Birger Moëll. Large language models and mathematical reasoning failures. Technical report, KTH Royal Institute of Technology, 2025. URL https://arxiv.org/abs/2502.11574.

Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3:77–101, 2006. ISSN 1478-0895. doi: 10.1191/1478088706qp063oa. URL https://www.tandfonline.com/action/journalInformation?journalCode=uqrp20.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 7 2020. URL http://arxiv.org/abs/2005.14165.

Zana Buçinca and Krzysztof Z Gajos. To trust or to think: Cognitive forcing functions can reduce overreliance on ai in ai-assisted decision-making. *Proc. ACM Hum.-Comput. Interact*, 5:21, 2021. doi: 10.1145/3449287. URL https://doi.org/10.1145/3449287.

Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel. *About Face: The Essentials of Interaction Design*. Wiley, Indianapolis, IN, 4 edition, 2014. ISBN 978-1-118-76657-6.

Josh Cowls, Luciano Floridi, and Mariarosaria Taddeo. The challenges and opportunities of ethical ai. In *Artificially Intelligent*. Alan Turing Institute, 2018. URL

https://digitransglasgow.github.io/ArtificiallyIntelligent/contributions/04_Alan_Turing_Institute.html.

Design Council. The double diamond, 2023. URL https://www.designcouncil.org.uk/our-resources/the-double-diamond/. Accessed: 2025-11-03.

Thomas Van Dyke, Vishal Midha, and Hamid Nemati. The effect of consumer privacy empowerment on trust and privacy concerns in e-commerce. *Electronic Markets*, 17: 68–81, 2007. ISSN 1019-6781. doi: 10.1080/10196780601136997.

Luciano Floridi and Josh Cowls. A Unified Framework of Five Principles for AI in Society. *Harvard Data Science Review*, 1(1), jul 1 2019. https://hdsr.mitpress.mit.edu/pub/l0jsh9d1.

Ben Grosser and Søren Bro Pold. Reading the praise/prompt machine: An interface criticism approach to chatgpt. In *Conference Proceedings - Computing X Crisis: 6th Decennial Aarhus Conference, AAR 2025*, pages 259–271. Association for Computing Machinery, Inc, 8 2025. ISBN 9798400720031. doi: 10.1145/3744169.3744194.

Peter A. Hancock, Deborah R. Billings, Kristin E. Schaefer, Jessie Y.C. Chen, Ewart J. De Visser, and Raja Parasuraman. A meta-analysis of factors affecting trust in human-robot interaction. *Human Factors*, 53:517–527, 10 2011. ISSN 00187208. doi: 10.1177/0018720811417254.

Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human Factors*, 57:407–434, 5 2015. ISSN 15478181. doi: 10.1177/0018720814547570.

Zainab Iftikhar, Amy Xiao, Sean Ransom, Jeff Huang, and Harini Suresh. How llm counselors violate ethical standards in mental health practice: A practitioner-informed framework. Technical report, 2025. URL https://www.trytherabot.com/.

Daniel Jurafsky and James H Martin. *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models Third Edition draft*. 3rd edition, 2025. URL https://web.stanford.edu/~jurafsky/slp3/.

Jonathan. Lazar, Jinjuan Heidi. Feng, and Harry. Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann Publishers, an imprint of Elsevier, 2017. ISBN 9780128053904.

John D. Lee and Katrina A. See. Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004. doi: 10.1518/hfes.46.1.50_30392.

Yugang Li, Baizhou Wu, Yuqi Huang, and Shenghua Luan. Developing trustworthy artificial intelligence: insights from research on interpersonal, human-automation, and human-ai trust. *Frontiers in Psychology*, 15, 2024. doi: 10.3389/fpsyg.2024.1382693.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Roger C Mayer, James H Davis, and F David Schoorman. An integrative model of organizational trust. 20:709–734, 1995. URL https://www.jstor.org/stable/258792?seq=1&cid=pdf-.

Melanie J. McGrath, Oliver Lack, James Tisch, and Andreas Duenser. Measuring trust in artificial intelligence: validation of an established scale and its short form. *Frontiers in Artificial Intelligence*, 8, 2025. ISSN 26248212. doi: 10.3389/frai.2025.1582880.

Siddharth Mehrotra, Chadha Degachi, Oleksandra Vereschak, Catholijn M. Jonker, and Myrthe L. Tielman. A systematic review on fostering appropriate trust in human-ai interaction: Trends, opportunities and challenges. *ACM Journal on Responsible Computing*, 1:1–45, 12 2024. doi: 10.1145/3696449.

Donald A. Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, New York, 2013. ISBN 978-0465050659.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, Hoboken, NJ, 4 edition, 2021. ISBN 978-1-292-40113-3.

Donald A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.

Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Hoboken, NJ, 5th edition, 2019. ISBN 978-1-119-54725-9.

Max Tegmark. *Life 3.0: Being Human in the Age of Artificial Intelligence*. Alfred A. Knopf, New York, 2017. ISBN 978-1101946596.

Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 6000–6010. Curran Associates, Inc, 2017. ISBN 1706.03762v7.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 1 2023. URL http://arxiv.org/abs/2201.11903.

Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9:36–45, 1 1966. ISSN 0001-0782. doi: 10.1145/365153.365168. URL https://doi.org/10.1145/365153.365168.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *ICLR 2023*, 3 2023. URL http://arxiv.org/abs/2210.03629.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93, 2020.

John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in hci. In *Conference on Human Factors in Computing Systems - Proceedings*, pages 493–502. Association for Computing Machinery, 2007. ISBN 1595935932. doi: 10.1145/1240624.1240704.

Appendices

NOTE: The questions, prompts, and statements below were asked in Swedish in the interviews.

A Initial Interview & Trial tasks

Consent: Users are aware they are being recorded, how the data is intended to be used, and that they will be anonymous in the report.

A.1 Warm-up

Purpose: Have a quick chat with the user, discuss their use of AI in general.

1. Intro: My research topic is trust in AI, in the context LLM applications. In the project, a platform-integrated chatbot is implemented on the Flowpass admin page. A pre-release version is available for the tests we are about to make.
2. Can you tell me about your work-related use of generative AI tools? (or outside work)? (e.g., ChatGPT, Copilot, Notion AI)
   (a) Follow-up: Any certain feature or workflow that you have been more successful with?
   (b) Follow-up: For which tasks does it bring the most consistent value?
3. "If you could have an AI assistant at work, can you picture something you would want help with?"

A.2 Trial Tasks

*The tasks do not necessarily need to be forced upon the user, he should get to interact with the AI as he likes.

A.2.1 Introduction

1.1 Flowpass (for those not involved with platform) Flowpass is a system for lending out and renting workspaces on short notice. If you are small tech company, for instance, you might only need a space for meeting or so once a month maybe. On the other hand, you have the owners of the space, this can be company with an increase of empty offices since the pandemic, or a long-term renter of a space that don't need the space, but has another couple of months on the contract. Members of the Flowpass platform can cut costs as hosts, and get more flexible office use as renters. The admin page caters to both guest and host organisations.

1.2 Explanation of Chatbot Here you see the interface of the AI, as it is integrated into the Flowpass admin dashboard. It is not just a mockup, it handles requests by using LLMs and integration with backend, so it has access to data on the platform. In the end-product, you as a workspace administrator can ask it any question about the data instead of clicking around on the page.

A.2.2 Task List

Note: Follow-up prompts are allowed to be customized by the user.

1. **Task 1 – Retrieve Workspaces** Ask the AI widget: *"Which workspaces are published?"* Follow-up questions: - *"what setups are under Canadian pride?"*
2. **Task 2 – Check bookings** Ask the AI widget: *"Do we have any upcoming bookings this week?"* Follow-up questions: - *"What about the past week?"* - *"Which space had the most bookings?"*
3. **Task 3 – Review access policy** Ask the AI widget: *"What access policy applies to the workspace at [Location X]?"* Follow-up questions: - "Who currently has access there?"
4. **Task 4 – Comparison** Try to execute one of the previous manually by navigating the GUI. (This is also allowed during the other tasks for validation)

A.3 Observation Points

The session is recorded via video call with face cam and screen sharing, so that the following observations can be noted. Additional observational notes could be taken.

A.3.1 General Interaction

- User appears comfortable initiating interaction with the widget.
- User understands the purpose of the chat interface.
- User formulates questions naturally (without rephrasing multiple times).
- User uses terminology similar to what they use in daily Flowpass work.
- User expresses confusion or uncertainty (e.g., long pauses, "I'm not sure what to ask…")?

A.3.2 Task Performance - noted for each task

- Task completed successfully (correct information retrieved).
- Number of prompts or reformulations needed.
- Evidence of trial-and-error interaction.
- Time to task completion with AI (seconds).
- Time to task completion with GUI (seconds).
- User compares AI results with GUI output (explicitly or implicitly).
  - How long does it take?
  - Which method feels more intuitive?

A.3.3 Perceived Trust and Confidence

- User spontaneously questions the AI's accuracy or data source.
- User asks for clarification or verification.
- User comments on tone, helpfulness, or confidence of responses.
- Emotional cues (e.g., smiles, frustration, surprise).

A.4 Post-Task Questionnaire

The aim is to discuss the AI widget briefly after the trial, and get quantitative measures on perceived trust, usability, and overall experience with it. All statements use a 5-point Likert scale: *(1 = Strongly disagree, 5 = Strongly agree).*

A.4.1 Trust and Confidence

1. I feel confident that the chatbot provided accurate information.
2. I trusted the chatbot to access the correct data from Flowpass.
3. The chatbot's tone and behavior made it feel reliable.

A.4.2 Usability and Experience

1. It was easy to phrase my questions so the widget understood me.
2. The responses were clear and easy to understand.
3. The response time made it felt natural and efficient.
4. I found the AI widget faster to use than the regular interface.

A.4.3 Overall Impression

1. I would like to see more AI features integrated in Flowpass.
2. I could imagine using this tool regularly in my daily tasks.

Final open-ended questions

- Which parts of its behavior felt confusing?
- Did it's communication feel clear? Why/why not?

B Second Interview & Evaluation

NOTE: The questions below were prepared in advance, and asked in Swedish in the interviews.

B.1 Warm-up

- What is your most common use of Flowpass?
- What is the core of your role, and what requires the most engagement?
- Is there any information in the platform or in external documents that you often need to double-check?

B.2 Prepared prompts (for live testing)

B.2.1 Basic

- Which setups exist at `[Workspace X]`?
- Which bookings did we have yesterday at `[Workspace X]`?

- Which companies have booked `[Setup X]` in the last 7 days?
- Which access policy applies for `[Guest org. Y]` at `[Workspace X]`?
- Does `[Organization X]` have access to `[Workspace X]`, and what are they allowed to do?

B.2.2 Complex

- Give me a short reception brief for today: important bookings, potential conflicts, and things to keep an eye on.
- Summarize utilization for `[Workspace X]` over the last two months: quiet vs. high-demand periods.
- Give me a quick report for `[Workspace X]` for the last 30 days: utilization + revenue + anomalies.
- Compare `[Setup X]` vs `[Setup Z]` per month for the last 6 months: what is trending up/down, and when?

B.3 Post-task questionnaire: S-TIAS (Short Trust in Automation Scale)

After running 1–2 prompts, participants rated the following statements on a 1–7 Likert scale (1 = strongly disagree, 7 = strongly agree).

1. I feel confident in the AI assistant.
2. The AI assistant is reliable.
3. I can trust the AI assistant.

B.4 Follow-up questions (TIAS-inspired)

B.4.1 Anchor and calibration

- Why did you choose `[X]` rather than a lower number?
- What would need to change for you to rate it one step higher?
- In which situations would you use it without double-checking? In which situations would you always double-check?

B.4.2 Potential distrust / risk cues (covers TIAS 1-5

- Did anything feel *off* or like it might be making things up? What specifically triggered that feeling?
- Did it ever seem like it could mislead you without you noticing? When would that be most harmful in reception work?
- Did you feel wary at any point? What made you cautious (tone, missing references, too much certainty, wrong entity/date)?
- If the system gave a wrong answer, what would the likely consequence be in your workday?

B.4.3 Dependability and consistency (covers TIAS 6, 9-11)

- Which parts felt robust or consistent?
- Does it feel equally dependable for basic lookups vs. summaries/reports? Why/why not?
- What makes a response feel "stable" to you (structure, references, concrete numbers, ability to verify)?

B.4.4 Integrity and "working in my interest" (covers TIAS 8)

- Did it feel like the assistant prioritized the right thing: correctness and usefulness over producing a smooth answer?
- When it is uncertain, how should it behave to feel more honest (admit uncertainty, ask a clarifying question, show assumptions)?

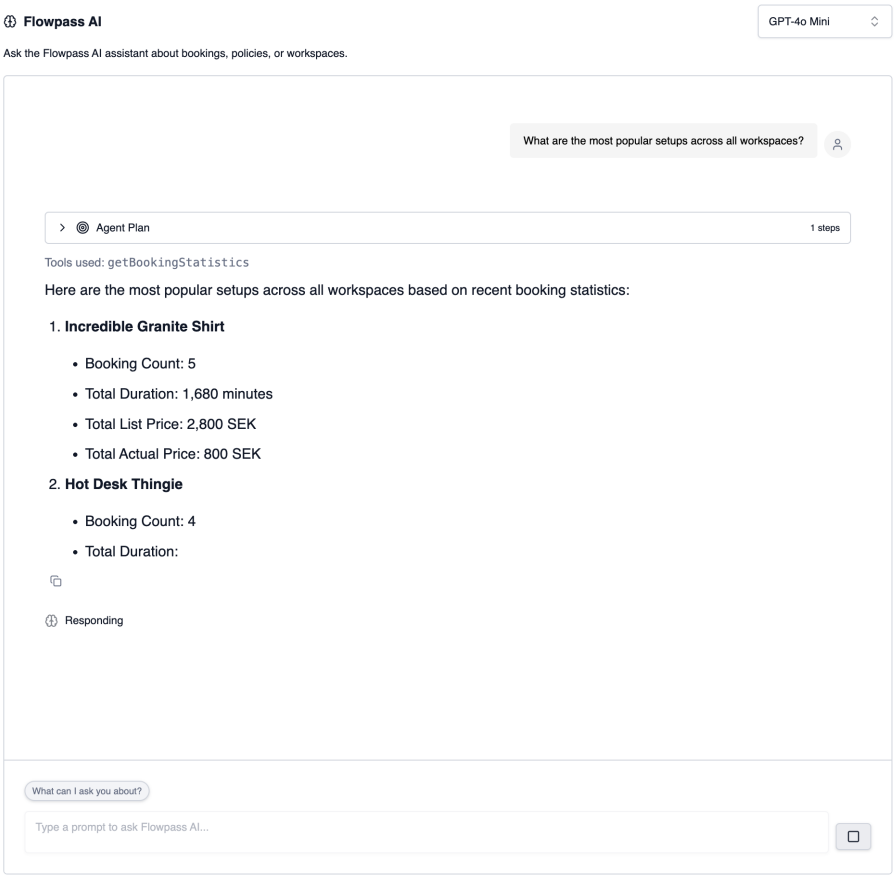
B.4.5 Security / feeling safe using it (covers TIAS 7)

- Do you feel safe using this in live situations? Why/why not?
- What UI elements would increase your feeling of safety (e.g., links to source views, tool-call traces, warnings, "stop" button)?

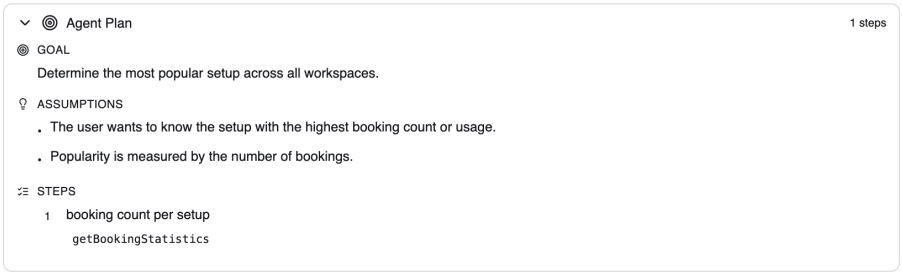
B.4.6 Understanding and mental model (covers TIAS 12)

- How do you think the assistant produces its answers (in broad terms)?
- What is hardest to predict about its behavior?
- What would help you build a clearer mental model (examples, explanations, showing which data it used)?

C Final implementation figures



(a) Flowpass AI response.



(b) Agent planning output.

Figure 11: Chatbot interaction

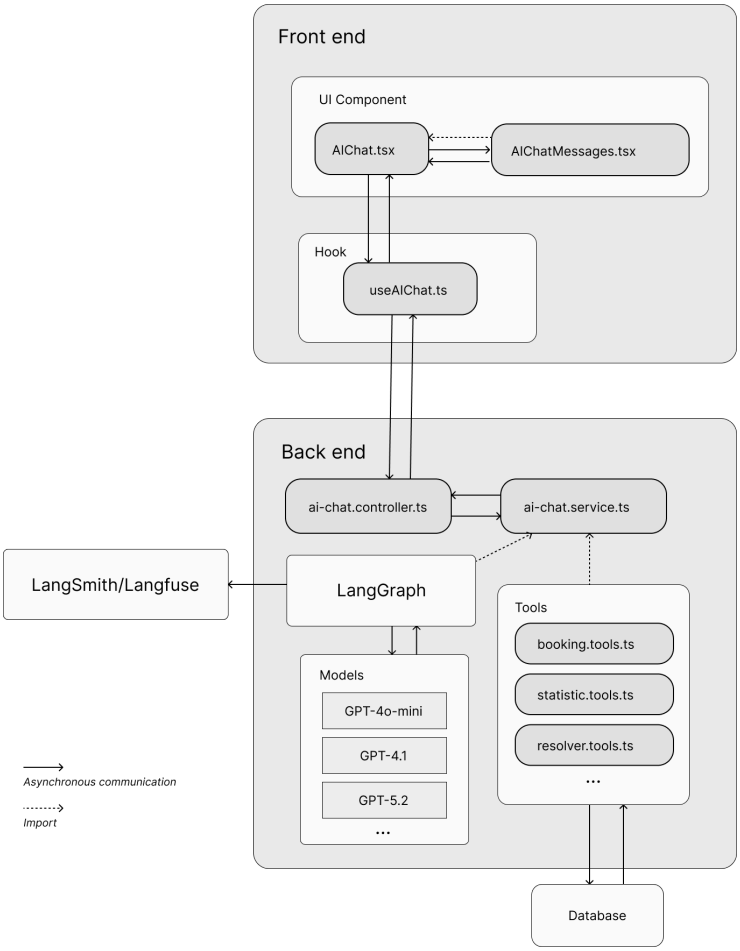


Figure 12: Chatbot Architecture

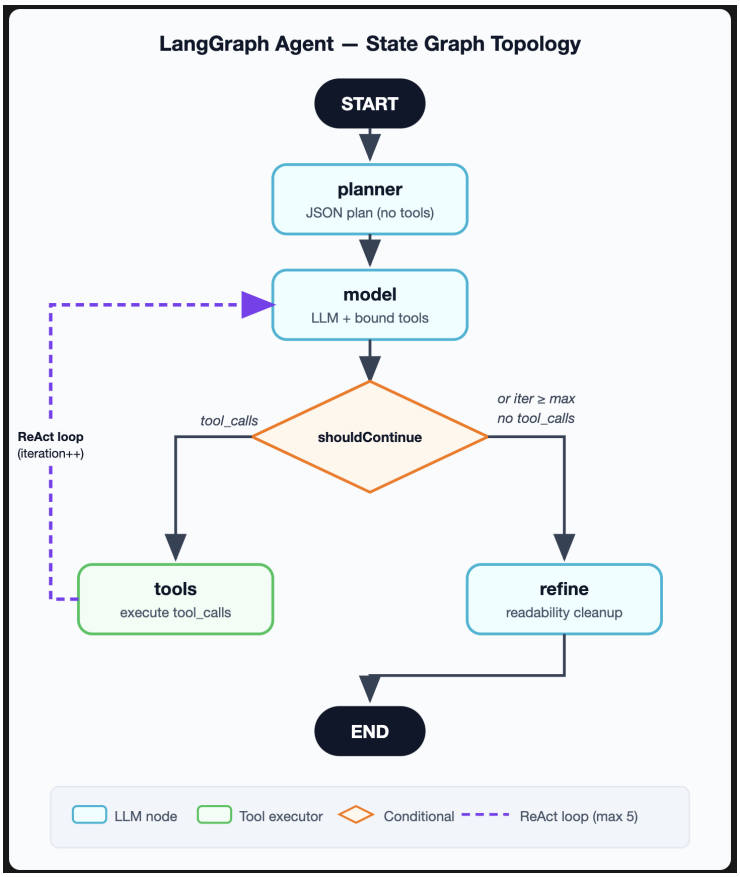


Figure 13: Final Agent's nodes