Electronic Communications of the EASST
Volume 85 Year 2025

deRSE25 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany

Edited by: René Caspart, Florian Goth, Oliver Karras, Jan Linxweiler, Florian Thiery, Joachim Wuttke

User Interface Innovations for LLMs – Insights from a Design Thinking Workshop at deRSE25

Maximilian Frank, Simon Lund

DOI: 10.14279/eceasst.v85.2716

License: 🅭 🅯 This article is licensed under a CC-BY 4.0 License.

User Interface Innovations for LLMs – Insights from a Design Thinking Workshop at deRSE25

Maximilian Frank[1]* **and Simon Lund**[2]*

[1]maximilian.frank@psy.lmu.de (ORCID: 0000-0002-8140-3519)
[2]simon.lund@lmu.de (ORCID: 0009-0006-5907-1577)

Ludwig-Maximilians-Universität München
Munich, Germany

**both authors contributed equally*

Abstract: Large Language Models have become widely adopted tools due to their versatile capabilities, yet their user interfaces remain limited, often following rigid, linear interaction paradigms. In this paper, we present insights from a design thinking workshop held at the deRSE25 conference aiming at collaboratively developing new user interface concepts for LLMs. It was motivated by our previous efforts for improving LLM user interfaces which we will describe in this work as well.

The workshop is embedded in the human-centered design (HCD) process, an iterative, user-focused methodology that emphasizes continuous development through user feedback. During the workshop, participants identified common use cases for LLMs as well as strengths and shortcomings of current LLM interfaces, and created visualizations of new interaction concepts emphasizing flexible context management, dynamic conversation branching, and enhanced mechanisms for user control.

Broader implications for future LLM interface development are discussed as we advocate for increased attention to UI innovation grounded in user-centered design principles.

Keywords: LLM, User Interface, UI, Interaction, Limitations,
Human centered design, HCD, Design thinking, Workshop

1 Introduction

The era of Large Language Models (LLMs) is rooted in the publication of *Attention Is All You Need* [VSP+17] in 2017, which introduced the transformer architecture for machine translation tasks. While the underlying components were not entirely new – they build upon years of basic research in fields in deep learning – the paper synthesized them into an effective design compared to existing architectures (e.g., recurrent neural networks). The broader impact of transformers and widespread adoption, however, came only after OpenAI released its GPT-3 model along with a web interface. This interface allowed users to interact with LLMs and brought the architecture into mainstream use [ZZL+25, NKQ+24].

Today, LLMs are used across diverse applications due to their versatile range of capabilities: world knowledge, multilingual support, text comprehension, instruction following, in-context learning, reasoning, user interaction, self-improvement, and tool utilization [MMN+25]. Applications include question-answering assistants, multistep reasoning agents, creative content generation, programming support, data analysis, research tools, language learning, and integration into games [KHM+23, MDW+23, KSK+23, GTZ+24].

While the capabilities of LLMs are rapidly evolving, the evolution of UI has not kept pace. Most LLM interfaces still follow a rigid conversation structure constrained by sequential integrity requirements – preventing contextual restructuring and masking parts of the conversation. This may increase the likelihood of model hallucinations, e.g., due to unalterable inconsistencies or focus shifts across multiple topics across extended interactions. For instance, in development contexts, when dialogue involves peripheral discussions – explaining code, identifying bugs, or managing feature requests – models struggle to discern and prioritize essential information. Quality may further degrade when conversations exceed context length limitations, resulting in information loss through truncation or compression. As a consequence, users may restart conversations and reconstruct context, potentially disrupting workflow continuity and affecting trust in model outputs.

In this paper, we present the results of a design thinking workshop at deRSE25 on the topic of user interface development for LLMs. After a brief introduction into the operational principles of LLMs, we highlight the strengths of a user-centered design for LLM interfaces and present the results of the workshop. We show how the design solutions generated in the workshop help steer the future development of an UI prototype for LLMs we previously developed, and discuss currently unutilized potentials in interface development.

2 Background and Motivation

2.1 Functioning of LLMs

LLMs are stochastic models that predict the next token based on preceding input [BGMS21]. A token is a basic unit of text that may correspond to part of a word, a complete word, or multiple words depending on the tokenization algorithm [KR18]. For example, the word ‘unbelievable’ might be split into tokens like ‘un’, ‘believe’, and ‘able’.

An LLM does not maintain internal state between interactions. Instead, it operates on a fixed-length context window containing the current prompt, conversation history, and system prompt (instructions about its role and answer behaviour). Context length limits stem from hardware constraints (computational resources, VRAM requirements) and training limitations (maximum token counts models process) [WWL$^+$24]. When processing input, the model applies transformer decoder blocks with self-attention mechanisms (Figure 1) to generate output token-by-token until producing a stop token. The generated response is then appended to the context window for subsequent interactions.

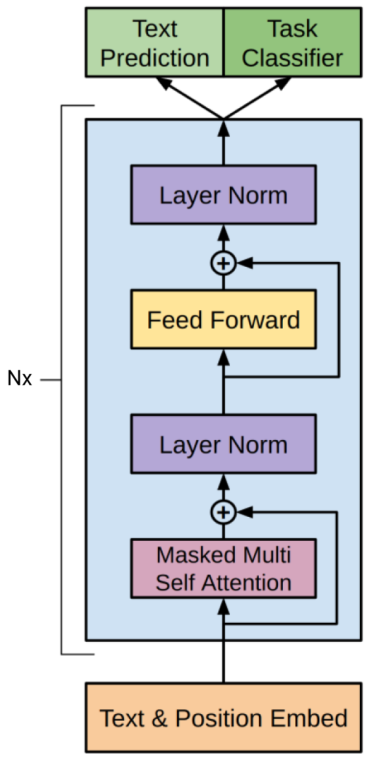

Figure 1: Decoder-only architecture, showing how LLMs process input tokens through multiple (N) transformer blocks to generate output tokens sequentially. The self-attention mechanism allows each token to attend to all previous tokens in the context window. [RNSS18]

The model's capability derives from parameters tuned during pre-training on large text corpora [GBB+20], encoding linguistic patterns, factual knowledge, and reasoning capabilities that enable contextually meaningful responses. However, this is no guarantee for factual accuracy or logical consistency, as models may generate plausible-sounding but incorrect or inconsistent information, commonly referred to as hallucinations[1] [HYM+25].

2.2 Limitations of Current LLM User Interfaces

Recent research has extensively documented various usability challenges and limitations inherent in current Large Language Models (LLMs). Prominent among these issues are the persistent occurrences of hallucinations, misinformation, and biased outputs, where LLMs confidently generate plausible yet incorrect information [BGMS21, SDG+24].

Such inaccuracies are exacerbated by inadequate calibration of trust and user confidence, leading users to either trust erroneous responses or to mistrust correct ones [VRM24, MMBK24].

Further compounding these challenges are mismatches between users' mental models and the operational realities of LLMs, resulting in unmet expectations [GSG21, WWPY25]. Additionally, accessibility and inclusivity remain significant concerns, with LLMs frequently perpetuating biases or failing to accommodate diverse user needs, further hindering their widespread, equitable adoption [GKD+23, MMR24].

While a considerable body of work focuses on enhancing the technical and functional reliability of LLMs through methods such as calibrating models and external knowledge integration [SDG+24, LPP+20], limited research has specifically addressed improvements to the user interface (UI) through which end-users interact with these systems.

The dominant paradigm in LLM interaction is the linear chat interface, where conversations progress sequentially without structural modification. Such interfaces limit users to a single conversation thread, with each message building linearly on previous exchanges. This approach spans commercial platforms (Claude, ChatGPT, Gemini, Perplexity), specialized applications (Google NotebookLM, DeepL Write), developer tools (GitHub Copilot, Cursor, Windsurf) and open-source alternatives (OpenWebUI, LMStudio, JAN, Aider).

To enhance user experience within this restrictive interaction framework, interfaces have implemented various enhancements. For example, some interfaces offer limited branching capabilities, allowing users to edit previous messages to create alternative paths, though typically at the cost of erasing the subsequent conversation history to the original message. Additional incremental improvements and interface features are summarized in Table 1.

However, these enhancements focus mainly on extending LLM capabilities rather than addressing interface limitations. Current designs cannot properly counteract context fragmentation, fail to distinguish between important long-term and merely short-term relevant information, and impose burdensome message management on users. For example, users cannot selectively mask irrelevant parts of previous exchanges, dynamically reorder messages, or optimize the context window based on relevance rather than recency. Furthermore, it also constrains exploration of problem spaces that would benefit from parallel inquiry paths, particularly for professional con-

[1] There is ongoing debate regarding this terminology. 'Hallucination' medically refers to 'the perception of an entity or event that is absent in reality' ([MP13] as cited in [HYM+25]). Some scholars suggest alternative terms like 'bullshitting' for inaccurate AI outputs [HHS24].

Table 1: Supplementary Features in LLM Interfaces

Feature	**Description**
Multimodal capabilities	Support for diverse input and output types (text, images, audio, video)
Tool integration	Web search, code execution, and protocols (like Model Context Protocol) to interact with external services
Separated content stages	Content areas (like Claude artifacts) distinct from conversation flows
Project workspaces	Environments where files are automatically added to conversation context
Writing style customization	Preset and custom writing styles to control tone, formality, or voice (e.g., concise, explanatory, formal)
Profile preferences	Account-wide settings that apply across all conversations

texts where users engage in extended problem-solving.

Recent research has begun addressing these limitations. Masson et al. (2024) proposed DirectGPT [MMCV24], a system that applies direct manipulation as an additional layer on linear chat interfaces like ChatGPT. Their approach includes an undo mechanism and a customizable toolbar to store frequently used commands (e.g., replace selected word with synonym) which can be used to update text in-place. So rather than progressing through continuation of the conversation, they allow users to update the current state continuously with visual feedback for changes. While DirectGPT demonstrates the potential benefits of moving beyond linear chat interfaces, our work explores a more comprehensive restructuring of the interaction model to address context management, parallel inquiry paths, and dynamic content organization.

2.3 Human-Centered Design Process

The human-centered design (HCD) process provides a methodological framework for developing interactive systems that prioritize user needs. Standardized in DIN EN ISO 9241-210 [fNB10], HCD diverges from linear development approaches like the waterfall model by Royce [Roy87] through its iterative, user-focused process.

The HCD process comprises the four phases: understanding the usage context, defining the usage requirements, drafting design solutions and evaluating the design solutions against these requirements. These phases are run through iteratively until an optimal result is achieved. The process is visualized in Figure 2.

For our work, HCD offers a structured approach to user-centric interface development for LLMs; whereby this paper focuses on the initial conceptualization phase of that process. The HCD paradigm was chosen because extensive standards and empirical work show that following HCD principles yields measurable benefits – improved usability, user satisfaction, and task performance, with fewer design errors and mismatches between system behaviour and user expectations [fS19, VMSC02, Mag01, GGB$^+$03, MVSC05]. It emphasizes iterative evaluation with real users, thereby reducing design errors and mismatches between system behaviour and

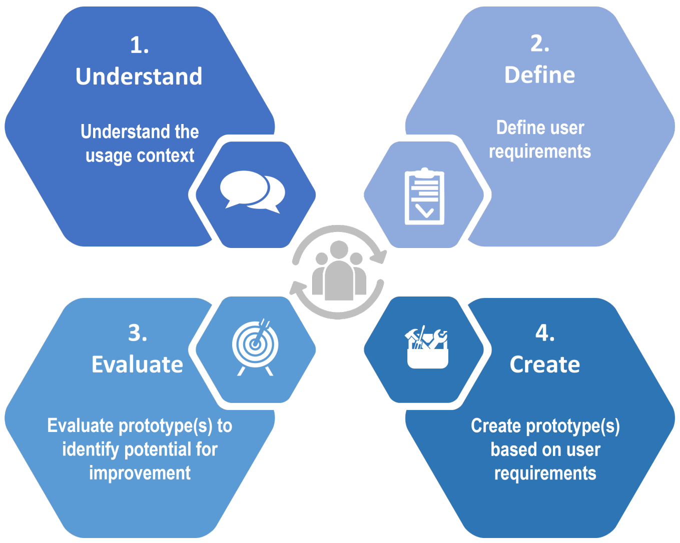
1.
Understand
Understand the usage context
2.
Define
Define user requirements
3.
Evaluate
Evaluate prototype(s) to identify potential for improvement
4.
Create
Create prototype(s) based on user requirements

Figure 2: Phases of the human-centered design process (own illustration)

user expectations [Nor86, Boy11]. Moreover, HCD enhances situation awareness, adaptability to varied user populations, and alignment with cognitive processes – factors that are highly relevant for the design of LLM interfaces [Boy11, End96]. By grounding interface design in the real needs and contexts of users, HCD contributes not only to system efficiency but also to safety and long-term system acceptance [Bil91, HW05].

3 Workshop Structure

To ensure user participation in the development of a new LLM interface, we have organized a design thinking workshop titled 'Breaking the Chat Barrier' at the deRSE2025 in Karlsruhe. Design thinking, utilizing the logic of the HCD process, is an iterative process of empathizing with users, defining their core challenges, ideating solutions, prototyping those ideas, and testing them for feedback [Dor11]. The workshop gathered LLM use cases and identified the challenges and barriers that users face. Based on these insights, we collaboratively identified requirements for a dynamic chat interface and developed different visualization concepts. The workshop therefore contributed to the first three phases of the HCD process, as described in the previous chapter. The workshop was held on the first day of the conference and lasted 60 minutes. The workshop materials are available on Zenodo.

Since deRSE is attended by scientists from diverse disciplines who work with research software – whether as programmers or in other roles – this audience provides two key advantages for our workshop: First, most attendees are likely already using LLMs in their daily or private work and therefore bring firsthand experience with this technology. Second, their general programming background might be helpful to imagine improvements for LLM interfaces. The workshop was very well attended, with an estimated 40 to 50 participants. Because the workshop was open to all conference participants and registration was voluntary, we cannot report any further demographic information about the attendees.

The workshop consisted of an introductory presentation and group work, which in turn was divided into several tasks. The introduction should bring all participants to a similar level, regardless of their prior knowledge. To this end, we gave a brief introduction to the architecture of LLMs, human-computer interaction frameworks and the HCD process. In addition, interfaces of existing LLMs were presented, and their limitations were demonstrated using the user story of a persona, the fictitious programmer Frank. For further details of the introduction, please refer to the slides linked above.

The group work consisted of three different tasks, each of which is reproduced here in the wording used in our presentation. At the beginning of the workshop, we asked the participants **'how you use LLMs'** and collected their answers on a whiteboard. This question is in line with the first phase of the HCD process to understand the context of use. No restrictions were placed on the type of answers, so participants could describe both business and personal use cases. In a follow-up question, we asked the participants to **'note what you like and dislike about working with LLMs'**. This question serves the purpose of defining user requirements for LLMs and corresponds to the second phase of the HCD process.

As mentioned above, screenshots of the UIs of various LLMs were shown in our introductory presentation, whereupon we asked the participants **'what do these [user] interfaces have in common'**? This question served as a ramp for the last and most important task, a group work, in which the participants were asked to create visualizations for new UIs based on the guiding question **'How to overcome the limitations of chat based LLM interfaces to enable new potentials?'**. This technique is also called a *How-Might-We-question* and is often utilized as a design thinking method to set a user centered mindset for brainstorming and prototyping. The participants were divided into seven small groups of 4 to 6 people each to visualize the UI using a flipchart. They were given 15 minutes for this task in total, 5 minutes for an initial discussion and

10 for the visualization process itself. The design solutions were then presented and discussed by the entire group, bringing the workshop to a close. This last task corresponds to the third phase of the HCD process, the prototyping[2].

[2] Please note that this workshop focuses solely on the first cycle of a HCD process. Accordingly, with the term "prototyping" we refer here to paper-based UI concept drawings rather than software implementation. In line with HCD, a functional prototype is developed only in subsequent cycles, after a clear design solution has emerged.

4 Results

4.1 Evaluation of the Workshop Results

For the evaluation of our workshop outputs, we first transcribed every written response – from the whiteboard notes and flip chart cards – verbatim into a single document. The documentation was done manually and without software for quantitative analysis, as the dataset was moderate in size. We preserved participants' original wording to avoid introducing interpretive bias. Next, we conducted a two-stage, inductive thematic grouping consistent with conventional qualitative content analysis (i.e., deriving categories directly from the data without imposing preconceived codes) as described by [HS05]. In the first stage, we reviewed each item and clustered those whose meaning was unmistakably identical, grouping them under provisional labels. In the second stage, we assigned these provisional clusters to higher-level categories, again merging only when semantic overlap was clear without interpreting latent meanings. Any responses that could not be confidently aligned with other items remained as standalone categories.

We followed the following rules for creating our response tables:

- **Ordering within categories:** Entries are listed in descending order by frequency of mention; ties are ordered randomly.
- **Whole-response assignment:** If a single response addressed multiple issues, it was not split across categories; instead, the response was preserved intact and assigned to the single most fitting category.
- **Alternative category notes:** Where an equally justifiable alternative category existed, we indicated it in italics within parentheses immediately after the entry.
- **Table conventions:** Column headers explain each category and display the number of responses. Table cells contain the participants' verbatim responses, so their specificity varies according to how each individual chose to express their point.

This conservative, semantic-level approach parallels the affinity-diagramming method widely used in human-computer interaction research to organize unstructured user data into theme clusters for design insights [BH97]. By staying very close to the text, our summary reflects the diversity of participants' answers while reducing redundancy and highlighting the most salient themes.

4.2 Participants' Experience with LLMs

Scientists' uses of LLMs fell into two main domains: *coding* (22 mentions) and *human-language tasks* (34 mentions). Within coding, the bulk of responses (14 mentions) described the coding process itself – writing snippets for apps and web interfaces, auto-completing functions, checking syntax on the fly, and even refactoring existing modules – while a smaller set (8 mentions) focused on code assisting, such as generating documentation, explaining library functions, and troubleshooting error messages.

In the realm of language, participants' needs split across five distinct uses. Seven attendees relied on the model for narrow, goal-oriented text creation – drafting emails, turning bullet points into prose, polishing paper titles, or even assembling a grocery list – while five embraced its wide-creative potential by brainstorming workout plans, interview questions, or exam prompts. Eleven mentions of text improvement highlighted LLMs' strength in proofreading, clarity edits, and stylistic rewrites, and another seven responses tapped into their ability to unpack and summarize complex material, answer detailed questions, or simplify technical jargon. Finally, four miscellaneous uses ranged from practicing a foreign language in tandem to extracting text via OCR for literature overviews.

Together, these patterns underscore that scientists lean on LLMs both as practical coding companions and as versatile language partners – able to deliver precise, task-driven outputs or to spark broader creative and analytical support.

Table 2: Responses to 'How you use LLMs'

Coding (22)		**Human language (34)**				
Coding process (14)	**Code assisting (8)**	**Text creation - goal narrow / task oriented (7)**	**Text creation - goal wide / creative oriented (5)**	**Text improvement (11)**	**Text understanding (7)**	**Text miscellaneous (4)**
Writing code (6x) (e.g., small apps, web interfaces, config files)	Writing documentation (2x)	Write text based on bullet points / keywords (2x)	Idea generation (2x)	Quality improvement (6x): (e.g., proofreading, rewriting)	Summarization (4x)	Tandem partner for language learning
Generate code snippet (5x)	Code explanation (2x)	Writing emails (2x)	Organizing workout schedule	Rephrasing (4x):	Provides context to certain topics	Gathering literature (paper, references) and brief summary for new topics
Code completion & generation	Finding & understanding library function (2x)	Job application writing	Interview preparation	Make suggestions on contents of a document	Explanation of research topics	Text recognition from images
Checking syntax while programming	Solutions for resolving programming issues	Generating / tweaking of paper titles	Homework creation / question creations for exams		Answer questions about content in our textbook (e.g., what is the CEO of a hospital responsible for)	The investigation of the analytical solution of the local strain around nanoparticles in incompressible elastic material
Code refactoring	Programming help *(could also belong to category 'Coding process')*	Preparing grocery list			Simplifying text for better understanding *(could also belong to category 'Text improvement')*	

Participants highlighted six ways in which LLMs enhance their workflows. First, on the *usability* front (3 mentions), users liked the models' simplicity and conversational feel – 'easy to use' (2×), straightforward setup even for local deployment, and an interactive, dialogue-like interface that feels more natural than form-based tools.

Closely related were comments about *user experience* (4): several appreciated that the AI is 'always in a helpful mood', and that engaging with it requires minimal mental overhead ('we don't need to think a lot'). Others noted how it unlocks new workflow possibilities and spares them from tedious tasks – one participant admitted, 'I don't like writing documentation, so AI is really great there'.

Efficiency and productivity were the most frequently cited advantages (16 mentions). Participants particularly emphasized time savings (9 mentions), but also highlighted the model's attention to detail, the ability to receive multiple alternative suggestions in a single response, and generally reliable accuracy.

Next, on the *quality of outputs* (9), attendees positively mentioned how the model consistently produces polished, ready-to-use artifacts. Two participants specifically noted that it 'improves text quality', while another pair singled out its 'professional communication style'. Beyond prose, users found the LLM 'works reasonably' for basic code refactoring, debugging, documentation, and translation tasks. The participants also liked its nuanced 'attention to detail and multiple suggestions', described outputs as 'mostly accurate', and highlighted its proficiency in correction, summarization, and stylistic edits – especially for concise summarization tasks.

On the *functional capabilities* side (8 mentions), participants noted that the LLM effectively follows common conventions and standards, retains instructions across multiple turns, and often produces useful output even from minimal prompts. Participants found the model to be adept at answering simple, direct questions and appreciated its depth of topic knowledge as well as its syntactic accuracy.

Please note, that we differentiate these two categories as follows: while **quality of outputs**, captures how well each individual response is crafted (e.g. accuracy, style, level of detail), **functional capabilities**, by contrast, describe the broader classes of tasks the model can solve reliably rather than the fine-grained quality of any single answer.

Finally, in *knowledge and learning support* (3), users pointed out how LLMs lower the barrier to exploring unfamiliar fields – providing broad coverage of topics, making it easy to dive into new areas, and even decoding cryptic error messages in R or other languages.

Together, these highlights show that scientists value LLMs not only for what they can do, but also for how seamlessly they integrate into both coding and research tasks – streamlining work while maintaining an approachable, user-friendly experience.

Table 3: Responses to ‘Note what you like about working with LLMs’

Usability (3)	**User experience (UX) (4)**	**Efficiency & productivity (16)**	**Quality of outputs (‘How good and usable is the actual generated output?’) (9)**	**Functional capabilities of LLMs (‘What kinds of tasks can the LLM perform effectively?’) (8)**	**Knowledge & learning support (3)**
Easy to use (2x)	Always in a helpful mood	Time savings (9)	Improves text quality (2x)	Follows common conventions and standards programming (3x)	Wide topic coverage and reduced effort for learning new topics
Ease of setup & local usage	We [the user] don’t need to think a lot	Fast (5x)	Professional communication style (2x)	GPT-Projects: Remembering the instructions (No need to instruct again for same task)	Easy to dig into new topics
Interactive ‘conversation’-like	New possibilities due to workflow changes	Efficiency	Works reasonably for simple forms of code refactoring, debugging, documentation, translation	Even with a small prompt, LLMs can usually expand a lot which is really good	Explains cryptic R error messages very well
	I don’t like writing documentation, so AI is really helpful in overcoming the burden	Sometimes increased productivity	Attention to detail and providing multiple suggestions	Very useful for simple questions with direct answers	
			Mostly accurate	Deep knowledge on the topic and syntax	
			Good at correction, summarizing, and style changes of text	Builds boilerplate code rapidly, recommends libraries and gives corrections for specific cases	
			Good for summarization [of] task		

Participants surfaced a range of dissatisfactions with LLMs that fall into five main areas. First, *functional capabilities and usability* (15 mentions) topped the list. Several users found outputs ‘wordy’ or ‘repetitive’ (3×), and many noted that the model sometimes ‘ignores coding instructions’ (2×) or ‘doesn’t understand the intention of the question’. Others mentioned persistence errors – losing context mid-conversation – and pointed out that some tools ‘only work well with widely used software’, limiting their flexibility. Only one person criticized a lack of usability, while mentioning that ‘copy / paste by hand via web interface is not very practical’.

Closely related was *output reliability and accuracy* (13). ‘Hallucinations’ (4×) and ‘faulty code’ (4×) were recurring pain points, alongside general ‘factual mistakes’ (3×) and the sense that ‘the solution suggested is wrong’. Even minor inaccuracies eroded trust, reinforcing that users need dependable, verifiable results.

With growing awareness of data practices, *privacy, environmental, and social concerns* also emerged (13). Three mentions each for ‘data safety’ and worries about ‘exploitation of labour and knowledge for training’ signalled unease about how user inputs and large-scale data harvesting occur. Some participants even questioned ‘companies using data for nefarious reasons’, underscoring ethical considerations beyond mere tool performance.

Under *explainability and transparency* (5), attendees criticized the opacity of model internals: ‘lack of transparency’ and ‘lack of provenance’ each featured twice, and several noted it’s ‘unclear what model is the best for a use case’. Without clear insight into how decisions are made, users felt handicapped when evaluating or debugging the LLM’s reasoning.

Finally, *disillusionment and saturation* (9) captured deeper, more subjective concerns: five participants described a ‘loss of cognitive engagement’ (‘if I always rely on AI, I learn less’), while others warned of ‘false security’, ‘AI dependency’, and fear that ‘people become dumb’ by outsourcing thinking. One user even remarked that they ‘can identify LLM-generated text, and its ever-present style everywhere’, hinting at early signs of fatigue and frustration.

Together, these critiques illuminate not just technical shortcomings, but broader worries about trust, ethics, and the evolving role of AI in society.

Table 4: Responses to ‘Note what you dislike about working with LLMs’

Lack of functional capabilities (15) / usability (1) [3]	**Output reliability & accuracy (13)**	**Data privacy & environmental & social concerns (13)**	**Explainability & transparency (5)**	**Disillusionment / saturation (9)**
Wordy / repetitive (3x)	Hallucinations (4x)	Data safety (3x)	Lack of transparency (2x)	Loss of cognitive engagement (5x) (e.g., if I am not mindful, I do not learn, encourages use without understanding)
Persistence errors (e.g., losing or forgetting information after long discussions) (2x)	Faulty code (4x)	Privacy (3x)	Unclear what model is the best for a use case (e.g., not specified on Huggingface leaderboard)	False security
Ignoring coding instructions (2x)	Factual mistakes (3x)	Copyright issues (2x)	Lack of explainability	AI dependency concerns
Only works well with widely used software	Not always accurate (2x)	Companies using data for nefarious reasons	Lack of provenance	People becoming dumb, using it everywhere
Doesn’t understand the intention of the question	The solution suggested from LLM is wrong	Exploitation of labour & knowledge for training AI-System (great tech, but a system based not on openness + quality, etc.)		I can identify LLM-generated text and it’s everywhere
Settings dependent on LLMs	Sometimes gives wrong conclusion	Waste of natural resources		
Might do incorrect debugging	Verification of output needed	Too much hardware required to train LLMs *(could also belong to category ‘Lack of functional capabilities’)*		

Table 4: Responses to 'Note what you dislike about working with LLMs'

Lack of functional capabilities (15) / usability (1) [3]	**Output reliability & accuracy (13)**	**Data privacy & environmental & social concerns (13)**	**Explainability & transparency (5)**	**Disillusionment / saturation (9)**
Tendency to please (rather give an incorrect answer than state that something doesn't work)	Often wrong (even so slightly)	It is often visible that the LLM was mainly trained around English language materials *(could also belong to category 'Lack of functional capabilities')*		
Finetuning didn't work for us				
Knowledge required on how to prompt efficiently				
Uncensoring is annoying *(could also belong to the category 'Data privacy & environmental & social concerns')*				
Copy / paste by hand via web interface is not very practical				

[3] Whilst *usability* is its own category, we have included the only answers to this area in the same column as the *functional capabilities* to ensure better readability of the whole table.

4.3 Participant-Designed UI Solutions

For the last task of the workshop, we divided the participants into groups and asked them to draw concepts for new LLM interfaces. This resulted in seven UI solutions, which are presented below. The drawings were then photographed and recreated in a standardized graphic style to enhance readability. The images of the original drawings can be found in the Appendix A.

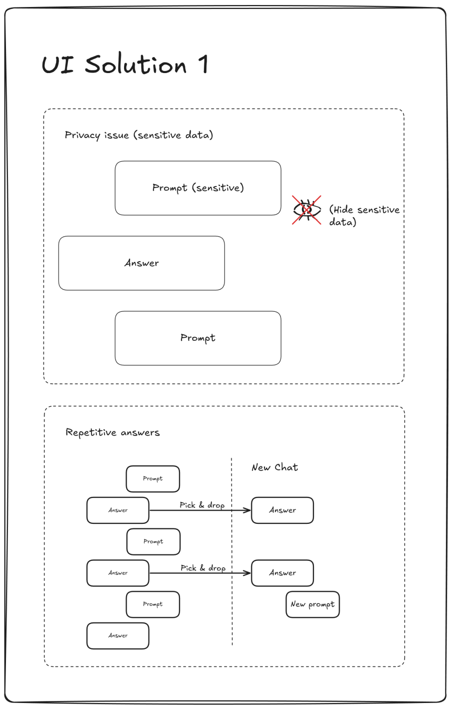

Figure 3: Digitized UI Solution 1

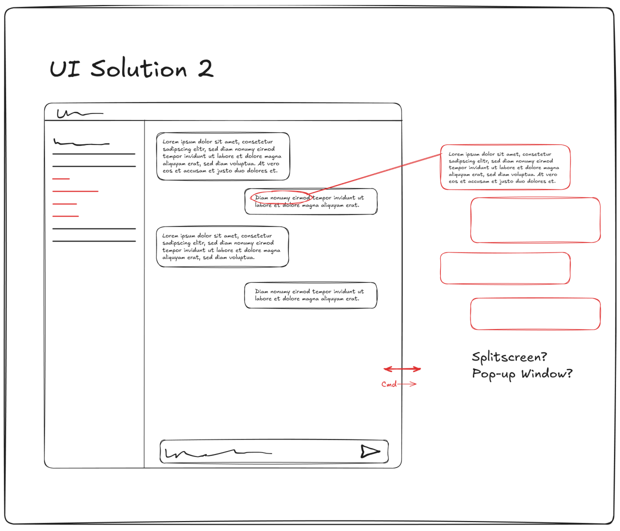

Figure 4: Digitized UI Solution 2

UI Solution 1 (see Figure 3). This concept introduces a horizontal panel-based chat interface, where each panel represents a distinct chat. Users can branch their conversation by dragging only the answers they find useful into a new panel to the right, creating a new chat context built from selected outputs. Repetitive or unsatisfactory answers are left behind, allowing users to curate and continue with a ‘clean’ conversation history. The upper part of the graphic, which sketches a privacy function for hiding sensitive data, is not considered here, as developing new privacy features is out of scope for our current interaction concepts.

UI Solution 2 (see Figure 4). This concept explores an interface where hovering over individual prompts or answers allows users to create branches, opening a secondary conversation as a sub-branch without leaving the main thread. These sub-branches function similar to threaded forms of communication, as in popular tools like Slack. Expanded sub-branches can be collapsed again to maintain clarity and reduce clutter. On the far left, a timeline visually summarizes the history of all ongoing and past LLM conversations.

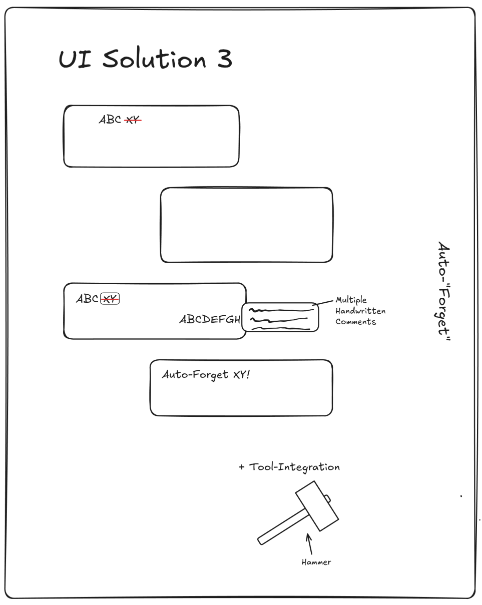

Figure 5: Digitized UI Solution 3

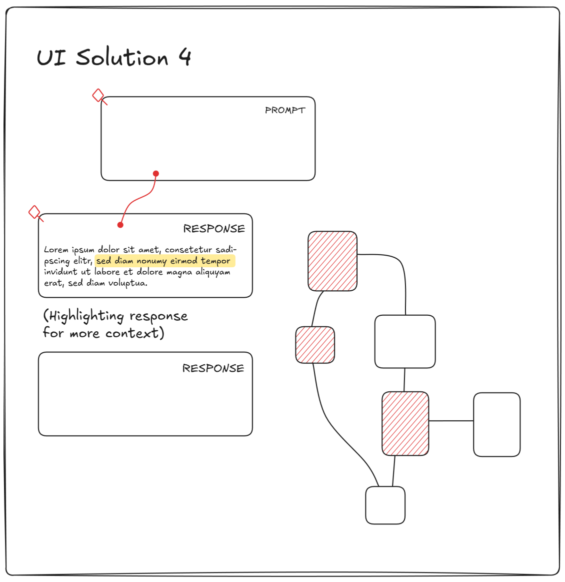
UI Solution 4
PROMPT
RESPONSE
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
(Highlighting response for more context)
RESPONSE

Figure 6: Digitized UI Solution 4

UI Solution 3 (see Figure 5). This concept introduces an 'auto-forget' function that allows users to explicitly instruct the LLM to forget specific elements or sections of the conversation, for example by typing a prompt like 'auto-forget [content]'. The system then automatically identifies and removes all mentions or context related to that element from the ongoing conversation, ensuring subsequent outputs do not reference it. The action is visualized with a hammer icon, reinforcing the idea of forcefully removing information.

UI Solution 4 (see Figure 6). This concept combines an LLM chat interface on the left with a branching visualization on the right, showing how the conversation structure evolves. Users can pin entire messages or highlight specific parts of a response to indicate their importance. These highlighted or pinned elements are intended to have increased influence on subsequent LLM outputs, effectively weighting them more in the model's context. The visualization on the right makes it easy to track how branches form whenever a new response is generated based on different highlights.

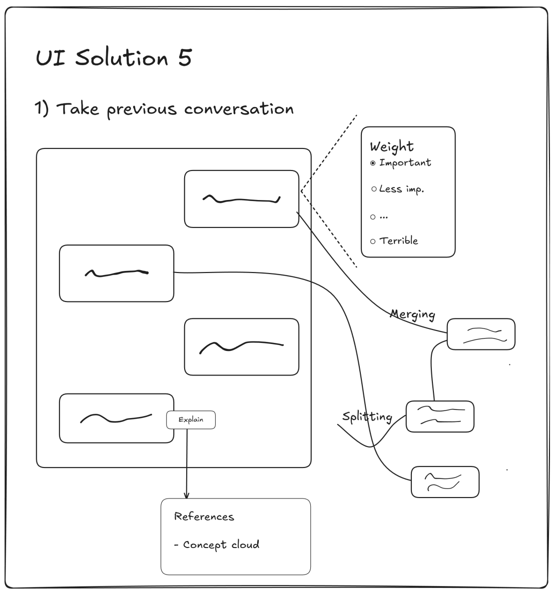

Figure 7: Digitized UI Solution 5

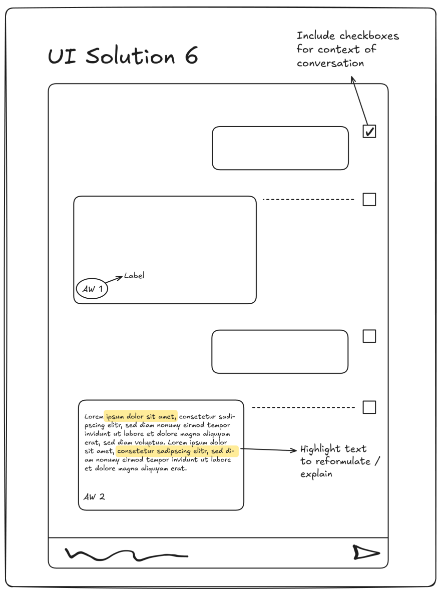
UI Solution 6
Include checkboxes for context of conversation
Label
AW 1
AW 2
Highlight text to reformulate / explain

Figure 8: Digitized UI Solution 6

UI Solution 5 (see Figure 7). This concept shares similarities with the previous branching and highlighting visualization in Figure 6 but introduces an explicit weighting mechanism: users can click on any prompt or message and assign it a value on a scale from ‘important’ to ‘unimportant’ or even ‘terrible’. These weights directly influence how much impact each part of the conversation has on subsequent LLM responses. Additionally, users can merge or split conversation threads, and there is a section for references or concept clouds to provide supporting information or context. In contrast to the previous concept – where the chat and the branching visualization are separate – here they are combined and can be displayed dynamically when hovering over the respective prompts.

UI Solution 6 (see Figure 8). This concept enables users to flexibly manage the context used by the LLM in generating responses. Each message in the conversation history has a checkbox, allowing users to include or exclude specific messages from the active context. Additionally, users can highlight individual words or sections – even in previous prompts or responses – to request reformulation or clarification of terms. When changes are made, outputs are regenerated to reflect the updated selections.

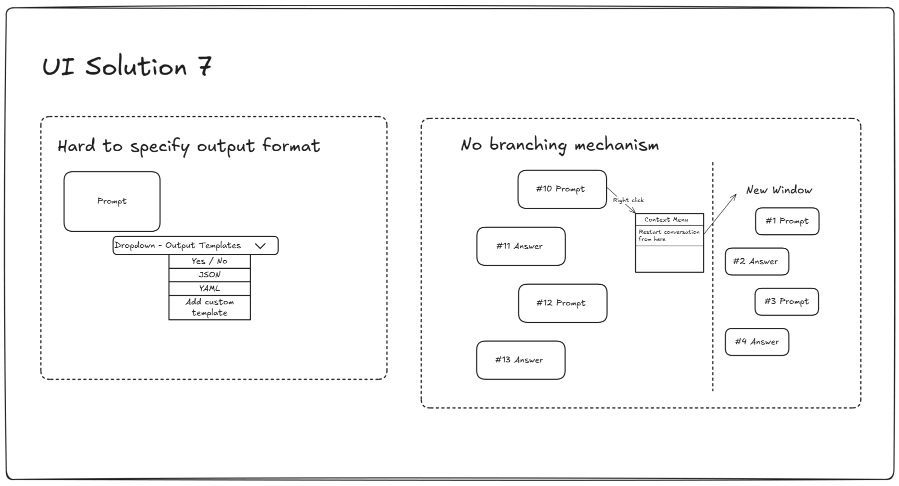

Figure 9: Digitized UI Solution 7

UI Solution 7 (see Figure 9). This concept tries to solve two problems: the difficulty of specifying the desired output format, and the lack of flexible conversation branching. These issues are addressed in the upper and lower panels, respectively. For output formatting (upper panel), users are given a dropdown menu alongside the prompt, allowing them to easily select or define how the LLM's response should be structured (for example, as JSON, YAML, or with a custom template), rather than relying on prompt engineering. For branching, shown in the lower panel, users can right-click on any previous prompt or answer to open a context menu and choose 'start conversation from here', which adds a new conversation thread to the right side of the UI – similar to concept number one.

5 Concept

5.1 Dynamic Whiteboard UI

Driven by the perception of unrealized potential in the current state of UIs of LLMs, the authors of this paper have created their own UI concept previous to the Workshop at deRSE25. This concept was awarded the AI-HUB@LMU Prize for the Most Innovative AI-based Research Project in 2024 at the Ludwig-Maximilians-Universität München (LMU). In this chapter, we present this concept, explain its functionality and show how its further development will benefit from the results of the design thinking workshop.

As mentioned in the prior chapter, current LLM are hampered by their rigid, chronological chat interfaces. In such systems, conversation unfolds chronologically, closely mimicking a dialogue between two parties. While this model is intuitive for basic exchanges, it imposes constraints on more complex tasks. As a result, users often struggle with cluttered, unfocused contexts, which can diminish the precision and usefulness of LLM outputs.

The core innovation of our approach is the introduction of a whiteboard-based UI with a context windows (see Figure 10), which departs from the linear chat paradigm. In this system, all user prompts and LLM outputs are represented as discrete, movable elements on a virtual whiteboard.

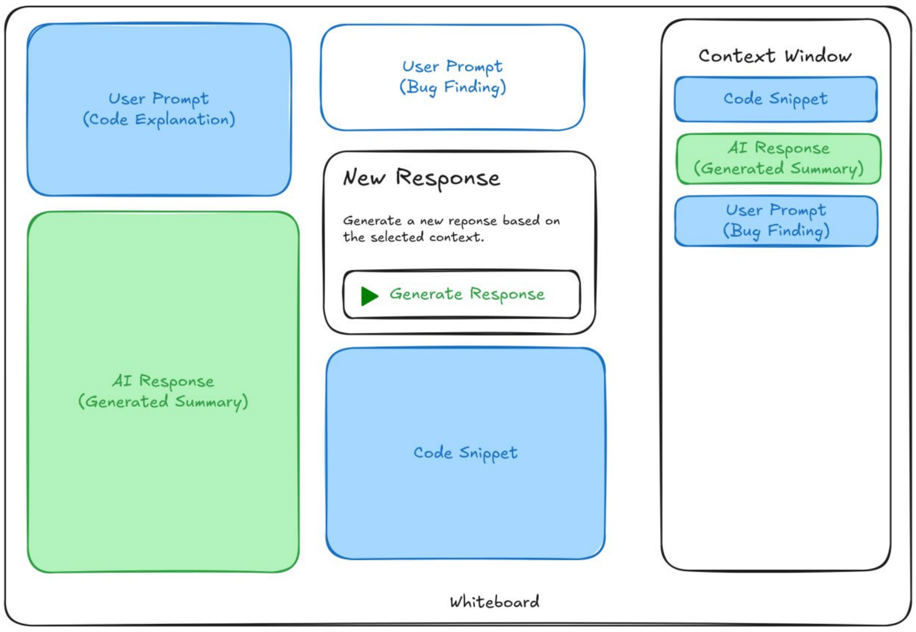
User Prompt (Code Explanation)
User Prompt (Bug Finding)
Context Window
Code Snippet
AI Response (Generated Summary)
User Prompt (Bug Finding)
New Response
Generate a new reponse based on the selected context.
Generate Response
AI Response (Generated Summary)
Code Snippet
Whiteboard

Figure 10: Concept for a dynamic whiteboard UI. Blue coloured boxes are user inputs, green coloured boxes are LLM outputs. Fully coloured boxes are active as part of the context window, while boxes only with a border represent deactivated context.

We envision the following features for our prototype:

- **Selective Context Management:** Users can add or remove any item (such as prompts, AI responses, or code snippets) to the context window. Only items present in this window are included as context for the next LLM response, enabling more specific interactions.
- **Dynamic Reordering:** Items within the context window can be freely rearranged via drag-and-drop (see the stack of blocks in the context window), allowing users to optimize the logical sequence of the provided information.
- **Visibility and Control:** The context window remains explicitly visible and manipulable at all times, providing transparency about what the LLM 'sees' and ensuring users can continuously adapt the context to their needs.
- **Blueprint Design:** Users can save particular arrangements of prompts and responses as reusable templates ('blueprints'), supporting the efficient handling of recurring tasks or workflows.

By moving beyond the linear chronological chat interface, our UI empowers users with a higher level of flexibility, transparency, and control in their interactions with LLMs. Importantly, the UI is designed for universal compatibility: rather than building a new language model, our focus is on providing a UI that can connect to a wide variety of commercial or open-source LLMs via their APIs. This ensures versatility and future-proofing, as users will be able to leverage different underlying AI models depending on their needs.

At present, this project is in the conceptualization phase where we are systematically exploring and refining the concept. Following this phase, we will proceed to prototype development and empirical testing, following the HCD process (see Chapter 2.3) to ensure the resulting UI is both usable and effective for a diverse range of users.

5.2 Design Patterns for Future Development

One of the aims of the workshop was to generate further ideas that would be beneficial for the subsequent development of our UI prototype. While analysing the UI concepts described in Chapter 4, we grouped them into three higher-level design patters we deem particularly valuable for future development. Table 5 shows the linkage between these patterns and the UI solutions

Table 5: Mapping of UI solutions to high-level design patterns

	Visualization of branching	**Weighting function**	**Hover-based interaction**
UI Solution 2 (Figure 4)	×		
UI Solution 4 (Figure 6)	×	×	×
UI Solution 5 (Figure 7)	×	×	
UI Solution 6 (Figure 8)			×
	3	2	2

from the workshop.

Visualization of branching. A first area of inspiration concerns the visualization of branching conversation paths. The first idea is a path diagram (see Figure 6), which visually represents the current position within a branched conversation. Users can see which path they are on and can easily switch to another branch by clicking within the visualization (similar to minimaps used in computer games). The second idea, inspired by Figures 4 and 7, is to display conversation branches as horizontally aligned panels on an infinite canvas. Every time a new branch is created, a new panel is added to the right, allowing users to scroll left and right to navigate between parallel branches.

Weighting function. Another idea is the implementation of a weighting function (see Figures 6 and 7) accessible for the user in the UI. This could be realized either by highlighting specific text segments within messages or by assigning weights to particular prompts using a scale. The weighted inputs would then influence the relative importance of those elements in subsequent LLM outputs, giving users more precise control over how their instructions and feedback shape the AI's responses.

Hover-based interaction. We also see potential in ideas around hover-based interactions for in-message highlights (see Figures 6 and 8). When users highlight parts of a prompt or response, the interface could present alternative formulations (useful for writing support) or offer explanations and tooltips with additional context or clarifications.

It is important to mention that it is not necessarily purposeful to integrate all generated ideas into one prototype: While individual ideas may increase the usability of the LLM interface, this does not always mean that in combination they will lead to a good user experience. Therefore, our next steps will involve evaluating which concepts best align with the overall design goals and user needs. Following the principles of the HCD process, it is useful to develop and test multiple prototypes – each with a specific set of integrated innovations – in parallel and test these with users to help identify which ideas deliver the most value in practice. This iterative, user-driven selection and refinement will ensure that the resulting UI concept is both innovative and effective.

6 Discussion

In this paper, we presented the results of a design thinking workshop held at the deRSE25 conference as part of our development process for innovative UIs for Large Language Models (LLMs). We explained the basic functioning of LLMs, highlighted their operational principles, and discussed key limitations of current UIs. We then detailed the structure and outcomes of our design thinking workshop, which aimed to collaboratively develop new UI concepts. Through participant feedback, we identified common use cases, strengths, and notable weaknesses of LLMs. The workshop results, visualized and documented by participants, revealed new UI concepts that offer potential enhancements over current linear chat interfaces, particularly in flexible context management, dynamic branching, and enhanced user control mechanisms. These insights now inform our ongoing UI development, reinforcing the value of HCD in optimizing user interaction with AI systems.

The insights derived from our workshop underline that LLMs are extensively utilized both in programming contexts and in everyday tasks. However, due to the nature of our conference audience, there was a distinct emphasis on applications of LLMs in software development (see Chapter 4). This focus provided rich insight into how users integrate AI tools within programming workflows, with productivity and efficiency gains cited as particularly valuable.

Participants valued the time-saving potential and efficiency offered by LLMs, especially their ability to handle repetitive and monotonous tasks, such as writing documentation for coding projects. However, significant concerns were also expressed, particularly regarding the reliability of outputs. Users frequently mentioned issues such as inaccuracies and occasional nonsensical outputs, which undermined their trust and required careful verification.

Interestingly, while these criticisms were abundant, critique specifically targeting the UI – apart from a solitary mention regarding the inconvenience of manually copying content between chat instances – was notably scarce.

We interpret this as evidence that participants perceived existing interfaces broadly functional and were generally accepting of current UI conventions. However, the absence of UI-focused criticism does not necessarily indicate complete satisfaction; it may instead reflect users' habituation to prevailing interaction paradigms. We also deliberately asked for all kinds of criticism – not only about the UI – to avoid narrowing participants' feedback too early in the development process.

Beyond these points, participants also raised important ethical and practical considerations, including data security, privacy, and the social and environmental impacts associated with AI. Concerns were particularly voiced regarding data privacy risks, potential misuse of data by companies, and broader social consequences, such as labor exploitation in the training of AI models. Environmental concerns were also prominent, highlighting the significant resource consumption and ecological footprint associated with training large-scale AI models. While these concerns cannot be addressed solely through alternative UIs, they highlight the need for broader social discourse about the implications of AI and the necessity for interdisciplinary collaboration to address these challenges in the future.

For us, the HCD process has proven to be a particularly valuable framework for structuring our development efforts, even though our work is still in the conceptual phase. Its iterative nature and explicit focus on user involvement enable a robust alignment of our UI concepts with user needs and expectations. Moving forward, we will refine our initial UI concept by integrating the ideas generated during the workshop. Subsequently, we will create wireframe prototypes and engage in multiple iterations guided by user evaluations. Through this development process, we aim to arrive at a fully functional UI, which we intend to share with the broader scientific community.

Underlying our future development effort is the following belief: We hypothesize that, although LLM capabilities have seen rapid advancement, characterized by increased parameters and expanded context windows, technical progress is likely to plateau in the near future. At this stage, interface usability and user experience will become decisive differentiators between competing models. We therefore emphasize that future development of LLMs should increasingly prioritize UI innovation, ensuring that UI design receives the attention necessary to fully exploit the potential of AI systems.

Acknowledgements:

Funding
We thankfully acknowledge financial support by the Münchner Universitätsgesellschaft awarded as part of the AI-HUB@LMU Prize for the Most Innovative AI-based Research Project in 2024.

Competing Interests
The authors declare no competing interests.

Author Contributions
All authors meet the criteria for authorship. The specific contributions are described below using the CRediT taxonomy (https://credit.niso.org).

Conceptualization:	Maximilian Frank, Simon Lund
Methodology:	Maximilian Frank, Simon Lund
Visualization:	Maximilian Frank, Simon Lund
Writing – original draft:	Maximilian Frank, Simon Lund
Funding acquisition:	Maximilian Frank, Simon Lund

Bibliography

[BGMS21] E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. Pp. 610–623. 2021. doi:10.1145/3442188.3445922

[BH97] H. Beyer, K. Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[Bil91] C. E. Billings. Human-centered aircraft automation philosophy. Technical report, NASA Technical Memorandum 103885, NASA Ames Research Center, 1991.

[Boy11] G. A. Boy. *Introduction to the Handbook of Human-Machine Interaction: A Human-Centered Design Approach*. Ashgate, 2011.

[Dor11] K. Dorst. The core of 'design thinking'and its application. *Design studies* 32(6):521–532, 2011. doi:10.1016/j.destud.2011.07.006

[End96] M. R. Endsley. Situation awareness measurement in test and evaluation. *Handbook of Human Factors Testing and Evaluation* 1:159–178, 1996.

[GBB^{+}20] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, C. Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*, 2020. doi:10.48550/arXiv.2101.00027

[GGB+03] J. Gulliksen, B. Göransson, I. Boivie, S. Blomkvist, J. Persson, Å. Cajander. Key Principles for User-Centered Systems Design. *Behaviour & Information Technology* 22(6):397–409, 2003.
doi:10.1080/01449290310001624329

[GKD+23] V. Gadiraju, S. Kane, S. Dev, A. Taylor, D. Wang, R. Denton, R. Brewer. ”I wouldn’t say offensive but...”: Disability-Centered Perspectives on Large Language Models. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’23, p. 205–216. Association for Computing Machinery, New York, NY, USA, 2023.
doi:10.1145/3593013.3593989

[GSG21] G. M. Grimes, R. M. Schuetzler, J. S. Giboney. Mental models and expectation violations in conversational AI interactions. *Decision Support Systems* 144:113515, 2021.
doi:10.1016/j.dss.2021.113515

[GTZ+24] R. Gallotta, G. Todd, M. Zammit, S. Earle, A. Liapis, J. Togelius, G. N. Yannakakis. Large Language Models and Games: A Survey and Roadmap. *IEEE Transactions on Games*, pp. 1–18, 2024.
doi:10.1109/TG.2024.3461510

[HHS24] M. T. Hicks, J. Humphries, J. Slater. ChatGPT is bullshit. *Ethics and Information Technology* 26(2):1–10, 2024.

[HS05] H.-F. Hsieh, S. E. Shannon. Three Approaches to Qualitative Content Analysis. *Qualitative Health Research* 15(9):1277–1288, 2005. PMID: 16204405.
doi:10.1177/1049732305276687

[HW05] E. Hollnagel, D. D. Woods. *Joint Cognitive Systems: Foundations of Cognitive Systems Engineering*. CRC Press, 2005.
doi:10.1201/9781420038194

[HYM+25] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* 43(2):1–55, 2025.
doi:10.1145/3703155

[KHM+23] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, R. McHardy. Challenges and Applications of Large Language Models. 2023.
doi:10.48550/arXiv.2307.10169

[KR18] T. Kudo, J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.
doi:10.48550/arXiv.1808.06226

[KSK+23] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, M. Stadler, J. Weller, J. Kuhn, G. Kasneci. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103:102274, 2023.
doi:10.1016/j.lindif.2023.102274

[LPP+20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Curran Associates Inc., Red Hook, NY, USA, 2020.
doi:10.48550/arXiv.2005.11401

[Mag01] M. Maguire. Methods to Support Human-Centered Design. *International Journal of Human-Computer Studies* 55(4):587–634, 2001.
doi:10.1006/ijhc.2001.0503

[MDW+23] P. Ma, R. Ding, S. Wang, S. Han, D. Zhang. Demonstration of InsightPilot: An LLM-Empowered Automated Data Exploration System. 2023.
doi:10.48550/arXiv.2304.00477

[MMBK24] L. Metzger, L. Miller, M. Baumann, J. Kraus. Empowering Calibrated (Dis-)Trust in Conversational Agents: A User Study on the Persuasive Power of Limitation Disclaimers vs. Authoritative Style. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI '24. Association for Computing Machinery, New York, NY, USA, 2024.
doi:10.1145/3613904.3642122

[MMCV24] D. Masson, S. Malacria, G. Casiez, D. Vogel. DirectGPT: A Direct Manipulation Interface to Interact with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI '24. Association for Computing Machinery, New York, NY, USA, 2024.
doi:10.1145/3613904.3642462

[MMN+25] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, J. Gao. Large Language Models: A Survey. 2025.
doi:10.48550/arXiv.2402.06196

[MMR24] P. Martínez, L. Moreno, A. Ramos. Exploring Large Language Models to generate Easy to Read content. 2024.
doi:10.48550/arXiv.2407.20046

[MP13] F. Macpherson, D. Platchias. *Hallucination: Philosophy and psychology*. MIT Press, 2013.
doi:10.7551/mitpress/9780262019200.003.0025

[MVSC05] J.-Y. Mao, K. Vredenburg, P. W. Smith, T. Carey. The State of User-Centered Design Practice. *Communications of the ACM* 48(3):105–109, 2005. doi:10.1145/1047671.1047677

[NKQ+24] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, A. Mian. A Comprehensive Overview of Large Language Models. 2024. doi:10.48550/arXiv.2307.06435

[Nor86] D. A. Norman. Cognitive engineering. *User Centered System Design*, pp. 31–61, 1986.

[fNB10] D. D. I. für Normung (Berlin). *ISO 9241-210:2019 Ergonomie der Mensch-System-Interaktion: Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme (ISO 9241-210:2019) : Ausgabe: 2019-04-01 ; Deutsche Fassung EN ISO 9241-210:2019*. Deutsche Norm. DIN Deutsches Institut für Normung, 2010.

[RNSS18] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever. Improving language understanding by generative pre-training. 2018. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[Roy87] W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*. Pp. 328–338. 1987.

[SDG+24] M. Shen, S. Das, K. Greenewald, P. Sattigeri, G. Wornell, S. Ghosh. Thermometer: Towards universal calibration for large language models. *arXiv preprint arXiv:2403.08819*, 2024. doi:10.48550/arXiv.2403.08819

[fS19] I. O. for Standardization. *Ergonomics of human-system interaction — Part 210: Human-centered design for interactive systems*. Volume ISO 9241-210:2019. International Organization for Standardization, 2019. https://www.iso.org/standard/77520.html

[VMSC02] K. Vredenburg, J.-Y. Mao, P. W. Smith, T. Carey. A Survey of User-Centered Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Pp. 471–478. 2002. doi:10.1145/503376.503460

[VRM24] K. Vafa, A. Rambachan, S. Mullainathan. Do large language models perform the way people expect? measuring the human generalization function. In *Proceedings of the 41st International Conference on Machine Learning*. ICML'24. JMLR.org, 2024. doi:10.48550/arXiv.2406.01382

[VSP$^+$17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. Attention Is All You Need. *CoRR* abs/1706.03762, 2017. doi:10.48550/arXiv.1706.03762

[WWL$^+$24] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang, M. Chowdhury, M. Zhang. Efficient Large Language Models: A Survey. 2024. doi:10.48550/arXiv.2312.03863

[WWPY25] X. Wang, X. Wang, S. Park, Y. Yao. Mental Models of Generative AI Chatbot Ecosystems. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*. IUI '25, p. 1016–1031. Association for Computing Machinery, New York, NY, USA, 2025. doi:10.1145/3708359.3712125

[ZZL$^+$25] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, J.-R. Wen. A Survey of Large Language Models. 2025. doi:10.48550/arXiv.2303.18223

A Drawn UI Solutions

During the workshop, participants created hand-drawn concepts for potential user interface solutions. These sketches, along with participant presentations, served as the foundation for developing digitized versions and the descriptions in the main text. The figures below present the original drawings created by the workshop participants.

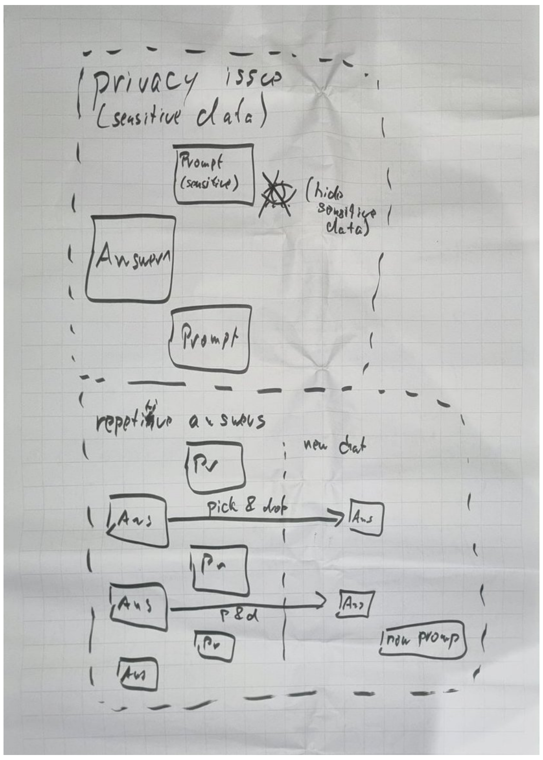

Figure 11: UI Solution 1

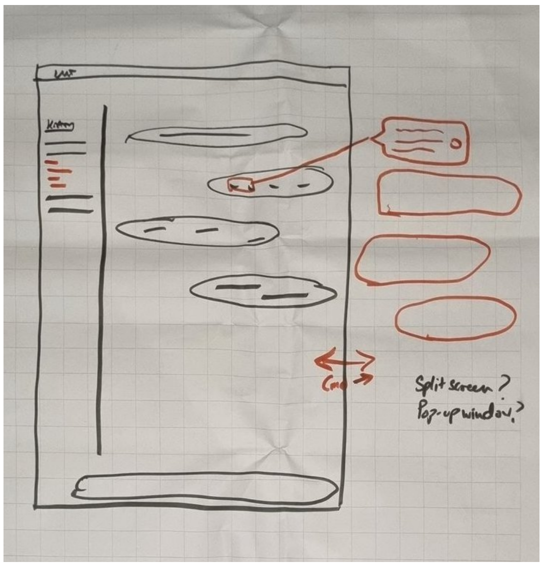

Figure 12: UI Solution 2

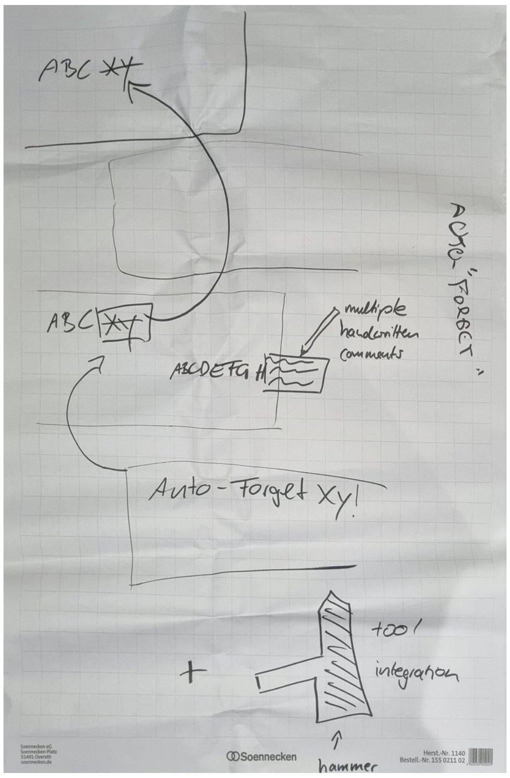

Figure 13: UI Solution 3

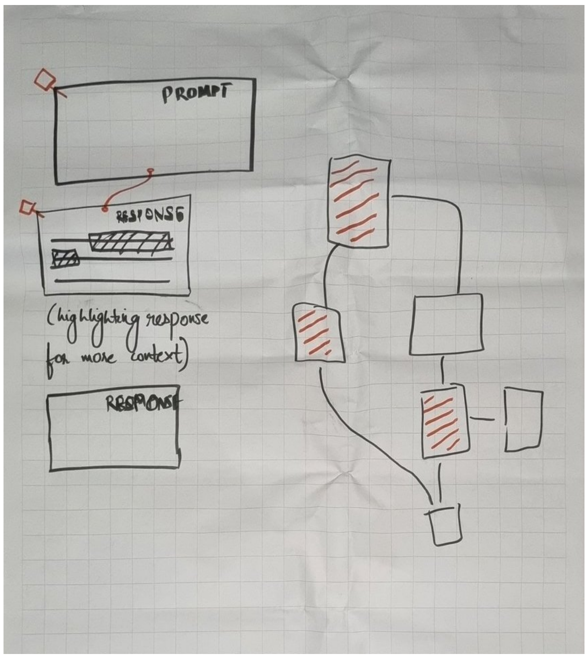

Figure 14: UI Solution 4

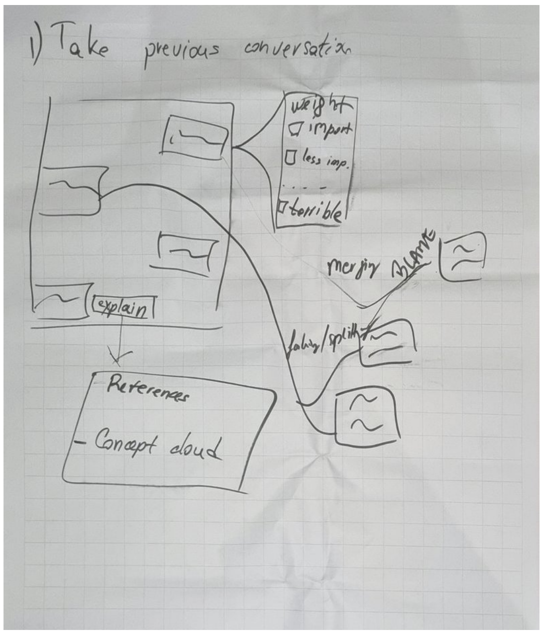

Figure 15: UI Solution 5

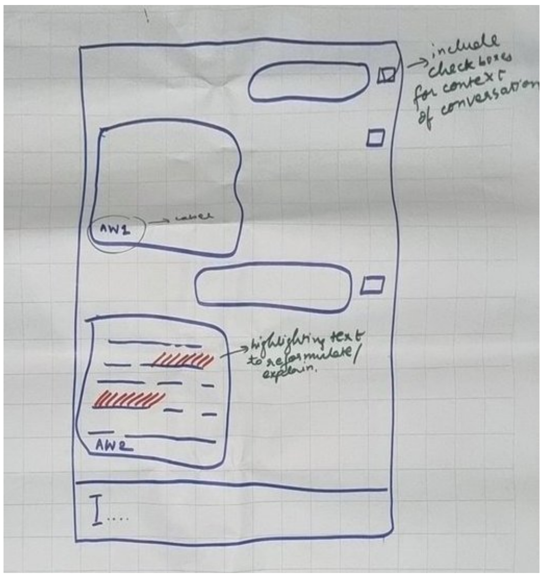
include check boxes for context of conversation
AW1
label
highlighting text to reformulate/explain.
AW2
I....

Figure 16: UI Solution 6

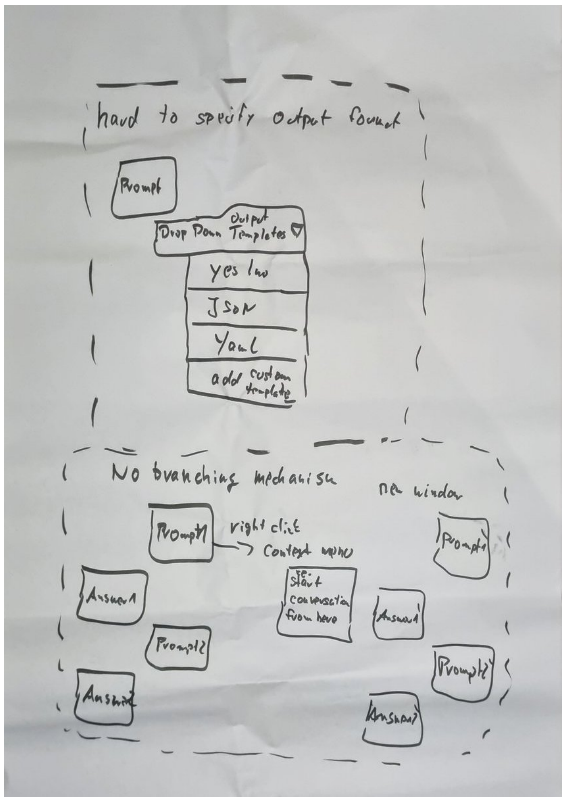

Figure 17: UI Solution 7