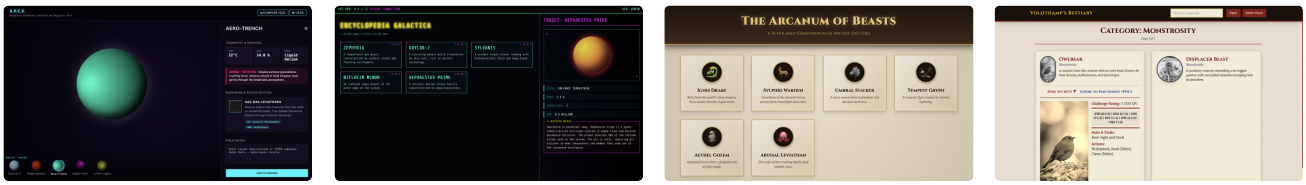
Thinking Inside the Box: Building User Interfaces with Question-Guided Inputs

Sejal Agarwal*
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
s97agarw@uwaterloo.ca

Helen Weixu Chen*
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
w352chen@uwaterloo.ca

Daniel Vogel
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
dvogel@uwaterloo.ca

Anamaria Crisan
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
ana.crisan@uwaterloo.ca



(a) (b) (c) (d)

Figure 1: Examples of participant-created interfaces produced using Question-Guided Format (QGF) and Free-Form Prompting (FFF) for the Planet Discovery Interface and the Fantasy Creature Encyclopedia tasks: (a) QGF for the Planet Interface, (b) FFF for the Planet Interface, (c) QGF for the Fantasy task, and (d) FFF for the Fantasy task.

Abstract

Generative Artificial Intelligence (GenAI) tools are rapidly being used to support user interface (UI) development. When building UIs, AI chatbots are highly effective at instantiating and designing projects; however, the quality of the AI response is highly dependent on the user's ability to translate their ideas into well-structured and defined prompts. Factors like varying AI literacy, can lead to vague inputs, inefficient iteration, and suboptimal outputs. Furthermore, there has been little innovation in how users engage with AI chatbots, as most chatbots rely on free-form inputs from the user. We investigate prompting format as an interaction design problem by comparing a free-form format (FFF) with a question-guided format (QGF) that helps users structure their initial requests before code generation. A within-subjects study with 12 participants completing two comparable UI-building tasks found that the QGF enabled significantly fewer iterations and revisions, illustrating that answering guided questions helps create a more satisfactory initial prompt. Interview results also revealed that the QGF encouraged more deliberate reflection on requirements, but also introduced extra time and a sense of constraint for some participants. Our findings highlight a fundamental trade-off between speed and structured guidance, informing the design of future GenAI interfaces for development.

Keywords

Generative AI, Prompt Engineering, Human–AI Interaction, User Interface Development, Interaction Design, Developer Tools

1 Introduction

Generative Artificial Intelligence (GenAI) is increasingly being used to support programming and user interface (UI) development, enabling users to generate, debug, and refine code through natural language interaction [9, 22, 46]. As these systems become integrated into educational contexts, they are reshaping how individuals engage with programming, design, and problem-solving [57, 58]. This shift has prompted growing attention in AI pedagogy and AI literacy research, which emphasizes not only what these systems can generate, but how users learn to communicate with them effectively, interpret outputs, and iteratively refine their ideas [25, 70].

Despite this, interacting with GenAI systems remains challenging, particularly during the initial prompt formulation stage [2, 4, 70]. Users must decide what information to include, how to structure their request, and how to translate abstract ideas into concrete instructions. Prior work demonstrates that users often struggle with this process due to limited AI literacy, weak mental models of system behaviour, and uncertainty about how prompts influence outputs [70]. This challenge is especially pronounced in open-ended tasks such as UI development, where requirements are often underspecified, too broad, or ill-defined. [38].

*These authors contributed equally as first authors.

Furthermore, most existing GenAI coding assistants rely on a free-form text chatbot interfaces. While this interaction style allows flexibility, it places the burden on users to formulate an effective and well-defined initial prompt. In many real-world interactions with GenAI systems, users begin with little prior context about how best to structure their request. This cold start problem can lead to vague prompts, inefficient trial-and-error interactions, and difficulty achieving the desired outcome [39, 70].

In this paper, we investigate whether restructuring the input interface of GenAI chatbots can support users in articulating clearer and more complete initial prompts by introducing a new interaction technique: Question-Guided Format (QGF). QGF is a structured input technique that prompts users to answer guided questions, encouraging reflection on key aspects of their design before generating code. These questions are grounded in principles from psychology and behavioural science that encourage users to consider context, goals, interactions, and constraints when describing an interface [41]. Prior work in AI-assisted learning and technology-enhanced education suggests that structured guidance and scaffolding play a critical role in supporting reflective thinking and improving task performance, particularly in open-ended design contexts [30, 37]. Additionally, recent research and industry practice has begun exploring more structured and clarification-oriented prompting workflows. Prior work has shown that asking clarifying questions can help language models elicit underspecified user preferences and improve downstream response quality [5, 29]. Commercial systems, such as Claude, increasingly ask follow-up questions when user requests are ambiguous [1], and emerging tools and libraries support guided prompt refinement [6]. Together, these developments suggest that users may benefit from scaffolding and guidance during prompt construction. Therefore, QGF operationalizes guided questioning as an interaction design strategy to support more deliberate and well-specified prompt formulation.

We present an exploratory within-subjects study ($N = 12$) comparing QGF with a standard free-form format (FFF) for GenAI-assisted UI development. Our findings provide preliminary evidence towards the effectiveness of structured prompting interfaces: **while QGF increases task time when constructing the first prompt, it leads to significantly fewer revisions and encourages more deliberate prompt construction.** Participants also reported that QGF supported reflection and idea generation, though it introduced creativity constraints and reduced flexibility for some users. These results suggest that prompt formulation is not just a user skill, but it can be shaped by interface design. Rather than constructing prompts through iterative trial-and-error, QGF reframes how users engage with generative systems during the initial prompt formulation stage. This shift has important implications for both productivity and learning, as it highlights how structured interaction can support users' reasoning.

Our research makes three key contributions:

- the QGF technique, which structures initial participant prompts through a set of questions grounded in psychology and behavioural science;
- findings from a preliminary empirical study comparing QGF with a standard FFF approach, highlighting the strengths and limitations of each;
- a broader reflection on the implications of QGF for supporting application development and programming pedagogy.

More broadly, our work contributes to emerging discussions in AI pedagogy and AI-assisted development by demonstrating that interaction design can play a central role in supporting AI literacy.

2 Related Work

2.1 GenAI Chatbots for Early-Stage Development

GenAI tools have been rapidly integrated into software development workflows supporting code generation, debugging, and interface design [14, 18, 19, 57–59]. Research illustrates that GenAI coding assistants can significantly reduce manual coding effort and accelerate task completion by reducing the time spent on coding, searching, and writing documentation [14, 57]. Furthermore, GenAI tools have been integrated into UI workflows, assisting with designing interfaces, creating mockups, and generating code [28].

Modern GenAI coding tools typically take two forms: integrated assistants embedded within development environments and standalone conversational chatbots. Integrated systems such as GitHub Copilot provide real-time suggestions during coding, raising challenges around when and how assistance should be presented [45]. In contrast, chatbot-based systems, like Gemini and ChatGPT, offer flexible, natural language interaction [7], but rely heavily on users constructing effective prompts.

Chatbot-based GenAI systems are particularly well-suited for early-stage development tasks, as they effectively support brainstorming, idea generation, and exploration by producing outputs in response to prompts [12, 33, 35]. These systems can also help translate high-level descriptions, ideas, and requirements into initial artifacts, such as code structures or interface concepts [28]. This makes conversational interfaces especially valuable at the beginning of tasks, where users may not yet have concrete implementations but can describe goals, features, or constraints in natural language.

However, this flexibility also introduces challenges. Chatbot-based systems rely entirely on user-provided prompts, making the quality of initial outputs highly sensitive to how well users articulate their intent [10, 70]. Despite them being well-suited for early-stage development, chatbot interfaces remain largely unchanged as free-form text inputs, providing limited support for translating vague, high-level ideas into well-specified prompts.

Our work addresses this gap by introducing a question-guided interface that not only changes how users interact with GenAI chatbots, but also supports clearer requirement articulation during UI development.

2.2 AI Literacy and Prompting

The effectiveness of GenAI tools in coding workflows depends not only on model capabilities but also on how users formulate prompts. Prior work highlights that users often struggle to specify requirements, constraints, and context when prompting GenAI systems, particularly at the start of an interaction, leading to suboptimal outputs and trial-and-error interactions [10, 61, 65, 70]. These

challenges are pronounced in open-ended tasks such as UI development, where AI solutions require substantial and frequent revision [32, 48]. To address these challenges, researchers have begun exploring prompt engineering strategies such as zero-shot, few-shot, and chain-of-thought prompting [36, 52, 68] to generate more effective AI outputs. While these methods can improve quality, they require users to learn specialized strategies.

More recently, research has explored structured prompting approaches to scaffold user input. Template-based systems include UI controls (e.g., dropdowns, fill-in fields, multiple choice questions) to guide prompt construction [43]. These templates are useful because they structure user input, reduce ambiguity, and draw attention to important elements that should be considered when forming a prompt. MacNeil et al. demonstrate how templates can standardize requests and reduce ambiguity by developing Prompt Middleware, a framework for creating prompts based on UI affordances [40]. Tools like ChainForge extend this idea by supporting reusable prompt templates, versioning, and comparison across variants, emphasizing reproducibility and evaluation of prompt designs [6]. While these approaches reshape how users interact with GenAI, they have primarily been studied in writing domains and often rely on static input structures. Additionally, there is limited work investigating how structured interfaces can support initial prompt formulation in complex, open-ended tasks such as AI-assisted UI development.

Our study addresses this gap by introducing the QGF to help users craft clearer, more deliberate, and effective initial prompts.

2.3 AI-Mediated Learning and Guided Inquiry

Recent research on GenAI in education suggests that its value extends beyond simply providing answers; the educational impact also critically depends on how learners are guided to engage with these systems [15, 16]. A growing body of work in learning sciences, instructional design, and AI-supported pedagogy positions iterative questioning and reflection as an important element of effective AI-enhanced learning [23, 56].

This emphasis aligns with long-standing educational theories on guided inquiry to support learner development. Approaches such as Socratic dialogue and iterative questioning encourage learners to make their thinking explicit, evaluate multiple perspectives, and refine their understanding [42, 49, 55]. To incorporate these pedagogical techniques in classroom AI tools, developers are increasingly designing these systems to include frictions and constraints to reinforce cognitive effort and engagement [66].

Empirical research from AI tutoring further illustrates the benefits of structured, question-driven interactions. Systems that guide learners through sequenced questions and provide feedback influence how students break down complex problems and reflect on their reasoning patterns [20, 26]. For example, SocraticAI enforces reflective engagement, leading learners to deliberate more carefully when decomposing problems and constructing solutions [60]. Comparisons between AI-simulated and human tutoring also show that purely answer-centric AI dialogues often lack the richness of guided questioning, highlighting the importance of embedding pedagogical structure in GenAI educational interactions [31].

These ideas also align with prior work in design education and human-centered specification practices, where structured briefs and guided questioning help individuals articulate ambiguous goals and shape problem framing [3, 24]. Similarly, research in human-robot interaction shows that interactive questioning can clarify vague intentions and improve collaboration between humans and intelligent systems [11]. These perspectives together suggest that structured questioning can help users refine intent and formulate clearer specifications when interacting with GenAI systems.

These insights align with broader perspectives on AI literacy, which frame GenAI use as a reflective cognitive practice rather than a shortcut to solutions. Learners benefit most when prompted to generate, critique, and refine ideas, and GenAI systems can support this through carefully guided, pedagogically grounded questioning.

Our work builds on these principles by incorporating AI pedagogical–based questioning in QGF, encouraging more deliberate and effective initial prompt formulation.

3 Design Goals

In this work, we use vibe coding to refer to a style of creating software with generative AI in which users begin with a high-level idea or intended feel of an interface [54], but may not yet have a complete specification of its logic, structure, or implementation details. Rather than writing code directly, users often describe what they want in natural language and iteratively refine the generated output. While this lowers the barrier to interface creation, it also introduces new challenges: users may lack the domain knowledge needed to specify important functional requirements [13], may struggle to progressively articulate design decisions [50], and may find it difficult to phrase effective prompts for downstream code generation [34]. Prior work on generative tools and prompt-based creation suggests that these challenges can limit users' ability to communicate meaningful requirements and maintain control over the resulting system [8, 17, 67]. Based on these challenges, we derived the following design goals to guide the design of QGF.

- **DG1: Support interface design when users have incomplete domain knowledge.** Users engaging in vibe coding may not fully understand the domain logic behind the system they want to build. For example, a user designing a diving-scoring interface may not know how scores are calculated, what data must be collected, or how different roles interact with the system. Prior work shows that when users work with generative tools, gaps in domain knowledge can make it difficult to express meaningful requirements [62, 71]. Therefore, the tool should help users clarify core domain concepts, data attributes, and workflows before moving into interface-level decisions.
- **DG2: Support progressive requirement specification.** Users often begin vibe coding with only a partial or vague idea of the interface they want to create. As a result, they may overlook important considerations such as use context, interaction flow, priorities, and constraints. Rather than expecting users to provide a complete specification upfront, the tool should help them progressively develop and refine their requirements through structured guidance.
- **DG3: Help users phrase effective prompts for downstream code generation.** Even when users have a reasonable idea of what they want to build, translating that idea

into an effective prompt for an LLM remains difficult. Prior work suggests that prompt-support tools often still assume that users know the right terminology and can clearly articulate task-specific details. Therefore, the tool should help users transform their project description, requirements, and preferences into a structured prompt that helps with downstream code generation over ad hoc writing.

4 Question-Guided Format (QGF) Chatbot

To address these design goals, we developed the QGF chatbot. QGF combines a lightweight project intake, domain-attribute step, category-based question workspace, and structured prompt-compilation process to help users construct robust initial prompts. Together, these components help users clarify domain concepts (**DG1**), progressively specify interface requirements (**DG2**), and transform those requirements into a code-generation-ready prompt (**DG3**).

4.1 Project Intake and Domain Attributes

The interaction begins with a chat-style prompt asking users to describe their project in 1-3 sentences (Figure 2a). Users can provide a short description of the interface they wish to build and provide some high-level details about what the interface should support, such as browsing multiple items, selecting entries, or updating content based on user input. This initial description is stored as the project summary and used to initialize later question generation.

After intake, QGF asks for domain-specific attributes and terminology needed to describe the application content. For example, in a hypothetical fantasy-creature encyclopedia application (a scenario we explore in our user study), users may specify what fields they would like to view, such as what species the creature is or its habitat, magical abilities, and danger level. Users who already have a schema in mind can skip this step through an explicit `SKIP` option. If provided, these attributes are stored as part of the project specification and reused in later prompts.

4.2 Category-Based Questions

After the initial intake, the interface transitions to a structured workspace organized into five categories: Usage Scenarios, Tone, Core Interactions, Salience, and Anti-features (Figure 2b).

These categories come from well-established findings in psychology and behavioural science. Questions about usage scenario draw on context-of-use and situated behaviour [41]; feeling and personality relate to emotional design and product personality [69]; core interactions align with goal-directed behaviour and mental models [27]; salience and flow reflect limited attention and visual hierarchy [53]; and anti-features capture avoidance [63], loss aversion [47], and reactance [51]. We chose to use these categories in our QGF design to ensure that participants would consider multiple dimensions of the interface design when describing their desired UI. By organizing prompts around this established framework, we help users to articulate aspects of their design that might otherwise remain implicit in free-form prompting. This structured approach also helps reduce ambiguity in user intent and supports more deliberate and comprehensive prompt formulation.

In the QGF interface, each category is shown as a separate section with a short description and includes three questions at a time. This paging strategy keeps the workspace compact and allows users to focus on one small set of decisions before moving on. Each question is presented as a card with a question stem, a set of predefined options, and an optional freeform text box for elaboration. The predefined options support fast selection and make answers easier to aggregate later. The freeform field allows users to add details that are not captured by the listed choices. At the end of the question block for a category, there is a `Generate more` button that lets users request additional questions within the current category.

4.3 Question Generation

To generate questions for the QGF chatbot, we use a Gemini model to generate questions from category-specific templates conditioned on the project description and any domain attributes collected earlier. The specific model used in our implementation is described in Section 4.5. Each category focuses on a different aspect of interface requirements. Usage Scenarios capture when and where the interface will be used. Tone/Vibe asks about the desired emotional and stylistic character of the interface. Core Interactions focuses on the main user actions the interface should support. Salience captures what should be visually prioritized or de-emphasized. Anti-features record patterns, elements, or behaviours the user wants to avoid.

4.4 From Answers to Code-Generation Prompts

As users answer questions, the technique stores responses in a structured representation containing the project summary and optional domain attributes, as well as the selected options and freeform elaborations from the category-based questions. Once enough information has been collected, the technique compiles these inputs into a code-generation prompt (Figure 3). The compiled prompt summarizes the application goal, key interface features, usage conditions, stylistic preferences, priorities, and non-goals, and formats them as instructions for Gemini to generate interface code.

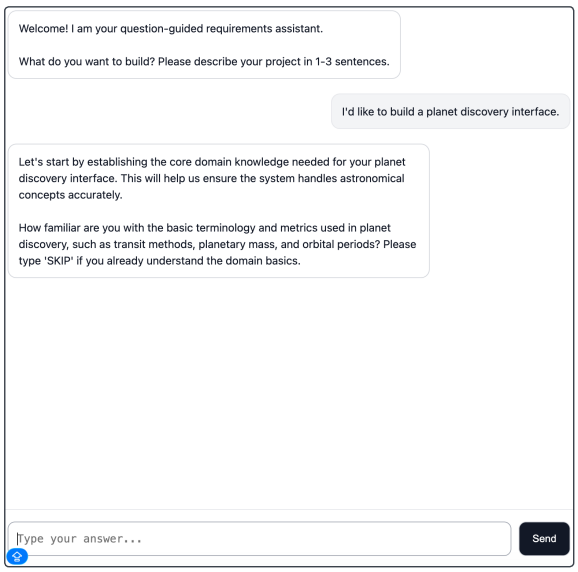
Instead of forwarding the raw interaction history, the tool produces a condensed specification that highlights the information most relevant for code generation. This helps preserve useful details from the elicitation process while reducing redundancy and ambiguity in the final prompt.

4.5 Implementation

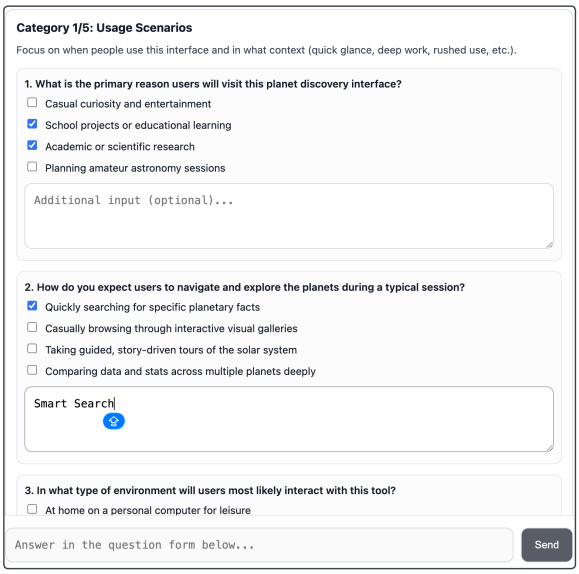
We implemented QGF as a web-based prototype using HTML, CSS, and JavaScript for the frontend and Node.js/Express for the backend. The frontend presents the chat-style intake, domain-attribute step, and category-based question workspace, while the backend manages question generation, response aggregation, and prompt construction. The current prototype uses `gemini-3.1-pro- preview` for question generation and downstream code-generation

5 User Study

We conducted a within-subjects study to examine how two AI chatbot formats, Free-form format (FFF; a standard baseline) and QGF (our technique), shaped users' experiences of building UIs with GenAI. In FFF, participants interacted with the chatbot through a conventional free-form text box, similar to widely used GenAI chat systems. In QGF, participants developed their prompts by responding to structured, category-based questions designed to



(a) Project intake and domain clarification.



(b) Category-based requirement questions.

Figure 2: A visualization of the user interface for QGF: (a) The chat-style project intake interface, where users briefly describe the application they want to build. The technique follows up with an optional domain-understanding question to clarify key concepts and terminology. (b) One of the five category-based question sets is shown here. This one asks questions regarding Usage Scenarios. Users answer the generated multiple-choice questions and optionally elaborate in the provided text field. Each category presents a small batch of questions at a time to support incremental requirement collection.

Figure 3: The final QGF code-generation view, where the system compiles user responses into an optimized prompt and displays generated interface code for downstream implementation.

scaffold requirement formulation. Through this comparison, we investigated the following research questions (RQs):

- **RQ1:** How does chatbot input structure influence users' prompt formulation strategies and efficiency during UI development?
- **RQ2:** How do different prompting interfaces shape users' perceptions of control, ownership, and satisfaction when working with GenAI?
- **RQ3:** What tradeoffs emerge between structured and free-form prompting in terms of efficiency, iteration, and user experience?

5.1 Participants

We recruited 12 participants from the university community (age range: 19 to 34; 6 self-identified as female and 6 as male). On 5-point scales, participants reported their programming experience (Mdn = 4, SD = 0.90), web development experience (Mdn = 3, SD=0.78), and exposure to GenAI tools (Mdn = 4, SD = 0.85). Participants reported using ChatGPT (10), Gemini (9), Claude (8), GitHub Copilot (5), Perplexity (2), and DeepSeek (1).

5.2 Apparatus

We ran the study in person. Participants used a web browser to access our custom web-based interface for interacting with the chatbots. The interface supported both study conditions: a baseline FFF (Figure 4), in which participants entered prompts in an open-ended chat interface, and our QGF, in which participants responded

to structured questions to help formulate their prompts before code generation. This setup allowed us to compare a familiar free-form prompting workflow against our guided prompting approach within the same experimental environment. The whole session was audio- and screen-recorded, and the study software logged participants' prompts and task duration. Screen and audio recordings were collected with consent. Participants took part voluntarily without compensation. The study is approved by a University of Waterloo research ethics board.

5.3 Procedure

Each participant completed two UI creation tasks (building a fantasy creature encyclopedia and building a planet discovery interface), one using the FFF chatbot and one using the QGF chatbot. The order of chatbot conditions, as well as the two different tasks, was counterbalanced using a Latin square design.

Introduction (~2 minutes). – Participants first completed a consent form and a demographic questionnaire. They were then introduced to the study procedure and told that they would use two different chatbot interfaces to complete UI-building tasks.

Task 1 (~25 minutes). – Participants were assigned to use either the FFF or QGF chatbot for the first UI-building task, and were encouraged to use the chatbot. Users were allowed to iterate on their interface by providing the chatbot with additional prompts.

Post-Task Survey (~3 minutes). – After completing the first task, participants completed the NASA-TLX questionnaire to assess perceived workload [21]. They also responded to two 5-point Likert-scale items (1 = strongly disagree, 5 = strongly agree): (1) The final result reflected my intentions. and (2) I felt ownership over the final interface that was created.

Task 2 (~25 minutes). – Participants then repeated the process with the chatbot they had not experienced before, and were tasked with a similarly scoped UI-building task.

Post-Task Survey (~3 minutes). – After completing the second task, participants again completed the NASA-TLX questionnaire and the same 5-point Likert-scale questions, allowing within-subjects comparison between the two chatbot formats.

Semi-Structured Interview (~10 minutes). – After completing both tasks, participants took part in a short semi-structured interview. We asked open-ended questions comparing their experiences with QGF and FFF, including which chatbot they preferred, what strengths and limitations they perceived in QGF, how they would redesign QGF, and how much control or ownership they felt when coding with each chatbot.

5.4 Design and Tasks

To avoid repetition effects, we designed two different but comparable UI-building tasks and assigned one to each chatbot condition. The two tasks were an Alien Planet Discovery Interface and a Fantasy Creature Encyclopedia. Full task prompts can be found in the Appendix A. We selected familiar and open-ended task topics so that participants would have some knowledge, but may not be deeply familiar with them or how to design an interface effectively for the task. To keep the tasks comparable in scope, both required participants to build a webpage in JavaScript, HTML, and CSS with the same three functional requirements: (1) a way to display multiple items, (2) a way for users to interact with or select items, and (3) a detailed view of a selected item. We intentionally defined these features at a high level rather than prescribing specific implementations, allowing us to better observe how participants interpreted the task and interacted with the chatbot while developing a solution.

5.5 Data Analysis

We conducted quantitative and qualitative analyses. Quantitatively, we compared task duration, prompt word count, NASA-TLX scores, and Likert scale ratings for how the final result reflected participants' intentions across conditions using Wilcoxon signed-rank tests. We also report 95% bootstrapped confidence intervals for the mean differences. Qualitatively, we conducted a thematic analysis [64] of the interview transcripts to identify perceived tradeoffs, prompting strategies, and user preferences.

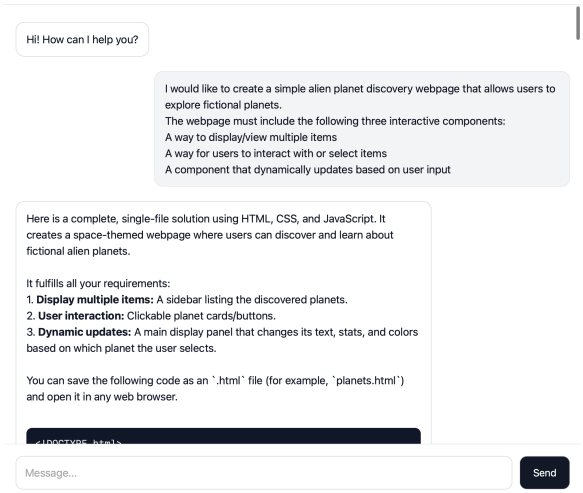


Figure 4: A visualization of the user interface for Free-form Format (FFF).

6 Results

This section presents the quantitative and qualitative findings comparing QGF and FFF across workload, task performance, and user experience. We report both statistical results and participant perceptions to provide a comprehensive understanding of how each approach shapes UI generation.

6.1 [RQ1:] Comparable Workload with Differences in Efficiency and Prompt Revisions

Participants completed the UI-building tasks without major difficulty in both conditions.

6.1.1 Cognitive Workload. Participants reported comparable workload between QGF and FFF across all six NASA_TLX dimension.

None of the differences between conditions reached statistical significance. Using mean differences computed as FFF − QGF, the largest differences were in Effort by -2.33 (95%CI[-7.049, 0.900], $p = .275$), Mental Demand by -2.25 (95%CI[-5.632, 1.020], $p = .193$), Physical Demand by -1.33 (95% CI[-4.612, 0.694], $p = .328$), Performance by -1.00(95% CI[-6.026, 4.034], $p = .691$), Frustration by 0.08 (95% CI[-3.939, 3.595], $p = .961$), and Temporal Demand by -0.08 (95% CI[-1.165, 1.162], $p = .750$).

6.1.2 Task Duration. Participants completed the tasks faster with FFF than with QGF (Figure 5). On average, task duration was 15.58 minutes shorter with FFF than QGF (95% CI [-25.91, -8.42], $p = .002$). This difference likely reflects that much of the additional time in QGF was spent on generating and working through the category questions before moving on to implementation.

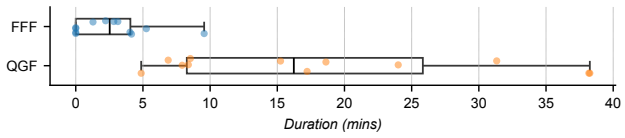


Figure 5: Task duration by technique.

6.1.3 Revision Prompts. Participants produced substantially more revision text with FFF than with QGF. We also examined the number of words participants typed when revising a prompt after being dissatisfied with an initial generation (Figure 6). This measure captures how much text participants entered to adjust the chatbot's next generation. The two conditions showed a difference of 24.25 words on average (95% CI [7.86, 55.69], $p = .012$).

(RQ1) Finding: While participants spent more time developing interfaces with QGF, they also made fewer revisions using subsequent prompts afterwards.

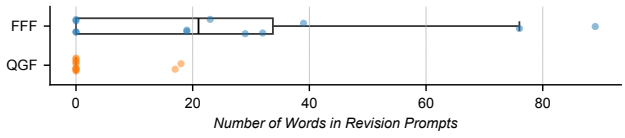


Figure 6: Revision prompts word count by technique.

6.2 [RQ2:] Satisfaction, Intention Reflection, and Ownership

Overall, most participants preferred using the QGF chatbot (66.6%, $n = 8$) over the FFF chatbot (33.3%, $n = 4$). Participants also reported comparable ratings of intention reflection and ownership between QGF and FFF. On a 5-point scale, ratings of whether the final results reflected participants' intentions were nearly identical across conditions, differing by only -0.08 points (95% CI[-0.925, 0.840], $p = .984$). Ratings of ownership were also similar, with FFF rated 0.50 points lower than QGF on average (95% CI[-1.116, 0.468], $p = .246$).

These results were also reflected in the interview. When asked whether participants felt more ownership over the generated website, several participants (P2, P4, P6, P11, P12) reported that they felt no difference in ownership when using either chatbots as they *"don't feel any ownership when I produce a system, a code with AI models"* (P11) and because *"you didn't interact with the code directly"* (P12). There were participants who said that they felt more ownership with QGF (P1, P3, P5, P8, P9), as they *"had more input in the process"* (P1); however, several did say that the ownership they felt using QGF over FFF was quite marginal.

(RQ2) Finding: Most participants preferred developing with QGF; however, they did not feel a significant degree of ownership over the final result when developing with either QGF or FFF.

6.3 [RQ3:] Participant Preferences and Perceived Differences Between QGF and FFF

6.3.1 Prompt Formulation and Guidance. Participants consistently described QGF as supporting more deliberate and structured prompt construction, while FFF required users to determine both the content and level of detail their prompts should include. Most participants ($n = 10$, 83.3%) described how QGF encouraged them to think more critically about what their website should include and look like. For instance, participants noted that QGF *"made them think more"* (P1) and that it was *"really good at forcing me to understand what my product would look like before the AI generated it"* (P10). Whereas with FFF, participants felt uncertain, as *"if you're just using the prompt, I'm afraid of like missing any details or any aspect."* (P6). Others similarly indicated that they *"just don't know how much information you have to put there"* (P7). These findings suggest that QGF provides guidance that supports more intentional prompt formulation, whereas FFF places greater responsibility on users to structure and scope their inputs.

6.3.2 Creativity and Design Exploration. Participants described both systems as supporting creativity in different ways. QGF was often seen as providing inspiration through its structured options, while FFF allowed for more open-ended exploration. Several participants (P5, P6, P7) highlighted how QGF supported creativity during the design process. As P6 explained, *"I can choose some options and also I can get some inspiration, [and] even if sometimes the options doesn't work for me, I can think of something better."* This indicates that the options provided by QGF not only guide users but can also serve as a source of inspiration, helping them expand or refine their ideas. However, other participants (P1, P7, P10, P11) felt that QGF constrained their creativity. They noted that *"the bounds for what the website would look like are framed by the questions"* (P1) and that *"QGF was kind of restrictive"* (P11). Three participants (P7, P10, P11) expressed a preference for more open-ended input, *"so that I could think [myself] and added... my own thoughts to it instead of choosing one of the suggested options by the system."* (P11). Conversely, FFF was associated with greater flexibility, allowing participants to explore ideas more freely, though sometimes at the cost of direction or clarity. For example, one participant noted that with *"[FFF], I didn't have anything in mind"* (P2), reflecting how the open-ended format may leave users without clear starting points

or ideas. Therefore, QGF and FFF support creativity in complementary ways: QGF provides structured inspiration that can guide idea development, while FFF enables open-ended exploration but with less support for forming initial concepts.

6.3.3 Iteration Speed and Workflow Efficiency. Participants described clear differences in how the two systems supported iteration and workflow speed. Many participants (P1, P4, P5, P8, P9, P10, P12) reported that QGF was time-consuming when developing an interface. For example, participants expressed a desire for *"a faster iteration cycle"* (P1) and noted that they were *"not [able] to see where it was going"* (P5), as it *"took a long time to get the results"* (P4). Meanwhile, FFF allowed participants to *"get an initial sense of what it output very quickly"* (P1). This immediacy supported rapid exploration and refinement, even if it sometimes required more revisions. These responses suggest that, while QGF supported more structured prompt construction, participants preferred a workflow that allowed for quicker iteration and more immediate feedback.

6.3.4 Interaction Friction and System Limitations. Participants identified different sources of interaction friction in QGF and FFF, related to structure, repetition, and familiarity. For QGF, a commonly reported issue was the repetition of questions throughout the interface (P1, P2, P3, P5). Participants pointed out *"repeat[s] of [earlier] questions"* (P1) and felt that *"some of the questions were really similar"* (P5). This repetition suggests a limitation in how the system maintains and utilizes prior context, leading to redundant interactions and a less fluid experience. In contrast, FFF was perceived as easier to use due to participants' prior familiarity with similar systems. Two participants (P1, P10) noted that interacting in a free-form format felt more natural, with P1 explaining that *"the other large language models that I've gotten used to working with... ask questions in more of a freeform format, and so it felt more familiar"*. This familiarity reduced interaction friction, even though FFF required more effort in constructing prompts. Overall, participants perceived QGF and FFF as introducing different types of interaction tradeoffs.

(RQ3) Finding: Participants liked QGF's support for reflection and structured prompt formulation, but disliked its increased time cost and had mixed views on its impact on creativity.

7 Discussion

This section synthesizes our key findings and contributions, explores design implications for AI-assisted development tools, and discusses limitations and directions for future work.

7.1 Key Findings and Contributions

This work investigates how structured and free-form prompting interfaces shape user behaviour and experience in GenAI-assisted UI development. Our findings highlight several key contributions.

We demonstrate that structuring prompt input through question-guided formats (QGF) fundamentally changes how users approach prompt construction. Rather than iteratively refining outputs through trial and error, participants using QGF engaged in more deliberate upfront planning, reflecting on features, constraints, and design goals before generating a UI. This suggests that interface design can meaningfully influence not just what users produce, but how they think through a task. Our findings extend prior work on structured prompting templates [40, 44], which showed that scaffolded inputs can reduce ambiguity. Our work demonstrates that these benefits transfer to complex, open-ended UI development tasks where requirements are often ill-defined [38].

Next, we identify a clear tradeoff between upfront guidance and iterative efficiency. While QGF increased task duration due to the time required to work through structured questions, it also reduced the need for subsequent revisions. In contrast, free-form prompting (FFF) enabled faster initial interaction and rapid iteration, but required more frequent and extensive prompt refinement. These findings highlight two distinct interaction paradigms: planning-oriented prompting versus iteration-oriented prompting. This aligns with Zamfirescu-Pereira et al. [70]'s observation that non-expert users struggle to construct effective prompts from scratch, and suggests that structured interfaces can compensate for limited AI literacy without requiring users to learn explicit prompt engineering strategies.

Our results also illustrate that structured guidance shapes users' creative processes in nuanced ways. QGF supported creativity by providing inspiration and prompting reflection, but was also perceived as constraining by limiting the space of possible designs. FFF, on the other hand, allowed for more open-ended exploration, though often at the cost of direction and clarity. Findings from AI-assisted ideation research similarly illustrate that structured scaffolding can both support and constrain divergent thinking [35, 62]. This demonstrates that creativity in GenAI systems is not solely a function of model capability, but it is also mediated by interface design.

Lastly, despite minimal differences in perceived ownership and control, most participants preferred QGF over FFF. This suggests that users may value guidance and structure in the design process, even when it introduces additional time costs. Together, these findings contribute to a better understanding of how input structuring influences prompting behaviour, user experience, and interaction strategies in AI-assisted development.

7.2 Design Implications: Structuring Guidance Without Limiting Creativity

Our findings suggest several important implications for the design of future GenAI coding interfaces.

Structuring prompting as a guide rather than a constraint appears to be particularly effective. QGF demonstrates that guiding users through well-organized questions can help them clarify goals, identify relevant details, and reduce uncertainty when constructing prompts. Interfaces that provide this type of guidance can be especially valuable for novice users or those less familiar with a domain. By prompting users to consider elements they might otherwise overlook, the process ensures that important or relevant information is included while still encouraging creative exploration.

At the same time, while QGF promotes more careful consideration of design choices, it requires additional time to answer all of the questions. Designers of GenAI interfaces may benefit from hybrid approaches that preserve the advantages of structured guidance while maintaining rapid feedback loops to support iterative exploration. The format of the questions can also be varied

to suit their purpose—for example, open-ended questions may be more suitable for capturing the overall atmosphere or style of a UI, whereas multiple-choice or selection-based questions can help specify concrete interactions with finite options.

Finally, our study highlights that no single prompting format is universally superior. Structured interfaces enhance deliberation and confidence, whereas free-form interfaces support speed and flexibility. Future designs might allow users to dynamically toggle between structured and free-form modes, or combine elements of both, balancing reflection with efficiency.

By evaluating QGF as a technique, we gain insight into how structured input shapes prompting behaviour and uncover strategies that can improve both user experience and effectiveness in AI-assisted UI development. More broadly, our findings carry implications for AI system designers beyond prompt engineering. Rather than treating prompt quality solely as a user skill to be cultivated through training or documentation, our results suggest that it can be directly shaped through interface-level design decisions. AI system designers can embed requirement elicitation strategies such as category-based questioning, progressive disclosure, and structured compilation directly into the interaction layer of GenAI tools. This reframes a traditionally user-side challenge as a system design opportunity, offering concrete guidance for how future AI development tools can be architected to support more intentional, reflective, and effective human-AI collaboration from the outset.

7.3 Limitations and Future Work

Our findings should be interpreted in light of several limitations. As a preliminary, exploratory study, this work was designed to investigate whether question-guided prompting could meaningfully shape how users formulate requirements and interact with an LLM during early-stage UI generation. Accordingly, these results should be interpreted as highlighting the potential benefits and trade-offs of this approach, rather than providing definitive evidence of its effectiveness across different contexts and settings.

First, this was a small within-subjects study with 12 participants recruited from one university community. Although this sample was appropriate for an initial investigation, it limits the generalizability of our findings to broader populations, such as professional developers, domain experts, or users with limited coding experience. Participants in our study were also relatively familiar with GenAI tools, which may have influenced how easily they adapted to both chatbot formats.

Second, our study focused on two short, controlled UI-building tasks in specific domains. While this allowed us to compare FFF and QGF under consistent conditions, it also limits the scope of our conclusions. The questions used in QGF were designed around the needs of these particular tasks and domains, and may not transfer directly to other types of interfaces or software projects. Real-world development often involves more open-ended goals, evolving requirements, debugging, and iterative refinement over time. As a result, our findings primarily reflect early-stage requirement articulation in a constrained setting, rather than end-to-end development practice across domains.

Third, we evaluated one specific implementation of a question-guided interface. The effects we observed may depend not only on the presence of structured guidance, but also on how that guidance was designed, including the choice of question categories, the wording and order of questions, pacing of the interaction, and the way responses were compiled into a final prompt. In addition, our prototype relied on a particular model configuration. Other LLMs may respond differently to both free-form and structured inputs.

Finally, our evaluation focused primarily on process-oriented measures, such as task duration, workload, prompting behavior, and perceived control. We did not systematically assess the quality of the final generated interfaces, such as their correctness, completeness, usability, or maintainability. Without such artifact-level analysis, it remains unclear whether the benefits of guided prompting for reflection and structure translate into better design outcomes.

These limitations point to several directions for future work. A first next step is to refine the design of QGF itself. For example, future iterations could reduce redundancy, simplify or merge overlapping questions, and make the questioning process more adaptive to the user and task. Rather than presenting the same fixed set of categories to all users, a next-generation QGF could ask fewer but more targeted questions based on the project description, the user's expertise, or the stage of the design process. This may help preserve the reflective benefits of structured guidance while reducing the sense of burden or interruption.

Future work should also examine how question-guided prompting performs in more realistic and longitudinal development settings, where users return to the same project over time and engage in broader workflows that include revising requirements, debugging, and refining generated code. In addition, it would be valuable to explore hybrid interfaces that combine the speed and openness of free-form prompting with the scaffolding benefits of question-guided input. For instance, users might begin with free-form ideation and then selectively invoke structured guidance when clarifying requirements or resolving ambiguity. Finally, future studies should evaluate not only user experience, but also the quality of generated artifacts across different domains, in order to better understand when and for whom guided prompting is most beneficial.

8 Conclusion

We presented a within-subjects study comparing two prompting formats for GenAI-assisted UI creation: FFF and QGF. Our findings show that the two formats supported different interaction styles and tradeoffs. FFF enabled faster task completion but more revision after initial generation, while QGF encouraged participants to reflect on design requirements and structure their requests more explicitly. At the same time, participants reported comparable workload, intention reflection, and ownership across the two conditions. Taken together, these findings suggest that prompting format is not a neutral interface choice. Rather, it shapes how users plan, communicate, and maintain agency when working with GenAI systems. By surfacing the trade-off between speed and guidance, our work contributes empirical evidence for the design of future AI-assisted development tools and points toward hybrid interfaces that better balance flexibility with reflective support.

Acknowledgments

We thank the anonymous reviewers for their feedback, which helped improve our work. We also thank our participants for their time and insights. This work was supported in part by NSERC Discovery Grant RGPIN-2025-01552.

References

[1] [n. d.]. Handle approvals and user input - Claude Code Docs — code.claude.com. https://code.claude.com/docs/en/agent-sdk/user-input?utm_source=chatgpt.com. [Accessed 12-05-2026].

[2] [n. d.]. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. doi:10.1145/3411763.3451760

[3] [n. d.]. THE FUNCTIONS OF THE DESIGN BRIEF. ([n. d.]).

[4] [n. d.]. What Guides Our Choices? Modeling Developers' Trust and Behavioral Intentions Towards GenAI | Proceedings of the IEEE/ACM 47th International Conference on Software Engineering. https://dl.acm.org/doi/10.1109/ICSE55347.2025.00087

[5] Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. 2024. STaR-GATE: Teaching Language Models to Ask Clarifying Questions. doi:10.48550/arxiv.2403.19154 arXiv:2403.19154 [cs].

[6] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3613904.3642016

[7] Amna Batool and Waqar Hussain. 2025. Evaluating the Usability and Ethical Implications of Graphical User Interfaces in Generative AI Systems. *Computers* 14, 10 (Oct. 2025), 418. doi:10.3390/computers14100418

[8] Stephen Brade, Bryan Wang, Maurício Sousa, Sageev Oore, and Tovi Grossman. 2023. Promptify: Text-to-Image Generation through Interactive Prompt Exploration with Large Language Models. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (2023). https://api.semanticscholar.org/CorpusId:258212979

[9] Félix Buendía-García and Javier Piris-Ruano. 2025. Using Generative AI to Support UX Design Students in Web Development Courses. *Applied Sciences* 15, 13 (Jan. 2025), 7389. doi:10.3390/app15137389

[10] Caner Börekci and Özgür Çelik. 2024. Exploring The Role of Digital Literacy in University Students' Engagement with AI through the Technology Acceptance Model. *Sakarya University Journal of Education* 14, Special Issue-AI in Education (Aug. 2024), 228–249. doi:10.19126/suje.1468866

[11] Maya Cakmak and Andrea L. Thomaz. 2012. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction (HRI '12)*. Association for Computing Machinery, New York, NY, USA, 17–24. doi:10.1145/2157689.2157693

[12] Hung-Fu Chang and Tong Li. 2025. A framework for collaborating a Large Language Model tool in brainstorming for triggering creative thoughts. *Thinking Skills and Creativity* 56 (June 2025), 101755. doi:10.1016/j.tsc.2025.101755

[13] Helen Weixu Chen and Lesley Istead. 2024. "Imagine a Dress": Exploring the case of task-specific prompt assistants for text-to-image AI tools. In *Proceedings of the 50th Graphics Interface Conference* (Halifax, NS, Canada) *(GI '24)*. Association for Computing Machinery, New York, NY, USA, Article 13, 8 pages. doi:10.1145/3670947.3670972

[14] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. doi:10.48550/arxiv.2107.03374 arXiv:2107.03374 [cs].

[15] Joanne Chia and Angela Frattarola. 2025. A design-based approach to analysing student engagement with a GenAI-Enabled brainstorming app. *Computers and Education: Artificial Intelligence* 9 (Dec. 2025), 100468. doi:10.1016/j.caeai.2025.100468

[16] Hui-Chun Chu, Yi-Chun Lu, and Yun-Fang Tu. 2025. How GenAI-supported multimodal presentations benefit students with different motivation levels: Evidence from digital storytelling performance, critical thinking awareness, and learning attitude. *Educational Technology & Society* 28, 1 (2025), 250–269. https://www.jstor.org/stable/48810718

[17] John Joon Young Chung and Eytan Adar. 2023. PromptPaint: Steering Text-to-Image Generation Through Paint Medium-like Interactions. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (2023). https://api.semanticscholar.org/CorpusId:260775964

[18] Anujkumarsinh Donvir, Sriram Panyam, Gunjan Paliwal, and Praveen Gujar. 2024. The Role of Generative AI Tools in Application Development: A Comprehensive Review of Current Technologies and Practices. In *2024 International Conference on Engineering Management of Communication and Technology (EMCTECH)*. 1–9. doi:10.1109/EMCTECH63049.2024.10741797 ISSN: 3064-9382.

[19] Anastasios A Economides and Maria Perifanou. 2024. Higher education students using GenAI tools to design websites. (2024).

[20] Caterina Fuligni, Daniel Dominguez Figaredo, and Julia Stoyanovich. 2025. "Would You Want an AI Tutor?" Understanding Stakeholder Perceptions of LLM-based Systems in the Classroom. doi:10.48550/arxiv.2503.02885 arXiv:2503.02885 [cs].

[21] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[22] Moritz Joerling. 2025. BUILDING GENAI-DRIVEN WEB APPLICATIONS FOR MARKETING RESEARCH: A METHODOLOGICAL GUIDE. doi:10.2139/ssrn.5256590

[23] August Jönsson. 2024. *Prompting for progression : How well can GenAI create a sense of progression in a set of multiple-choice questions?* https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-346350

[24] Engin Kapkın and Sharon Joines. 2021. The Design Brief as a Creativity Catalyst in Design Education: Priming through Problem Statement. *International Journal of Art & Design Education* 40, 1 (2021), 126–145. doi:10.1111/jade.12339 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jade.12339.

[25] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (April 2023), 102274. doi:10.1016/j.lindif.2023.102274

[26] Greg Kestin, Kelly Miller, Anna Klales, Timothy Milbourne, and Gregorio Ponti. 2025. AI tutoring outperforms in-class active learning: an RCT introducing a novel research-based design in an authentic educational setting. *Scientific Reports* 15, 1 (June 2025), 17458. doi:10.1038/s41598-025-97652-6

[27] Jeffrey L Krichmar, Tiffany Hwu, Xinyun Zou, and Todd Hylton. 2019. Advantage of prediction and mental imagery for goal-directed behaviour in agents and robots. *Cognitive Computation and Systems* 1, 1 (2019), 12–19.

[28] Tarika Kumar, Matteo Zallio, and Xinyi Tu. [n. d.]. How Generative AI is reshaping UI/UX Design Workflows: A Systematic Review. ([n. d.]).

[29] Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. [n. d.]. ELICITING HUMAN PREFERENCES WITH LANGUAGE MODELS. ([n. d.]).

[30] Molly Li and Joshua Wilson. 2025. AI-Integrated Scaffolding to Enhance Agency and Creativity in K-12 English Language Learners: A Systematic Review. *Information* 16, 7 (July 2025), 519. doi:10.3390/info16070519

[31] Ruijia Li, Yuan-Hao Jiang, Jiatong Wang, and Bo Jiang. 2025. How Real Is AI Tutoring? Comparing Simulated and Human Dialogues in One-on-One Instruction. https://arxiv.org/abs/2509.01914v1

[32] Shuang Li, Yuntao Cheng, Jinfu Chen, Jifeng Xuan, Sen He, and Weiyi Shang. 2026. Performance analysis of AI-generated code: A case study of Copilot, Copilot Chat, CodeLlaMa, and DeepSeek-Coder models. *Empirical Software Engineering* 31, 3 (May 2026), 62. doi:10.1007/s10664-025-10776-1

[33] Sitong Li, Stefano Padilla, Pierre Le Bras, Junyu Dong, and Mike Chantler. 2025. A Review of LLM-Assisted Ideation. doi:10.48550/arxiv.2503.00946 arXiv:2503.00946 [cs].

[34] Yunyao Li, Ishan Chaudhuri, Huahai Yang, Satinder Singh, and HV Jagadish. 2007. Enabling domain-awareness for a generic natural language interface. In *AAAI*. 833–838.

[35] Gionnieve Lim and Simon T. Perrault. 2024. Rapid AIdeation: Generating Ideas With the Self and in Collaboration With Large Language Models. doi:10.48550/arxiv.2403.12928 arXiv:2403.12928 [cs].

[36] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–31. doi:10.1145/3544548.3580817

[37] Qingtang Liu, Yanli Wang, and Yubei Chang. 2025. Exploring the effects of scaffolded reflective learning on student teachers' design performance and reflective thinking. *Thinking Skills and Creativity* 56 (June 2025), 101709. doi:10.1016/j.tsc.2024.101709

[38] Diana Loeffler, Anne Hess, Andreas Maier, Joern Hurtienne, and Hartmut Schmitt. 2013. Developing Intuitive User Interfaces by Integrating Users' Mental Models

into Requirements Engineering. BCS Learning & Development. doi:10.14236/ewic/HCI2013.14
[39] Francesca Lucchetti, Zixuan Wu, Arjun Guha, Molly Q. Feldman, and Carolyn Jane Anderson. 2024. Substance Beats Style: Why Beginning Students Fail to Code with LLMs. doi:10.18653/v1/2025.naacl-long.433 arXiv:2410.19792 [cs].
[40] Stephen MacNeil, Andrew Tran, Joanne Kim, Ziheng Huang, Seth Bernstein, and Dan Mogil. 2023. *Prompt Middleware: Mapping Prompts for Large Language Models to UI Affordances.* arXiv:2307.01142 [cs] doi:10.48550/arxiv.2307.01142
[41] Martin Maguire. 2001. Context of use within usability activities. *International journal of human-computer studies* 55, 4 (2001), 453–483.
[42] B. Barry Mahoney, R. Ron Oostdam, H. Hessel Nieuwelink, and J. Jaap Schuitema. 2023. Learning to think critically through Socratic dialogue: Evaluating a series of lessons designed for secondary vocational education. *Thinking Skills and Creativity* 50 (Dec. 2023), 101422. doi:10.1016/j.tsc.2023.101422
[43] Yuetian Mao, Junjie He, and Chunyang Chen. 2025. From Prompts to Templates: A Systematic Prompt Template Analysis for Real-world LLMapps. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering.* Association for Computing Machinery, New York, NY, USA, 75–86. https://dl.acm.org/doi/10.1145/3696630.3728533
[44] Yuetian Mao, Junjie He, and Chunyang Chen. 2025. From prompts to templates: A systematic prompt template analysis for real-world LLMapps. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering.* 75–86.
[45] Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2024. When to Show a Suggestion? Integrating Human Feedback in AI-Assisted Programming. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 9 (March 2024), 10137–10144. doi:10.1609/aaai.v38i9.28878
[46] Veera Harish Muthazhagu and Surendiran B. 2024. Exploring the Role of AI in Web Design and Development: A Voyage through Automated Code Generation. In *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE).* 1–8. doi:10.1109/IITCEE59897.2024.10467409
[47] Nathan Novemsky and Daniel Kahneman. 2005. The boundaries of loss aversion. *Journal of Marketing research* 42, 2 (2005), 119–128.
[48] Julian Oertel, Jil Klünder, and Regina Hebig. 2025. Don't Settle for the First! How Many GitHub Copilot Solutions Should You Check? 183 (2025), 107737. doi:10.1016/j.infsof.2025.107737
[49] Yu Pan, Lixun Wang, and Yidan Zhu. 2024. Strategic questioning for formative assessment in TEFL: insights from blended synchronous learning environments. *Humanities and Social Sciences Communications* 11, 1 (Nov. 2024), 1519. doi:10.1057/s41599-024-04086-y
[50] Veronica Pimenova, Sarah Fakhoury, Christian Bird, Margaret-Anne Storey, and Madeline Endres. 2025. Good vibrations? A qualitative study of co-creation, communication, flow, and trust in vibe coding. *arXiv preprint arXiv:2509.12491* (2025).
[51] Brian L Quick, Lijiang Shen, and James Price Dillard. 2013. Reactance theory. *The SAGE handbook of persuasion* (2013), 167–183.
[52] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2025. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. doi:10.48550/arxiv.2402.07927
[53] Peter A Sandon. 1989. An attentional hierarchy. *Behavioral and Brain Sciences* 12, 3 (1989), 414–415.
[54] Advait Sarkar and Ian Drosos. 2025. Vibe coding: programming through conversation with artificial intelligence. *arXiv preprint arXiv:2506.23253* (2025).
[55] David Scott. [n. d.]. Inquiry-Based Learning: A Review of the Research Literature. ([n. d.]).
[56] H. E. Shanyun and Shen Yan. 2025. Learn to Question: Study on the Pattern of Student-GenAI Collaborative Learning in Higher Education. *Frontiers of Education in China* 20, 4 (Dec. 2025), 450. doi:10.3868/s110-020-025-0024-1
[57] Md Istiak Hossain Shihab, Christopher Hundhausen, Ahsun Tariq, Summit Haque, Yunhan Qiao, and Brian Wise Mulanda. 2025. The Effects of GitHub Copilot on Computing Students' Programming Effectiveness, Efficiency, and Processes in Brownfield Coding Tasks. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V.1.* ACM, Charlottesville USA, 407–420. doi:10.1145/3702652.3744219
[58] Md Kamrul Siam, Huanying Gu, and Jerry Q. Cheng. 2024. Programming with AI: Evaluating ChatGPT, Gemini, AlphaCode, and GitHub Copilot for Programmers. doi:10.1145/3723178.3723224 arXiv:2411.09224 [cs].
[59] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An Analysis of the Automatic Bug Fixing Performance of ChatGPT. doi:10.1109/apr59189.2023.00012 arXiv:2301.08653 [cs].
[60] Karthik Sunil and Aalok Thakkar. 2025. SocraticAI: Transforming LLMs into Guided CS Tutors Through Scaffolded Interaction. https://arxiv.org/abs/2512.03501v1
[61] Ningzhi Tang, Meng Chen, Zheng Ning, Aakash Bansal, Yu Huang, Collin McMillan, and Toby Jia-Jun Li. 2023. An Empirical Study of Developer Behaviors for Validating and Repairing AI-Generated Code. (March 2023). doi:10.1184/R1/22223533.v1
[62] Sirui Tao, Ivan Liang, Cindy Peng, Zhiqing Wang, Srishti Palani, and Steven Dow. 2025. DesignWeaver: Dimensional Scaffolding for Text-to-Image Product Design. *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (2025). https://api.semanticscholar.org/CorpusId:276394430
[63] Davide Tavernini, Efstathios Velenis, and Stefano Longo. 2017. Feedback brake distribution control for minimum pitch. *Vehicle System Dynamics* 55, 6 (2017), 902–923.
[64] Gareth Terry, Nikki Hayfield, Victoria Clarke, Virginia Braun, et al. 2017. Thematic analysis. *The SAGE handbook of qualitative research in psychology* 2, 17-37 (2017), 25.
[65] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22).* Association for Computing Machinery, New York, NY, USA, 1–7. doi:10.1145/3491101.3519665
[66] Mireia Vendrell and Samantha-Kaye Johnston. 2026. Scaffolding critical thinking with generative AI: Design principles for integrating large language models in higher education. *Computers and Education: Artificial Intelligence* 10 (June 2026), 100572. doi:10.1016/j.caeai.2026.100572
[67] Zhijie Wang, Yuheng Huang, Da Song, Lei Ma, and Tianyi Zhang. 2024. PromptCharm: Text-to-Image Generation through Multi-modal Prompting and Refinement. *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (2024). https://api.semanticscholar.org/CorpusID:268264261
[68] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. doi:10.48550/arxiv.2201.11903 arXiv:2201.11903 [cs].
[69] Hui Yun Yen, Po Hsien Lin, and Rungtai Lin. 2014. Emotional product design and perceived brand emotion. *International Journal of Advances in Psychology (IJAP)* 3, 2 (2014), 59–66.
[70] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23).* Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3544548.3581388
[71] Sai Zhang, Zhenchang Xing, Ronghui Guo, Fangzhou Xu, Lei Chen, Zhaoyuan Zhang, Xiaowang Zhang, Zhiyong Feng, and Zhiqiang Zhuang. 2024. Empowering Agile-Based Generative Software Development through Human-AI Teamwork. *ACM Transactions on Software Engineering and Methodology* 34 (2024), 1 – 46. https://api.semanticscholar.org/CorpusId:271328890

A Tasks

A.1 Alien Planet Discovery Interface

You are asked to create a simple planet discovery webpage that allows users to explore fictional planets. Your webpage must include the following three interactive features:

- A way to display/view multiple items
- A way for users to interact with or select items
- A detailed view of a selected item

Please use JavaScript, HTML, and CSS to develop the website. You will use theAI tool provided to assist with your development.

We are not evaluating your coding ability. Instead, we are studying how you interact with the AI tool while completing a development task. Please try to spend no more than 25 minutes on this task. You may stop at any time.

A.2 Fantasy Creature Encyclopedia

You are asked to create a simple fantasy creature encyclopedia webpage that allows users to explore fictional creatures. The creatures can be entirely fictional (for example: dragons, forest spirits, shadow beasts, etc.). You may design the creatures and interface however you like. Your webpage must include the following three interactive features:

- A way to display/view multiple items
- A way for users to interact with or select items
- A detailed view of a selected item

Please use JavaScript, HTML, and CSS to develop the website. You will use the AI tool provided to assist with your development.

We are not evaluating your coding ability. Instead, we are studying how you interact with the AI tool while completing a development task. Please try to spend no more than 25 minutes on this task. You may stop at any time.