Closing the Loop between User Stories and GUI Prototypes: An LLM-Based Assistant for Cross-Functional Integration in Software Development

Felix Kretzer*
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
felix.kretzer@kit.edu

Kristian Kolthoff*
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
kristian.kolthoff@tu-clausthal.de

Christian Bartelt
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
christian.bartelt@tu-clausthal.de

Simone Paolo Ponzetto
Data and Web Science Group
University of Mannheim
Mannheim, Germany
ponzetto@uni-mannheim.de

Alexander Maedche
human-centered systems lab (h-lab)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
alexander.maedche@kit.edu

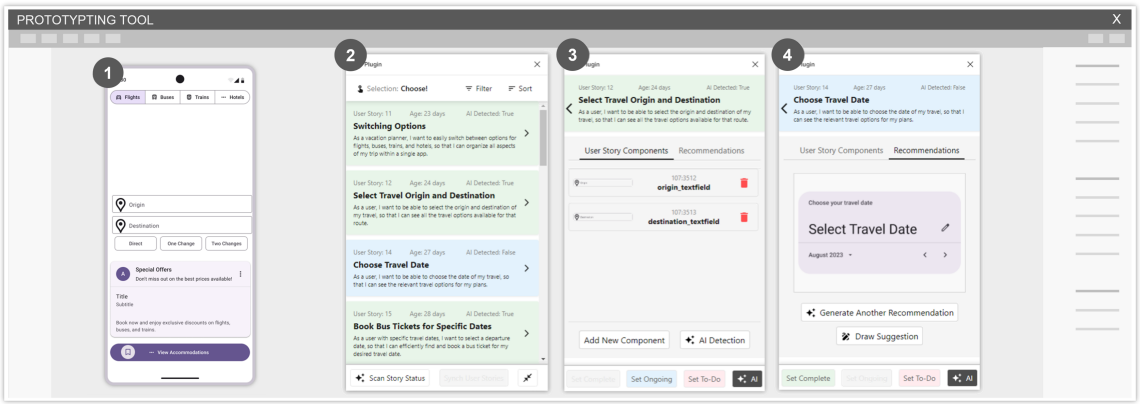

Figure 1: GUI prototype (*1*) and three views (*2-4*) of our assistant for GUI prototype designers integrated as a plug-in into a prototyping tool. Our assistant displays user stories (*2*) imported from collaboration tools (e.g., *JIRA*) for prototype designers to reference while working. It detects whether a user story is implemented (*3, 4*), identifies relevant GUI components (*3*), and generates GUI components for user stories (*4*). Figure uses *Google Material 3 Design Kit* [24] components under *CC BY 4.0*.

Abstract

Graphical user interfaces (GUIs) are at the heart of almost every software we encounter. GUIs are often created through a collaborative effort involving UX designers, product owners, and software developers, constantly facing changing requirements. Historically, problems in GUI development include a fragmented, poorly integrated tool landscape and high synchronization efforts between stakeholders. Recent approaches suggest using large language models (LLMs) to recognize requirements fulfillment in GUIs and automatically propose new GUI components. Based on ten interviews with practitioners, this paper proposes an LLM-based assistant as a *Figma* plug-in that bridges the gap between user stories and GUI prototyping. We evaluated the prototype with 40 users and 40 crowd-workers, showing that the effectiveness of GUI creation is improved by using LLMs to detect requirements' completion and generate new GUI components. We derive design rationales to support cross-functional integration in software development, ensuring that our plug-in integrates well into established processes.

CCS Concepts

• **Software and its engineering** → **Software usability**; **Requirements analysis**; *Consistency*; • **Human-centered computing** → *User interface toolkits*; Laboratory experiments.

*Both authors contributed equally to the paper.

CHI '25, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3713932

Keywords

GUI Prototypes; User Stories; Requirements; Assistance

ACM Reference Format:
Felix Kretzer, Kristian Kolthoff, Christian Bartelt, Simone Paolo Ponzetto, and Alexander Maedche. 2025. Closing the Loop between User Stories and GUI Prototypes: An LLM-Based Assistant for Cross-Functional Integration in Software Development. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3706598.3713932

1 Introduction

Graphical user interfaces (GUIs) are ubiquitous and are at the heart of today's software. At their best, they allow us to interact effectively with the multitude of software applications. Both in practice and in research, there has been an extensive exploration of methods and approaches that facilitate creating better software with even more effective GUIs. An important technique is the creation of GUI prototypes, which are subsequently employed as a point of reflection, for example, during the elicitation and refinement of requirements with stakeholders [3, 40, 46]. Many aspects of prototyping have been investigated, including low- to high-fidelity prototypes [53, 57], throw-away or evolutionary prototypes [3], or agile vs. more traditional methods [7, 36]. While the rapid creation of GUI prototypes for communicating requirements provides many benefits, it simultaneously carries challenges. Typically, the creation of GUI prototypes necessitates the collaboration of UX designers, requirements analysts (e.g., product owners), and software developers. In practice, software development teams often encounter different tools that are only partially integrated [47]. Although efforts are being made to integrate workflows (e.g., *Figma's* developer mode [21]), core processes are still entirely separate. Furthermore, the requirements are subject to continuous adaption and extension during the development [15]. Changing requirements leading to extensive communication effort has been described in literature, and was one of the most frequently mentioned topics during interviews conducted as part of this study. Due to the sustained modification of requirements, not only synchronization efforts between the roles in the development teams are additionally increased, but also the effort to continuously update the respective GUI prototypes.

In practice, UX designers create prototypes based on requirements provided in various forms. For example, requirements are sometimes only verbally discussed and quickly turned into paper prototypes. However, a prominent approach is to explicitly articulate requirements, e.g., as user stories [14]. Requirements are sometimes formalized after building initial GUI prototypes, causing ambiguity in early prototyping. Software developers typically work with GUI prototypes and formalized requirements, where changes during the technical implementation can create a synchronization effort for both the formalized requirements and GUI prototypes. We found that updating requirements in prototypes can be overlooked, making the implemented version the de facto latest version.

Prior research proposed several approaches to adapt or improve the GUI prototyping process. For instance, GUI prototyping assistants such as *GUIComp* [37] retrieve similar GUI screens to stimulate design inspirations and provide complexity metrics to enhance prototypes. Moreover, a plethora of approaches for GUI retrieval has been proposed before, utilizing various input formats such as natural language [5, 33, 34], sketches [27], or screenshots [38]. More recently, research proposed GUI generation approaches, for example, based on training or fine-tuning an LLM [8, 20]. However, these approaches lack the capability of integrating user requirements and the respective GUI prototype and cannot directly generate GUI component implementations based on user stories for fine-grained GUI prototyping support. Meanwhile, UX professionals have shown to be open for AI-based support systems and see merit in their potential for addressing practical challenges [30]. Yet, relying solely on chat-based LLMs to create GUI prototypes from user stories raises several issues, such as how to provide the current prototype state to the LLM, and limitations of static prompt-response interactions - most notably - integrating recommendations into prototyping tools. Additionally, current GUI prototyping often requires designers to repeatedly customize the same GUI components, adding significant effort to the prototyping process. New requirements further demand manual adaptations to align with evolving user stories, adding to the workload. Meanwhile, untrained designers struggle to start from scratch and create effective prototypes [37].

While a first approach proposed the utilization of LLMs for integrating requirements and GUI prototypes via detection of user story implementation, matching of GUI components and generating GUI components from user stories [35], these approaches have not yet been implemented in a system, nor systematically evaluated with actual users. To address this research gap, we follow the research question of *how to design an assistant to increase the effectiveness of GUI prototyping under consideration of cross-functional integration in software development*? Our overall contribution is twofold, *(i)* supporting UX designers during their GUI prototyping efforts, and, *(ii)* enhancing cross-functional integration in software development teams, focusing on the roles of UX designers, product owners, and software developers. In detail we contribute by:

- Providing insights into GUI prototyping challenges from practice and providing design rationales for an effective LLM-based assistant tailored at UX designers working within software development teams.
- Presenting a novel plug-in for *Figma* with role-specific functionalities. The interface for UX designers integrates an approach within a prototyping environment (e.g., *Figma*) that utilizes a state-of-the-art LLM to *(i)* assess whether a requirement is completed, *(ii)* identifies GUI components completing the requirement, and *(iii)* generates recommendations for UI components, fulfilling a given requirement.
- An empirical evaluation of the functionalities of our proposed GUI prototyping assistant for UX designers and the resulting prototype quality and user story completion.
- A novel approach and technical evaluation to derive additional requirements from GUI design prototypes tailored at product owners working in software development teams.

Our findings demonstrate that GUI prototypes created with our LLM-based approach achieve higher quality, particularly in selecting appropriate components, using correct labels and completing given requirements in contrast to manually created GUI prototypes. Additionally, user stories automatically derived from existing GUIs with our approach were rated high for accuracy and specificity.

2 Related Work

In the following we present insights into relevant prior research in the fields of (automated) analysis assistants for GUI prototyping, automated GUI generation, and GUI retrieval approaches (e.g., to efficiently draw design inspirations).

2.1 GUI Prototyping Assistants

To support the GUI prototyping process, different approaches have been proposed in research. For example, *DesignScope* [42, 45] facilitates the design process by actively recommending layout refinements and design suggestions. Moreover, *SketchPlorer* [54] represents a sketching approach that integrates an ad-hoc layout optimizer to rapidly provide layout improvement suggestions to users. In addition, *GUIComp* [37] provides assistance for novices during GUI prototyping through a multi-faceted support system including the retrieval of similar GUIs from the *Rico* dataset [17]. *GUIComp* offers a visualization of several GUI complexity metrics such as GUI component alignment, balance and density, and an attention map for the created GUI prototype. In contrast, our approach targets rapid generation of GUI prototypes (components) based on fine-grained user stories that can instantly be integrated into the current working GUI prototype. We close this loop and provide a tight integration of user requirements (in the form of user stories) and resulting GUI prototypes. To ensure consistency between user requirements and GUI prototypes, an ontology-based approach was proposed before [51]. However, this approach necessitates the availability of a respective ontology for the domain and enables mere validation of the requirements, thus, cannot actively support during the creation of the GUI prototype. More recently, an approach for interlinking user stories and GUI prototyping has been proposed [35] introducing the LLM-based detection of user stories in GUI prototypes and the matching to the corresponding GUI components within the prototype. With our research, we extend and build on this work by proposing an LLM-based approach for rapidly generating the implementations for user stories in an editable form as well as an integration of the LLM-based assistant in the form of a fully-fledged plug-in in *Figma*.

2.2 Automated GUI Generation

Prior research on automatically generating GUI prototypes followed different approaches. For example, *GUIGAN* [59] generates new GUI images based on combining existing GUI screenshot excerpts automatically extracted from the *Rico* dataset [17] by employing a generative adversial network [22]. Moreover, the two similar approaches *LayoutTransformer* [25] and *Variational Transformer Networks* [1] mainly generate low-fidelity GUI layouts from much simpler compositional elementary graphical units such as differently sized rectangles by employing a self-attention approach. Another similar approach regarding the generated low-fidelity GUI layout artifacts utilizing a transformer encode-decoder model was proposed before [28]. More recently, research focused on generating low-fidelity GUI prototypes from natural language requirements with LLMs. For example, *MAxPrototyper* [58] requires a predefined layout and a short textual description of the GUI in order to generate a low-fidelity GUI prototype in the form of a novel domain-specific language (DSL) utilizing a combination of retrieved GUIs from *Rico* and a zero-shot prompting approach for the LLM. However, their approach is mostly focused on creating content (i.e., images and text) fitting the short text description. Furthermore, the *Instigator* approach [8] enables the creation of low-fidelity GUI prototypes from brief text descriptions by training a *minGPT* [29] model via utilizing a large-scale dataset of web sites (transformed to low-fidelity variants) automatically scraped from the web. Another related approach instead focuses on fine-tuning a pre-trained LLM using the large-scale *Rico* GUI dataset [17] based on a custom DSL for the generation of low-fidelity GUI prototypes and propose post-processing of the generated artifacts to improve their quality [20]. Moreover, instead of creating GUI prototypes in the form of a custom DSL, the *UIDiffuser* approach [55] directly generates GUI images utilizing *stable diffusion* [52]. While the resulting GUI images might be useful to rapidly obtain initial and coarse design inspirations, they lack clarity and usually functioning GUI components cannot be identified, only coarse layouts. Therefore, existing work focuses mainly on generating low-fidelity GUI prototypes, neglecting the more complex structure and characteristics of high-fidelity GUI prototypes and components included in our work. Additionally, related work usually produces proprietary output formats that require training or fine-tuning of the LLM and that are hard to reuse or integrate into traditional workflows (e.g., such as in *Figma*). In contrast, our approach generates a highly detailed *Material Design* specification that enables the instantiation of directly editable components within the visual *Figma* editor. This specification extends beyond selecting individual components (e.g., buttons, check-boxes) to include detailed parameters such as size, positioning, margins, elevation levels (e.g., flat, or raised), states (e.g., hover or pressed), labels, icons, or shapes, while we adhered to design system standards for typography, as well as primary and secondary colors. In addition, our proposed approach also is novel in terms of the specific technique used to generate the components. In particular, we employ a zero-shot LLM which requires no training or fine-tuning and simultaneously reduce the required amount of context tokens drastically through a RAG-based two-stage generation approach.

2.3 GUI Retrieval

Many approaches have been proposed for NL-based GUI retrieval. For example, *Guigle* [5] proposed the first GUI search approach based on extracted screenshots from automatically collected Android apps. Their approach facilitates the rapid retrieval of GUI screenshots from brief text descriptions, utilizing multiple fractions from the GUI hierarchy within *Lucene* [6], which employs classic *TF-IDF* and *BM25* [50] retrieval. Another approach proposed employing pooled BERT embeddings for text-only retrieval and additionally introduced a multi-model embedding space for text-based GUI retrieval [28]. More recently, *GUI2WiRe* [32, 33] and *RaWi* [34] introduced novel BERT-based Learning-to-Rank (LTR) models for enhancing NL-based GUI retrieval. While these retrieval approaches facilitate rapidly obtaining relevant GUI prototypes based on textual requirements, retrieval systems inherently are not able to create individualized GUI prototypes or components. Most often, these retrieval approaches additionally merely produce GUI prototype artifacts in non-editable formats such as images, impairing their reusability in the GUI prototyping process. Besides

NL-based GUI retrieval methods, research proposed several GUI retrieval approaches utilizing other input formats such as hand-drawn sketches [27, 41], wireframes [11], GUI screenshots [38], or entire apps [4]. In comparison to our approach, these methods lack the ability to create fine-grained implementations for user stories and retrieve GUIs solely from a fixed repository and additionally the retrieved GUI artifacts are non-editable, reducing the usability within the prototyping process. *Gallery DC* [13] harnesses a large-scale GUI design repository crawled from real-world applications and extracts GUI components in order to provide multi-faceted GUI component search capabilities. While this approach similarly supports to rapidly obtain relevant GUI components by inputting text, our LLM-based method enables the processing of entire user stories to generate relevant and editable GUI components.

3 System Design

To solve challenges from practice through an effective support system tailored at UX designers working within software development teams, we conducted formative interviews for which we interviewed ten professionals. Five participants (two female, three male, with an average age of 30.00 years, σ = 2.74) were professional UX designers, constantly utilizing dedicated prototyping tools (such as *Figma*). The UX designer group had an average experience in creating GUI prototypes of 5.70 years (σ = 1.64), 4.50 years (σ = 2.12) of evaluation of GUI prototypes, and high (four interviewees) to very high (one interviewee) self-reported knowledge in GUI prototyping. Three of our participants were professional product owners, two of whom were female and one male, with an average age of 31.33 years (σ = 2.89). They had 3.50 (σ = 1.73) years of experience in creating and 2.50 years (σ = 1.73) of experience evaluating GUI prototypes, showcasing their expertise in this area. Additionally, we interviewed two professional software developers (one female and one male, with an average age of 35 years, σ = 10.61). Both, on average, had 12 years (σ = 10.61) of experience in software and 10.50 years (σ = 8.49) working with GUI prototypes. We did not ask participants about their prior experience with LLMs or Generative AI beforehand, to avoid biasing their responses (ensuring they approached the problem space in the first half of the interviews without already thinking about AI). However, during later stages of the interviews most participants expressed experience with tools like *ChatGPT*, and a few had tried image generators. We chose professional UX designers, product owners, and software developers as our interviewees since these three roles usually collaborate extensively using GUI prototypes. Our objective was to ensure that a solution tailored to UX designers would not create challenges in collaboration with other roles and ideally offer advantages for all roles involved.

We conducted semi-structured individual interviews. After agreeing to participate in the study and stating demographic data, the participants were asked to describe their experience with GUI prototyping. The participants were asked multiple questions to explore the problem and solution space. For the second half of the interviews, UX designers and product owners were shown our initial system design as described in section 3.2. The participants first explored and described the initial design independently before receiving an introduction to the various functions. The interviewees then reflected on the functions, visualizations, information architecture, and general usability. Interviews with UX designers and product owners lasted 45 minutes on average. Interviews with software developers lasted 30 minutes and did not include a demonstration of the initial system design. Interviews with UX designers and product owners were transcribed and coded independently by a research assistant and a paper author. After coding, ambiguities were discussed, and disagreements were resolved.

3.1 Design Rationales

Subsequently, we present the fundamental design rationales derived from the interviews that motivated the design of our assistant. For each design rationale, the roles of interviewees mentioning these rationales are stated.

(1) *Integration of user stories into GUI prototyping tools from practice, such as Figma [21] (UX, PO, SD).* The assistant should be able to integrate new and changed user stories into GUI prototyping tools from practice (e.g., from *JIRA* [2]) and thereby minimize tool switches in work processes.
(2) *The user should retain control over the decisions of the assistant (UX, PO).* As long as the LLM-supported assistant can produce imperfect solutions, users should decide which suggestions to implement.
(3) *The user should be able to gain some understanding into the LLM-based*[1] *decisions (UX, PO).* Users must be able to understand decisions made by the LLM-based assistant, especially when imperfect solutions are created.
(4) *(Automated) mapping of user stories to components of GUIs (UX, PO, SD).* The assistant should facilitate linking user stories and GUI components to enhance transparency while implementing GUI prototypes in software code, and evaluate which components correspond to specific requirements.
(5) *(Automated) recommendation of GUI components (UX, PO).* The assistant should automatically suggest GUI components and draw them directly in the prototyping tool at correct positions.
(6) *(Automated) user story generation from GUI prototypes (PO).* In our interviews, product owners expressed the wish for automated generation of user stories from GUI prototypes for cases where less documentation was available.

The design rationales were derived from a qualitative analysis of the interview data. During the coding process, themes were identified based on recurring statements, such as participants highlighting challenges with existing tools or expressing a desire for specific functionalities. For example, the rationale of integrating user stories into prototyping tools was derived from interviewees describing missing connections between *Figma* and requirements management tools, e.g., "*I think it ultimately comes down to this: in the end, there are different tools that you use, and the more strongly they are interconnected, the better. I think Figma is great, but if Figma had a strong integration with JIRA, that would be even better [...] and these [requirements] should then ideally be linked to the corresponding prototyping pages, sections, or areas [..].*"

[1] During the first half of the sessions, some participants discussed "understanding the AI's decision" in general terms rather than understanding *LLM-based* decisions. After we explained our initial design and introduced the possibility of an LLM serving as the underlying technology at the end of the second half, they expressed a need to understand the LLM's decision, though AI was often used interchangeably.

3.2 Initial System Design

At the beginning of the interviews with UX designers and product owners, we explored the problem space in general, followed by an exploration of an existing prototype intended to motivate reflections on the features. The initial features and design were motivated by prior research and built upon Kolthoff et al. [35]. Figure 2 shows a schematic screen that we used for reflection with the participants. We adapted the LLM-based approach of Kolthoff et al. [35] for detecting and matching user stories in GUI prototypes to specifically support *Figma* GUI prototypes. Furthermore, we extend their work by a GUI component generation approach to rapidly create editable GUI designs. Additionally, we evaluated the extended approach and thereby provide an evaluation for the approach of Kolthoff et al. as well, since they only evaluate it using a technical train-test split.

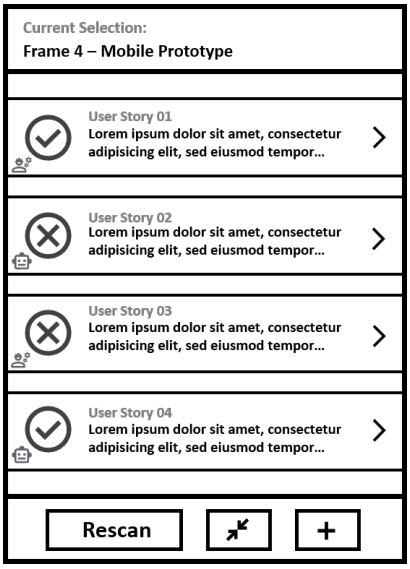

Figure 2: First schematic of prototype used in interviews as point of reflection.

3.3 Refined System Design

We developed two variants of our GUI prototyping assistant to evaluate its effectiveness in our lab study: a control and a treatment configuration. The treatment configuration builds on the baseline functionality of the control version by introducing advanced features enabling LLM-based automation. In the following, we describe the shared functionality of the control configuration, followed by the unique capabilities of the treatment configuration. Table 1 provides a comparative overview of these differences.

3.3.1 Control Configuration. This section outlines the baseline functionality of the GUI prototyping assistant, as implemented in the control configuration. These core features are also present in the treatment configuration.

Tool Integration. Our assistant directly integrates into the popular prototyping tool *Figma*. We based the size of the plug-in on popular plug-ins for *Figma* and integrated a minimization function (*C6* and *T6* in figure 3), allowing the plug-in to shrink and display only numerical values for open, ongoing, and completed user stories to optimize screen space. This feature is particularly useful during tasks such as graphical fine-tuning of design components, where workspace is a priority. With the assistants integration as a plug-in, we aim at supporting UX designers, product owners and software developers within tools of practice. By facilitating tasks in tools like *Figma* such as synthesizing textual descriptions into GUI components, and picking and contextualizing fitting GUI components, we aim at allowing UX designers to focus more on the creative and conceptual aspects of their work, such as sketching rough designs and crafting the overall user experience.

Table 1: Comparison of assistant's features in control and treatment configuration.

Feature	Control	Treatment
Tool Integration	Plug-in integrated in Figma	Plug-in integrated in Figma
User Story Listing	Lists all user stories	Lists all user stories
Change Intervention	Synchronizes user story updates	Synchronizes user story updates
User Story State Mang. & Detection	Manual	Manual and LLM-based
User Story Matching	Not available	LLM-based matching of GUI comp. to user stories
GUI Comp. Generation	Manual	LLM-based GUI component generation allowing user control over suggestions

Listing User Stories in Figma. The assistant lists user stories directly in the plug-in, with their status visually highlighted as completed *(green)*, ongoing *(blue)*, or still a to-do *(red)*. By default, all user stories for all app screens currently opened in *Figma* are displayed. If a single frame (i.e., the screen of an app) is selected (selection indicator: *C1* and *T1* in figure 3), only the user stories associated with that frame are shown in the plug-in. Additional information, such as the age and identifier of user stories, is displayed alongside status indicators. Users can update the status of a user story (completed, ongoing, or to-do) directly in the plug-in (*C3*, and *T13* in figure 3). Based on the interviews, we implemented filtering (based on *state*, and *AI Detected* in the treatment configuration) and sorting (based on *age* and *state* of the user story).

Change Intervention. One identified design rationale guiding our implementation is the assistant's ability to deal with new and constantly changing requirements. To fulfill this rationale, we developed a feature synchronizing user stories (e.g., with an external database). For the evaluation conducted in this paper, we developed an intervention, as shown in figure 4. As part of this intervention, users were notified of updates after a specified interval, when the "Sync User Stories" button became active, simulating a synchronization with user story databases (such as *JIRA*). Clicking the button led to the new or updated user stories being displayed in the plug-in and highlighted with dots.

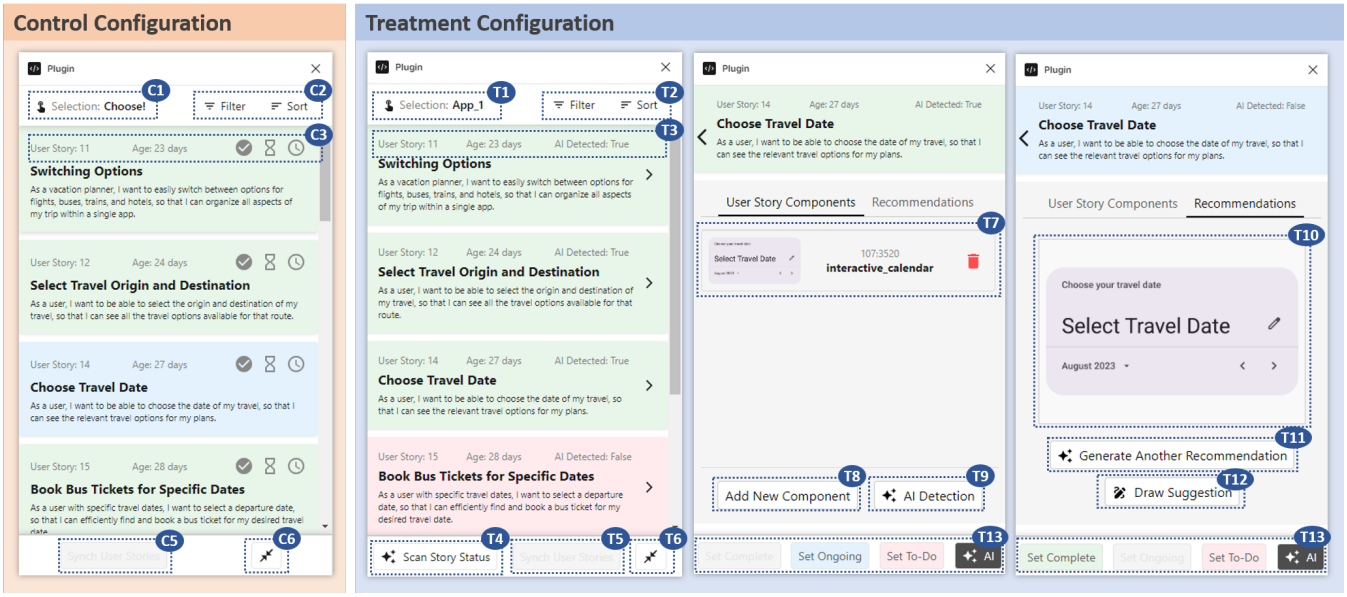

Figure 3: Assistant for GUI prototype designers in the control (left side) configuration and treatment (right side) configuration. In both configurations, the assistant shows user stories based on a selected frame in the prototyping tool *(C1/T1)*, allows to filter and sort user stories (*C2/T2*), can synchronize the user stories with a database (*C5/T5*), and be minimized (*C6/T6*). User stories can be marked completed, ongoing or as a to-do (*C3 and T13*). In the treatment configuration, the assistant can identify components completing a user story (*T7*), manually added (*T8*) or identified using an LLM (*T9*). Additionally, an LLM can detect the completion status of a user story (*T13*), can recommend GUI components completing a user story (*T10 and T11*) and draw them into the GUI prototype (*T12*). Figure contains *Material 3 Design Kit* [24] components from Google, used under CC BY 4.0.

3.3.2 Treatment Configuration. The treatment configuration extends the baseline features of the control configuration by integrating advanced functionalities guided by specific design rationales. This section outlines the unique features exclusive to the treatment configuration, as shown on the right side of figure 3.

User Story Detection. The assistant can automatically detect whether a user story has already been implemented in a GUI prototype. Leveraging our LLM-based detection approach, the method automatically analyzes the implementation status of the user story in a selected prototype. Users can apply this feature at two levels: individually for specific user stories (T13 in figure 3) or collectively for all stories in a frame (T4 in figure 3). Detected implementation states are visually indicated by color changes. After clicking *"Scan Story Status"* (*T4* in figure 3), discrepancies between user-assigned and AI-detected states are flagged for review in a dialog box, where users confirm or override AI suggestions. To ensure transparency, each user story in the list displays a label indicating whether the completion state was assigned by the user or detected by the AI (e.g., "AI Detected: True/False," as shown in T3 in figure 3). By automating detection, this feature aims at minimizing manual effort and ensuring a reliable overview of the implementation progress.

User Story Matching. For our assistant, we also implemented the function to automatically recognize which components of a GUI prototype in *Figma* fulfill a user story. This feature is shown in the third screen in figure 3. Matched components are displayed in a dedicated field (T7 in figure 3), showing their names as assigned in *Figma* alongside an isolated image of each component. Users can manage these associations by adding components manually ("Add New Component," T8), using the AI to detect components ("AI Detection," T9), or removing incorrectly associated components. The objective of this feature is to provide users a better understanding of the components influencing the decision of the LLM for automatically assessing the user story implementation status and to enable the direct interlinking between user stories and their counterparts in GUI prototypes. Thereby providing transparency regarding the features that still necessitate attention and enabling the verification of whether the appropriate GUI components have been utilized.

GUI (Component) Generation. Our assistant also allows users to generate components based on user stories. This feature is shown in figure 3 in the treatment screens (fourth screen from the left). To ensure that users retain control over the decisions of the assistant and decide for themselves whether LLM-based component proposals are to be incorporated, recommendations are first previewed in the assistant (*T10* in figure 3). Users can then decide whether they want to use the button *"Generate Another Recommendation"* (*T11* in figure 3) to generate another recommendation or use the button *"Draw Suggestion"* (*T12* in figure 3) to insert these components as already correctly placed GUI component in *Figma*. In this case, either correctly contextualized and parameterized material design assets, material icons, labels, or rectangles are drawn into the existing GUI prototype. Our assistant utilizes the existing design as a context for correct dimensioning and positioning.

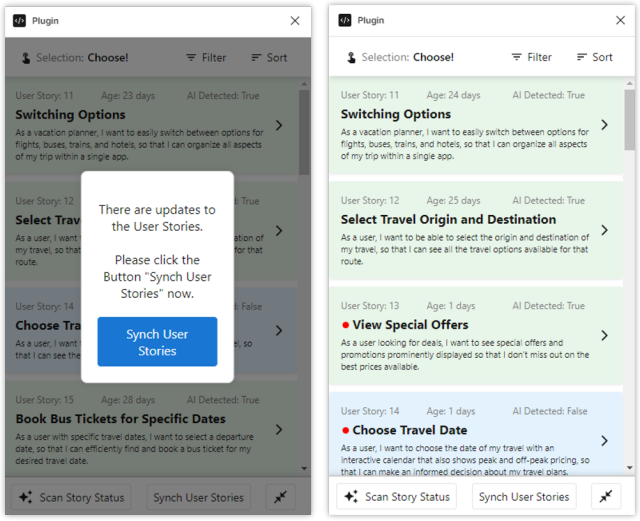

Figure 4: New and changed user stories intervention shown to participants after 30 minutes in both the treatment and control configuration in lab experiments (here: treatment config.). Intervention shown as pop-up message, and red dots next to updated user stories, mimicking requirement updates and synchronization with collaboration tools, such as *JIRA*.

3.4 LLM-Based Approaches

Our underlying LLM framework is composed of several components. An overview of the approach is illustrated in figure 5. In alignment with the figure, we describe the steps of the approach sequentially in a top-down manner. First, *(A)* we propose a component transforming the GUI prototype represented by *Material Design* components in *Figma* to an abstract and compressed string representation as the input to the LLM. Second, *(B)* the user story detection component consisting of a zero-shot prompted LLM for deciding whether a provided user story is implemented in the GUI prototype. Third, *(C)* the user story matching component enabling the coupling of GUI components corresponding to a given user story. In addition, *(D)* the user story generation component for automatically creating user stories for the GUI prototype. Fifth, *(E)* the two-stage GUI component generation approach enabling the rapid creation of implementations for a provided user story. Finally, *(F)* the component transforming the generated intermediate prototype representation back to a rendered GUI component and appropriate placement in the GUI prototype. All detailed prompts can be found in our supplementary materials. Subsequently, we present each of the introduced components of our LLM framework for supporting the GUI prototyping process in detail.

3.4.1 GUI Prototype Representation (A). Our approach focuses on GUI prototypes created within the popular prototyping tool *Figma* and supports prototypes implemented with the *Material Design* component library. This extensive component library encompasses over 85 distinct GUI components including elementary individual components such as *Checkbox*, *Slider*, and *Button*, to more intricate GUI elements assembled from multiple individual GUI components such as *Dialogues*, *List-Item*, *Search-Bar*, and *Card*. Each library GUI component possesses numerous configuration options including, for example, *Icon* and *Main-Text*, whereas the assembled components encompass several sub-components in a multi-level fashion. To further enhance the versatility of supported GUI components, we incorporated additional more generic component types (e.g., *Label*, *Image-Placeholder*, *Rectangle* and, *Icon*). To enable the previously discussed LLM-based assistance, the prototype requires to be transformed to an abstract and minified string representation to provide an efficient and effective input to the LLM. An intermediate JSON representation of the prototype is reduced to the abstract variant ensuring not only substantial increases in token efficiency, but also removal of details unnecessary for the proposed tasks to potentially increase effectiveness. The abstract string representation is constructed as a multi-level bullet point list, each GUI component being represented as an individual item using an abstract pattern[2], providing basic information of the component such as the *group*, *type*, *position*, and *size* as well as a list of component-specific attributes. Multiple levels are introduced in the bullet point list when assembled elements contain sub-components and form nested structures to ensure and retain the appropriate grouping of the components.

3.4.2 Implementation Detection (B). To address the challenge of automatically detecting whether a given user story is already implemented in the GUI prototype, we exploit the text understanding and reasoning capabilities of recent LLMs [10, 18, 49, 56] and employ an LLM-based method with *Zero-Shot (ZS)* prompting [31, 39] to formulate the problem as a binary classification task. ZS prompting enables instructing an LLM with a novel task without requiring resource-intensive training or fine-tuning and the creation of high-quality examples for *Few-Shot (FS)* prompting [10]. Our approach builds upon the ZS method proposed by Kolthoff et al. [35] showing the highest effectiveness with ZS prompting and we provide an adaption for the editable *Material Design* GUI prototypes in *Figma*. In particular, we create a ZS prompt template with *(i)* providing clear instructions on the provided information and the detection task, *(ii)* including the previously created abstract GUI prototype, and *(iii)* the user story to classify. The LLM is instructed to output a single token (either *0* or *1*) as its classification response and provide an additional short reasoning description.

3.4.3 GUI Component Matching (C). Similarly to the previously discussed implementation detection, we employ a ZS prompt adapted from Kolthoff et al. [35] to match a given user story to its corresponding GUI components within the GUI prototype. We follow the same ZS prompt template structure as described previously, but extend each GUI component in the abstract GUI representation with an identifier, enabling the LLM to reference the relevant GUI components. Therefore, we instruct the LLM to extract a list of identifiers of the GUI components relevant for implementing the user story, which enables the marking of the respective GUI components in the GUI prototype.

[2] *component-group (component-type) (position) (size) |attribute name:"attribute value"| (id)*, for example, *Button (SimpleButton) (x:25|y:790) (width:359|height:54) |Icon:"add"|Label Text:"Search"|Style:"Filled"|State:"Enabled"|Show Icon:"True" (id=27)*

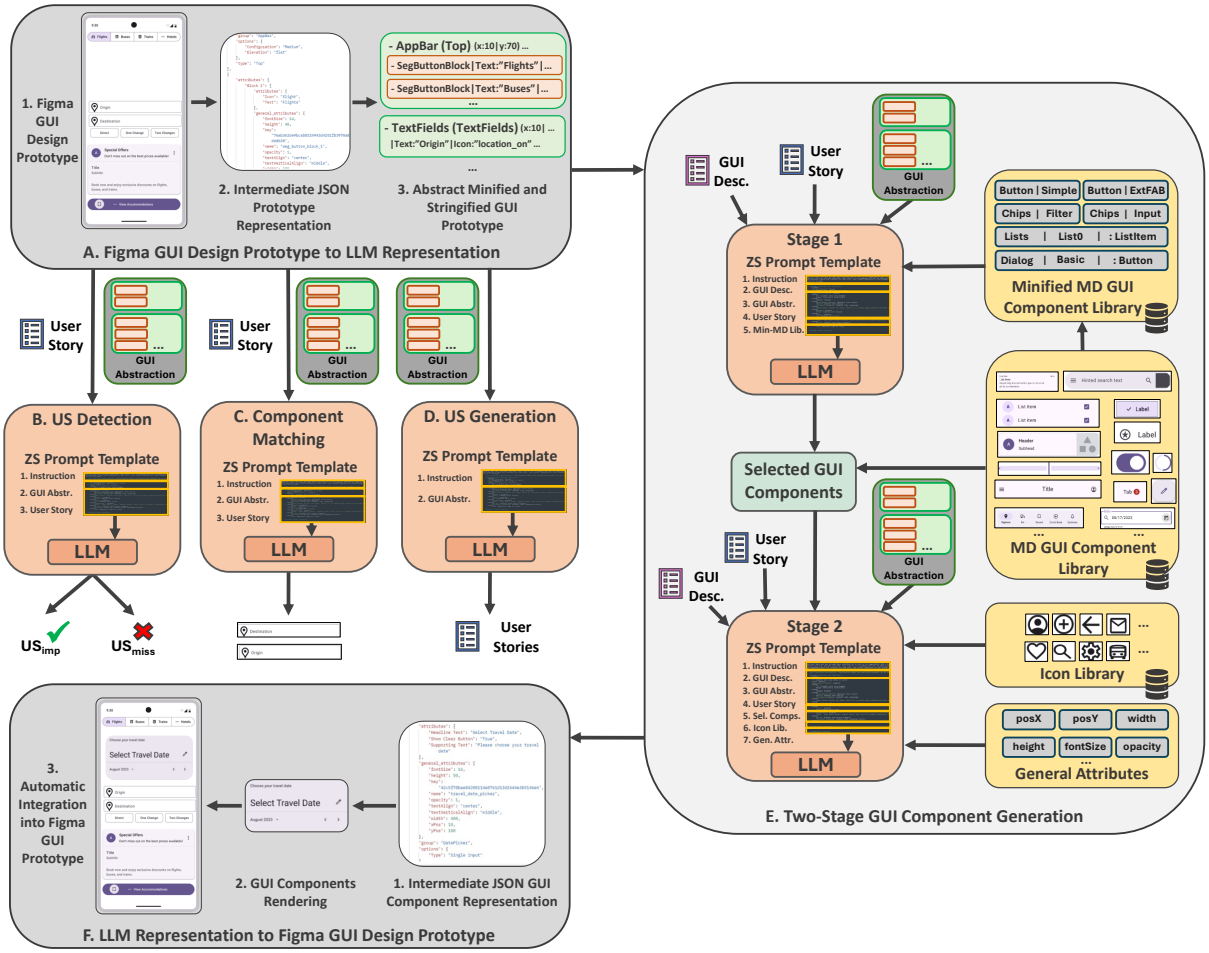

Figure 5: LLM framework for automatic detecting user story implementation, matching GUI components for user story, generating user story for GUI prototypes and two-stage GUI component generation for user story including the transformation of GUI prototypes in *Figma* to a compressed representation for inputting into the LLM and vice-versa rendering of generated GUI components within the prototype. Fig. contains *Material 3 Design Kit* [24] components from Google, used under CC BY 4.0.

3.4.4 User Story Generation (D). To enable the creation of user stories from a fraction or entire GUI prototype which currently is not covered by any user story within the collection, we also employ ZS prompting with an LLM. In this prompt template, we *(i)* clearly instructed the model to extract all present user stories with their respective GUI components and *(ii)* provided the abstract GUI representation with GUI component identifiers. In particular, the LLM is tasked with creating a JSON providing objects with the user story text and a list of corresponding GUI components.

3.4.5 GUI Component Generation (E). In addition to the detection, matching, and US generation methods, we propose to facilitate the prototyping process by enabling the contextualized GUI component generation for a given user story based on the current GUI prototype. To tackle the challenge of generating GUI component recommendations for a given user story, we employed another ZS prompting approach utilizing an LLM. While the LLM is pretrained on large amounts of text corpora from the web and potentially includes various information about *Material Design* (MD) utilized in our approach, the LLM lacks the specific *Material Design* component library and configuration options employed in our approach. Therefore, we manually constructed the comprehensive component library as input to the model. However, to increase token efficiency for the LLM, we propose a two-stage GUI component generation method. As depicted in Figure 5E, we first derive a minified MD component library containing only the information about available component types and sub-component references. Afterwards, we construct our first stage ZS prompt template by instructing the LLM to select all required components from the minified library and encompass *(i)* the user story to generate the implementation for, *(ii)* the abstract GUI representation, *(iii)* a brief textual description of the main functionality of the GUI, and *(iv)* the minified MD component library. Afterwards, we construct the second stage ZS prompt template instructing the LLM to generate the intermediate GUI component representation by utilizing the same information as before, but instead of the minified MD library, we additionally

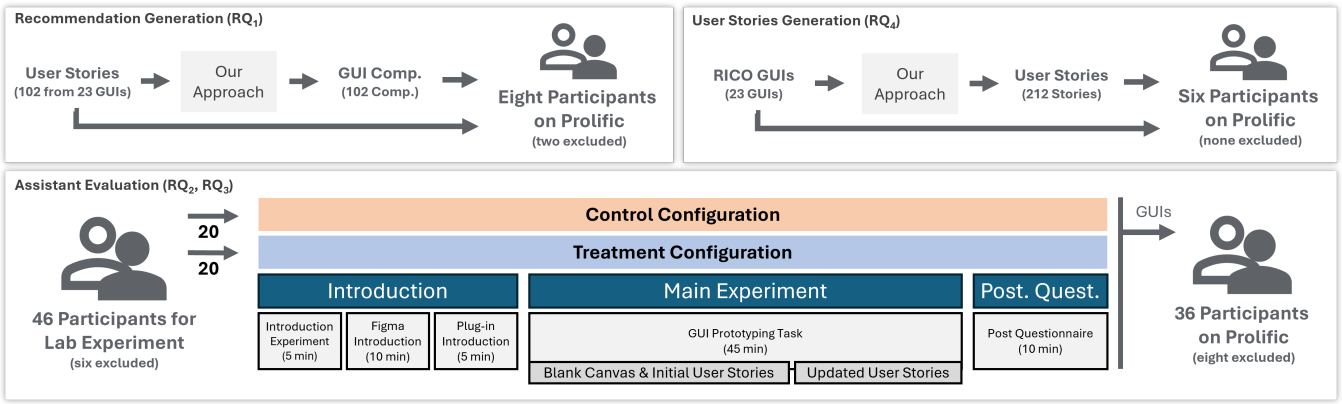

Figure 6: Overview of experiment procedures for evaluating the recommendation generation (RQ_1), the assistant (RQ_2 and RQ_3), and the user story generation (RQ_4). For RQ_1, GUI components were generated with our approach from user stories and evaluated by crowd-workers on *Prolific*. For RQ_2 and RQ_3, participants were recruited from a student panel, randomly assigned to either the control or treatment group to create GUI prototypes. GUI prototypes were then evaluated by a second set of participants sourced through *Prolific*. For RQ_4, user stories were generated with our approach from *Rico* GUIs and evaluated by crowd-workers on *Prolific*.

provide *(i)* the full specifications from the MD GUI component library for the selected components, *(ii)* an icon library, and *(iii)* specifications for general attributes that are shared among all GUI components.

By conducting this two-stage procedure, we avoid inputting the entire large MD component library for each generation and instead reduce it to a minimal representation. To ensure correctness of the generated component specifications of the first and second stage, we set the temperate of the LLM to zero for a more probable and deterministic output. Moreover, we implemented an automatic verification of the generated specification by matching them against the MD specifications. The component position and size is generated by the LLM via the general attributes and influenced by the components and their positions in the current prototype. Finally, the intermediate representation is rendered within our *Figma* plugin and the generated GUI component can directly be integrated into the existing GUI prototype design, including the automatic positioning of the component within the GUI prototype. Although the focus of our approach lies on generating functional GUI prototypes, the LLM is able to generate also different component styles (e.g., different buttons such as *IconButton* or *FloatingActionButton*), since different styles are represented as different components in the MD specification. In the future, we plan to integrate more fine-grained styling options (e.g., *background color*, *font color*, *font style*) by extending the general attributes. With this functionality, custom CI such as a color palette or special fonts could be incorporated into the generation process by enabling users to provide textual style requirements additionally to the textual functional requirements.

3.4.6 LLM Configuration. As the LLM in our approach, we utilize the most recent *GPT-4o* model [44] with 128k tokens context length (accessed in August, 2024), which extends the preceding *GPT-4* model [43] with multi-modal functionalities. We decided for *GPT-4o* since it provides state-of-the-art performance in many text evaluation tasks [44].

4 Evaluation Studies

In this chapter, we present the underlying methodology of our evaluation studies. In particular, we first focused on measuring the ability of our approach to create relevant GUI components matching the context. Subsequently, we focused on the question if our LLM-based assistant improves the effectiveness of GUI prototyping. In addition to the resulting quality of the GUI prototypes, we were also interested in the subjective satisfaction of the participants and the use of the assistant. Finally, we focused on the question of how effective our approach is in generating user stories from components of GUI prototypes. Overall, we articulated the following research questions for our evaluation:

- **RQ_1** How effective are LLM-based approaches for generating GUI prototype components completing a user story?
- **RQ_2** How does our LLM-based assistant influences GUI prototype quality and user story completion?
- **RQ_3** How does our LLM-based assistant influences perceived user experience?
- **RQ_4** How effective are LLM-based approaches for generating user stories from components of GUI prototypes?

In the following we will describe the underlying methods, procedures and evaluation datasets employed to provide answers to the research questions articulated above by a series of evaluation studies. With figure 6, we present an overview of our three evaluations, investigating RQ_1, RQ_2 together with RQ_3 and lastly RQ_4.

4.1 Recommendation Generation (RQ_1)

First, we present our evaluation study, which examines the extent to which the recommendations generated by our LLM-based approach are suitable for fulfilling the respective user stories.

4.1.1 Procedure. In order to measure the extent to which the recommendations created match the respective user stories, we employed the publicly available user story dataset from Kolthoff et

al. [35] at random. Recommendations were then generated using our approach with the assistant in the prototyping tool, rendered as images, and presented to crowd-workers on *Prolific* [19, 48] for evaluation. As there was no existing *Figma* context in this setup (no previous GUI prototype for which more recommendations are generated), a short, high-level text description as context created by students was utilized in addition to the user stories (e.g., *"a screen from a travel app showing flight search results"*). Created GUI components were presented to the crowd-workers as they were created by the LLM-based assistant, i.e., without post-editing.

4.1.2 Participants. We invited eight crowd-worker from *Prolific* who self-reported UI or UX experience, had more than 30 previous submissions and an high approval rate (>99%). No crowd-worker participated in any of our other studies. We had to exclude two participants based on failing our attention checks. The remaining participants (4 male, 2 female with an average age of 35.7 years) had, on average, 6.7 years (σ = 5.59) of experience in creating and 3.7 (σ = 3.40) years of experience in evaluating visual design (GUI prototyping). We did not collect information about participants' LLM or GenAI experience, as such expertise was not required for evaluating the GUI components and revealing that the GUI components were LLM-generated could have introduced bias.

4.1.3 Data Collection. The crowd-workers were shown the GUI components in a survey as pictures and the respective user stories. The crowd-workers were then asked whether the GUI components fully meet the functional requirements described in the user story, the GUI excerpt contains all the necessary components (e.g., buttons, input fields) to fulfill the user story, and whether textual descriptions within the GUI excerpt (e.g., labels, instructions, messages) are clear and appropriate for fulfilling the user story.

4.2 Assistant Evaluation (RQ_2, RQ_3)

To evaluate our assistant, we conducted a lab experiment in which participants were asked to create GUI prototypes based on predefined user stories in a controlled environment. Subsequently, we evaluated the resulting GUI prototypes with crowd-workers. We explicitly decided against a remote structure, e.g., with crowd-workers, to control boundary conditions (such as context and environment as well as processing time for the GUI prototyping tasks).

4.2.1 Procedure. We created two versions of the assistant and evaluated both in a between-subject design experiment. For this purpose, GUI prototypes were created by participants in a lab-based study for predefined user stories. User stories were derived from the publicly available user story dataset of Kolthoff et al. [35] to employ high-quality user stories that have already been evaluated in previous studies. While the user stories are focused on single GUIs and do not span across multiple GUIs of an app, the dataset contains multiple coherent user stories describing different functionalities of a GUI screen. The participants in the lab had help from one or the other version of the assistant (control or treatment). The GUI prototypes created were then evaluated by crowd-workers with UX experience sourced on *Prolific* [19, 48]. In the following, we present insights into the lab experiment and crowd-working related procedures.

Lab Experiment. In the 75-minute laboratory study, after agreeing to the data collection, participants read a briefing on the study task (5 min), watched a video explaining the GUI prototyping tool used (10 min), and a video demonstrating the assistant implemented as a plug-in for the GUI prototyping tool (5 min). Since our assistant in the treatment configuration generated *Material Design* components, we therefore decided to give both groups an introduction to *Figma* including the use of *Material Design* components. Then, participants started the GUI prototyping task (45 min) and finished the experiment with a post-hoc questionnaire (10 min). Participants worked with mobile screen templates and had pre-loaded icons and *Material Design* assets available. For the GUI prototyping task, the laboratory study participants were tasked with creating up to three GUI prototypes based on user stories. For each GUI prototype, there were eight user stories to complete. After 30 minutes, there were updates for two existing user stories, and two new user stories were added to the assistant. The update was intended to simulate changed requirements, as is usual in practice. Participants were instructed to start with the first GUI prototype and the first eight user stories for this GUI prototype and continue with the second and third GUI prototypes only when the previous prototype was completed. For our control group, which used the assistant without generative component creation, producing three prototypes as part of the study task is clearly extensive. We deliberately opted for an extensive task so that no participant would finish early, even in the treatment with generative component creation, and would continue to prototype in the 45-minute GUI prototyping phase. We measured the load in pretests to find the right amount of GUI prototyping tasks. The study included attention and comprehension checks. The study design was carefully evaluated with the university's Institutional Review Board (IRB).

Evaluation of GUI prototypes. We conducted an evaluation study on *Prolific* with crowd-workers to evaluate the generated GUI prototypes. We chose *Prolific* since comparative studies indicated that crowd-workers on *Prolific* produce higher data quality compared to other crowd-working platforms [19]. Participants from *Prolific* received a questionnaire to evaluate the GUI prototypes created in the lab sessions. After consenting to data processing, participants received a study description and evaluated 20 GUI prototypes each, randomly drawn. On average, each participant received 13 GUI prototypes for the first GUI prototyping task (five for the second and two for the last GUI prototyping task) created with the assistant in the control or treatment configuration, respectively. The study lasted 65 minutes on average and included six attention checks. Furthermore, the study design was carefully evaluated with the university's IRB.

4.2.2 Participants. Following, we describe the participants of the lab study and the evaluation of the GUI prototypes.

Lab Experiment. We recruited 46 participants from a university panel for the lab study randomly assigned to the control group (23 participants) or treatment group (23 participants). In both groups, three participants had to be eliminated (one technical problem, three times failure to complete attention checks, two failures to meet the 45-minute GUI prototyping task time) ex-post. Eventually, 20 participants were admitted to the study in both groups. The participants in the control group (14 male, 6 female) were, on average,

25.40 years (σ = 4.65) old. The participants in the treatment group (12 male, 8 female) were, on average, 23.85 years (σ = 2.67) old. Participants in both groups studied, on average, little over four years and had the same average experience with creating and evaluating visual design (such as GUI prototypes).

Evaluation of GUI prototypes. On *Prolific*, we originally invited 36 participants who self-reported UI or UX experience, had more than 30 previous submissions on *Prolific* and an high approval rate (>99%). Submission from eight participants were excluded for failing one or multiple attention checks. Only data from the remaining 28 (22 male, 6 female) participants were considered. *Prolific* participants were 28.25 years old (σ = 6.78) and had 4.25 years (σ = 5.87) of experience creating visual design and 3.86 (σ = 4.04) years experience evaluating visual design on average. We did not collect information about participants' LLM experience, as such expertise was not required for evaluating the GUI prototypes and revealing that the GUI prototypes creation was partially assisted by our LLM-based assistant could have introduced bias.

4.2.3 Data Collection. We collected a range of data, both in the lab setting and on *Prolific*.

Lab Experiment. During the lab experiment, we logged the usage data of the assistant (navigation in the assistant, clicks, and web-calls in the treatment group) and the GUI prototypes generated during the sessions. In addition, the participants filled out a post-hoc questionnaire in which Task Load (NASA TLX Raw [26]), System Usability (SUS [9]), and Creativity Support (CSI [12]) were recorded. In addition, we asked for further Likert items (e.g., ease of use and effectiveness of the assistant). We also surveyed subjective use of the features (detection of user story completion, recognition of the GUI components of a fulfilled user story, and recommendation of GUI components) in the treatment, and in both groups positive and negative aspects of the assistant via open text fields.

Evaluation of resulting GUI prototypes. For each GUI and user story shown, the crowd-workers assessed the extent to which the GUI fully meets the functional requirements described in the user story, whether the GUI contains all the necessary components (e.g., buttons, input fields) to fulfill the user story and whether text within the GUI (e.g., labels, instructions, messages) is clear and appropriate for fulfilling the user story. Additionally, they rated each GUI for overall consistency with user stories, visually appealing design, clear information organization, intuitive interaction, whether the GUI looks like an app page, minimal prototype errors, and satisfaction with GUI prototype.

4.3 User Stories Generation (RQ_4)

In addition, we evaluated the extent to which our approach can create user stories from existing GUI prototypes.

4.3.1 Procedure. To evaluate the ability to create suitable user stories, research assistants recreated *Rico* GUIs [16] for the prototyping tool. 23 GUIs were randomly chosen based on the GUIs employed in the user story dataset of Kolthoff et al. [35]. We then utilized these GUI prototypes to create LLM-based user stories for the GUI components using our proposed assistant. Created user stories were presented to the crowd-workers as they were created, i.e., without editing. These were then evaluated by crowd-workers with UI or UX experience. Each crowd-worker rated ten GUI prototypes and its related user stories. On average, 9.2 user stories were created by the LLM for each GUI.

4.3.2 Participants. We recruited six crowd-workers through *Prolific* [19, 48], applying the same selection criteria as in the lab study. None of the crowd-workers had participated in any of our previous studies. No participant failed any of our attention checks. The final sample (6 male, average age of 33.83 years) had an average of 6.00 (σ = 4.12) years of experience in creating and 4.33 (σ = 2.98) years of experience in evaluating visual design (GUI prototyping). Once more, we did not inquire about participants' LLM experience, since such was not required for assessing the user stories.

4.3.3 Data Collection. The user stories created and GUI prototypes were presented to crowd-workers for comparison. Subsequently, they assessed the user stories in combination with the GUI prototype for the following four items: *(i) "The user story accurately describes functionality found in the presented GUI."*, *(ii) "The user story is written with sufficient clarity and precision."*, *(iii) "The user story is specific enough to describe a particular feature of the presented GUI."*, *(iv) "The level of detail in this user story meets the standards in a professional project."*

5 Results

In this section, we provide the results on our LLM-based approach's capability to create GUI components based on user stories through measuring how well the generated components fulfill the user stories (RQ_1). Afterwards, we present the results of the lab experiment using the assistant, including the evaluation of the lab results on *Prolific* (RQ_2, RQ_3). In addition, we provide the results of the evaluation for automatically creating user stories (RQ_4).

5.1 Recommendation Generation (RQ_1)

To maximize requirement diversity, we took 102 user stories from 27 GUIs in the publicly available dataset by Kolthoff et al. [35] and generated components for each user story using our LLM-based approach. These were presented individually (i.e., as single components with the user stories) to crowd-workers on a basic schematic mobile app outline. We collected 40 pairs of user story component ratings from each of the six crowd-workers, resulting in an average of 2.35 ratings per user story and GUI component.

Crowd-workers rated user stories and derived GUI components assessing whether the components *(1)* allow the user to perform the task in the user story, *(2)* all necessary components for fulfilling the user story are presented, *(3)* appropriate components were chosen and *(4)* text of components were clear and appropriate. All ratings were obtained on nine-point Likert scales (1: *Strongly Disagree*, 3: *Disagree*, 5: *Neutral*, 7: *Agree*, 9: *Strongly Agree*). Figure 7 shows violin plots for each of the four assessed aspects. All four aspects were rated high, with mean (median) values of 7.5 (8) for functional requirement fulfillment, 7.51 (8) for user story fulfillment, 7.04 (8) for component correctness and 6.86 (7) for textual clarity. In addition, to evaluate the efficiency benefits, we compared the proposed two-stage GUI component generation approach against a similar one-stage GUI component generation

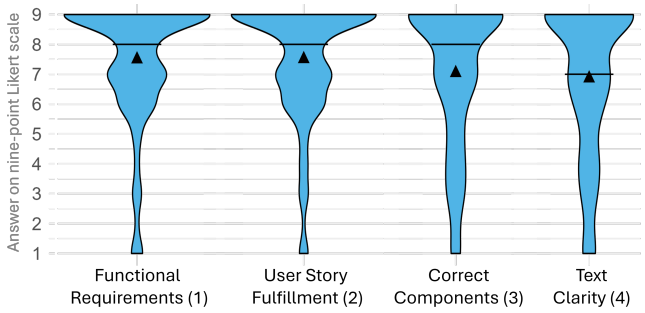

Figure 7: Violin plots of crowd-worker ratings (y-axis: 9-point Likert scale) for components generated by our LLM-based approach in the prototyping tool based on over 100 user stories. Crowd-workers rated functional requirements fulfillment, user story fulfillment, component correctness and text clarity for user stories and derived GUI components. Triangles represent means, black lines medians.

ZS prompt, which always incorporates the full *MD* component library. On the 102 user stories dataset, the two-stage GUI generation approach accomplished a considerable 64.35% reduction of the consumed input tokens (Mean #One-Stage-Tokens=8559.49|Mean #Two-Stage-Tokens=3050.78) and a significant 60.82% reduction of the total consumed or generated tokens (Mean #One-Stage-Tokens=9359.44|Mean #Two-Stage-Tokens=3666.54) highlighting the efficiency improvements[3].

5.2 Assistant Evaluation (RQ_2, RQ_3)

Our analysis of the data collected in the lab experiment is twofold. Firstly, we examine the completion of user stories and the quality of the GUI prototypes created by study participants, as assessed by crowd-workers on *Prolific*. Afterwards, we report the self-assessments of the lab participants, including both qualitative and quantitative aspects. In addition, we evaluated the assistant's usage, highlighting which features and functions were used by the participants. Figure 8 shows three GUI prototypes created by the control and treatment groups for the first, second and third tasks.

5.2.1 User Story Completion and Prototype Quality. Our results for user story completion and overall GUI prototype quality are based on data from crowd-workers on *Prolific*. The results for evaluating the completion of the user stories are shown in figure 9. The participants in the lab study were tasked with creating three app prototypes based on ten user stories each. Participants were instructed to start with the first app and move on to the next one once the user stories had been completed. As a result, a different number of GUI prototypes were created for each of the apps in the control group and the treatment group. In the control group, all 20 participants created a first app, five a second, and two a third. In the treatment group, 19 participants created the first app, 11 the second one, and five participants the third app. On average, 1.35 apps were created in the control group and 1.75 apps in the treatment group.

[3]At the time of writing, this reduction in input and output tokens leads to a reduction in cost from $0.052 to $0.024 per generated GUI component using the *GPT-4o* [44] flagship model from *OpenAI*.

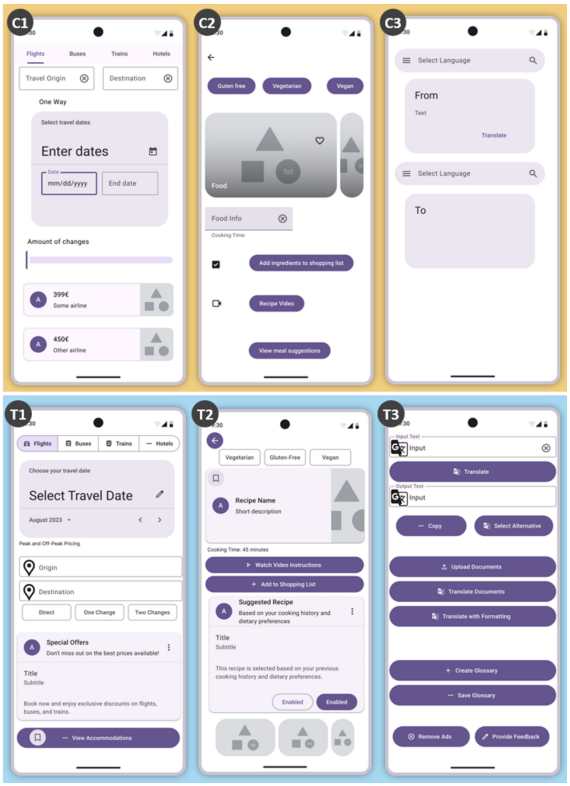

Figure 8: Examples of GUI prototypes created in the lab sessions with the control (orange, top, C1 - C3) and treatment (blue, bottom, T1 - T3) configuration of the assistant for all three tasks (*(1)* travel booking app, *(2)* recipe app and *(3)* translation app). Figure contains *Material 3 Design Kit* [24] components from Google, used under CC BY 4.0.

To analyze the degree of fulfillment of the user stories, we solely investigated apps that participants had worked on. Apps that were not started were excluded from the evaluation in *Prolific* (i.e., we did not count user stories for apps that were not started as unfulfilled). An app was classified as started when changes were made to the schematic app background in *Figma* and GUI components added (simply moving an schematic app background in *Figma* would not count as starting the app).

For the first app, for six (three significant)[4] user stories, the treatment group produced GUI prototypes with a higher rating for user story completion in contrast to four (two significant) user stories in the control configuration that led to higher user story completion. For the second app, seven (three significant) user stories using the treatment, and three (one significant) user stories with the control configuration were measured higher. For the third app, all

[4]Participants in the treatment group started with just the schematic app background without any components, but could quickly use the recommendation feature to generate components. To give participants in the control group an idea of how to pick and adjust *Material Design* components, the introduction video demoed the use of an app bar demonstrating the creation of the first user story. This component was also already drawn to the schematic app background in *Figma* when participants started. Since this component completes the first user story, technically participants in the control group did not complete the first user story by themselves.

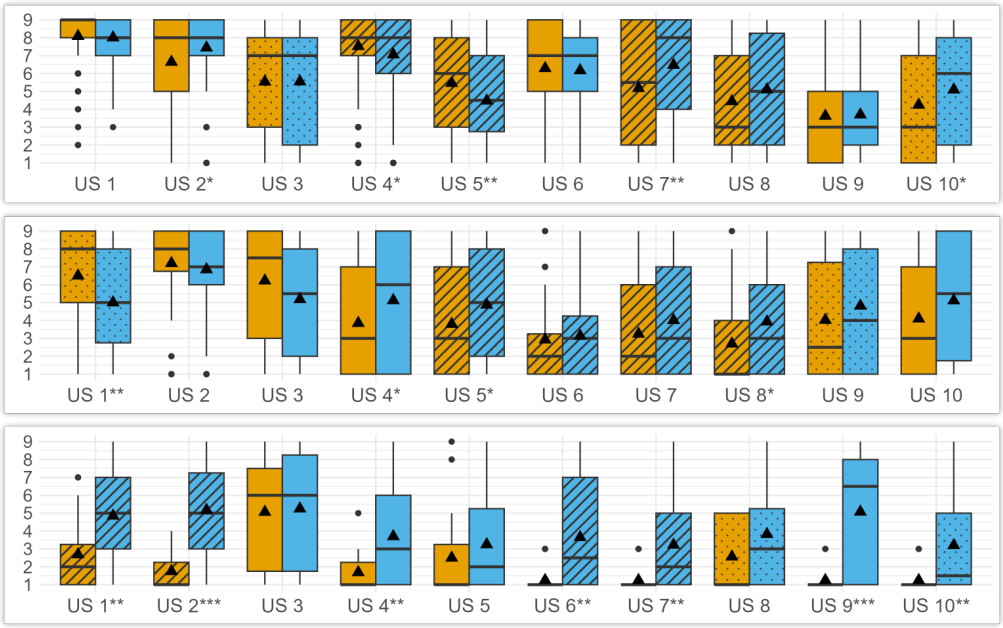

Figure 9: Boxplots showing user story completion as rated by crowd-workers for the three apps (top: first app, middle: second app, bottom: third app) to be created by participants in the lab experiment using the control (orange) or treatment configuration (blue). For each app ten users stories were to be created (US 1 to US 10). Dotted (new user story) and dashed (extended user story) bars represent user stories with an update intervention after 30 minutes. Triangles represent means. One, two and three stars represent significant results (<0.05, <0.01, <0.001) for two-sided Wilcoxon rank-sum tests.

Table 2: Means of crowd-workers' ratings (Likert scale 1-9) of GUI prototypes for control and treatment configuration, and results of Mann-Whitney-U-Tests. Crowd-workers were asked if the GUI design is consistent with the overall user stories, whether the visual GUI design is appealing, organization of information is clear, the GUI allows for intuitive interaction, the GUI prototype looks like a screen from a complete app, the GUI prototype only has minimal errors, and whether crowd-workers were satisfied with the GUI.

	Consistent Design	Appealing Design	Information Organization	Intuitive Interaction	Screen from Complete App	Minimal Errors	Overall Satisfied
Control	4.583	4.454	4.820	4.528	3.465	4.190	3.819
Treatment	5.046	4.586	5.025	4.761	3.614	4.225	4.254
p-value	**0.0151***	0.5105	0.2600	0.1849	0.5540	0.7485	**0.0423***

ten (seven significant) user stories with the treatment configuration were rated as more complete in contrast to the control group.

For both other aspects that were evaluated by crowd-workers for each user story, the following picture emerged: For the selection of the right components to fulfill the respective user story, the control group was rated higher on average for nine user stories, and the GUIs created with the treatment for 21 cases. Regarding the selection of correct descriptions (e.g., texts), the results of the control group were rated higher for 8 user stories, and the results of the treatment group for 22 user stories.

Not all user stories were available to participants from the start. In our intervention, the user stories were updated after 30 minutes of processing. Two user stories were made available for the first time as new user stories (dotted boxplots in figure 9). For two user stories, an update of the previous user stories was displayed that expanded the scope of the previous user story (dashed in figure 9). In figure 9, stories 4/5 and stories 7/8 represent pairs in the first app, in which the second user story represents the update. For app two (app three), these are 5/6 (1/2) and 7/8 (6/7). As expected, the updated user stories were, on average, rated as less completed in contrast to their initial versions, since the update of user stories merely extended the first version of the user story. In our discussion in section 6.2, we provide insight into examples and discuss cases where the control results were assessed as more completed.

In addition to completing the user stories, crowd-workers also independently evaluated whether *(1) the GUI design is consistent with all user stories for this GUI prototype, (2) whether the visual GUI design is appealing, (3) organization of information is clear, (4) the GUI prototype allows for intuitive interaction, (5) the GUI prototype looks like a screen from a complete app, (6) the GUI prototype has only*

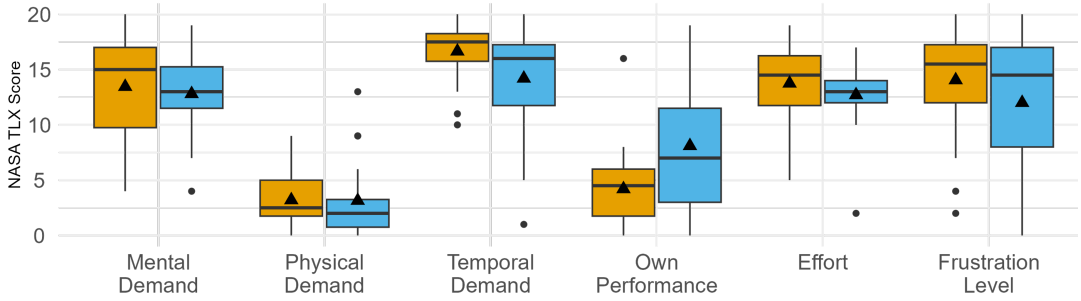

Figure 10: Results of NASA TLX (raw [26]) for participants in control (orange) and treatment group (blue). *Own Performance* significant at 5% level. Triangles represent means.

Table 3: Rating means and p-values (Mann-Whitney-U-Tests, one-sided) for seven questions from lab study participants using Likert-scales for: *satisfaction with final designs, ease of use and navigation of plug-in, confidence in creating professional GUI designs, efficiency in design process, future use consideration, impact on design quality,* and *difficulty in creating GUI prototypes.*

	Satisfaction Designs	Ease of Use	Confidence Creating Professional GUIs	Efficiency in Design Proc.	Future Use	Impact Design Quality	Difficulty Creating GUI Prototypes
Likert Scale	*Strongly Disagree* (1) - *Strongly Agree* (9)				*Def. Not* (1) *Def Yes* (5)	*Not at All* (1) *Extremely* (5)	*Very Easy* (1) *Very Difficult* (5)
Control	3.00	6.20	4.35	5.35	3.25	2.90	3.75
Treatment	3.75	6.30	5.65	6.85	3.80	3.50	3.45
p-values	0.2543	0.4400	**0.0277***	0.0573	**0.0392***	**0.0462***	0.8408

minimal errors, and *(7) whether crowd-workers were satisfied with the GUI prototype* for each of the created GUIs. Results are shown in table 2. For each aspect, means in the treatment configuration were considerably higher, but only *consistent GUI design* and *overall satisfaction* are statistically significant.

5.2.2 Participants Perceptions. We asked our participants qualitatively and quantitatively how they rated using the plug-in in the control and treatment groups. Table 3 shows the results for seven Likert scale items. The control group achieved a lower mean value for each aspect, except for *difficulty creating GUI prototypes*, where a higher value is associated with greater perceived difficulties. Particularly noteworthy are the significantly better results of the treatment group for the questions: *confidence creating professional GUI prototypes*, *consideration for future use*, and *the impact on design quality*.

We also measured the task load via the NASA TLX (raw) questionnaire. Figure 10 shows the values for the task load. On average, the treatment configuration was rated lower for *Mental Demand*, *Physical Demand*, *Temporal Demand*, *Necessary Effort*, and *Frustration Level*. The value for *Own Performance* proved to be significantly higher for the treatment configuration. The measured System Usability Scale (SUS) [9] and Creativity Support Index (CSI) [12] showed no significant difference, however, for both scales the treatment configuration rated higher than the control configuration. For SUS means were 52.13 for the control and 55.50 for the treatment configuration, reflecting a steep learning curve of participants using *Figma* and GUI component libraries in *Figma*. For CSI, means were 47.63 and 56.13, respectively.

5.3 User Stories Generation (RQ_4)

To investigate the effectiveness of our LLM-based approach to create user stories from GUIs, we created 212 user stories from 23 *Rico* [16] GUI prototypes directly in *Figma* with our assistant (an average of 9.22 per GUI prototype). Both were rated by our experts on Likert scales (1: *Strongly Disagree*, 3: *Disagree*, 5: *Neutral*, 7: *Agree*, 9: *Strongly Agree*). Participants rated the user story while being shown the GUI prototype for *(1)* the user story being found in the GUI prototype, as well as *(2)* clarity and precision, *(3)* specificity, and *(4)* professionalism. Figure 11 shows violin plots for each assessed aspect for all user stories. Means (medians) for each the four questions were 6.70 (7), 6.63 (7), 6.53 (7), and 6.02 (6). However, while the total average and median values were high (medians of *"Agree"* for the first three questions in the Likert scales), there was considerable variance of the ratings for the user stories derived from individual GUIs, we therefore included user story ratings for four particularly selected GUIs in figure 11.

6 Discussion

Using our prototyping assistant in the treatment configuration, participants created GUI components rated by crowd-workers as, on average, completing the user stories to a higher degree. Nevertheless, some user stories were rated as better completed in the control configuration. Subsequently, we discuss some of the results and put them into context. In addition, we discuss selected recommendations for the design of automated LLM-based support systems for GUI prototyping and how future researchers and practitioners might leverage the results and findings from this work.

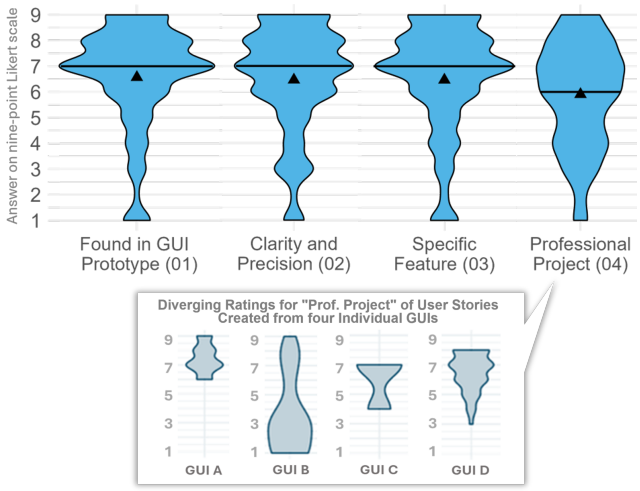

Figure 11: Top: Crowd-worker ratings of generated user stories from GUI prototypes accurately describing functions of the GUI, user stories being precise, for a identifiable features, and formulated as in projects from practice. Y-Axis presents Likert scale items (1-9). Triangles represent means, lines medians. Bottom: Crowd-worker ratings of generated user stories from GUI prototypes for the fourth question, for four chosen GUI prototypes (A-D) and user story parings.

6.1 Effectiveness of Generating GUI Prototype Components (RQ_1)

In our first study, we evaluated the ability of our LLM-based approach to generate GUI prototype components, which received high ratings from crowd-workers on functional requirements fulfillment, user story fulfillment, and component correctness (three times median of eight, one median of seven on a nine-point Likert scale). These results indicate that our approach successfully produced components that effectively fulfill functional requirements and align with user stories. This supports the thesis that LLM-based methods can assist early-stage prototyping by generating functional and relevant elements with minimal manual intervention. However, slightly lower ratings for textual clarity suggest room for improvement in how text is generated for GUI components. We propose two potential avenues for refinement: *(i)* technical improvements, such as domain-specific few-shot prompting or fine-tuning, and *(ii)* procedural improvements, such as integrating human feedback loops into the creation process. The presented efficiency gains from the two-stage approach hint at feasibility in large-scale design workflows where cost and computation are bottlenecks.

6.2 Impact of the LLM-Based Assistant on Prototype Quality and User Experience (RQ_2, RQ_3)

Our investigation into how our LLM-based assistant affects the quality of GUI prototypes, user story completion, and perceived user experience showed that experts rated GUI prototypes created with the LLM-based assistant as having a higher quality, especially in selecting the right components and utilizing correct descriptions. User stories were also rated as completed to a higher degree for prototypes created with the LLM-based assistant (23 cases for the treatment versus seven cases for the control group). These results suggest that the assistant effectively supports task completion and improves the overall GUI prototype quality. As expected, participants who used the LLM-based assistant started or completed more app screens as part of the study tasks. The automatic generation of GUI components likely contributed to this efficiency benefit and enabled faster task progression. Participants also reported a higher perceived user experience when using the LLM-based assistant, such as significantly higher ratings for future use, perceived impact on design quality, and confidence in creating professional GUIs. While NASA TLX scores for mental and temporal demands, effort, and frustration were lower for the treatment group, these differences were not significant. However, both groups experienced high mental and temporal demands potentially due to the extent of the tasks (e.g., creating three GUIs, each with ten user stories). This intentional aspect of the study design was fine-tuned in pre-tests to ensure that the treatment group supported by LLM-based GUI component generation would not run out of tasks.

When analyzing the cases in which crowd-workers rated GUI prototypes created in the control group as completing the user story to a higher degree in comparison to the treatment group, the presence of a text label often played a crucial role. Figure 12 illustrates such an example where a button with a text label (*"save for later"*) was created for the user story by a participant in the control group (left). At the same time, an icon button with a *bookmark* icon was used in the treatment group (right). Almost all GUI prototypes in the treatment group featured a *bookmark* icon for this user story, and our tests have shown that our LLM-based approach generates *bookmark* icon buttons almost consistently for this user story.

This motivates two discussion points. Firstly, to what extent were crowd-workers influenced by explicit textual mentions in the components, especially when comparing them with aesthetically styled components such as the illustrated icon button? Secondly, how the temperature setting in our LLM-based approach led to the generation of similar recommendations when the solution space was somewhat limited, particularly when participants clicked *"Generate Another Recommendation"*. We recommend balancing the generation of more diverse additional recommendations for further studies. Potential solutions include: evaluating adjusting the temperature setting in iterative component generation, enabling the approach to generate multiple implementations of the same user story while instructing it to produce distinct variants, or allowing participants to provide further LLM instructions in addition to the user story. Future users could, for example, restrict the next version of the same component generation by specifying instructions such as *"include a label in the next generation"*. While participants expressed great appreciation for the automated component generation, e.g., P12: *"Straightforward to use and it gave good advice most of the time, excellent starting point for most stories"*, P19: *"Really good recommendations."*, P10: *"So much work was reduced"*, there were improvements mentioned in regard to the discussed variations of recommendations. Participants, such as P7: *"[I] would have liked to have had an option to adjust the task that is sent to the AI"*, frequently requested more variation in the recommendations. In fact, this suggestions was among the most mentioned improvements by participants - second only to reducing component generation time.

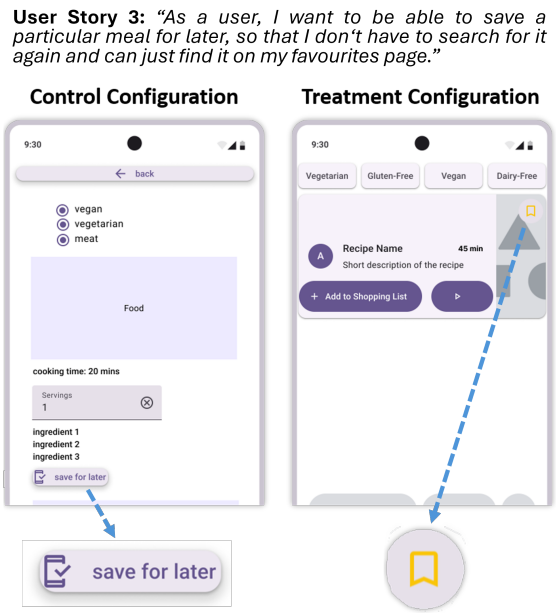

Figure 12: Example of an instance where a user story (second app, the third user story) was rated more complete in the control group. The control group tended to use buttons with a clear call to action (e.g. *"save for later"*), whereas in the treatment group our approach mostly created *bookmark* icon buttons representing a concise but more implicit implementation of the user story. Figure contains *Material 3 Design Kit* [24] components from Google, used under CC BY 4.0.

6.3 Effectiveness of Generating User Stories from GUI Components (RQ_4)

In the third study, we investigated the effectiveness of our approach for automatically deriving user stories from GUI prototypes. Crowdworkers rated the user stories created by the approach high on relevance, clarity, precision, and specificity (all with a median of seven on a nine-point Likert scale), indicating that the generated stories align well with the functionality and purpose of the GUIs. However, ratings for professionalism (six-point median) were slightly lower, raising questions about, e.g., the tone and depth of the generated user stories. While overall median ratings across user stories and GUIs were high, it is worth looking at the ratings of user stories derived from individual GUIs. In the results in figure 11, we have intentionally shown ratings of user stories that emerged from four individual GUI prototypes. Here, the presented GUIs resulted in variations in the ratings, showing that the quality of the user stories created varies depending on the underlying GUI. This variability may stem from differences in GUI complexity, ambiguity, or the LLMs' ability to generate user stories for specific tasks, which were more likely to be included in its training data. Despite these limitations, the findings underscore the potential of LLM-based tools to support early-stage documentation and design workflows. Integrating such tools into iterative design processes could reduce manual effort, enabling teams to focus on higher-level design challenges.

6.4 Generating GUI Prototypes from User Stories or Vice-Versa?

Our results show that our LLM-based approach enables the effective creation of GUI prototypes (components), mainly desired by UX designers and partly by product owners, and effectively derives user stories from GUI prototypes, which product owners mainly desired in our interviews. Readers may wonder that it is certainly not possible to do both simultaneously and might sense a certain chicken-and-egg problem. Based on our interviews, we identified two options for integrating both features. We see our assistant (in the treatment configuration) as a mediator for both roles through the two described features. While product owners define user stories, the GUI prototyping assistant could generate GUI components in parallel. Vice versa, the UX designer could create design drafts from which the assistant derives user stories for the product manager. In addition, GUI prototypes are often created in an iterative process, so it is conceivable that product owners formulate an initial user story, and the assistant creates a design proposal, which UX designers subsequently refine. The created refinement is then used to generate more precise user stories. Therefore, we regard the features of automatic GUI (component) creation and user story creation as one of the leading collaboration features.

6.5 LLM-Based GUI Prototyping for Future Research and Practice

Our study provides insights that can inform both researchers and practitioners. Next, we discuss three aspects in which our findings can contribute to future research and design practices.

First, our qualitative interviews revealed important design rationales for AI-based GUI prototyping. While prior research has explored AI integration in UX workflows [30], our findings highlight additional requirements emerging from cross-functional collaboration between UX designers, product owners, and software developers. Notably, we did not expect the sixth design rationale — the need for product owners to automatically derive user stories from GUIs. This underscores the necessity for AI-based prototyping tools to accommodate diverse stakeholder needs when multiple roles interact in the prototyping process (see section 6.4).

Second, our study provides insights into the use of AI-based prototyping tools under controlled laboratory conditions. Findings indicate that participants effectively understood the iterative approach of generating individual GUI components from granular user stories (in contrast to, e.g., creating a full GUI from a single text description). This structured method allowed them to incrementally build more coherent and higher-quality GUI prototypes compared to a group without AI-based prototyping support. Additionally, our study revealed unexpected requirements, such as the need for future generative prototyping tools to offer greater variation in generated GUI components (see section 6.1).

Lastly, a recurring challenge across disciplines is to structure abstract problems as textual representations that enable LLMs to process them effectively. In our case, this involved deriving user stories for GUIs in prototyping tools like *Figma* at the component level or, conversely, recognizing fulfilled user stories and generating corresponding components. Our work contributes to future

research by providing a structured approach to bridging user stories and GUI prototyping through LLMs. We demonstrate textual representations to support requirement recognition and component generation, addressing gaps in the fragmented tool landscape of GUI development. Our findings enabling further exploration of how LLM-based GUI prototyping support can be integrated into software development workflows. Our work not only provides insights into structuring textual representations based on the example of Material Design (see figure 5), but also offers insights into optimizing these representations to reduce token usage.

7 Threats to Validity

In the following, we provide a collection of threats to internal validity, such as selection biases, and external validity, such as threats to generalizability. Furthermore, we briefly explain how they may influence the results of our paper.

7.1 Internal Validity

Measuring Treatment Completeness. We evaluated several aspects such as the detection, matching, and component generation mechanisms as well as the overall design of the proposed plug-in within an user study. Therefore, the contribution of each of these aspects for the quality of the created GUI prototypes is not entirely clear. To address this issue, we initially conducted the evaluation of LLM-based generation of GUI components based on user stories ex ante and showed its effectiveness, indicating that a large contribution for the improvement of the quality of the GUI prototypes and the improvements in terms of efficiency are mainly due to the automatic GUI component generation.

7.2 External Validity

Artificial Lab Setting. The user study was conducted in an artificial lab setting including a less representative population of participants as a proxy for UX designers regarding the group (i.e. undergraduate and graduate students) as well as the young age distribution (25.4 years on average). However, we primarily utilized this conducted user study to evaluate various usability aspects associated with the proposed approach. To address the cross-functional nature of GUI prototyping, we conducted interviews with stakeholders from different roles (UX, PD, SD) to ensure that the assistant's design rationals reflect the perspectives of diverse stakeholders. To strengthen the evaluation of the quality of generated user stories and generated GUI components, we conducted an additional annotation of the generated artifacts with self-reported UI/UX professionals on *Prolific* [48]. The obtained results show the generation effectiveness. Nonetheless, we did not the evaluate the assistant in a co-design situation involving multiple stakeholders at the same time, which represents an important direction for future work to explore its potential in collaborative prototyping contexts.

Artificial Functional User Stories. To evaluate our proposed approach, we heavily relied on the publicly available user story dataset from Kolthoff et al. [35], which is the only dataset available combining user stories with GUI prototypes. These user stories solely focus on functional aspects of a single GUI prototype. In more natural settings, user stories often span across several GUI prototypes as well as encompass non-functional aspects. However, in our work we decided to initially focus on functional user stories for reducing complexity, yet still provide valuable support and facilitate the creation of GUI prototypes. In addition, the employed US dataset is validated and filtered for quality [35].

High-Level GUI Descriptions for Component Generation. To provide additional context for the generation of GUI components as part of its evaluation, we employed additional high-level GUI descriptions as an input to the ZS prompt for the LLM. Moreover, in the user study, the context of the current GUI prototype status could meaningfully be utilized by the LLM, for example, to adapt the generated GUI components to the GUI prototype and to predict accurate positioning of the GUI components. While these brief GUI descriptions can easily be obtained in practical GUI prototyping settings, in the conducted evaluation they were created by a research assistant and evaluated for accuracy by the paper authors.

Participants Interview. There is a potential sample bias in recruiting UX professionals, product owners and software developers for the initial interviews which possess limited diversity in terms of company size, geographic location and experience. However, we included multiple distinct participants for each of the roles.

8 Limitations And Future Work

Mobile GUIs and Limited Component Library. Since our approaches focuses solely on mobile GUIs, other GUI types with varying characteristics (e.g., screen size, domains etc.) are disregarded. In addition, we heavily relied on mobile GUIs taken from the *Rico* dataset [17], which has a restricted scope and usefulness. However, the *Rico* dataset is the largest publicly available GUI dataset and encompasses GUIs from over 27 different domains. In the future, we plan to extend our approach to more diverse GUI types. In addition, the considered GUI component library is restricted in size and configuration complexity. To enable a meaningful support, we already included over 89 different *MD* GUI components and 100 icons, however, we plan to further expand both libraries to additionally enhance the support capabilities.

Static GUI Prototypes. In this work, we proposed a support approach for creating single static GUI prototypes and the resulting format does not allow for interaction. Usually, the interactions considered in practice span over multiple GUIs and maintaining consistency across screens within an application is of high importance. In order to improve the usefulness and application scenarios of the proposed approach, we plan to further extend our LLM-based approaches and plug-in to additionally enable the support of user stories spanning across several GUI prototype screens.

Functional User Stories. Since the current approach solely provides support for functional user stories, the application scenarios are limited. However, particularly for creating initial GUI prototypes for rapidly obtaining customer feedback regarding the functionality, our approach can be applied. In such an initial requirements elicitation scenario where the GUI prototypes act as a functional requirements communication artifact, the detailed design aspects are of less significant role. For future work, we plan to extend the support to also non-functional user stories, for example, including styling aspects such as coloring and corporate design, among others. However, styling aspects of GUI components could also be integrated by replacing the component library within our approach.

Therefore, the assistant could be extended to enable providing textual prompts for stylistic or design requirements. To increase flexibility, another extension could be that the assistant provides multiple variants of the generated component.

Handover, Versioning and Commenting. One additional limitation is that we could only thoroughly implement selected collaboration requirements for the version of the assistant in this paper. In the interviews, both UX designers and product owners described the versioning of user stories, commenting on user stories combined with their components in the GUI prototype, and an improved handover (e.g., of parameters to software developers) compared to the existing functionalities in *Figma* as further requirements. Versioning was addressed in our intervention, but the implementation of our approach in its current form is limited to creating new components and cannot adapt existing components. We plan to address this in the future. Commenting functionality was not implemented, but as *Figma* already supports commenting components, it is feasible to link user stories to components and leverage *Figma's* commenting feature. Finally, the extraction and handover of data to developers were not included, but tools like *Figma* and *Relay* [23] already provide solutions for this.

9 Conclusion

In this paper, we provide insights into challenges from practice and provide design rationales for an effective LLM-based GUI prototyping assistant tailored for UX designers working within software development teams. We extend and further evaluate the LLM-based approach of Kolthoff et al. [35], enabling user story detection, user story matching, and GUI (component) generation for GUI prototyping. Building on this work, we show that the LLM-based approach can successfully generate GUI component recommendations from user stories. Furthermore, we evaluate and optimize the textual representation of the GUI prototypes from *Figma* and component library for the LLM, reducing the necessary amount of tokens and making our enhanced approach feasible for contexts with limited available resources. We present a novel LLM-based assistant as a plug-in for *Figma* with role-specific functionalities. Particularly, we found that participants using our assistant completed the given user stories to a higher degree. Using the treatment, our assistant also led to significantly higher satisfaction of experts with the created GUI prototype designs. Furthermore, using the user story detection, matching, and automated GUI (component) generation, our participants reported lower task load values, significantly higher confidence in creating professionally appearing GUI prototypes, high interest in future use of the assistant, and a higher impact on design quality. Inspired by real-world requirements from product owners collected in our interviews, we successfully expanded the approach to derive user stories directly from GUI prototypes that can then be directly refined by product owners. In our evaluation, large parts of the user stories were rated as specific, explicit, and derived from identifiable GUI components.

Our results show that LLM-based user story detection, matching, and the automated GUI (component) generation can be integrated directly into *Figma* providing a resource-efficiently prototyping assistant to increase user story completion and facilitating the integration of user stories and GUI prototypes.

References

[1] Diego Martin Arroyo, Janis Postels, and Federico Tombari. 2021. Variational Transformer Networks for Layout Generation . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 13642–13652. doi:10.1109/CVPR46437.2021.01343

[2] Atlassian Corporation. 2024. JIRA. www.atlassian.com/software/jira Accessed: 2024-09-11.

[3] Dirk Baumer, W Bischofberger, Horst Lichter, and Heinz Zullighoven. 1996. User interface prototyping-concepts, tools, and experience. In *Proceedings of IEEE 18th International Conference on Software Engineering*. IEEE, IEEE Computer Society, Los Alamitos, CA, USA, 532–541. doi:10.1109/ICSE.1996.493447

[4] Farnaz Behrang, Steven P Reiss, and Alessandro Orso. 2018. GUIfetch: supporting app design and development through GUI search. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems* (Gothenburg, Sweden) *(MOBILESoft '18)*. Association for Computing Machinery, New York, NY, USA, 236–246. doi:10.1145/3197231.3197244

[5] Carlos Bernal-Cárdenas, Kevin Moran, Michele Tufano, Zichang Liu, Linyong Nan, Zhehan Shi, and Denys Poshyvanyk. 2019. Guigle: a GUI search engine for Android apps. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings (ICSE '19)*. IEEE, IEEE Press, Montreal, Quebec, Canada, 71–74. doi:10.1109/ICSE-Companion.2019.00041

[6] Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. 2012. Apache lucene 4. In *SIGIR 2012 workshop on open source information retrieval* (Portland, Oregon) *(Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval)*. Department of Computer Science, University of Otago, Dunedin, New Zealand, 17.

[7] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. 2011. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering* (Lancaster, United Kingdom) *(AREW '11)*. Association for Computing Machinery, New York, NY, USA, Article 3, 5 pages. doi:10.1145/2068783.2068786

[8] Paul Brie, Nicolas Burny, Arthur Sluÿters, and Jean Vanderdonckt. 2023. Evaluating a large language model on searching for gui layouts. *Proceedings of the ACM on Human-Computer Interaction* 7, EICS (2023), 1–37.

[9] John Brooke. 1996. SUS: A 'Quick and Dirty' Usability Scale. In *Usability Evaluation In Industry* (0 ed.), Patrick W. Jordan, B. Thomas, Ian Lyall McClelland, and Bernard Weerdmeester (Eds.). CRC Press, London, England, 207–212. doi:10.1201/9781498710411-35

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[11] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 423, 14 pages. doi:10.1145/3411764.3445762

[12] Erin A. Carroll, Celine Latulipe, Richard Fung, and Michael Terry. 2009. Creativity factor evaluation: towards a standardized survey metric for creativity support. In *Proceedings of the seventh ACM conference on Creativity and cognition*. ACM, Berkeley California USA, 127–136. doi:10.1145/1640233.1640255

[13] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery DC: Design Search and Knowledge Discovery through Auto-created GUI Component Gallery. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–22.

[14] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development.* Addison Wesley Longman Publishing Co., Inc., USA.

[15] Sourav Debnath, Paola Spoletini, and Alessio Ferrari. 2021. From Ideas to Expressed Needs: an Empirical Study on the Evolution of Requirements during Elicitation . In *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE Computer Society, Los Alamitos, CA, USA, 233–244. doi:10.1109/RE51729.2021.00028

[16] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 845–854. doi:10.1145/3126594.3126651

[17] Biplab Deka, Zifeng Huang, Chad Franzen, Jeffrey Nichols, Yang Li, and Ranjitha Kumar. 2017. ZIPT: Zero-Integration Performance Testing of Mobile App Designs. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. Association for Computing Machinery, New York, NY, USA, 727–736. doi:10.1145/3126594.3126647

[18] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2022. A Survey on In-context Learning. arXiv:2301.00234 [cs.CL] https://arxiv.org/abs/2301.00234

[19] Benjamin D Douglas, Patrick J Ewell, and Markus Brauer. 2023. Data quality in online human-subjects research: Comparisons between MTurk, Prolific, CloudResearch, Qualtrics, and SONA. *Plos one* 18, 3 (2023), e0279720.

[20] Sidong Feng, Mingyue Yuan, Jieshan Chen, Zhenchang Xing, and Chunyang Chen. 2023. Designing with Language: Wireframing UI Design Intent with Generative Large Language Models. arXiv:2312.07755 [cs.HC] https://arxiv.org/abs/2312.07755

[21] Figma, Inc.. 2024. Figma. www.figma.com Accessed: 2024-09-11.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[23] Google LLC. 2024. Relay. www.relay.material.io Accessed: 2024-09-11.

[24] Google LLC. 2025. Material 3 Design Kit. https://m3.material.io/blog/material-3-figma-design-kit Accessed: 2025-01-20.

[25] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. 2021. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. IEEE Computer Society, Los Alamitos, CA, USA, 1004–1014. doi:10.1109/ICCV48922.2021.00104

[26] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (Oct. 2006), 904–908. doi:10.1177/154193120605000909

[27] Forrest Huang, John F Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based user interface retrieval. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3290605.3300334

[28] Forrest Huang, Gang Li, Xin Zhou, John F. Canny, and Yang Li. 2021. Creating User Interface Mock-ups from High-Level Text Descriptions with Deep-Learning Models. arXiv:2110.07775 [cs.HC] https://arxiv.org/abs/2110.07775

[29] Andrej Karpathy. 2022. minGPT - GitHub Repository. https://github.com/karpathy/minGPT. Accessed: 2024-07-20.

[30] Tiffany Knearem, Mohammed Khwaja, Yuling Gao, Frank Bentley, and Clara E Kliman-Silver. 2023. Exploring the future of design tooling: The role of artificial intelligence in tools for user experience professionals. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, Article 384, 6 pages. doi:10.1145/3544549.3573874

[31] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[32] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2021. Automated Retrieval of Graphical User Interface Prototypes from Natural Language Requirements. In *International Conference on Applications of Natural Language to Information Systems*. Springer, Springer International Publishing, Cham, 376–384.

[33] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2021. GUI2WiRe: rapid wireframing with a mined and large-scale GUI repository using natural language requirements. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (Virtual Event, Australia) *(ASE '20)*. Association for Computing Machinery, New York, NY, USA, 1297–1301. doi:10.1145/3324884.3415289

[34] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2023. Data-driven prototyping via natural-language-based GUI retrieval. *Automated Software Engineering* 30, 1 (2023), 13.

[35] Kristian Kolthoff, Felix Kretzer, Christian Bartelt, Alexander Maedche, and Simone Paolo Ponzetto. 2024. Interlinking User Stories and GUI Prototyping: A Semi-Automatic LLM-Based Approach. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, Reykjavik, Iceland, 380–388. doi:10.1109/RE59067.2024.00045

[36] Marja Käpyaho and Marjo Kauppinen. 2015. Agile requirements engineering with prototyping: A case study . In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE Computer Society, Los Alamitos, CA, USA, 334–343. doi:10.1109/RE.2015.7320450

[37] Chunggi Lee, Sanghoon Kim, Dongyun Han, Hongjun Yang, Young-Woo Park, Bum Chul Kwon, and Sungahn Ko. 2020. GUIComp: A GUI Design Assistant with Real-Time, Multi-Faceted Feedback. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3313831.3376327

[38] Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 578, 15 pages. doi:10.1145/3411764.3445049

[39] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[40] Markus Mannio and Uolevi Nikula. 2001. Requirements Elicitation Using a Combination of Prototypes and Scenarios. In *Workshop em Engenharia de Requisitos*. Workshop on Requirements Engineering 2001, Buenos Aires, Argentina, 283–296. https://api.semanticscholar.org/CorpusID:15021206

[41] Soumik Mohian and Christoph Csallner. 2022. PSDoodle: Searching for App Screens via Interactive Sketching . In *2022 IEEE/ACM 9th International Conference on Mobile Software Engineering and Systems (MobileSoft)*. IEEE Computer Society, Los Alamitos, CA, USA, 84–88. doi:10.1145/3524613.3527807

[42] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1221–1224. doi:10.1145/2702123.2702149

[43] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[44] OpenAI. 2024. GPT-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2024-09-11.

[45] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2014. Learning layouts for single-pagegraphic designs. *IEEE transactions on visualization and computer graphics* 20, 8 (2014), 1200–1213.

[46] Carla Pacheco, Ivan García, and Miryam Reyes. 2018. Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. *IET Software* 12, 4 (2018), 365–378. doi:10.1049/iet-sen.2017.0144

[47] Srishti Palani, David Ledo, George Fitzmaurice, and Fraser Anderson. 2022. "I don't want to feel like I'm working in a 1960s factory": The Practitioner Perspective on Creativity Support Tool Adoption. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 379, 18 pages. doi:10.1145/3491102.3501933

[48] Prolific Academic Ltd. 2024. Prolific. www.prolific.co Accessed: 2024-09-11.

[49] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[50] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[51] Thiago Rocha Silva, Marco Winckler, and Hallvard Trætteberg. 2019. Ensuring the Consistency Between User Requirements and Graphical User Interfaces: A Behavior-Based Automated Approach. In *Computational Science and Its Applications – ICCSA 2019*, Sanjay Misra, Osvaldo Gervasi, Beniamino Murgante, Elena Stankova, Vladimir Korkhov, Carmelo Torre, Ana Maria A.C. Rocha, David Taniar, Bernady O. Apduhan, and Eufemia Tarantino (Eds.). Springer International Publishing, Cham, 616–632.

[52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models . In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 10684–10695. doi:10.1109/CVPR52688.2022.01042

[53] Jim Rudd, Ken Stern, and Scott Isensee. 1996. Low vs. high-fidelity prototyping debate. *interactions* 3, 1 (1996), 76–85.

[54] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 543–555. doi:10.1145/2901790.2901817

[55] Jialiang Wei, Anne-Lise Courbis, Thomas Lambolais, Binbin Xu, Pierre Louis Bernard, and Gerard Dray. 2023. Boosting GUI Prototyping with Diffusion Models . In *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE Computer Society, Los Alamitos, CA, USA, 275–280. doi:10.1109/RE57278.2023.00035

[56] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs.CL] https://arxiv.org/abs/2206.07682

[57] Peter Windsor and Graham Storrs. 1992. Prototyping user interfaces. In *IEE Colloquium on Software Prototyping and Evolutionary Development*. IET, Institution of Electrical Engineers, Savoy Place, London WC2R OBL, 4/1–4/4.

[58] Mingyue Yuan, Jieshan Chen, and Aaron Quigley. 2024. MAxPrototyper: A Multi-Agent Generation System for Interactive User Interface Prototyping. arXiv:2405.07131 [cs.HC] https://arxiv.org/abs/2405.07131

[59] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. 2021. Guigan: Learning to Generate GUI Designs Using Generative Adversarial Networks. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE '21)*. IEEE Press, Madrid, Spain, 748–760. doi:10.1109/ICSE43902.2021.00074