MorphGUI: Real-time Natural Language Interface Customization with Large Language Models

Author 1[a,*], Author 2[a]

[a]*Department of Control and Computer Engineering, Politecnico di Torino, Torino, 10129 Italy*

Abstract

User interface customization traditionally relies on predefined options or learned patterns from historical user data, limiting the scope of possible personalizations. We introduce MorphGUI, a framework that leverages Large Language Models to enable free-form interface customization through natural language interaction. MorphGUI dynamically generates React components based on users' expressed preferences, eliminating the need for predefined customization options or user models. Through a pilot implementation in a calendar application and user study (n=18), where participants reshaped interfaces through natural language descriptions, we demonstrate that through MorphGUI users can successfully achieve their personalization goals.

Keywords: Keywords goes here with, separator

1. Introduction

User interfaces (UIs) are crucial in a digital society – providing access to essential services, information, or entertainment. These are, traditionally, created following a one-size-fits-all approach aimed at an average user, with designers detaining full control of the UI crafting process. On the other side, each individual has unique needs and experience using the software but limited agency to control how UIs are presented. UI personalization [1] represents a solution to adjust UIs to the needs and characteristics of users. Customization, particularly, enables users to directly manipulate the graphical objects on the screen – defining new properties and values. It improves user experience [2] and allows people to feel in control, express their identity, or create a connection with UIs [3, 4, 5, 1]. However, not everyone is skillful enough to customize their UIs or wants to invest time doing it [6]. Moreover, people have different expertise and tech savviness levels that influence how they customize interfaces [1]. As such, solutions are needed to empower non-expert users to personalize UIs without requiring extensive knowledge or effort [7]. In this work, we present MorphGUI, a framework that leverages LLMs to enable free-form interface customization through natural language interaction [8]. Unlike existing approaches that ground on predefined customization options or user models, MorphGUI allows users to create or select specific interface components describing the functional ("what it should do") and visual ("how it should appear") aspects through natural language. MorphGUI integrates with React's component architecture through a specialized middleware layer that modifies the application's component tree at runtime. By structuring the customization process into these distinct steps and providing clear templates for different aspects of the UI, we help users articulate their preferences more precisely while maintaining the flexibility of natural language interaction. We validate MorphGUI through a user study with 18 participants tasked with customizing a calendar interface to match a target design. In each task, participants were shown a reference calendar interface and asked to use MorphGUI to transform a basic calendar into one matching the target layout, functionalities and visual style through natural language instructions. The study results indicate that users without technical expertise could effectively generate personalized calendar views by expressing their desired interface changes. Results from semi-structured interviews highlight areas for further improvement in the natural language interaction.

2. Related Work

2.1. User Interface Personalization

Traditional one-size-fits-all user interfaces fail to accommodate the diverse contexts and needs of individual users, often resulting in suboptimal user experiences [9]. Users' personalization needs are highly individual and context-dependent [10]. Existing work showed the benefits of having UIs adapted to people's needs [2], characteristics [11, 12], or cultural backgrounds [13]. In addition, studies of interface customization behavior reveal that users often want to modify interfaces in ways not anticipated by designers [14, 15]. For instance, users might want to reorganize interface elements based on their workflow [16], adjust visual properties for better accessibility [17], or modify interaction patterns to match their expertise level [18].

Current approaches to user-driven interface personalization can be categorized into three main strategies: manual customization, done by the user [15, 19, 20, 21, 2, 22], adaptation, driven by the system that learn from user behavior [17, 23, 24],

*Corresponding author
Email addresses: mail 1 (Author 1), mail 2 (Author 2)

and hybrid approaches that combine user control with system suggestions [25]. While manual customization gives users direct control, it requires user expertise and technical knowledge. Adaptive systems can reduce this burden but may make changes that users find unpredictable or undesirable [10]. Hybrid approaches attempt to balance automation with user control but can struggle to capture complex adaptation needs [26].

Adaptation approaches rely on predefined rules and user data to automatically adjust UIs. Adaptive UIs (AUIs) have evolved to respond to various factors including display parameters [27], user abilities [28, 29], context [30], preferences [17, 25], and tasks [17]. Implementation methods range from pre-programmed frameworks [31, 32] to context-aware [30] and mixed-reality interfaces [33]. Model-based approaches provide abstract specifications of interfaces through formal models that describe UI elements and their relationships independently of specific platforms, enabling automatic generation of interfaces for ubiquitous [34] and multi-device computing [35]. In contrast, optimization-based techniques like SUPPLE [17] and ARNAULD [36] automatically generate layouts by optimizing interface configurations based on user preferences and abilities. Recent work has extended these concepts using semantic properties [37, 38] and developer-defined objective functions [39].

Customization generally allows users to manually control the personalization. One example is CrowdAdapt [2], a direct manipulation toolkit that allows customization with intuitive operations such as move, resize, spacer, hide, collapse, font size, and multi-column. Customization stands out from other approaches as it does not require the collection of personal data and generally allows for more personalizations. This is done at the expense of more time invested by users, which may, however, outweigh customization benefits [20, 6]. Therefore, enabling rapid and effortless customization becomes crucial for encouraging users to personalize their interfaces.

Sundar and Marathe [1] found that the tech savviness level impacts people's customization perspective. Less tech-savvy users have negative attitudes toward an interface when asked to customize it but a positive attitude when presented with an already personalized one. Tech-savvy users, on the other hand, showed more positive attitudes when allowed to customize. Current popular customization tools like Stylish [40] and Tampermonkey [41] enable CSS and JavaScript customization, overlooking non-expert users by requiring coding knowledge.

In addition, traditional customization approaches typically constrain users to predefined options [14] or require them to learn complex configuration languages [42]. While these methods provide some flexibility, they often fail to capture users' actual adaptation needs and mental models [16].

Our work aims to bridge this gap by introducing an approach that leverages Large Language Models to enable interface customization through structured natural language inputs. Unlike previous solutions that either require technical expertise (like Stylish [40] and Tampermonkey [41]) or limit users to predefined options (like CrowdAdapt [2]), our framework allows users to express their customization preferences through a guided interface where they can select specific components and describe desired changes in natural language. Our approach allows users to maintain fine-grained control over their personalizations without needing to understand CSS, JavaScript, or complex configuration options.

2.2. Computational UI Development Tools

As our system translates abstract customization objectives into concrete interface implementations, it builds upon prior research in automated UI Development. Numerous computational techniques, such as layout optimization and generation, have been integrated into tools that aid in the development process. The space of possible UI layouts is very large, so some work has proposed surfacing relevant design examples to inspire new designs [43]. Webzeitgeist, for example, collected a large dataset of webpage layouts and mined the common design patterns used with different web pages [44]. RICO is a similar dataset collected from Android apps, and introduced an efficient machine learning model for finding designs with similar layouts [45]. Improvements have been made to improve the efficiency of exemplar search [46], and several tools have been developed that integrate these search capabilities using example galleries [47] and sketches [48]. An alternative to finding existing UI designs is to generate them. Numerous systems have applied optimization methods to generate a set of candidates or surface interesting design alternatives. Early work on layout design generation and suggestion focused on recommending graphic designs (e.g., posters) based on detected content types and a set of pre-defined optimization objectives [49]. Sketchplore was a UI prototyping/sketching tool that integrated a layout optimizer to provide design suggestions and alternative designs, which enabled users to quickly and efficiently explore a range of design alternatives [50]. Similarly, Scout was a system that enabled users to narrow a search space by specifying UI constraints for automated development [51]. GRIDS applied similar techniques earlier in the design process and supported creativity by producing a diverse set of starting points [52]. Recent advances in natural language processing and LLMs offer new possibilities for interface adaptation. Instead of learning complex configuration options or relying on system-driven adaptation, users can express their adaptation preferences in natural language [53, 54]. This approach allows for more flexible and expressive adaptation while maintaining user control over the process. Misty [55] has leveraged large language models for UI generation and modification by enabling developers to interactively blend concepts from UI examples into their work through a multimodal AI system. Misty demonstrates how modern AI models can be used to both understand design intent and generate UI code, though it focuses specifically on design blending rather than direct interface customization. Very close to our work, ReactGenie [56] provides a framework for developing multimodal applications that leverages LLMs to enable natural language interactions. However, while both approaches use LLMs and integrate directly with React's underlying framework to generate and modify UI components, ReactGenie focuses on translating voice and touch inputs into function calls through a neural semantic parser, our work specifi-

cally targets interface customization by allowing users to express UI preferences through a structured interface.

LLMs approach opens new challenges in ensuring consistent interpretation of user intent and maintaining interface usability across adaptations [57]. Recent work has shown that prompt engineering plays a crucial role in translating abstract user goals into concrete interface realizations, yet crafting effective prompts remains unintuitive for many users [8, 58]. Key principles for effective prompting through LLMs include providing clear instructions, using relevant examples, breaking tasks into smaller steps, and precisely specifying layout and styling requirements [59, 60, 61]. Our framework builds on these insights by implementing a prompting approach mediated through an input system designed for intuitiveness where users can specify both the functional ("what it should do") and visual ("how it should appear") aspects of each interface component. By breaking down interface descriptions into specific components and providing clear templates for different aspects of the UI, we help users overcome common prompting challenges while maintaining the flexibility of natural language interaction.

3. System Design and Implementation

3.1. Architecture Overview

We design our system with the architecture shown in Figure 1. The user interacts with the system through a web-based front-end that combines application-specific functionality with interface customization capabilities. The dynamic component module acts as the core visualization engine, managing both the rendering of the current interface and the generation of updated views based on user instructions. This module directly communicates with the front-end for real-time interface updates. It implements a component-based architecture that allows for dynamic modification and re-rendering of interface elements without requiring page reloads. The server processes user preferences, manages database operations, keeps track of previous system modifications and orchestrates interactions with the AI module. The AI component, implemented using a Large Language Model, processes natural language descriptions and converts them into concrete interface specifications. These specifications undergo validation before being transformed into executable interface code by the dynamic component system. A database layer handles persistence of user preferences, interface versions, and component configurations. The system maintains both current and previous interface states, enabling version control functionality that allows users to track and revert changes as needed. This versioning system provides a safety net for users experimenting with interface modifications.

3.2. Natural Language Interface

The natural language interface, shown in Figure 2, implements a structured approach to capturing user preferences through a dedicated customization panel. Rather than providing users with complete freedom in describing their desired

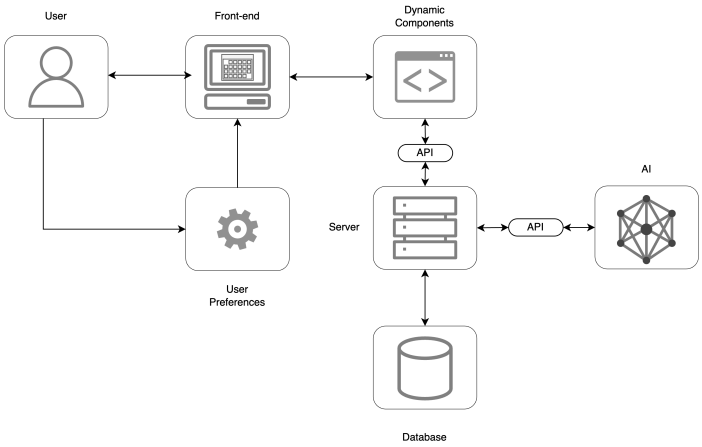

Figure 1: System architecture showing the interaction between user interface, server components, and AI services

changes, which could lead to ambiguous specifications, the system guides users through a systematic customization process. For each selected component, the interface presents two distinct input fields that separate functional and aesthetic requirements. The first field, labeled "What it should do," captures behavioral modifications and functional requirements. The second field, "How it should appear," focuses on visual aspects like styling and layout. is dual-field approach helps users articulate their preferences more precisely by providing distinct contexts for functionality and appearance while enabling the system to maintain separate prompt contexts for processing behavioral changes versus visual modifications. A gradient-styled generate button triggers the customization process, while a "Previous" option enables users to undo changes if needed.

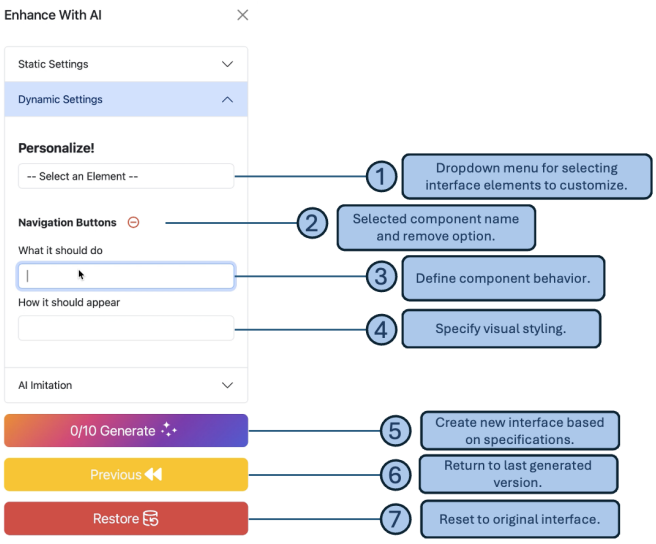

Figure 2: Natural language customization interface showing component selection and preference input fields

The prompt engineering strategy underlying this interface combines the selected component's context, user inputs, and

technical constraints into structured prompts for the Large Language Model. These prompts ensure that generated modifications maintain component functionality while implementing the requested changes within the technical framework's constraints.

3.3. Dynamic Component System

The dynamic component system enables real-time interface updates by managing component generation, evaluation, and rendering during runtime. At its core lies a specialized React [62] component that serves as a bridge between the generated interface code and the application's runtime environment. The runtime component management is implemented through a combination of code evaluation and dynamic rendering. When new interface specifications are received, the system first preprocesses the code to ensure compatibility with the runtime environment. This preprocessing includes handling import statements, resolving dependencies, and ensuring proper integration with the application's existing component ecosystem. The component employs Babel [63] for code transformation and compilation, enabling it to safely evaluate and execute the generated code. State preservation represents a critical aspect of the system. When updating components, the dynamic component system maintains the application's state through React's lifecycle methods. This ensures that user data and interaction states persist across interface updates. The system implements a state tracking mechanism that preserves important values during component regeneration. Code generation and evaluation occur through a pipeline that ensures security and reliability. Generated code undergoes validation before being transformed into executable components. The system employs a structured approach to component creation. Initially, code preprocessing handles dependencies and imports, ensuring all required resources are properly managed. Following this, the code undergoes transformation using Babel, which converts modern JavaScript features into compatible code. The system then creates a new function component through dynamic evaluation, allowing for runtime component generation. Finally, this newly created component is integrated into the React component tree, enabling rendering within the application's interface. The styling and layout control is managed through both inline styles and dynamic CSS generation. The system supports various styling approaches, including direct style objects and class-based styling, while maintaining consistency with the application's existing style system. Style definitions are processed alongside component code to ensure proper rendering. This approach to dynamic component management enables the system to handle complex interface updates while maintaining application stability and performance. The separation of concerns between component generation, state management, and styling control allows for flexible and reliable interface customization.

4. Evaluation

4.1. Study Design

To evaluate our approach to natural language interface customization, we conducted a within-subjects experiment where 18 participants completed two customization tasks of increasing complexity. Figure 3 shows the starting calendar interface and the target designs for both easy and complex customization tasks. The design allowed direct comparison of participant performance and behavior across different levels of interface complexity while controlling for individual differences in technical background and LLM experience.

We collected both quantitative and qualitative data through multiple methods. Quantitative measures included System Usability Scale (SUS) scores and task completion metrics. Qualitative feedback was gathered through open-ended questions during post-task interviews, focusing on participants' experiences with the natural language customization process. Additionally, we evaluated the quality of customization outcomes by assessing the visual and functional similarity between participants' generated interfaces and the target designs.

4.2. Participants

We conducted the study with 18 participants through university networks and professional channels. Table 1 presents the demographic distribution of our sample across age, professional background, and experience with Large Language Models. The age distribution showed 12 participants in the 20-30 age range, 5 participants aged 30-60, and 1 participant under 20. Professional backgrounds demonstrated diversity, with 9 participants from technology-related fields. The remaining participants came from various sectors including legal (2), healthcare (2), design (1), marketing (1) and other fields (3). This diversity provided perspectives from different professional contexts. Regarding LLM experience, 7 participants reported occasional usage, while 6 were regular users. 4 participants had no prior knowledge of LLMs, and 1 had only heard of them without direct experience.

4.3. Procedure

We conducted the study remotely via video conference calls. Initially, participants received an overview of the study and completed informed consent forms. The study was structured in three phases:

Setup and Training. Before interacting with the system, participants were asked an open-ended question about their calendar interface customization preferences and needs. This preliminary inquiry aimed to understand users' expectations regarding interface personalization without biasing them with existing solutions, following established HCI methodologies for user-centered design. After collecting their responses, participants received a brief introduction to the interface customization system and a short demo to familiarize themselves with the natural language input mechanism and generation process.

Customization Tasks. Participants completed two tasks in a fixed order. Each participant began with the simple task and then proceeded to the complex task. Each task had a 20-minute time limit and a maximum of 10 generation attempts. For each task, participants started with

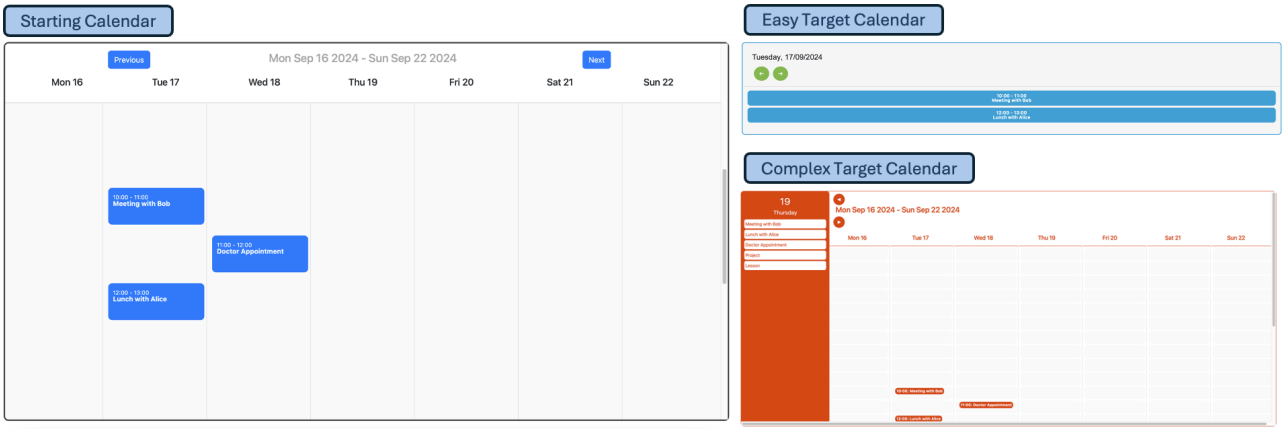

Figure 3: Calendar interface examples showing the starting calendar (left) and target calendars for easy (top right) and complex (bottom right) customization tasks.

the same base calendar interface and attempted to replicate a provided target design. Participants could revert to previous versions or reset to the original interface at any time. The system logged all interactions and generation attempts. We designed the tasks to be completed sequentially to allow participants to familiarize themselves with MorphGUI's natural language customization capabilities during the initial task. The consistent task sequence ensured that all participants had the same learning opportunities, making the observed improvements in performance attributable to increased familiarity with MorphGUI.

Evaluation and Interview. After completing both tasks, participants filled out the System Usability Scale questionnaire and participated in a semi-structured interview about their experience with the system. The interview focused on their customization strategy, challenges encountered, and suggestions for improvement.

Each session lasted approximately 45 minutes (M = 43 minutes, SD = 8 minutes). All sessions were recorded with participant consent for subsequent analysis of interaction patterns and customization strategies.

4.4. Measurements

During the study, we collected three types of measurements. First, system logs recorded participant interactions, including session duration, generation attempts (out of 10 available per task), interface resets, reversions to previous versions, and system errors or warnings. These quantitative metrics provided insights into participants' interaction patterns and system performance.

Second, after completing both tasks, participants completed a System Usability Scale (SUS) questionnaire to evaluate the overall usability of the natural language customization system. The SUS questionnaire consisted of 10 items rated on a 5-point Likert scale, covering aspects such as system complexity, ease of use, and learning curve.

Third, participants answered open-ended questions about their customization experience, focusing on their approach to natural language descriptions, challenges encountered, and suggestions for improvement. These responses provided qualitative insights into participants' mental models and interaction strategies.

4.5. Data Analysis

System logs were analyzed to compute mean session duration, average number of generation attempts, and frequency of resets and reversions across participants. These metrics were compared between tasks using paired t-tests to assess differences in interaction patterns between simple and complex customization scenarios.

Interface evaluation scores (0-5 scale) were calculated for each component and aggregated into a target adherence score. We computed mean scores and standard deviations for each component category and the overall adherence score. Pearson correlation coefficients were calculated to examine relationships between participant characteristics (LLM experience, technical background) and customization success. For categorical variables, we employed Wilcoxon rank-sum tests to analyze differences in performance across groups.

SUS questionnaire responses were processed following standard scoring procedures. Raw scores were normalized to a 0-100 scale, with mean scores and standard deviations calculated. These were compared against established SUS benchmarks to evaluate system usability. Additionally, we analyzed individual question scores to identify specific usability strengths and concerns.

For qualitative feedback from interviews and open-ended questions, we report representative quotes and observations that participants' customization strategies and experience with our system. Results are presented using descriptive statistics and, where applicable, statistical significance is reported at p ¡ .05.

We evaluated the final generated interfaces against the target designs using a structured assessment framework examining layout accuracy, component positioning, color scheme im-

Table 1: Overview of study participants (N=18), showing individual demographics, LLM usage and UI development experience levels.

ID (Age)	Professional Background	LLM Usage Experience*	UI Development Experience**
P1 (20-30)	Technology	●●●○	●○○
P2 (20-30)	Technology	●●●●	●○○
P3 (20-30)	Technology	●●●●	●○○
P4 (20-30)	Design	●●●○	●●○
P5 (20-30)	Technology	●●●○	●○○
P6 (20-30)	Technology	●●●●	●●●
P7 (20-30)	Healthcare	●●●●	●○○
P8 (20-30)	Marketing	●●●○	●○○
P9 (20-30)	Healthcare	●●●○	●○○
P10 (20-30)	Technology	●●●●	●○○
P11 (<20)	Other	●●●●	●○○
P12 (30-60)	Other	●●○○	●○○
P13 (30-60)	Healthcare	●○○○	●○○
P14 (20-30)	Healthcare	●●●○	●○○
P15 (30-60)	Legal	●○○○	●○○
P16 (30-60)	Legal	●○○○	●○○
P17 (20-30)	Technology	●●●○	●○○
P18 (30-60)	Other	●○○○	●○○

* ●○○○= No Knowledge, ●●○○= Heard Of, ●●●○= Used Occasionally, ●●●●= Regular User
** ●○○= No Knowledge, ●●○= Basic Knowledge, ●●●= Proficient,

plementation, navigation functionality, and event display formatting. These scores were aggregated into a target adherence score to evaluate participants' ability to achieve their customization objectives through MoldGUI.

5. Results

5.1. Task Performance

Participants successfully completed both customization tasks, with higher performance on the simple task (M=74.7%, SD=12.9%) compared to the complex task (M=70.2%, SD=9.6%). Task completion times decreased between the simple (M=16.7 min, SD=7.4) and complex tasks (M=13.6 min, SD=4.6), suggesting a learning effect. An overview of task performance and prompt behavior across participants is illustrated in Figure 4.

Participants completed both customization tasks in a fixed order, starting with the simple task followed by the complex task. Despite the increased complexity in the second task, participants maintained strong performance (M=70.2%, SD=9.6%) compared to the simple task (M=74.7%, SD=12.9%). Task completion times decreased from simple (M=16.7 min, SD=7.4) to complex tasks (M=13.6 min, SD=4.6), indicating that participants effectively transferred their learned skills to more challenging scenarios. This skill transfer is particularly noteworthy given that participants achieved comparable accuracy scores even as task complexity increased, suggesting robust learning of the natural language interface paradigm.

5.2. Interface Generation Patterns

Participants used an average of 5.0 (SD=2.0) generation attempts for the simple task and 6.0 (SD=1.9) for the complex task. Reset operations were rare (M=1.1, SD=0.8 per task), while reversions to previous versions were more common (M=2.1, SD=1.2). Analysis showed no significant correlation between number of generation attempts and final interface quality ($r = .21, p > .05$).

5.3. Natural Language Input Analysis

Average prompt length increased between simple (M=12.7 words, SD=8.1) and complex tasks (M=13.3 words, SD=8.1). Optimal results were achieved with descriptions of 15-20 words, with performance declining for both shorter and longer prompts. Figure 5 illustrates the correlation between average scores and prompt lengths, showing no strong linear relationship between these metrics. Technical background showed no significant effect on prompt effectiveness ($\chi^2(2) = 3.42$, $p > .05$).

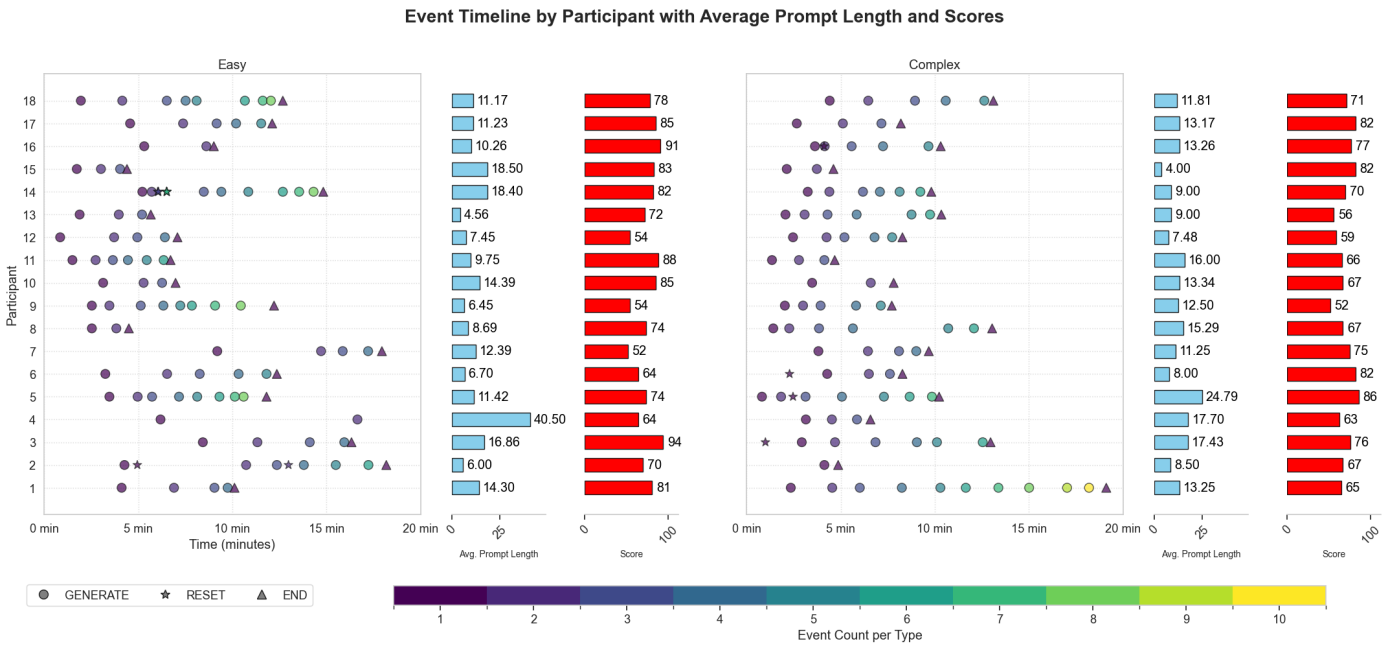

Figure 4: Event Timeline by Participant with Average Prompt Length and Scores.

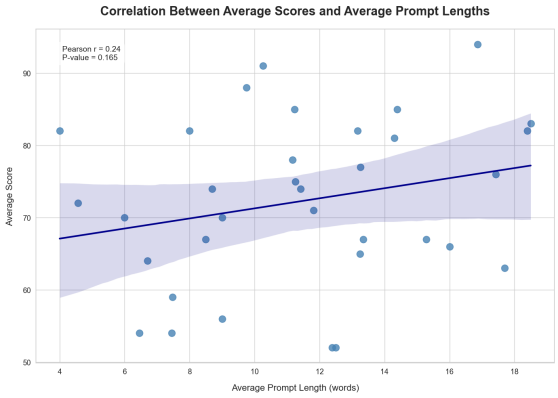

Figure 5: Correlation Between Average Scores and Average Prompt Lengths.

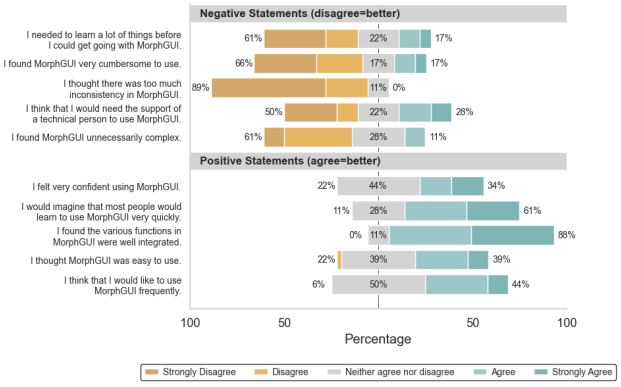

Figure 6: System Usability survey results showing agreement levels with positive and negative statements about MorphGUI (N=18).

5.4. System Usability

The system achieved a mean SUS score of 68 (SD=12.3), meeting the threshold for acceptable usability. Highest-rated aspects included "easy to learn" (M=4.2/5) and well integrated" (M=4.0/5). Lower scores were observed for "technical support needed" (M=2.8/5) and "system complexity" (M=2.6/5). Figure 6 presents detailed user feedback on various aspects of the system.

5.5. Analysis of Template Usage Patterns

Through our analysis of participants' interactions with MorphGUI's natural language interface, we identified several patterns in how users approached the separation between functionality ("What it should do") and styling/layout ("How it should appear") instructions. We used a confusion matrix approach to quantify the accuracy of template usage, as shown in Figure 7.

Our analysis revealed several key patterns in template usage, as shown in Figure 7. On average, participants correctly placed $M = 3.5$ ($SD = 0.9$) functionality instructions in the "What it should do" field, while correctly placing $M = 8.1$ ($SD = 4.8$) styling/layout instructions in the "How it should appear" field. However, we observed a consistent tendency to misplace styling and layout instructions in the "What it should do" field ($M = 7.7$, $SD = 4.7$), while functionality misplacements in the "How it should appear" field were less common ($M = 1.5$, $SD = 1.3$).

The most common error pattern was the placement of styling and layout instructions in the "What it should do" field, with participants (particularly P3, P10, and P11 with FS ¿ 12) struggling with button modifications (e.g., "transform buttons to circular shape") and component positioning (e.g., "align elements to the left"). This suggests users tend to think of visual modifi-

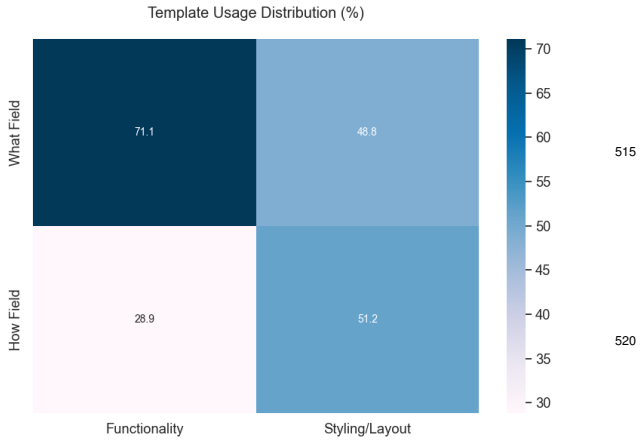

Figure 7: Confusion matrix showing template usage patterns across participants. The matrix tracks: (1) True Functionality (TF): Correct placement of functionality instructions in "What it should do" field; (2) True Styling (TS): Correct placement of styling/layout instructions in "How it should appear" field; (3) False Styling (FS): Incorrect placement of styling/layout instructions in "What it should do" field; (4) False Functionality (FF): Incorrect placement of functionality instructions in "How it should appear" field.

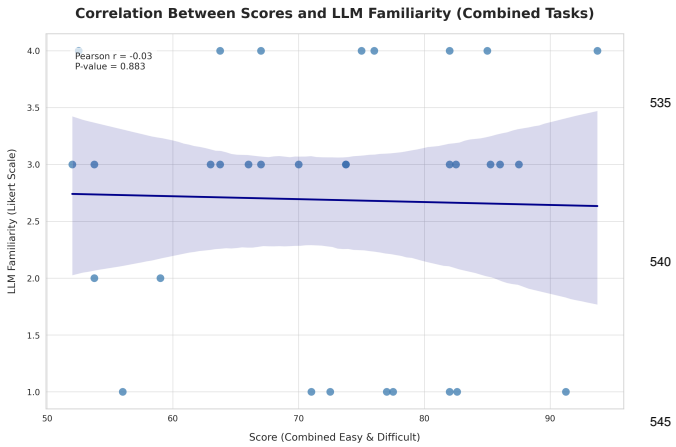

Figure 8: Correlation Between Scores and LLM Familiarity (Combined Tasks) showing no significant relationship (r = -0.03, p = 0.883)

cations as actions to be performed rather than appearance specifications.

The secondary error pattern involved functionality descriptions appearing in the "How it should appear" field, though this was less frequent. These errors typically involved repetition of functionality requirements (e.g., "should show one day at a time") rather than new functional specifications. P9 and P18 showed the highest rates of this error type (FF = 4 and FF = 5 respectively).

Notably, participants with higher rates of correct styling placement (P1, P4, P5 with TS ¿ 12) generally showed lower rates of styling misplacement (FS ¡ 5), suggesting that understanding the proper use of the "How it should appear" field correlates with better template usage overall.

5.6. Impact of LLM Experience

Analysis revealed no significant correlation between participants' LLM familiarity and their task performance (r = -0.03, p = 0.883), as shown in Figure 8. This suggests that prior experience with language models did not impact users' ability to effectively customize interfaces using MorphGUI's natural language interface.

5.7. Qualitative Feedback

The qualitative analysis of participant interviews revealed several interconnected themes that provide deeper insights into users' experiences with MorphGUI and opportunities for system enhancement. Five primary themes emerged from the data, these themes highlight both the strengths of the natural language approach and areas where additional support could enhance the user experience.

Learning Curve and Initial Experience. Most participants noted an initial adjustment period that decreased with system usage. While the interface itself was considered straightforward, users needed time to understand how to effectively formulate their customization requests. As P07 explained: "The system isn't complicated to use, but it would be useful to clarify its usage methods to understand from the start the level of specificity and complexity [...] needed in the input information to obtain the desired results." P05 similarly noted: "Initially it's more complicated to find the most comprehensive way to give instructions to the system, after various generations it became faster and easier to use."

Interface Clarity and Input Visibility. Several participants highlighted issues with input visibility and suggested improvements for interface clarity. A common concern was expressed by P14: "I would prefer to be able to view the entire text of the instructions I'm giving to the AI and to have an example of what I could write." P08 observed that "the most complicated part is understanding how to write instructions correctly for generating various parts, while understanding how to use the input system is immediate."

Request for Examples and Guidance. A recurring theme was the desire for example prompts and better guidance during the customization process. P16 stated: "the system works well, but examples with explanatory phrases could be useful to understand commands more quickly." P17 suggested to "automate the personalization process by adding examples or preset modifications to speed up the operation." P10 explicitly requested "examples before modifying something that shows how it works," while noting that "using the program often, you learn how to communicate with it."

System Effectiveness and Control. Despite initial challenges, participants generally found the system effective once familiar with it. P18 emphasized: "The system is absolutely intuitive and allows almost total control with increased use." P04 provided a nuanced observation: "The interface is quite intuitive to use even though results can vary based on how a person usually expresses themselves, and can offer a different experience for all users."

Suggestions for Improvement. Participants offered several constructive suggestions for system enhancement. P17 and P18

both suggested "highlighting the elements being modified to better understand how to proceed." P11 requested "a signal for wrong categories when they don't appear as I would like," while P04 suggested "using keywords to standardize certain functions that allow specific actions."

These qualitative insights complement the quantitative findings presented earlier and provide valuable direction for future system improvements. The feedback particularly emphasizes the importance of providing better initial guidance while maintaining the flexibility and power of natural language interaction. The themes that emerged suggest that while MorphGUI successfully enables natural language interface customization, additional scaffolding could further improve the user experience, particularly during initial system encounters.

6. Discussion

The findings from our user study indicate that the presented approach to natural language-based UI customization holds promise in reducing the complexity and effort required for non-technical users to adapt interfaces to their individual needs. Participants demonstrated that, with minimal familiarization, they could specify desired changes to a calendar interface using natural language instructions. Although performance levels were generally high across both simple and complex tasks, subtle differences in task completion rates and the qualitative feedback highlight areas where additional scaffolding and iterative refinements are needed.

From the quantitative metrics collected, we observed a modest decrease in performance as the complexity of the customization tasks increased. This suggests that while users are capable of transferring learned interaction patterns to more challenging scenarios, an optimal prompt design or interface guidance mechanism could further mitigate performance decrements under more demanding conditions. The prompt analysis revealed no significant correlation between prompt length and outcome quality. This result is somewhat encouraging, as it suggests that users do not necessarily need extensive or highly detailed descriptions to achieve satisfactory results. However, participants frequently expressed a desire for examples and structured hints, indicating that guidance on effective prompt construction remains a key area for improvement.

Qualitative feedback from the interviews reinforced the notion that the initial learning curve could be eased by integrating example prompts and visual cues directly into the interface. Participants also emphasized the importance of being able to view and revise their instructions easily, suggesting that clearer input fields, preview functionalities, or step-by-step prompts could enhance the user experience. Furthermore, although the system was generally well-received and considered intuitive over time, users felt that more explicit instructions on how to engage with the interface could accelerate the learning process.

In considering how this approach compares to existing solutions, it is instructive to contrast natural language-based customization with traditional manual configuration tools or WYSIWYG editors. While these conventional approaches often provide a set of well-defined interaction patterns and immediate visual feedback, they may require users to navigate through structured menus, toolbars, and configuration options. Such methods can be efficient for straightforward changes but may limit the creativity and expressiveness of user modifications. In contrast, natural language interaction allows users to describe their desired changes more freely, potentially enabling more nuanced or extensive modifications without requiring technical skills. Although the present study did not explicitly compare performance against these traditional approaches, incorporating such a comparison in future research could illuminate differences in time-to-completion, success rates, and perceived user satisfaction. Highlighting these distinctions would help clarify the unique value that a natural language-driven framework offers, particularly for users who prefer a more flexible and intuitive interaction paradigm.

Our analysis of template usage patterns reveals an interesting cognitive model in how users approach natural language interface customization. The tendency to place styling instructions in the functional context ("What it should do" field) suggests that users naturally conceptualize interface modifications as actions rather than declarative specifications. This finding aligns with previous research on end-user programming, where novice users often think in terms of imperative commands rather than declarative descriptions. The lower error rate in functional specifications suggests that users more readily understand the separation of behavioral aspects, while visual modifications present a more nuanced cognitive challenge.

The lack of correlation between LLM familiarity and task performance is particularly noteworthy, as it suggests that MorphGUI successfully abstracts the complexity of language model interaction. This democratization of interface customization indicates that the system effectively bridges the gap between natural language understanding and interface modification, regardless of users' prior exposure to language models. This finding supports our goal of making interface customization more accessible to non-technical users through natural language interaction.

These observations align with prior work on user-driven customization and prompt engineering. The combination of a free-form natural language approach with structured input fields is a step forward compared to systems that rely heavily on predefined customization options or require code-level intervention. Yet, this study underscores that even in a natural language setting, ensuring that users feel adequately supported remains paramount. Ultimately, enhancing these supportive measures may lead to a more seamless and approachable customization paradigm, increasing overall user acceptance and broadening the spectrum of potential adopters.

7. Conclusion

This paper introduced MorphGUI, a framework leveraging large language models to support natural language-driven interface customization. By unifying user intent, component-level customization directives, and dynamic code generation, MorphGUI enables users to adapt complex interfaces without requiring specialized technical knowledge. The user study

demonstrated that participants could successfully modify a calendar application to meet specific target designs by expressing their desired changes through natural language instructions. The results suggest that this approach can democratize the process of interface personalization, making it accessible to a wider range of users. However, several areas deserve further attention. Improved guidance, such as integrated example prompts or more explicit instructions, may help users quickly grasp the system's capabilities. Additionally, exploring the performance of this method across diverse application domains, device types, and user populations would help validate its scalability and robustness. In conclusion, MorphGUI highlights the potential of bridging large language models and UI customization, contributing to a richer, more inclusive vision of interactive experiences. By refining the system to better support user needs, we take another step toward empowering individuals to shape their digital environments through natural, intuitive means.

References

[1] S. S. Sundar, S. S. Marathe, Personalization versus customization: The importance of agency, privacy, and power usage, Human Communication Research 36 (3) (2010) 298–322.

[2] M. Nebeling, M. Speicher, M. C. Norrie, Crowdadapt: enabling crowdsourced web page adaptation for individual viewing conditions and preferences, in: Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems, ACM, 2013, pp. 23–32.

[3] A. Jameson, Adaptive interfaces and agents, in: The human-computer interaction handbook, CRC press, 2007, pp. 459–484.

[4] S. Marathe, S. S. Sundar, What drives customization? control or identity?, in: Proceedings of the SIGCHI conference on human factors in computing systems, 2011, pp. 781–790.

[5] S. S. Sundar, Self as source: Agency and customization in interactive media, Routledge (2008).

[6] W. E. Mackay, Triggers and barriers to customizing software, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 1991, pp. 153–160.

[7] F. Paternò, End user development: Survey of an emerging field for empowering people, International Scholarly Research Notices 2013 (2013).

[8] L. Reynolds, S. Wiseman, D. Sharma, R. Bendor, Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts, arXiv preprint arXiv:2305.14724 (2023).

[9] J. Hussain, A. U. Hassan, H. S. M. Bilal, R. Ali, M. Afzal, S. Hussain, J. Bang, O. Banos, S. Lee, Model-based adaptive user interface based on context and user experience evaluation, Journal on Multimodal User Interfaces 12 (1) (2018) 1–16.

[10] K. Z. Gajos, M. Czerwinski, D. S. Tan, D. S. Weld, Predictability and accuracy in adaptive user interfaces, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2008, pp. 1271–1274.

[11] K. Gajos, D. S. Weld, Supple: Automatically generating user interfaces, in: Proceedings of the 9th international conference on Intelligent user interfaces, ACM, 2004, pp. 93–100.

[12] K. Z. Gajos, D. S. Weld, J. O. Wobbrock, Automatically generating personalized user interfaces with supple, Artificial Intelligence 174 (12-13) (2010) 910–950.

[13] K. Reinecke, A. Bernstein, Improving performance, perceived usability, and aesthetics with culturally adaptive user interfaces, ACM Transactions on Computer-Human Interaction (TOCHI) 18 (2) (2011) 8.

[14] W. E. Mackay, Patterns of sharing customizable software, Proceedings of the 1990 ACM conference on Computer-supported cooperative work (1991) 209–221.

[15] T. Page, Personalisation of web interfaces: A study of user preferences and implications for user interface design, Journal of Design Research 14 (1) (2016) 22–53.

[16] L. Findlater, J. McGrenere, Preferences for interface adaptations: Understanding individual differences in customization behavior, ACM Transactions on Computer-Human Interaction 16 (3) (2009) 1–31.

[17] K. Z. Gajos, J. O. Wobbrock, D. S. Weld, Automatically generating personalized user interfaces with supple, in: Proceedings of the 9th international conference on Intelligent user interfaces, 2007, pp. 93–100.

[18] A. Cockburn, P. Quinn, C. Gutwin, Understanding and developing user interfaces that respect the user's limited attention, Communications of the ACM 57 (6) (2014) 76–80.

[19] N. Bila, T. Ronda, I. Mohomed, K. N. Truong, E. De Lara, Pagetailor: reusable end-user customization for the mobile web, in: Proceedings of the 5th international conference on Mobile systems, applications and services, ACM, 2007, pp. 16–29.

[20] A. Bunt, C. Conati, J. McGrenere, Supporting interface customization using a mixed-initiative approach, in: Proceedings of the 12th international conference on Intelligent user interfaces, 2007, pp. 92–101.

[21] J. Häkkilä, C. Chatfield, Personal customisation of mobile phones: a case study, in: Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles, ACM, 2006, pp. 409–412.

[22] M. d. Q. Proença, V. G. Motti, K. R. d. H. Rodrigues, V. P. d. A. Neris, Coping with diversity - a system for end-users to customize web user interfaces, Proceedings of the ACM on Human-Computer Interaction 5 (EICS) (2021) 1–27.

[23] K. Z. Gajos, K. Chauncey, The influence of personality traits and cognitive load on the use of adaptive user interfaces, in: Proceedings of the 22nd International Conference on Intelligent User Interfaces, ACM, 2017, pp. 301–306.

[24] R. Kumar, J. O. Talton, S. Ahmad, S. R. Klemmer, Bricolage: example-based retargeting for web design, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2011) 2197–2206.

[25] K. Todi, G. Bailly, L. Leiva, A. Oulasvirta, Adapting user interfaces with model-based reinforcement learning, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–13.

[26] K. Todi, J. Jokinen, K. Luyten, A. Oulasvirta, Familiarisation: Restructuring layouts with visual learning models, in: 23rd International Conference on Intelligent User Interfaces, 2018, pp. 547–558.

[27] M. Nebeling, M. C. Norrie, Responsive design and development: methods, technologies and current issues, in: Web Engineering: 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings 13, Springer, 2013, pp. 510–513.

[28] M. Nebeling, M. Speicher, M. Norrie, W3touch: metrics-based web page adaptation for touch, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013, pp. 2311–2320.

[29] J. O. Wobbrock, S. K. Kane, K. Z. Gajos, S. Harada, J. Froehlich, Ability-based design: Concept, principles and examples, ACM Transactions on Accessible Computing (TACCESS) 3 (3) (2011) 1–27.

[30] A. K. Dey, G. D. Abowd, D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human–Computer Interaction 16 (2-4) (2001) 97–166.

[31] Adobe, Adobe DreamWeaver (2023).
URL `https://www.adobe.com/products/dreamweaver.html`

[32] Bootstrap (2023).
URL `https://getbootstrap.com/`

[33] S. Krings, E. Yigitbas, I. Jovanovikj, S. Sauer, G. Engels, Development framework for context-aware augmented reality applications, in: Companion Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2020, pp. 1–6.

[34] F. Paterno', C. Santoro, L. D. Spano, Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments, ACM Transactions on Computer-Human Interaction (TOCHI) 16 (4) (2009) 1–30.

[35] G. Mori, F. Paterno, C. Santoro, Design and development of multidevice user interfaces through multiple logical descriptions, IEEE Transactions on Software Engineering 30 (8) (2004) 507–520.

[36] K. Gajos, D. S. Weld, Preference elicitation for interface optimization, in: Proceedings of the 18th annual ACM symposium on User interface software and technology, 2005, pp. 173–182.

[37] Y. Cheng, Y. Yan, X. Yi, Y. Shi, D. Lindlbauer, Semanticadapt: Optimization-based adaptation of mixed reality layouts leveraging virtual-physical semantic connections, in: The 34th Annual ACM Symposium on User Interface Software and Technology, 2021, pp. 282–297.

[38] D. Lindlbauer, A. M. Feit, O. Hilliges, Context-aware online adaptation of mixed reality interfaces, in: Proceedings of the 32nd annual ACM symposium on user interface software and technology, 2019, pp. 147–160.

[39] J. M. B. Evangelista, M. N. Lystbæk, A. M. Feit, K. Pfeufer, P. Kán, A. Oulasvirta, K. Grønbæk, Auit–the adaptive user interfaces toolkit for designing xr applications, in: Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology, 2022, pp. 1–16.

[40] SimilarWeb, Stylish - custom themes for any website, `https://userstyles.org/help/stylish_chrome`, accessed: 2022-03-05 (2022).

[41] J. Biniok, Tampermonkey, `https://www.tampermonkey.net`, accessed: 2022-03-05 (2022).

[42] D. S. Weld, C. Anderson, P. Domingos, O. Etzioni, K. Gajos, T. Lau, S. Wolfman, Automatically personalizing user interfaces, IJCAI 3 (2003) 1613–1619.

[43] S. R. Herring, C.-C. Chang, J. Krantzler, B. P. Bailey, Getting inspired! understanding how and why examples are used in creative design practice, in: Proceedings of the SIGCHI conference on human factors in computing systems, 2009, pp. 87–96.

[44] R. Kumar, A. Satyanarayan, C. Torres, M. Lim, S. Ahmad, S. R. Klemmer, J. O. Talton, Webzeitgeist: design mining the web, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013, pp. 3083–3092.

[45] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, R. Kumar, 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, 2017, pp. 845–854.

[46] S. Bunian, K. Li, C. Jemmali, C. Harteveld, Y. Fu, M. S. El-Nasr, Vins: Visual search for mobile user interface design, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–14.

[47] B. Lee, S. Srivastava, R. Kumar, R. Brafman, S. R. Klemmer, Designing with interactive example galleries, in: Proceedings of the SIGCHI conference on human factors in computing systems, 2010, pp. 2257–2266.

[48] F. Huang, J. F. Canny, J. Nichols, Swire: Sketch-based user interface retrieval, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 2019, pp. 1–10.

[49] P. O'Donovan, A. Agarwala, A. Hertzmann, Designscape: Design with interactive layout suggestions, in: Proceedings of the 33rd annual ACM conference on human factors in computing systems, 2015, pp. 1221–1224.

[50] K. Todi, D. Weir, A. Oulasvirta, Sketchplore: Sketch and explore with a layout optimiser, in: Proceedings of the 2016 ACM conference on designing interactive systems, 2016, pp. 543–555.

[51] A. Swearngin, C. Wang, A. Oleson, J. Fogarty, A. J. Ko, Scout: Rapid exploration of interface layout alternatives through high-level design constraints, in: Proceedings of the 2020 CHI conference on human factors in computing systems, 2020, pp. 1–13.

[52] N. R. Dayama, K. Todi, T. Saarelainen, A. Oulasvirta, Grids: Interactive layout design with integer programming, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–13.

[53] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain of thought prompting elicits reasoning in large language models, arXiv preprint arXiv:2201.11903 (2022).

[54] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[55] Y. Lu, A. Leung, A. Swearngin, J. Nichols, T. Barik, Misty: Ui prototyping through interactive conceptual blending (2024). `arXiv:2409.13900`.
URL `https://arxiv.org/abs/2409.13900`

[56] J. Yang, et al., Reactgenie: A development framework for complex multimodal interactions using large language models, Proceedings of the CHI Conference on Human Factors in Computing Systems 1 (2024) 1–23.

[57] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A prompt pattern catalog to enhance prompt engineering with chatgpt, arXiv preprint arXiv:2302.11382 (2023).

[58] M. Suzgun, M. Scales, J. Ni, E. Wang, B. McArthur, M. Shridhar, N. Schärli, J. Schulman, Promptbreeder: Self-referential self-improvement via prompt evolution, arXiv preprint arXiv:2309.16797 (2023).

[59] Q. Liu, X. Chen, H. Deng, A systematic survey of prompt engineering, arXiv preprint arXiv:2307.05242 (2023).

[60] J. Wu, S. Wang, S. Shen, Y.-H. Peng, J. Nichols, J. P. Bigham, Webui: A dataset for enhancing visual ui understanding with web semantics, Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (2023) 1–14.

[61] F. Huang, G. Li, X. Zhou, J. F. Canny, Y. Li, Creating user interface mockups from high-level text descriptions with deep-learning models, arXiv preprint arXiv:2110.07775 (2021).

[62] Meta, React - a javascript library for building user interfaces, accessed: 2024-12-09 (2024).
URL `https://react.dev`

[63] Babel Team, Babel - the compiler for next generation javascript, accessed: 2024-12-09 (2024).
URL `https://babeljs.io`