MorphGUI: Real-time GUIs Customization with Large Language Models

Tommaso Calò[1,*], Andrea Sillano[1], Luigi De Russis[1]

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

Abstract

Graphical user interface (GUI) customization relies on predefined configuration options and settings, constraining diverse individual needs and preferences within predetermined boundaries and often requiring technical expertise. To address these limitations, this work introduces MorphGUI, a framework leveraging Large Language Models (LLMs) to enable interface customization through natural language. By allowing users to express desired changes using their own words and harnessing the generative capabilities of LLMs, MorphGUI mitigates the limitations of predefined options and reduces the need for technical expertise. The framework translates functional and stylistic requests into either modifications of existing application components or generation of new ones. Through a use case implementation with a calendar application and a user study (n=18), where participants were tasked with modifying interfaces towards a target goal, we investigate if MorphGUI can enable effective natural language-driven interface customization for non-expert users through both functional and visual modifications. Results show that participants successfully customized interfaces using natural language. Users found the system intuitive and achieved good performance regardless of technical background, we report analysis of optimal prompt length, challenges in separating functional and visual instructions in structured templates, correlation between LLM experience and success, and learning effects. The study revealed opportunities for enhanced guidance, examples, and scaf-

*Corresponding author

Email addresses: tommaso.calo@polito.it (Tommaso Calò), andrea.sillano@polito.it (Andrea Sillano), luigi.derussis@polito.it (Luigi De Russis)

folding to help users structure their customization requests more effectively.

Keywords: User Interface Customization, Natural Language Interaction, Large Language Models, Interactive Systems, UI Personalization, Interface Generation

1. Introduction

Traditional one-size-fits-all graphical user interfaces (GUIs) fail to accommodate the diverse contexts and needs of individual users, often resulting in suboptimal user experiences [1]. Personalization refers to the process of tailoring a user interface (UI) to better meet the specific needs, preferences, or behaviors of individual users. Users' personalization needs are highly individual and context-dependent [2]. Existing work showed the benefits of having UIs adapted to people's needs [3], characteristics [4], or cultural backgrounds [5]. In addition, studies of interface personalization behavior reveal that users often want to modify interfaces in ways not anticipated by designers [6, 7]. For instance, users might want to reorganize interface elements based on their workflow [8] or adjust visual properties for better accessibility [9].

Personalization can take two primary forms: customization and adaptation. Customization refers to *explicit personalization*, where users actively modify the interface to suit their specific needs and preferences. In contrast, adaptation involves *implicit personalization*, where the system automatically adjusts the interface based on user behavior, preferences, or context. While adaptation can enhance usability through automation, customization stands out for its ability to improve user experience [3] and to allow people to feel in control, express their identity, or create a connection with GUIs [10, 11, 12, 13]. Recent work on natural language interaction for interface customization, such as Stylette [14], has shown promise but faces challenges with ambiguous user specifications and is limited to a fixed set of styling attributes, preventing users from realizing more complex interface modifications or generating new components. As such, solutions are needed to empower users to personalize graphical user interfaces (GUIs) without requiring extensive technical knowledge or effort, while providing the freedom to realize their true customization intentions [15] without being restricted to a limited set of predefined customization options.

In this work, we present MorphGUI[1], a framework that combines traditional GUI controls (colors, fonts, text size, and accessibility settings) with LLM-powered customization, unlike existing approaches that rely solely on predefined customization options or user models. Through a structured input approach, the LLM-powered customization disentangles stylistic ("how it should appear") from functional ("what it should do") modifications to guide users through the customization process and allows targeting either specific components or the overall interface. To evaluate MorphGUI's effectiveness, we conduct a within-subjects experiment with 18 participants using a calendar application where participants customize interfaces toward specific target designs. In particular, we investigate (RQ1) can users achieve their intended interface modifications across different complexity levels? (RQ2) what is the relationship between prompt characteristics, interaction patterns, and successful customization outcomes? and (RQ3) how do users perceive system usability through both quantitative measures and qualitative feedback? Our results show that participants achieved substantial target adherence across both simple and complex customization tasks, with performance remaining consistent regardless of participants' prior technical experience. Users successfully completed interface modifications using natural language instructions, developing strategies for iterative refinement. While participants rated the system as acceptably usable, qualitative feedback highlighted challenges in formulating effective prompts, with users expressing a need for clearer guidance on how to structure their customization requests to achieve intended outcomes.

The primary contributions of this work are: (1) MorphGUI, a novel framework that leverages Large Language Models to enable natural language-driven interface customization. The framework employs a structured dual-input approach that separates functional requirements from visual specifications, thereby reducing ambiguity in natural language customization requests; (2) a comprehensive user study (n=18) demonstrating that non-expert users can effectively customize interfaces through natural language instructions; and (3) empirical insights into factors influencing customization success and user experience in LLM-powered interface personalization systems.

[1]An interactive demonstration of MorphGUI is available at `https://elite.polito.it/research/initiatives/morphgui`

2. Related Work

Current approaches to interface personalization can be categorized into three main strategies: customization, where users manually modify the interface to suit their preferences, giving them full control but often requiring expertise, effort, and technical knowledge [7, 16, 17, 3, 18]; adaptation, where the system automatically adjusts the interface by analyzing user behavior and context, which can reduce effort for users but may result in changes that feel unpredictable or undesirable [9, 19, 20, 2]; and hybrid approaches, which combine user-driven control with system-generated suggestions [21, 22], but can struggle to accommodate complex and evolving personalization needs.

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 [23], user abilities [24, 25], context [26], preferences [9, 21], and tasks [9]. Implementation methods range from preprogrammed frameworks [27, 28] to context-aware [26] and mixed-reality interfaces [29]. 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 [30] and multi-device computing [31]. In contrast, optimization-based techniques like SUPPLE [9] and ARNAULD [32] automatically generate layouts by optimizing interface configurations based on user preferences and abilities. Recent work has extended these concepts using semantic properties [33, 34] and developer-defined objective functions [35].

Additionally, recent studies have focused on data-driven and behavior-driven UI personalization. Todi et al. [21] leveraged model-based reinforcement learning to optimize interface layouts adaptively, while Vidmanov and Alfimtsev [36] proposed a multi-agent reinforcement learning framework where each UI widget acts as an agent, learning adaptations based on a usability-oriented reward model. These approaches exemplify the integration of advanced machine learning techniques for real-time, user behavior-aware personalization.

Customization allows users to manually personalize the interface by providing direct control over various options, such as rearranging layout elements, resizing components, changing visual properties like colors and fonts, or hiding unnecessary features. CrowdAdapt [3] is 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 freedom. This is done at the expense of more time invested by users, which may, however, outweigh customization benefits [17, 37]. Therefore, enabling rapid and effortless customization becomes crucial for encouraging users to personalize their interfaces. Sundar and Marathe [13] 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 [38] and Tampermonkey [39] enable CSS and JavaScript customization but require coding knowledge. In addition, traditional customization approaches typically constrain users to predefined options [6] or require them to learn complex configuration languages [40]. While these methods provide some flexibility, they often fail to capture users' actual adaptation needs and mental models [8].

Recent advances in natural language processing and LLMs offer new possibilities for interface customization [41, 42, 43]. Instead of learning complex configuration options or relying on system-driven customization, users can express their customization preferences in natural language [44, 45]. For instance, Li et al. [46] showed that off-the-shelf LLMs can be prompted to directly apply interface changes across webpages based on natural language instructions, automatically identifying relevant UI components and modifying them as required. Vaithilingam et al. [47] introduced DynaVis, which combines NL commands with dynamically synthesized UI widgets for visualization editing, offering the flexibility of NL input with immediate GUI feedback. Similarly, Zhang et al. [48] proposed NLDesign, a tool that empowers designers to control UI element layout and properties through natural language commands, including the definition of interaction logic. Although these systems highlight a clear trend toward natural, language-based end-user control, they are mostly oriented toward design prototyping and did not explore interactive, real-time end-user driven interface customization.

Close to our work, Stylette [14] allows website customization using natural language. Users express their goals in free-form voice commands and interact with a set of design alternatives presented by the system. While this work shows the promise of natural language for interface customization, Stylette faces challenges with ambiguous user specifications and focuses solely on styling modifications, being limited to a fixed set of CSS attributes,

preventing users from realizing more complex interface modifications. Their system does not support functional modifications or the creation of new interface components.

Our work builds on these foundations and addresses the identified gaps in several key ways. First, unlike tools oriented toward design prototyping, MorphGUI is designed for real-time, interactive customization by end-users. Second, while prior systems such as Stylette focus exclusively on stylistic modifications, our framework supports both visual and functional changes, allowing users to alter not only the appearance but also the behavior of interface components. Third, prior work often lacks comprehensive evaluations focused on the practical usability of these tools for non-technical end-users. We address this gap by conducting a user study with this specific demographic, demonstrating that the generative power of LLMs can move beyond fixed attribute sets and empower users to achieve complex customizations that more closely match their intent.

3. System Implementation

3.1. Use Case

Consider Sarah, a content manager who frequently uses a calendar application to schedule and track team meetings. While the application offers standard customization settings for colors, fonts, and basic layout options, Sarah finds herself unable to adapt the interface to her team's specific workflow needs. She wants to modify how meetings are displayed to highlight upcoming deadlines and distinguish between different project categories.

Initially, Sarah attempts to use the traditional GUI controls, adjusting color schemes and text sizes through predefined settings. However, she realizes these static options cannot achieve her desired outcome: combining deadline information with meeting titles and applying conditional formatting based on project types. Using MorphGUI's structured input approach, Sarah first selects "Global GUI" modification and specifies in the "What it should do" field: "Show meeting deadlines next to titles and group events by project category." In the "How it should appear" field, she describes: "Use bold red text for urgent deadlines and apply different background colors for each project category." Through this natural language interaction, MorphGUI generates a new calendar view that integrates deadline information with meeting displays and implements the requested visual categorization. Sarah then selects the specific header component to add a new element,

describing in the functional field: “Add a color legend showing project categories” and in the visual field: “Display as a horizontal strip at the top with colored boxes and category labels.” The structured input fields guide her through each modification while maintaining the consistency of the overall interface.

This use case demonstrates how MorphGUI extends beyond traditional static customization options by allowing users to express and implement more complex, context-specific interface modifications through natural language instructions, at both global and component-specific levels.

3.2. 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 system is implemented using a Node.js server with Express.js framework for HTTP request handling and WebSocket support for real-time updates. The server maintains a PostgreSQL database for storing user preferences, interface versions, and component configurations, enabling persistent storage and version control functionality. The AI component, implemented using OpenAI’s GPT-4o model [2] with temperature=0.7 and max_tokens=4096, 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 al-

[2] version GPT-4o-2024-05-13

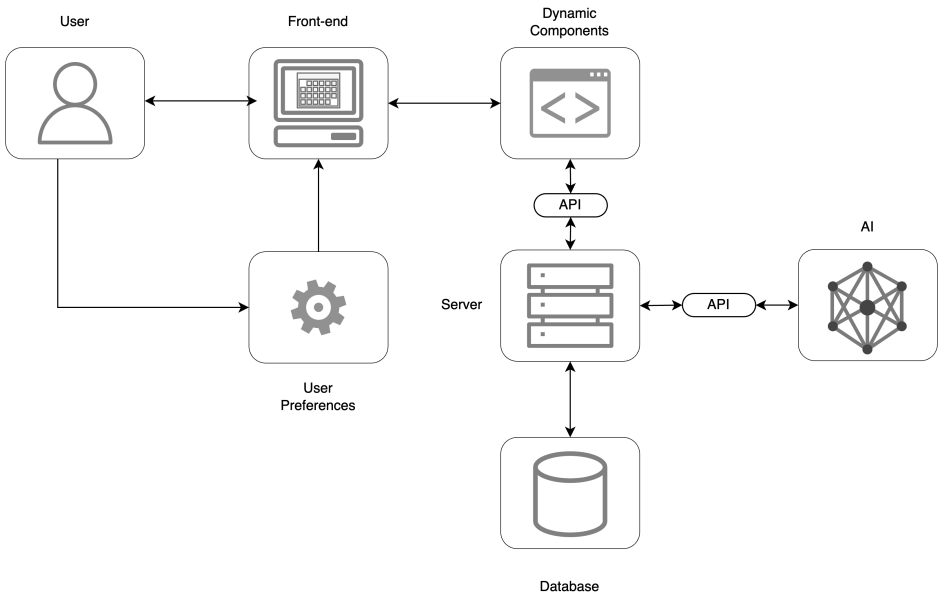

Figure 1: System architecture showing the interaction between user interface, server components, and AI services

lows users to track and revert changes as needed. This versioning system provides a safety net for users experimenting with interface modifications.

3.3. MorphGUI Customization Interface

MorphGUI's customization interface, shown in Figure 2, consists of two main sections: traditional static settings and natural language-driven customization. Through the "Static Settings" dropdown, users can access conventional GUI controls for basic modifications like colors, fonts, and layout options. The "Dynamic Settings" section enables more complex customizations through natural language input.

LLMs approach opens new challenges in ensuring consistent interpretation of user intent and maintaining interface usability across customizations [49]. 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 [50, 51]. 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 [52, 53, 54].

Our framework builds on these insights by implementing a natural language interface where users can operate at different levels of customization. In Figure 2, under "Personalize," users can either select existing interface elements for modification, add new components, or choose "Global GUI" to refine the entire interface. For any selected scope, users express their desired changes through two separate input fields specifying their 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. Our choice of two separate input boxes over a unified field is guided by recent literature on end-user prompting and LLM-assisted design [50, 55]. This approach supports more efficient and less error-prone interaction by providing distinct contexts for functionality and appearance. Separating these inputs helps both the user to articulate their intent more clearly and the system to more reliably process behavioral changes versus visual modifications. This dual-field approach helps users articulate their preferences 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.

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.

3.4. 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 [56] 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

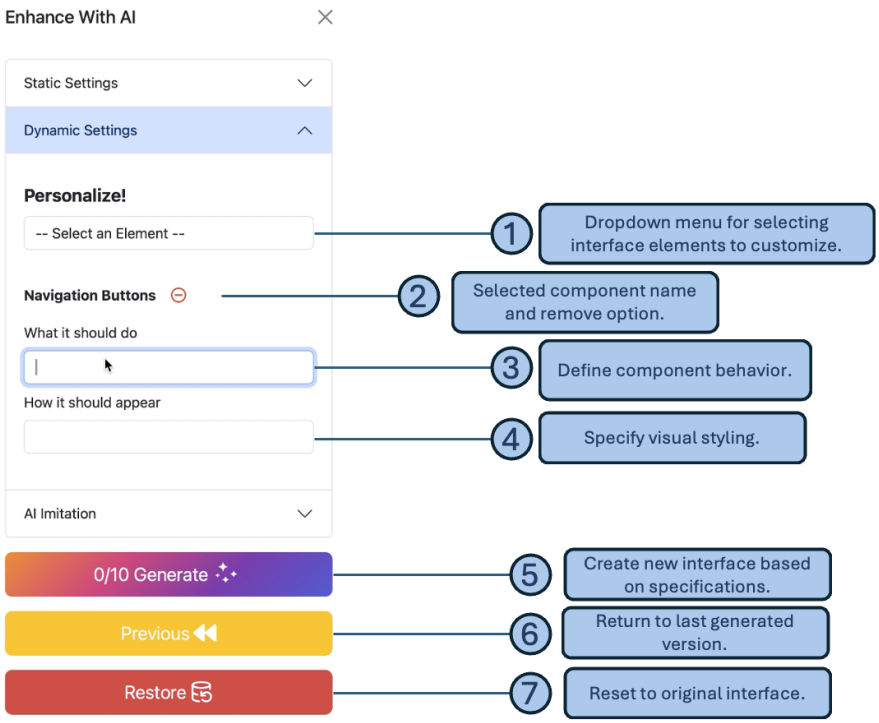

Figure 2: Natural language customization interface showing component selection and preference input fields

integration with the application's existing component ecosystem. The component employs Babel [57] 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 con-

verts 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.

3.5. Broader Applicability

MorphGUI's applicability can be extended beyond the calendar use case to other common web and application design patterns. For example:

Dashboard Widgets: For dashboard interfaces, users can customize data visualization components through commands like "Show the revenue chart as a donut instead of bars" in the functional field and "Use blue gradients with white text labels" in the visual field. MorphGUI's component-level targeting allows users to modify specific widgets while maintaining the overall dashboard layout and data connections.

E-commerce Product Cards: In e-commerce interfaces, users might specify "Display user ratings as stars below the price" functionally and "Use larger product images with rounded corners" visually. The system can generate modified product card components that maintain essential e-commerce functionality (add to cart, product links) while implementing the requested visual and layout changes.

Multi-step Form Wizards: For complex forms, users can request functional modifications like "Add a progress indicator showing completed steps" and visual changes such as "Highlight the current step with a colored background and bold text." MorphGUI's structured approach ensures that form validation and navigation logic remain intact while implementing the customization requests.

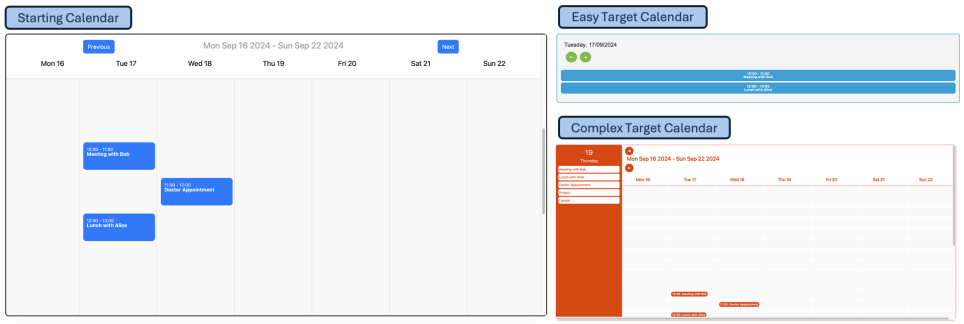

Figure 3: Calendar interface examples showing the starting calendar (left) and target calendars for easy (top right) and complex (bottom right) customization tasks.

To demonstrate these capabilities, an interactive demonstration is available at `https://elite.polito.it/research/initiatives/morphgui`, where readers can explore the calendar application used in our study and experiment with customizations across the design patterns discussed above (dashboard widgets and e-commerce product cards). Future work will evaluate MorphGUI's effectiveness across these diverse interface patterns to establish its generalizability as a universal natural language customization framework.

4. Evaluation

4.1. Study Design

To evaluate MorphGUI, 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. We employed a within-subjects design to maximize statistical power and control for individual differences in technical background, LLM experience, and customization preferences. This design provides greater sensitivity to detect effects of interface complexity compared to between-subjects designs, which is particularly important given our focus on understanding how task complexity affects user performance. All participants completed tasks in the same order (simple task first, then complex task) to ensure consistent learning opportunities and eliminate confounding effects from task sequence variations. We used a fixed target layout rather

than free customization to provide objective performance measures through target adherence scoring. 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 feedbacks, focusing on participants' experiences with the 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. Complete qualitative questionnaires used in the study are provided in Appendix A.

4.2. Participants

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

ID (Age)	Profession	LLM Usage Exp.*	UI Development Exp.**
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,

We conducted the study with 18 participants recruited using a combination of snowball and convenience sampling. Table 1 presents the demographic distribution of our sample across age, professional background, and experience with using 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 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 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 Scal (SUS) questionnairee [58] and provided open-ended feedback about their experience with the system. This feedback 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. Measures and Analyses

During the study, we collected several measurements to answer our research questions. To understand the factors influencing user success (RQ2), 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 logs were analyzed to compute mean session duration, average number of generation attempts, and frequency of resets and reversions, which were compared between tasks using paired t-tests.

To assess usability and gather qualitative feedback (RQ3), participants completed the System Usability Scale (SUS) questionnaire and answered open-ended questions about their customization experience. SUS scores were processed following standard procedures and compared against established benchmarks. The qualitative feedback was thematically analyzed to identify common patterns regarding customization strategies, challenges, and suggestions for improvement.

To evaluate user performance and the effects of interface complexity (RQ1), we assessed the final generated interfaces against the target designs using a structured assessment framework. This framework examined layout accuracy, component positioning, color scheme implementation, navigation functionality, and event display formatting. These scores were aggregated into a target adherence score. We computed mean scores and standard deviations for each component and the overall score. Paired t-tests were used to compare performance between simple and complex tasks, while Pearson correlation coefficients were calculated to examine relationships between participant characteristics and customization success. For categorical variables, we employed Wilcoxon rank-sum tests to analyze differences in performance across groups.

5. Results

This section presents our findings organized around the three central research questions established in our methodology.

5.1. RQ1: User Performance and Interface Complexity Effects

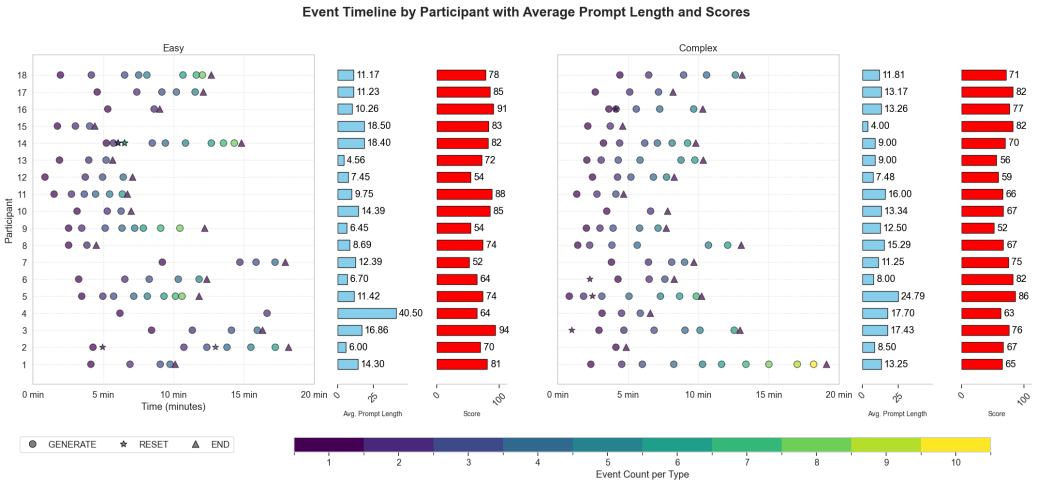

Figure 4: Event Timeline by Participant with Average Prompt Length and Scores.

RQ1 investigated the extent to which non-expert users can steer MorphGUI toward a target interface and how task complexity affects their performance. To answer this, we analyzed target adherence scores and task completion times.

The findings indicate that participants were largely successful in using MorphGUI for interface customization. On average, they achieved a target adherence score of M=74.7% (SD=12.9%) on the simple task and M=70.2% (SD=9.6%) on the complex task. These scores demonstrate a proficient level of control, confirming that users, even those without technical expertise, can effectively steer the system to generate desired UI modifications. A paired t-test confirmed that the decrease in adherence score for the complex task was statistically significant ($t(17) = 2.15$, $p < 0.05$), indicating that as interface complexity increases, achieving precise customization becomes more challenging, yet remains high.

Task completion times improved from the simple task (M=16.7 min, SD=7.4) to the complex task (M=13.6 min, SD=4.6), despite the latter's

higher complexity. This suggests that participants became more efficient with the system. Figure 4 provides a detailed overview of individual performance, illustrating these trends across participants. In summary, while complexity introduces challenges, non-expert users can effectively control MorphGUI and become progressively faster at using it.

5.2. RQ2: Factors Influencing Customization Success

RQ2 focused on identifying the factors that influence users' success in customizing the interface. We analyzed system interaction logs, and natural language inputs to uncover usage patterns and determinants of performance.

Generation Patterns and System Interaction: On average, participants used M=5.0 (SD=2.0) generation attempts for the simple task and M=6.0 (SD=1.9) for the complex task, indicating a measured and iterative approach. The low number of reset operations (M=1.1, SD=0.8 per task) compared to the more frequent use of reversions to previous versions (M=2.1, SD=1.2) suggests that participants preferred to refine their designs incrementally rather than starting over. This pattern implies that they found the iterative feedback loop effective for steering the system. Interestingly, we found no significant correlation between the number of generation attempts and the final interface quality (r=.21, $p > .05$), suggesting that success was not determined by sheer persistence but by the quality of interaction.

Natural Language Input Analysis: The analysis of natural language inputs revealed nuanced patterns. While the average prompt length was similar for simple (M=12.7 words, SD=8.1) and complex tasks (M=13.3 words, SD=8.1), we observed a non-linear relationship between prompt length and adherence scores. As illustrated in Figure 5, optimal results were often achieved with prompts between 15-20 words. Shorter prompts tended to lack sufficient detail for the system to act upon, while overly long prompts may have introduced ambiguity or conflicting instructions, leading to lower performance. This finding suggests that prompt conciseness and clarity are more critical factors for success than mere length. Furthermore, a user's technical background showed no significant effect on prompt effectiveness ($\chi^2(2) = 3.42$, $p > .05$), reinforcing the idea that MorphGUI is accessible to non-experts.

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

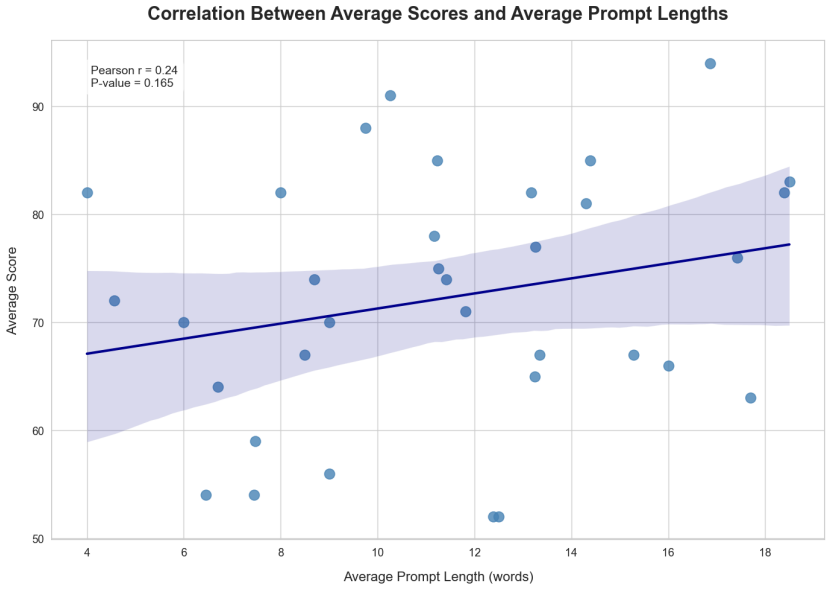

Figure 5: Correlation Between Average Scores and Average Prompt Lengths.

used a confusion matrix approach to quantify the accuracy of template usage, as shown in Figure 6.

Our analysis revealed several key patterns in template usage, as shown in Figure 6. 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 modifications 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

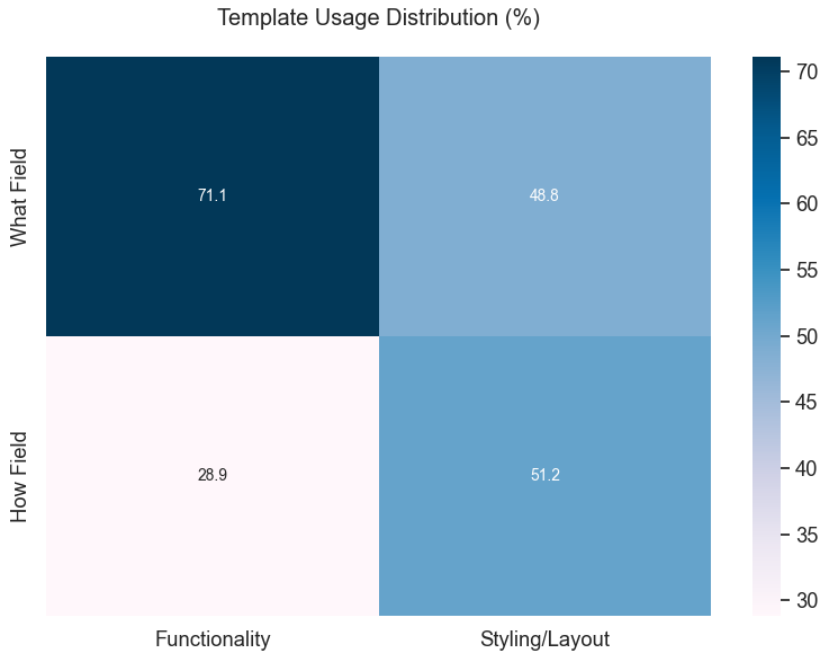

Figure 6: Confusion matrix showing template usage patterns across participants. The matrix tracks: (1) True Functionality (TF): Correct placement of functionality instructions in "What it should do" field; (2) True Styling (TS): Correct placement of styling/layout instructions in "How it should appear" 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.

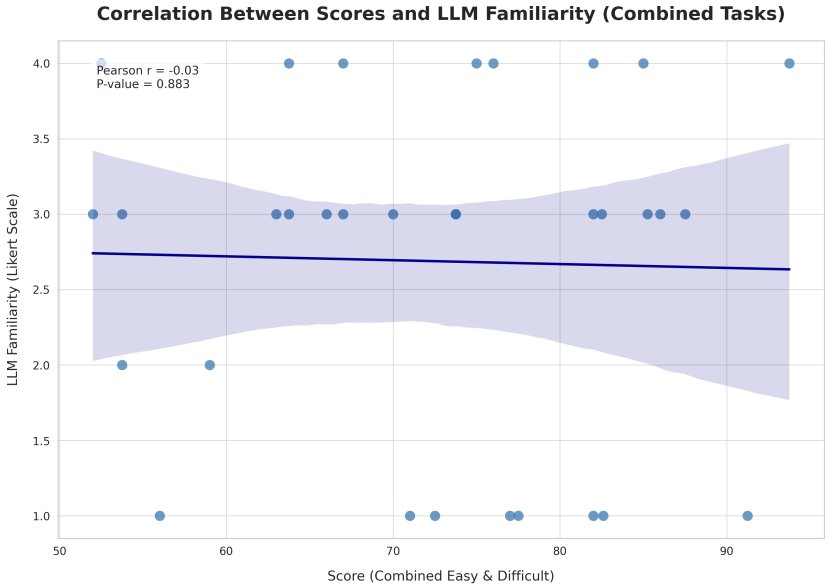

Figure 7: Correlation Between Scores and LLM Usage Experience (Combined Tasks) showing no significant relationship (r = -0.03, p = 0.883)

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.

LLM Experience Impact: Analysis revealed no significant correlation between participants' LLM familiarity and their task performance (r = -0.03, p = 0.883), as shown in Figure 7. This suggests that we cannot conclusively state whether prior experience with language models impacted or did not impact users' ability to effectively customize interfaces using MorphGUI's natural language interface.

5.3. RQ3: User Perceptions and Qualitative Feedback

To answer our third research question regarding user perceptions of MorphGUI's usability and their qualitative feedback, we analyzed SUS scores and open-ended feedback data.

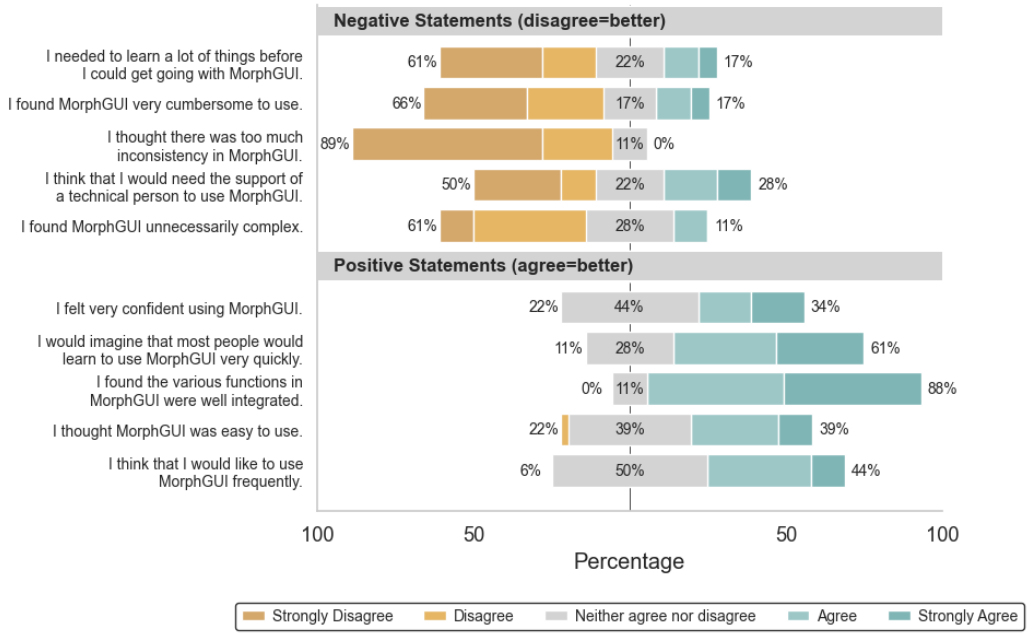

Figure 8: System Usability survey results showing agreement levels with positive and negative statements about MorphGUI (N=18).

The system's usability was found to be acceptable, achieving a mean SUS score of M=68 (SD=12.3). Participants rated the system as "easy to learn" (M=4.2/5) and "well integrated" (M=4.0/5), highlighting its intuitive nature. Lower scores were observed for "technical support needed" (M=2.8/5) and "system complexity" (M=2.6/5), suggesting that while the core system is usable, clearer guidance could reduce the need for support. Details are reported in Figure 8.

Qualitative analysis of participant interviews revealed several themes that provide deeper insights into users' experiences with MorphGUI and opportunities for system enhancement. We grouped the responses by five recurring themes, highlighting 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 success-

fully 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. These paradigms serve complementary purposes: direct manipulation tools provide precise control through well-understood interface metaphors, while natural language approaches offer expressiveness and accessibility through intent-driven interaction. Incorporating such a comparison in future research 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. Limitations

While our study provides valuable insights into natural language-driven interface customization, several limitations should be acknowledged. First, our sample size of 18 participants, while meeting established usability guidelines for SUS evaluation [58], limits our ability to conduct detailed subgroup analyses or detect smaller effect sizes. This constraint affects the generalizability of findings across different user populations and technical backgrounds.

Second, our evaluation focused exclusively on calendar interface customization, which may not fully represent the complexity and diversity of interface types encountered in real-world applications. While we demonstrated broader applicability through design patterns in Section 3, empirical validation across different interface domains (e.g., dashboards, e-commerce, forms) remains to be conducted. The calendar domain's inherent structure and familiar interaction patterns may have influenced user performance in ways that might not transfer to less structured or more novel interface types.

Third, the controlled laboratory setting and specific task scenarios may not fully capture the variability and complexity of authentic customization needs that users encounter in their daily workflows. Participants worked with predefined target interfaces and specific customization goals, which may differ from the more exploratory and iterative customization processes that occur in naturalistic settings.

Finally, our study examined short-term usability and immediate task performance, but did not assess longer-term adoption patterns, learning curves over extended use, or the maintenance of customized interfaces over time. Future work should address these limitations through larger-scale studies, cross-domain validation, longitudinal evaluation, and in-the-wild deployment to establish the broader applicability and sustained effectiveness of natural language-driven interface customization through MorphGUI.

8. 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. Future research could also include comparative evaluations with direct manipulation tools to understand the trade-offs between interaction paradigms, and the integration of automated testing capabilities to verify that customizations maintain functionality and consistency across different scenarios. 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.

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.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

In accordance with the journal's policy on the use of AI and AI-assisted technologies, we provide the following declaration regarding the use of such tools in the preparation of this manuscript.

Use in Manuscript Preparation. Generative AI tools (specifically, large language models) were used solely for language editing and improving readability of the manuscript. These tools helped refine grammar, sentence structure, and clarity in specific passages that had already been drafted by the authors. No AI tools were used to generate new scientific content, perform data analysis, interpret results, formulate arguments, or write original sections of the paper. All research design, data collection, analysis, interpretation of findings, and substantive intellectual contributions were performed exclusively by the human authors.

Use in Code Development. AI-assisted code generation tools were used to support the development of data analysis scripts and visualization code. However, all analytical decisions, statistical interpretations, and validation of results were performed by the authors. The AI tools served as coding assistants under continuous human supervision and verification.

Author Responsibility. The authors have carefully reviewed, edited, and validated all content in this manuscript and take full responsibility for its accuracy, integrity, and originality. The use of AI tools was limited to improving readability and coding efficiency, not for generating substantive scientific content or analysis. All work presented represents the original intellectual contribution of the authors.

References

[1] 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. doi:10.1007/s12193-018-0258-2.

[2] K. Z. Gajos, K. Everitt, D. S. Tan, M. Czerwinski, 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. doi:10.1145/1357054.1357252.

[3] 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, 2013, pp. 23–32. doi:10.1145/2494603.2480304.

[4] K. Gajos, D. S. Weld, Supple: Automatically generating user interfaces, in: Proceedings of the 9th international conference on Intelligent user interfaces, 2004, pp. 93–100. doi:10.1145/964442.964461.

[5] 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) 8:1–8:29. doi:10.1145/1970378.1970382.

[6] W. E. Mackay, Patterns of sharing customizable software, in: Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work, ACM, 1991, pp. 209–221. doi:10.1145/99332.99356.

[7] 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. doi:10.1504/JDR.2016.076309.

[8] L. Findlater, J. McGrenere, Preferences for interface adaptations: Understanding individual differences in customization behavior, ACM Transactions on Computer-Human Interaction 16 (1) (2009) 1–44. doi:10.1145/1502800.1502801.

[9] 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. doi:10.1145/1216295.1216313.

[10] A. Jameson, Adaptive interfaces and agents, in: The Human-Computer Interaction Handbook, CRC Press, 2007, pp. 459–484. doi:10.1201/9781410615862.ch22.

[11] S. Marathe, S. S. Sundar, What drives customization? control or identity?, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2011, pp. 781–790. doi:10.1145/1978942.1979056.

[12] S. S. Sundar, Self as source: Agency and customization in interactive media, Routledge, 2008, pp. 58–74.

[13] 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. doi:10.1111/j.1468-2958.2010.01377.x.

[14] T. S. Kim, D. Choi, Y. Choi, J. Kim, Stylette: Styling the web with natural language, in: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, CHI '22, Association for Computing Machinery, New York, NY, USA, 2022. doi:10.1145/3491102.3501931. URL https://doi.org/10.1145/3491102.3501931

[15] F. Paternò, End user development: Survey of an emerging field for empowering people, International Scholarly Research Notices 2013 (2013) 1–11. doi:10.1155/2013/532659.

[16] 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. doi:10.1145/1247660.1247664.

[17] 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, ACM, 2007, pp. 92–101. doi:10.1145/1216295.1216317.

[18] 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. doi:10.1145/3457151.

[19] 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. doi:10.1145/3025171.3025192.

[20] R. Kumar, J. O. Talton, S. Ahmad, S. R. Klemmer, Bricolage: example-based retargeting for web design, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2011, pp. 2197–2206. doi:10.1145/1978942.1979262.

[21] K. Todi, G. Bailly, L. A. 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. doi:10.1145/3411764.3445497.

[22] K. Todi, J. Jokinen, K. Luyten, A. Oulasvirta, Familiarisation: Restructuring layouts with visual learning models, in: Proceedings of the 23rd International Conference on Intelligent User Interfaces, 2018, pp. 547–558. doi:10.1145/3172944.3172949.

[23] 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, Springer, 2013, pp. 510–513. doi:10.1007/978-3-642-39200-9_47.

[24] 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. doi:10.1145/2470654.2481319.

[25] 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. doi:10.1145/1952383.1952384.

[26] 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. doi:10.1207/S15327051HCI16234_02.

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

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

[29] 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. doi:10.1145/3393672.3398640.

[30] F. Paternò, 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. doi:10.1145/1614390.1614394.

[31] 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. doi:10.1109/TSE.2004.44.

[32] 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. doi:10.1145/1095034.1095063.

[33] Y. Cheng, Y. Yan, X. Yi, Y. Shi, D. Lindlbauer, Semanticadapt: Optimization-based adaptation of mixed reality layouts leveraging virtual-physical semantic connections, in: Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology, 2021, pp. 282–297. doi:10.1145/3472749.3474750.

[34] 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. doi:10.1145/3332165.3347945.

[35] 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. doi:10.1145/3526113.3545651.

[36] D. Vidmanov, A. Alfimtsev, Mobile user interface adaptation based on usability reward model and multi-agent reinforcement learning, Multimodal Technologies and Interaction 8 (4) (2024) 26. doi:10.3390/mti8040026.

[37] 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. doi:10.1145/108844.108867.

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

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

[40] D. S. Weld, C. Anderson, P. Domingos, O. Etzioni, K. Gajos, T. Lau, S. Wolfman, Automatically personalizing user interfaces, IJCAI Proceedings 3 (2003) 1613–1619, available online. doi:10.5555/1630659.1630906.

[41] K. Dong, A. Liu, R. Patel, Widgetcoder: Generating mobile app interfaces from natural language, Proceedings of the ACM on Human-Computer Interaction 8 (CSCW1) (2024) 1–24.

[42] D. Chen, J. Kim, A. Brown, Sketch2code: Converting ui sketches and natural language to code, in: Proceedings of the 2022 International Conference on Software Engineering, IEEE, 2022, pp. 456–467.

[43] W. Zhang, S. Johnson, M. Lee, Ui-bert: Transformer-based models for ui understanding and generation, ACM Transactions on Computer-Human Interaction 30 (3) (2023) 1–31.

[44] 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). doi:10.48550/arXiv.2201.11903.

[45] 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. doi:10.5555/3495724.3495975.

[46] A. Li, J. Wu, J. P. Bigham, Using LLMs to Customize the UI of Webpages (2023). doi:10.1145/3586182.3616671.

[47] P. Vaithilingam, E. L. Glassman, J. P. Inala, C. Wang, DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing (2024). doi:10.1145/3613904.3642639.

[48] T. Zhang, P. Fu, J. Liu, Y. Zhang, X. Chen, NLDesign: A UI Design Tool for Natural Language Interfaces (2024). doi:10.1145/3674399.3674455.

[49] 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). doi:10.48550/arXiv.2302.11382.

[50] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, Q. Yang, Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts, in: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23, Association for Computing Machinery, New York, NY, USA, 2023. doi:10.1145/3544548.3581388.
URL https://doi.org/10.1145/3544548.3581388

[51] 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). doi:10.48550/arXiv.2309.16797.

[52] Q. Liu, X. Chen, H. Deng, A systematic survey of prompt engineering, arXiv preprint arXiv:2307.05242 (2023). doi:10.48550/arXiv.2307.05242.

[53] 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) 1–14doi:10.1145/3544548.3580954.

[54] F. Huang, G. Li, X. Zhou, J. F. Canny, Y. Li, Creating user interface mock-ups from high-level text descriptions with deep-learning models, arXiv preprint arXiv:2110.07775 (2021). doi:10.48550/arXiv.2110.07775.

[55] Y. Lu, A. Leung, A. Swearngin, J. Nichols, T. Barik, Misty: Ui prototyping through interactive conceptual blending, in: Proceedings of the

2025 CHI Conference on Human Factors in Computing Systems, CHI '25, Association for Computing Machinery, New York, NY, USA, 2025. doi:10.1145/3706598.3713924.
URL https://doi.org/10.1145/3706598.3713924

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

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

[58] J. Brooke, Sus: A 'quick and dirty'usability scale, Usability evaluation in industry 189 (194) (1996) 4–7.

Appendix A. Study Script and Questionnaires

This appendix provides the complete study script and questionnaires used in the user study to enhance transparency and reproducibility. The study was conducted remotely via video calls with participants using a calendar application customization system.

Introduction Script. The following introduction was provided to participants at the beginning of each session: "Good morning/afternoon, my name is [Researcher] and I will guide you through today's experiment. Before we begin, I have some information for you. I am working on research regarding the possibility of personalizing the appearance and use of a web application based on personal preferences through the use of generative models. Today we will test its functionality specifically on a calendar application. First, I want to clarify that we are testing the application, not your abilities. The objective is to verify whether the system functions as expected and whether it can satisfy user needs. During the experiment, there are no real errors that can be made. If you have any questions during the experiment, feel free to ask. I can only respond to those that do not involve the application's functionality, as one of the objectives is to verify how users interface with the system without assistance. Do you have any questions so far?"

Demographic Information. Participants provided demographic information through the following questions: (1) "How old are you?" with options: ¡20, 20-30, 30-60, ¿60; (2) "What is your gender?" with options:

Male, Female, Other or prefer not to specify; (3) "What is the highest level of education you have completed?" with options: High school, Bachelor's degree, Master's degree, Doctorate, Other; (4) "Which sector do you work in?" with options: Technology, Design, Marketing, Education, Healthcare, Legal, Other; (5) "How comfortable do you feel using technology (e.g., computers, smartphones, etc.)?" with options: Not comfortable at all, Not very comfortable, Neutral, Quite comfortable, Very comfortable.

Calendar Application Usage. The background questionnaire included calendar-specific questions: (1) "How often do you use calendar applications (e.g., Google Calendar, Apple Calendar)?" with options: Never, Rarely (once a month), Often (once a week), Daily; (2) "What type of calendar application do you usually use?" with options: Google Calendar, Apple Calendar, Microsoft Outlook Calendar, Other, None; (3) "How important is it for you to be able to customize the appearance of your calendar (e.g., colors, fonts)?" with options: Not important at all, Not very important, Neutral, Quite important, Very important.

Large Language Model Experience. Participants answered questions about their LLM familiarity: (1) "How familiar are you with LLMs (e.g., ChatGPT)?" with options: I don't know what they are, I've heard about them but never used them, I've used them a few times, I use them regularly; (2) "Have you ever written prompts to use an artificial intelligence model (e.g., ChatGPT, MidJourney)?" with options: Yes, often, Yes, sometimes, No, never; (3) "If you have written prompts in the past, for what purpose did you use them?" with options: To generate text, To generate images, To automate activities, Other, I have never written prompts; (4) "How comfortable do you feel creating specific prompts to obtain desired results from artificial intelligence?" with options: Not comfortable at all, Not very comfortable, Neutral, Quite comfortable, Very comfortable; (5) "How much do you trust artificial intelligence tools to personalize user experience?" with options: Not at all, A little, Neutral, Quite a bit, Very much; (6) "If you have had experience using LLMs, could you briefly describe how you used them and in what application contexts?" (open-ended response).

Experiment Instructions. The following instructions were provided to participants before the customization tasks: "Now that we have finished with the questions, we can proceed to the next step. This will take place as follows: You will be provided with two images of two possible interfaces generated previously through the application. Your task will be to replicate their appearance using the available functionalities. You will have 5 minutes

to view the images, which will remain available throughout the task. You will have 10 minutes of time and 10 generations for each task to achieve the objective. The time starts as soon as you log in and stops as soon as you log out. Once you have reached a satisfactory level, please log out."

System Usability Scale Questionnaire. After completing both customization tasks, participants completed the SUS questionnaire with the introduction: "Thank you for completing the experiment. Now we move to the final part, which consists of completing a questionnaire about the system's usability." The ten SUS items were: (1) "I think I would use this system frequently"; (2) "I found the system too complex"; (3) "I thought the system was easy to use"; (4) "I think I would need technical support to use this system"; (5) "I found that various functions were implemented correctly"; (6) "I thought there was too much inconsistency in the system"; (7) "I imagine that many people could learn to use this system very quickly"; (8) "I found the system too intricate"; (9) "I felt very confident using the system"; (10) "It is necessary to learn several things before being able to use the system correctly." Each item was rated on a 5-point Likert scale from 1 (Strongly Disagree) to 5 (Strongly Agree).

Additional Feedback. Participants were given the opportunity to provide additional feedback through an open-ended section: "In this section you can write additional considerations on customization strategy, challenges encountered, and suggestions for improvement."

Performance Metrics. The following metrics were collected during the study for interface accuracy assessment: Calendar elements (month name, day cells, day labels) and visual properties (font, style, layout, color scheme) were evaluated using a rating scale of Accurate, Almost Accurate, Needs Adjustments. Task performance metrics included critical errors, non-critical errors, time per task, and number of generation attempts used. This structured approach ensured consistent data collection across all participants while maintaining the authenticity of user interactions with the natural language customization system.