Misty: UI Prototyping Through Interactive Conceptual Blending

Yuwen Lu*
Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana, USA
ylu23@nd.edu

Alan Leung
Apple
Seattle, Washington, USA
alleu@apple.com

Amanda Swearngin
Apple
Seattle, Washington, USA
aswearngin@apple.com

Jeffrey Nichols
Apple
Seattle, Washington, USA
jwnichols@apple.com

Titus Barik
Apple
Seattle, Washington, USA
tbarik@apple.com

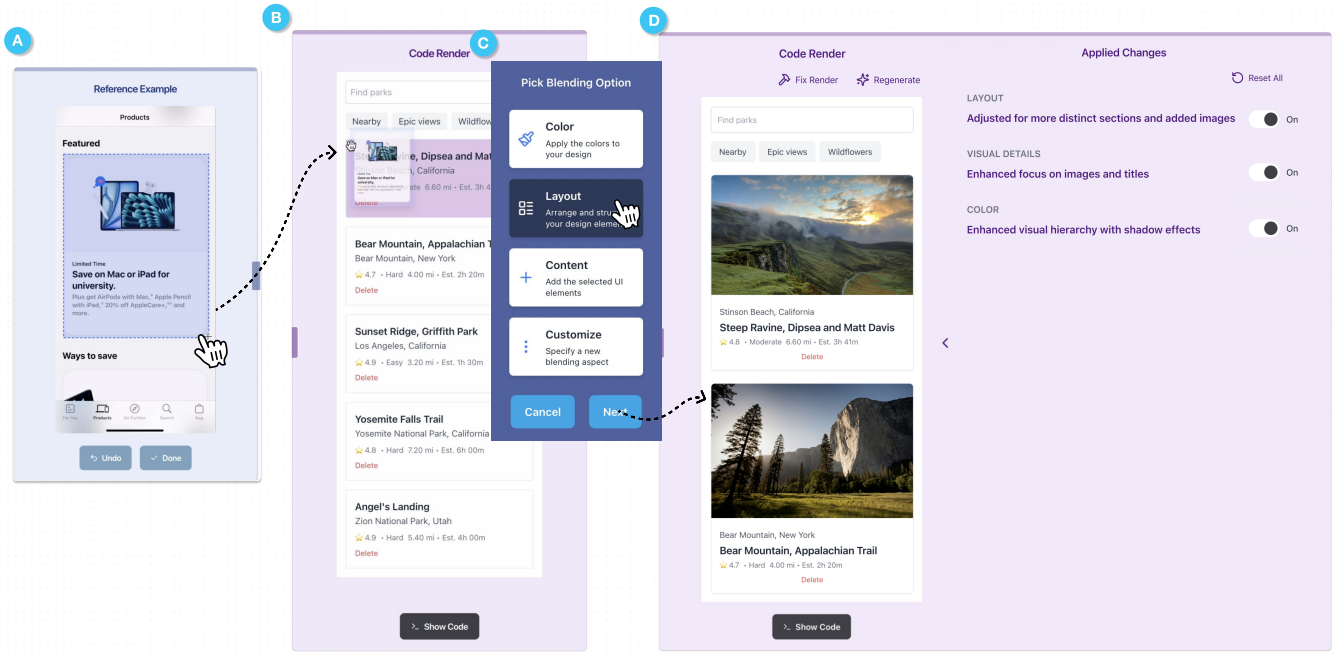


Figure 1: The Misty system. Misty embodies the interactive conceptual blending interaction for UI prototyping, where UI developers *blend* selected sections of an example image A into work-in-progress UI code B. Users can *specify* aspect(s) of the image to blend in C. After viewing the blending output D, the user can further tweak details by toggling on/off within the *semantic diff* (that is, semantically categorized code changes) between the updated and original UI versions. Initially, the work-in-progress UI shows basic cards with some park information. The user selects the card layout from the example UI, drags it onto their current design's cards, and chooses to blend the "Layout". Misty then generates updated UI code, preserving the original content but presenting it in a new card layout with images and hierarchical text descriptions, mirroring the example's layout.

Abstract

UI prototyping often involves iterating and blending elements from examples such as screenshots and sketches, but current tools offer limited support for incorporating these examples. Inspired by the cognitive process of conceptual blending, we introduce a novel UI workflow that allows developers to rapidly incorporate diverse aspects from design examples into work-in-progress UIs. We prototyped this workflow as Misty. Through a exploratory first-use study with 14 frontend developers, we assessed Misty's effectiveness and gathered feedback on this workflow. Our findings suggest

*Work done during internship at Apple.

This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.
CHI '25, Yokohama, Japan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3713924

that Misty's conceptual blending workflow helps developers kick-start creative explorations, flexibly specify intent in different stages of prototyping, and inspires developers through serendipitous UI blends. Misty demonstrates the potential for tools that blur the boundaries between developers and designers.

CCS Concepts

• **Human-centered computing → Systems and tools for interaction design**; **User interface programming**.

Keywords

UI prototyping, UX design, conceptual blending, artificial intelligence

ACM Reference Format:
Yuwen Lu, Alan Leung, Amanda Swearngin, Jeffrey Nichols, and Titus Barik. 2025. Misty: UI Prototyping Through Interactive Conceptual Blending. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3706598.3713924

1 Introduction

I can't speak, afraid to jinx it
I don't even dare to wish it
But your eyes are flying saucers from another planet
Now I'm all for you like Janet
Can this be a real thing? Can it?

Taylor Swift, *Snow on the Beach* (2023)

When prototyping user interfaces (UIs), developers routinely turn to reference examples—such as sketches, screenshots, or existing design languages—as sources of inspiration. These examples aid developers in ideation [13, 17, 35, 52], help them to avoid design fixation [26], and spur creative exploration [62, 65]. As with other forms of creative expression, developers then selectively iterate and blend various elements from these examples—such as layout or color—in this case, to design and produce a novel, distinctive UI.

This imaginative cognitive process of *conceptual blending* plays a fundamental role in the construction of meaning in everyday life, and it is so natural to the way we reason that we are often not even consciously aware that we are blending. "The essence of the operation," as described by Fauconnier and Turner, "is to construct a partial match between two input mental spaces, to project selectively from those inputs into a novel 'blended' mental space, which then dynamically develops emergent structure" [28]. For example, when singer-and-songwriter Taylor Swift tells us "your eyes are flying saucers from another planet," we intuit that the eyes are not literally flying saucers, but instead readily project selectively the two inputs—eyes and flying saucers—to blend that their gaze has a mesmerizing and otherworldly quality [67].

While it's unsurprising that other research has demonstrated the utility of conceptual blending in computational creativity support for various domains [16, 100], there are nevertheless limited tools for blending user interfaces. Blending visual inspirations is common in UI creation, but designers and developers often have to manually implement these blends from scratch [35]. How then, can we bring conceptual blending into our tools for user interface creation?

Towards addressing this gap, we created a web prototype system, Misty[1], that embodies and operationalizes conceptual blending. With Misty, users can upload example UI screenshots or hand-drawn sketches, specify specific regions and aspects of interest, blend them into their work-in-progress UIs, and use a *semantic diff* to make additional refinements to the blend (Figure 1). We postulated that because generative AI models encode certain characteristics of human ingenuity [45, 83], these latest multi-modal AI models would have within them some capacity to automatically interpret and apply conceptual blending. In Misty, we evaluated conceptual blending in developing individual mobile UI screens.

To evaluate Misty, we conducted a user study with 14 frontend developers. The study results revealed three key insights: First, participants found Misty's varied levels of input granularity valuable, supporting different stages of the design process from early ideation to detailed refinement. Second, while the current AI models showed limitations in generating perfect conceptual blends, participants appreciated our semantic diff and the code editor to tweak generation results, often using imperfections as springboards for further ideation. Third and finally, conceptual blending through Misty enhanced UI prototyping workflows by supporting both focused iteration on single designs and broad exploration of diverse alternatives.

The contributions of this paper are:

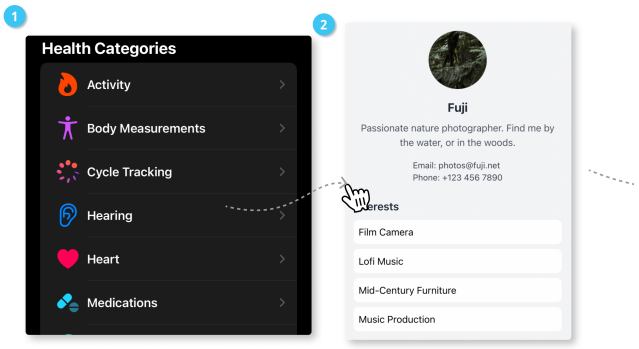
- A novel UI prototyping approach, conceptual blending, which integrates diverse, meaningful aspects of reference examples into the user's current UI.
- Misty, a system that supports conceptual blending, where UI developers interactively blend example images into work-in-progress code.
- User studies that demonstrated the value of Misty to kick-start and inspire UI development, and dissolve designer—developer boundaries.

2 Example Usage Scenario For Misty

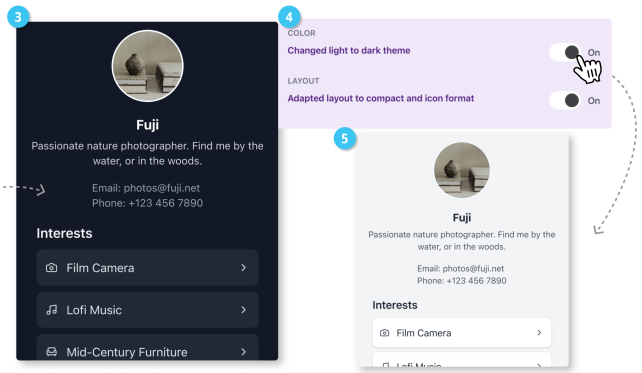
Fuji is a frontend developer creating a user profile management page in a new mobile app. Using React Native and Tailwind CSS, Fuji has laid out some basic components for the profile page, including a circular avatar, the user's name, a brief bio, contact information, and a list of the user's interests. While functional, Fuji feels the current UI is too basic and lacks visual appeal. The layout is simple, the color scheme is monotonous, and there's no clear visual hierarchy. Fuji decides to look online for inspiration to enhance the design.

Fuji has recently installed Misty, which streamlines the incorporation of design examples through conceptual blending. With some inspiring UI designs found online, Fuji is ready to use Misty to explore improvement directions for the profile UI.

[1] Misty is an acronym for Mixing Interfaces by Semantic Transformation. Plus Y, because why not.

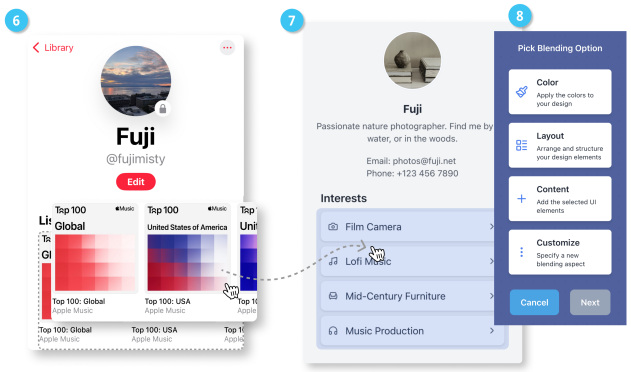


Global blending for initial exploration. Fuji began by uploading the code for their current profile UI to Misty, along with screenshots of a few inspiring profile UI designs. To get an overview of possibilities, Fuji used Misty's global blending feature to quickly explore alternatives. They connected the example UI 1 to their source UI 2, triggering a global, screen-level blending operation. Misty generated updated code that took Fuji's current profile UI content and *blended* into it the visual styles of the example UI 3.

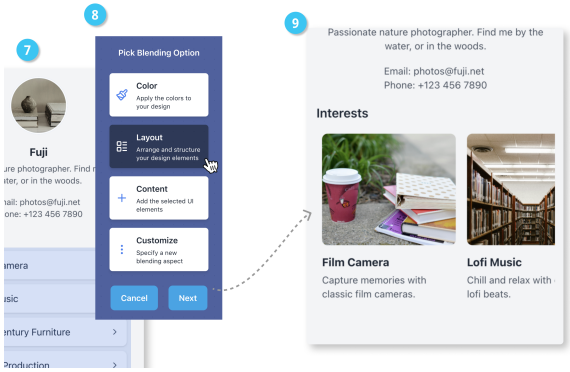


Misty provided a list of changes alongside the outputs 4, summarizing the code updates. Fuji noticed that Misty helped to turn their UI into dark theme and adopt the list item layout in the example. Fuji decided to keep the updated list item layout, but turn off the dark theme. By switching off the toggle corresponding to dark theme, Fuji received an updated result from Misty 5, which returned the UI back to the previous light theme but kept the other changes. This gave Fuji more control in exploration, by picking the desired categories of change and directly seeing the visual result.

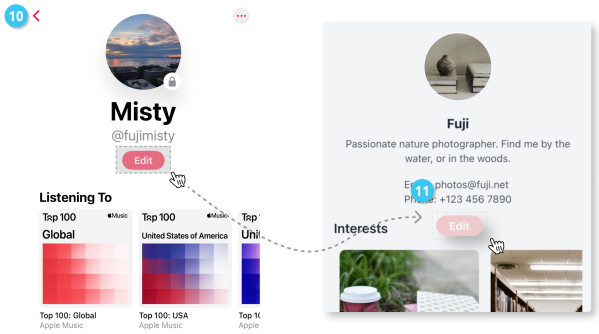
Usually, Fuji would manually re-create the example using code, which is very time-consuming. Moreover, without seeing their own content reflected in these design variations, Fuji would struggle to get a direct impression of how each design would work in their scenario. But now, with Misty, Fuji can get a quick visual preview of multiple design possibilities, all populated with their own content. Misty's global blending feature supports a more comprehensive exploration of the design space.



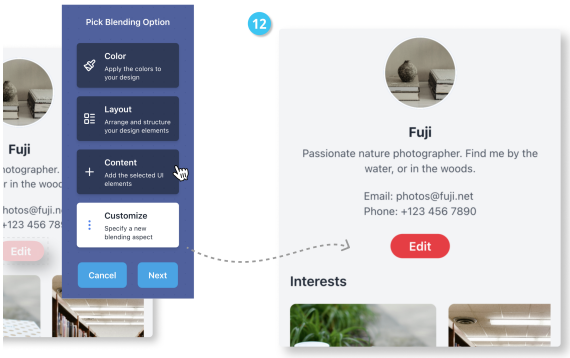
Targeted blending for sections of interest. As Fuji looked through other examples, they decided to adopt a design that is good at presenting information in a clean, card-based layout using a horizontal scrollable carousel 6. Fuji decided to incorporate this layout more precisely into the "Interest" section of their UI. Using Misty's drag-and-drop feature, Fuji selected the horizontal scrollable list from the example UI 6, then dragged it onto the corresponding area in their current design 7.



In the popup that appeared 8, Fuji specified that they wanted to blend the "layout" of the example UI. Misty processed this request and generated a new version of the interests section 9, featuring a sleek, horizontally scrollable list of users' interest items, while also adding contextually relevant details of each interest to fit in the example UI.



Adding new UI elements through blending. While refining their design, Fuji realized it would be beneficial to add a new "edit" button they saw in the previous example UI (10), to let users update their profile. This section wasn't part of Fuji's original design, but they thought it would add value to the profile page. Fuji dragged this section from the example UI onto the designated area of their current design (11).



In the popup that appeared, Fuji specified the "color", "layout", and "content" aspects to be blended, preserving the original design in the example UI. Misty successfully added this new button into Fuji's UI at the specified location (12).

Fuji is impressed with how quickly they were able to explore and iterate the UI using Misty. What would have taken hours of manual coding and design work , they were able to accomplish in a fraction of the time. The conceptual blending approach allowed Fuji to easily experiment with different design elements and see how they would look in the context of their UI. Moreover, Fuji appreciated that Misty didn't just provide a static design but generated UI code. They could now directly take the updated code back to their development environment. This seamless integration between design and code is particularly valuable for Fuji as a developer. Fuji realized that Misty can significantly speed up their UI development process in the future, allowing for rapid prototyping and iteration.

3 Related Work

In this section, we review prior work in example-driven UI prototyping, conceptual blending, and AI-enabled UI prototyping tools. These three lines of work characterize the main ideas behind Misty and build towards future tools that enable conceptual blending in UI/UX prototyping.

3.1 Using Examples in UI Prototyping

Finding inspiration from examples is extremely common in UI ideation [13, 17, 26, 35, 38, 52, 65]. UI designers and developers use examples to understand best practices, gather inspirations, and refine design concepts. Online galleries like Mobbin and Dribbble[2] are useful in providing examples with different styles, components, and interactions.

Prior research has introduced systems for retrieving design examples [13, 17, 38, 57, 65]. These systems use various methods, such as image recognition, generative networks, and metadata tagging to categorize and recommend design examples in a way that is easily searchable and accessible. Other prior works generate examples from constraints [80] or layout diffusion models [20, 21]. Misty instead focuses on blending examples, but can potentially enhance these systems by providing an interface for UI developers for blending examples generated and presented by these tools.

More related to Misty's core contributions are tools that visually transfer styles between UI designs [9, 29, 59, 79]. Uniform [60] presents a framework for ensuring similarity between automatically generated cross-platform interfaces. Bricolage [44] helps web developers to transfer the content from one full webpage into the design of the other. Similarly, VST [91] supports interactive style transfer between vector graphics. These approaches present algorithms for computing semantic correspondances between design elements. Compared to Misty, these works have primarily transferred the *visual styles* between designs. With Misty, we expand the transfer to both visuals and content between UIs. We also enable *interactive* transfer, where developers can iteratively specify sections and aspects of an example screen to transfer. While prior works [44, 91] require code and SVG as input, Misty only requires screenshots from developers. This significantly expands the set of examples developers can use and makes Misty more practical in real workflows.

3.2 Conceptual Blending

Conceptual blending is a prevalent theory in cognitive linguistics, establishing *blending* as a cognitive mechanism for the construction of meaning in people's everyday life [16, 28]. It involves identifying common elements between two concepts, selectively combining their features, and creating a new concept that inherits and transforms aspects of both inputs.

Previous studies have demonstrated conceptual blending's utility in computational creativity support for various creative domains [16, 100]. In music, conceptual blending has been applied to inspire music production by blending unique concepts drawn from music and language, two very distinct domains [100]. Conceptual blending has also been used as the basis for studying creativity and building computational implementations of creative acts [16]. Tools such as CreativeConnect have established blending as an effective operation in creative exploration for graphic design [23]. Researchers have also designed direct manipulation interfaces for AI to support creative analogical reasoning [74].

Conceptual blending is intrinsic to the inspiration-seeking processes in UI prototyping [35, 52]. Current practices such as moodboarding also reflects conceptual blending's value for exploration and ideation [17, 65]. In this work, we apply conceptual blending to UI development. We built our system Misty on the insight that the latest multi-modal AI models are capable of interpreting user intents, writing UI code [73, 97], and have the potential to operationalize conceptual blending because they encode certain characteristics of human ingenuity—for example, in metaphor reasoning [45, 83] and conceptual blending of pop culture images [88].

3.3 AI Support for UI/UX Prototyping

Many researchers have investigated the use of AI in UI design workflows [43, 49, 54, 55]. Emergent startups including Galileo,

[2]http://mobbin.com, http://dribbble.com

Magic Patterns, and Vercel have also demonstrated compelling uses of AI in UI design, yet their interactions are mostly restricted to chat interfaces and natural language instructions.

For UI development, large multi-modal AI models have exhibited capabilities in code generation given UI screenshots [73] or natural language descriptions [48, 53, 97]. While techniques for code generation from natural language have advanced rapidly in recent years [46, 50, 99], generation based on provided UI design screenshots remains limited to simplistic outputs [73].

Our system Misty expands existing work in two main directions. First, Misty does not adopt a chat-based interface, but primarily uses direct manipulation for users to specify their blending intents (i.e. which sections and aspects to blend in). Moreover, instead of code generation based on only screenshots or natural language descriptions, Misty's usage scenario provides two main input sources: developers existing code (i.e. work-in-progress), and example UI screenshots to be blended into the code. This provides a more well-scoped, multimodal code editing task to fully utilize AI models' code generation capabilities.

4 The Misty System

To evaluate how conceptual blending can help developers prototype user interfaces, we developed Misty as a web-based system that supports interactive conceptual blending.

4.1 System Design Goals

Misty embodies three design goals supported by findings from a literature review of three key areas: UI/UX design processes and tools [42–44, 52, 55, 63, 91, 98], AI for creativity support [5, 18, 23, 24], and Human-AI Interaction [3, 37, 56, 72, 77]. We formulate the design goals as follows:

- **DG1** **Support flexible and familiar input formats.** Because users employ representations at varying levels of fidelity depending on design focus [14, 38, 61] (e.g., sketches, wireframes, or high-fidelity prototypes), users should be given an input mechanism flexible enough to express intent across those representations. We identify UI examples in the form of screenshot images and hand-drawn sketches as natural mediums that agrees with common developer practices [35].
- **DG2** **Preserve developer agency in creative interaction.** Maintaining designer agency is crucial in AI-assisted design tools [43, 49, 55]. By offering granular control over the blending process, we ensure that the tool remains predictable and aligned with the user's vision, addressing key concerns raised in previous studies on AI adoption in design workflows [43, 55]. Studies have also shown that users prefer visual interactions over purely textual prompts when working with AI tools [52, 56, 77].
- **DG3** **Promote exploration, reflection, and iteration.** Creative design processes are inherently iterative and reflective [14, 61], so conceptual blending should support reflection by allowing the user to visually compare different blended results and to critically evaluate choices before making further changes. This iterative approach aligns with established design practices, such as parallel prototyping [23, 26, 63], that help users refine ideas to arrive at more innovative solutions.

4.2 Interaction Affordances

Misty realizes the above design goals through three primary interaction affordances, which we now describe.

4.2.1 Conceptual blending control for implicit intent disambiguation. A primary challenge in Misty's interaction design is the disambiguation of user intent, as the space of valid results for a conceptual blending operation is potentially large. For example, when a user wants to blend an example with their source UI, they might intend to blend the color, layout, content, or a combination of these aspects. Additionally, the user might only wish to blend from specific sections of the example, rather than the entire screen.

To address this challenge, Misty provides interactions that implicitly disambiguate user intent. Concretely, Misty offers two primary blending modes, each respectively providing different levels of user control:

(1) **Global blending through node connection**: Users connect example nodes to source nodes, triggering a global, screen-level blending operation. This mode is suitable for more exploratory phases of design and exploits the language model's implicit world knowledge from pre-training [1].
(2) **Localized blending using drag-and-drop**: For more refined control, users can drag subsections of example UIs onto specific areas of the source UI. As the drag operation hovers over parts of the source UI, a highlight animation occurs which indicates affected areas and appropriately communicates expectations of the operation's outcome to the user [61]. Finally, upon dropping, Misty presents the user a dialog to specify which aspect(s) to blend.

While global-level conceptual blending is suitable for exploratory phases of design, it can yield a low sense of control for users with specific goals in mind. Misty's drag-and-drop interface for targeted blending operations allows users to precisely specify which elements they wish to blend.

4.2.2 Node-Based Interface and Infinite Canvas. Misty employs a node-based system interface with two primary node types: example nodes (A in Figure 1) showing example images and source nodes rendering the source UI code (B in Figure 1). During a conceptual blending operation, a user blends an example node into a source node, creating a new source node with a rendering of the updated code. This node-based approach facilitates parallel prototyping [26] via exploration, visual comparison, and iteration of design alternatives. The infinite canvas, as also found in contemporary design tools such as Figma, Sketch, and Freeform, provides a familiar setup for users (DG1). Moreover, this architecture helps the user to trace the evolution of ideas, as nodes represent different stages or variations of a design. By visually representing the design process, the node-based interface inherently supports creative thinking and iterative refinement (DG3).

We additionally embed a live code editor, expandable and collapsable to the right side of the screen, for editing the code underlying any source node. In this way, developers can apply manual tweaks as desired (DG2). Note that code editors are only for source nodes and *not* example nodes. Crucially, Misty does not require access to the underlying code for example images, different from

previous research prototypes [44, 91]. Users need only upload example *images*, which may be more easily obtained than code.

4.2.3 Dynamic UI for Semantic Refinement. After a blending operation, Misty presents a summary of its changes organized into *"semantic diffs"*—groups of semantic changes to the source UI describing the outcome of a conceptual blending operation. *Semantic diffs* extend the idea of program diffs [40] by organizing changes based on their semantic impact on the UI, not on their character-level changes to the underlying code.

As a motivating example, consider the impact of converting a UI to dark mode: while conceptually a simple change, its realization might require multiple distinct edits to various CSS styles corresponding to different UI components. Misty summarizes the combination of these edits into a dynamic widget whose content is generated on-the-fly based on the meaning of the change. Crucially, the generated widget affords both coarse-grained control (enabling or disabling dark mode), or fine-grained control (varying individual colors). This approach provides the user with a high-level, conceptual understanding of the blending outcome, while also preserving user control at different levels of granularity.

This implementation of semantic diffing, inspired by recent work in dynamic UI generation [22, 84, 85], ensures that users maintain agency throughout the blending process (DG2). It allows for rapid exploration of design alternatives while preserving the ability to make precise adjustments, thus supporting both divergent and convergent thinking in the design process (DG3).

Combined affordances. The above affordances, when combined, create a spectrum of user agency ranging from full agent automation (global blending) to full direct manipulation (dynamic UI widgets and manual code editing) [72] in which the user can flexibly switch between levels of control (DG2). Global blending allows broad exploration in early stages, while granular controls (localized blending and semantic diff) allow refinement of specific aspects of the design at a later stage (DG3). By offering this range of interaction options, Misty maintains the users' sense of agency and control throughout the process [68] (DG2).

4.3 Implementation Details

Misty is implemented in React, React Flow, Tailwind CSS, and React Live, with OpenAI's GPT-4o as the underlying large multimodal model servicing prompt requests. We employ few-shot prompting [12] with chain-of-thought reasoning [92] to improve response quality (see Appendix A.3 for full details on the prompts used). To detect and repair malformed React syntax, Misty employs heuristic rules implemented using regular expressions. However, in cases where these heuristics are insufficient, we note that users may opt to either re-generate the code or use the code editor to apply manual repairs.

To enable conceptual blending in Misty, we used GPT-4o [1]. GPT-4o's capabilities in code generation, multimodal input processing, and broad world knowledge make it suitable for context-aware conceptual blending across various UI design domains.

We asked the AI model to generate *semantic diff* summaries together with each blending outputs. Afterwards, when the user toggle on/off each *semantic diff* options, Misty triggers another API call to GPT-4o to update the code.

4.4 System Limitations

Our implementation of Misty focuses on UI content and visuals. These aspects are foundational elements of UIs; in the future, we plan to build on these static aspects and expand conceptual blending to UI interactivity. Conceptual blending for interactive UX prototypes presents its own unique challenges and opportunities, as it involves dynamic states and transitions between multiple screens.

While GPT-4o and other large pre-trained multimodal models are capable and low-cost backbones for prototyping, it is likely that a custom model fine-tuned specifically for conceptual blending would offer more precise results that better follow user expectations. Recent technical advancements have shown great promise in creating a customized model in this direction [70, 97]. However, developing such a model is beyond the scope of our current study. We identify this as a direction for future work (see Section 7).

5 Exploratory First-Use Study

To evaluate Misty, we conducted an exploratory, first-use user study with 14 experienced frontend engineers. Our study answers three primary research questions:

RQ1 How do developers utilize the different features in Misty?
RQ2 To what extent do Misty's outputs satisfy developers' expectations of conceptual blending?
RQ3 How does Misty and conceptual blending support developers' workflow?

5.1 Participants

For our study, we recruited 14 frontend developers who knew React and Tailwind CSS. Participants worked at a large US technology company and were recruited via internal communication channel advertisements and word of mouth. Our participants varied in frontend development expertise, ranging from two to fifteen years of experience and novice to advanced knowledge of React and Tailwind CSS. Table 1 shows a detailed breakdown of participant demographics. We carried out two study sessions in person and fourteen online, based on the participants' preferences. Study sessions lasted 60 minutes, and we compensated each participant with a $12 digital gift card.

Since Misty uses frontend code to represent UIs, we recruited frontend engineers as our participants. Thus, they were able to fully use all features of Misty, especially the code editor, and understand the limitations of current AI models in generating frontend code [73]. All participants had experience creating and improving UIs by themselves, rather than only implementing designs created by designer colleagues (see more discussion in Section 4).

5.2 Procedure

We divided our study into the following three steps. When participants performed the tasks, we asked them to think aloud to provide insights into their cognitive process [3] [27]. We also piloted

[3] Appendix A.1 and A.2 provide a detailed list of questions in our interview and usability questionnaire.

Table 1: The participant demographics of our user studies. All participants are frontend developers with at least basic knowledge of React and Tailwind CSS, and experience in creating and improving the design of UIs.

ID	YOE	Main Frontend Tools	React & Tailwind
P1	8	React	Advanced
P2	7	React	Novice
P3	14	React, TailwindCSS	Advanced
P4	12	ReactJS, SvelteJS	Proficient
P5	2	React	Intermediate
P6	5	React, TypeScript	Proficient
P7	13	React, TailwindCSS	Advanced
P8	4	SwiftUI, React	Advanced
P9	11	React, Vue	Proficient
P10	11	React, NextJS	Proficient
P11	11	React, TailwindCSS	Advanced
P12	15	React, TailwindCSS	Advanced
P13	13	React Ecosystem	Advanced
P14	5	SwiftUI, React	Advanced

the study twice to refine the process and validate the tasks were realistic and engaging. These two pilot studies were not included in the above 14 participants.

5.2.1 Step 1 (10 min): Understanding current practices on using examples. We began by exploring participants' current practices for finding and using design examples. We asked participants about their use of examples in their work, how they find them, and examples that significantly inspired their work. We also asked about their workflow to incorporate these examples into their projects.

5.2.2 Step 2 (35 minutes): Use Misty to perform frontend development tasks through conceptual blending. We then introduced participants to Misty and asked them to perform four frontend development tasks through conceptual blending:

- **T1** Blending a full example UI into the current source UI,
- **T2** Adding the header section of an example UI to the source UI,
- **T3** Updating the source UI layout by blending in a hand-drawn screen sketch,
- **T4** Looking for examples online, and open-ended improvement of the source UI through conceptual blending, until satisfied.

We designed these tasks to progress from *constrained* to *open-ended* exploration for the participants. The initial tasks familiarized the participants with Misty's core functionalities, while the later tasks encouraged application of Misty to more subjective and open-ended design scenarios. The final open-ended task helped us understand how participants might naturally incorporate Misty into their existing workflows. This progression lets us naturally onboard our participants while assessing Misty's usability, functionality, and potential impact across various design scenarios. Participants iterated on each task until they were satisfied with the results. We picked three representative UI layouts and components (i.e., book lists, information cards, phone apps) as the basis of these tasks. We asked the participants to perform the first three tasks using one example, but we asked them to pick a different example for the last task to have a clean slate for open-ended exploration.

After each blending operation, participants provided (1) a 5-scale value of how much the results met their expectations, and (2) qualitative feedback on ways to improve the results. These results helped us further understand developers' expectations and current multimodal AI models' limitations in UI conceptual blending.

5.2.3 Step 3 (15 minutes): Usability questionnaire and follow-up interview. After completing the tasks, participants completed a questionnaire on their experience using Misty. We adapted the questionnaire from the standard System Usability Scale [11] and included 5-point Likert scale questions evaluating Misty on its usability, user control and agency, output quality, value for design exploration, and real-world adoption potential. After completing the questionnaire, we conducted a semi-structured interview with participants. The interview questions covered scenarios where participants would or would not use Misty, desired improvements or additional features, concerns about using Misty, and thoughts on how tools like Misty might influence their work.

6 User Study Results

In this section, we provide the visualizations of participants' conceptual blending operations (Figure 2) and their responses to the usability questionnaires (Figure 3). We also conducted a qualitative analysis of the user study sessions' transcripts. These results together helped to answer our research questions regarding the value of Misty in UI development workflows.

6.1 Quantitative Results

To answer RQ1 and RQ2—how developers utilize Misty's different features and to what extent Misty's outputs satisfy developers' expectations—we visualized the blending operations conducted by participants in Figure 2. The average time each participant spent interacting with Misty was 36.3 minutes (median 36, sd 7.0). On average, each participant conducted 7.4 blending operations during the session (median 8, sd 2.5). Some participants conducted fewer blending operations (e.g. P1, P5, P9), because they provided extensive feedback and rationale. For the total of 103 blending operations, participants used Global blending (36/103) and Drag-and-drop (35/103) the most, constituting 34.95% and 33.98% of all operations, respectively. Users also used Regenerate (19/103, 18.5%), Semantic diff (9/103, 8.74%), and Code editing (4/103, 3.88%) during the study.

After each blending operation, participants also evaluated how satisfied they were with each output on a 5-point Likert scale from "Very unsatisfied (1)" to "Very satisfied (5)". The average satisfaction across all participants was 3.2 (median 3, sd 1.3). Out of all 103 activities, 31 received a satisfaction score below 3 (Unsatisfied or Very unsatisfied), 44 received a score above 3 (Satisfied or Very satisfied), with the remaining 28 rated 3 (Neutral). We noticed that some participants primarily expressed negative valence towards the outputs, for example, P7 and P14. These were mostly due to their high expectations of the outputs' visual details, or the random nature of Misty's underlying model's behavior. Meanwhile, certain participants (e.g. P2, P6) showed generally positive valence towards the outputs. We observed that users' responses to the usability

questionnaire (Figure 3) were influenced by their blending output quality in Misty.

Nine out of the fourteen participants rated some blending outputs as "Very satisfied", where Misty followed their expectations and captured the essence of examples in the updated code. Meanwhile, some blending outputs did not meet the users' expectations, mostly because they either did not reflect the users' expected changes or created UIs with non-ideal layout or alignment.

Regarding the usability questionnaire responses (Figure 3), most participants find Misty generally easy to use and would like to use it frequently in their work. However, opinions were mixed on how well it supported expressing user intents, with only 5 out of the 14 participants (35%) agreeing or strongly agreeing. Follow-up questions revealed that participants' main concerns for this question were centered around usability issues in the prototype, such as cumbersome interaction details in drag-and-drop and imperfect feature discoverability. Moreover, participants explained that the strong disagreement responses came from receiving Misty outputs that failed to follow their instructions or didn't meet their expectations for visual details (e.g., alignment, element sizing). These undesired outputs also reflected as low satisfaction scores in Figure 2.

For most participants who rated low on "Misty supports me well in expressing my intents", they requested features to continuously improve Misty's output after conceptual blending. For example, after receiving an output, participants wanted to ask the AI model to further iterate on output details. These iterations do not involve reference examples, and are therefore not directly related to our focus on conceptual blending.

When specifically asked about Misty's features for intent expressions, such as cropping and drag and drop, participants did not raise any issues. They felt these features addressed their needs in blending operations, as we will discuss later in Section 6.2.2.

6.2 Qualitative Themes

To answer RQ3—understanding how Misty's interactive conceptual blending workflow supports developers—we conducted a qualitative analysis of our user study by transcribing the interviews, segmenting the interview content into quotations, and organizing these quotations into meaningful themes. Our analysis resulted in six themes, and we present a summary of these themes in Table 2.

6.2.1 Kickstarting nascent creative explorations. All participants reported the use of examples in their UI development workflows. Developers would look for relevant UI examples online or find examples from component libraries. Then, they would often manually recreate these examples from scratch using the framework for their own projects (P3, P6, P8–10, P11, P14). The open source culture in web development—as well as browser inspection tools—allowed developers to easily "borrow aspects of the design inspirations from others" (P3) and build their UI from these starting "patterns" (P4).

Misty supports developers by scaffolding and accelerating exploratory UI prototyping in early stages of development workflows, especially "being useful in very fast development" (P3). This is triangulated by the usability questionnaire results, where most participants agreed that Misty helps to "iterate on designs faster" and "explore more alternative design options". Participants "would want to use this (Misty) to do the grunt work of setting up something"

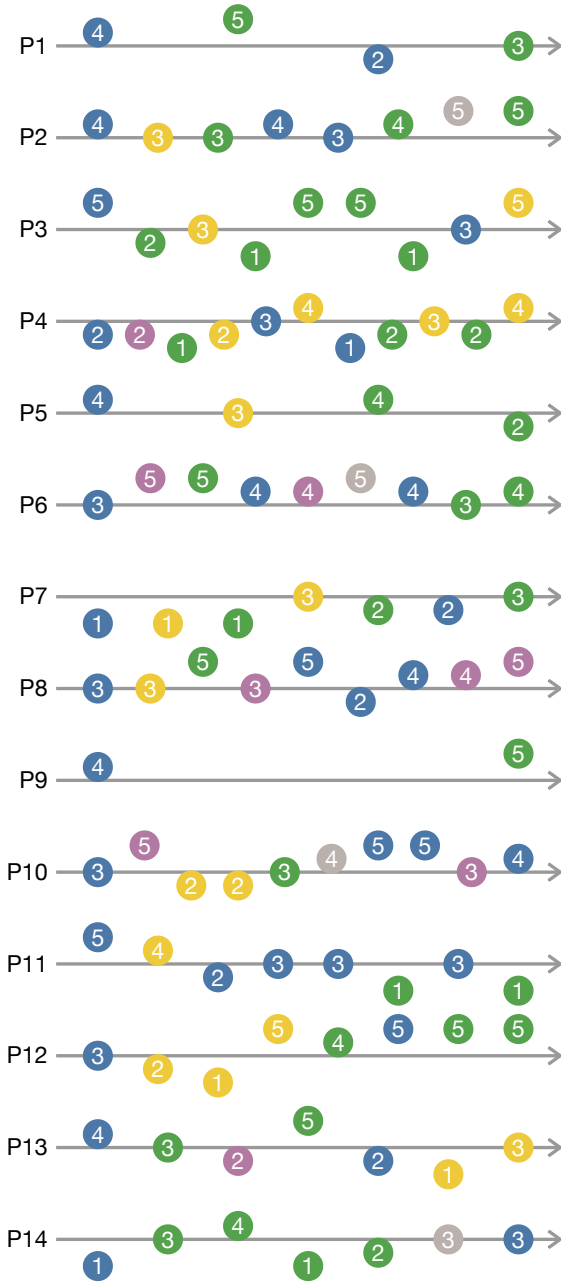


Figure 2: Event trace of all blending operations and user evaluation scores during the user study sessions. Each circle represents a blending operation, laid out from left to right based on the time it occurred during the study. The number inside is the participant's evaluation of how well the blending output matched expectations, with 1 as not satisfied and 5 very satisfied. Based on these evaluation scores, we also varied the circles' vertical positions. The color of each circle represents the type of operation in Misty: Global blend, Drag and drop, Semantic diff, Regenerate, Code edit. We normalize the space between operations.

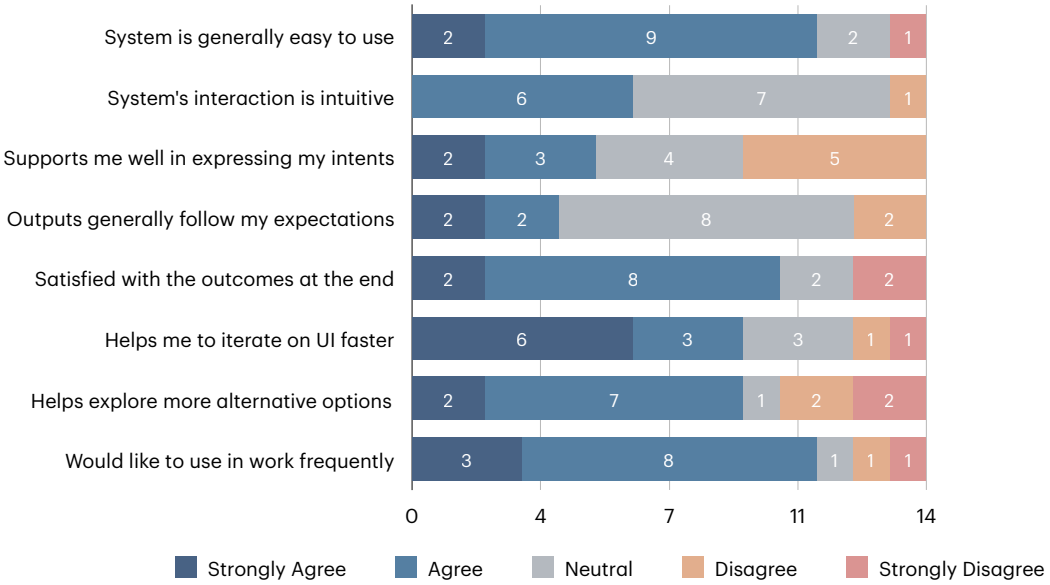


Figure 3: The usability questionnaire results from our user study. User satisfaction survey results for Misty: most participants found Misty easy to use and would like to use it in their work regularly. Misty is generally well-received, with potential for targeted improvements in user intent interpretation and interface refinement.

(P11). P4 estimated that Misty "will add productivity boost to our work by 20%–30%", since 30%–40% of their current work time is spent on creating CSS and static UI mockups. Participants prefer to take Misty's output versus creating UI manually, as it can automate tasks that would take them 20–30 minutes to do otherwise (P10).

6.2.2 Allowing flexible intent specification. Participants appreciated Misty's diverse input mechanisms for specifying design intent, which supported various stages of their development process. The ability to blend in full screenshots, specific aspects of example UIs, and hand-drawn sketches proved efficient for different phases of UI development. For early ideation, participants valued the option to start exploring with generally blending in full screens or hand-drawn sketches. Particularly, hand-drawn sketches are widely used in roughly laying out design concepts and communicating design ideas. The ability to blend in sketches fits well with developers' current practices (P3, P6, P11, P12), especially "in the case (when) I have an idea in mind, for which I cannot find any relevant template online" (P6). P3 noted "a lot of my time is just translating a sketch... this (Misty) can save me time on making the code or the visual look, so I don't have to fight with CSS and layout". As designs progressed, participants found value in having more specific input options. P3 noted, for example, that drag-and-drop is very good for updating existing products, instead of starting brand new ones.

6.2.3 Generating code as a blending substrate. Participants proposed utilizing the generated code in different ways, including as another avenue for refining the UI blending results (Section 6.2.5) to "easily fixing little things" (P2) or adjust other "minute aspects like font or spacing that's quicker to just do myself" (P14). For some participants, authoring user interface code for presentation is tricky, and Misty can help developers avoid "having to fight with CSS" (P3) and the tool "would help with a lot of the presentation boilerplate" (P1).

Other participants desired to "take the code and just use it" (P4) or to uplift an "existing legacy app interface to generate compatible but modern React and Tailwind" (P3). P11 elaborated on their expectations around this code generation hand-off: "With these kinds of systems, I wouldn't necessarily expect it to make pixel perfect code, but I would want the code that it resulted in to be something that I can then take from there and then bulletproof myself."

A gap in Misty is that some participants wanted the code generation to emit UI components that map to their own libraries and design systems (P1, P5, P6, P10, P12). For example, P12 suggested organizing the output code into components, to better fit into the Atomic Design principles [31]: "If it (Misty) could build that as a component, or at least tell me that this thing is repetitive (reusable), that would solve a major problem." In some situations where Misty complements designers, "developers might not have the freedom to output any design they like" (P6). A future version of Misty could allow preloading a set of design components for the tool to draw from or take advantage of custom data schemas—that is, "being able to fill in the UI with my own list of data" (P1)—when these are made available to the Misty.

6.2.4 Inspiring through serendipity. A benefit of Misty was its ability to generate surprising blending outputs that inspire developers. Six participants (P2, P5, P7, P8, P10, P14) reported enjoying serendipitous outputs that sparked new ideas. During the study, P2 received a particularly unexpected yet inspiring blend "that sparks joy". This sentiment was echoed by P5, who found Misty's surprise blends "to the most useful to my work, making me think of things that I have not thought of." These experiences suggest that

Table 2: Summary of qualitative participant feedback, organized by themes.

Theme	Description	Representative Examples	Participants
Kickstarting nascent creative explorations (Section 6.2.1)	Participants found Misty valuable to accelerating early stages of UI exploration.	"I would want to use this to do the grunt work of setting up something." "I would definitely use Misty for giving me a head start which I can later improvise on." "It saves a good amount of time and effort from writing the code from scratch through any starting templates."	P1–P4, P6, P10, P11, P14
Allowing flexible intent specification (Section 6.2.2)	Participants appreciated the flexible intent specification mechanisms of Misty for various usage scenarios.	"It's very useful when I can't find any relevant templates to blend, I can just draw it as a sketch and use Misty to generate the UI." "The drag-and-drop is really good, it's basically an alternative to manually fixing it." "Being able to come in and toggle these on and off through dynamic widgets to make the AI output manageable like this is really cool."	P1–4, P6–12, P14
Generating code as a blending substrate (Section 6.2.3)	Participants used the generated code in a variety of ways and noted their expectations of the generated code.	"Designers can do the prototyping design phase and we can just take the code and use it." "This section of the code will be reused many times and will componentize." "I wouldn't necessarily expect it to make pixel perfect code, but I would something that I can then take from there and then bulletproof."	P1, P3, P4, P12
Inspiring through serendipity (Section 6.2.4)	Participants enjoyed unexpected blending outputs from Misty, which helps to provide inspirations they have not thought of.	"This one blend sparks joy because it is unexpected." "(Misty would be) useful to my work because it makes me think of things that I have not thought of."	P2, P5, P7, P8, P10, P14
Refining iteratively and visually (Section 6.2.5)	After their initial explorations, participants desired additional capabilities to support further iterative refinement of their UIs.	"I just want to tell it (Misty) to fix it without dragging and dropping" "A designer wants to be pixel perfect." "I'd love for the ability to add text to the prompt like I could when I was adding parts of a design to full styles."	P1, P2, P8–10
Blending boundaries between designers and developers (Section 6.2.6)	Developers can see Misty as a tool for both themselves and designers, complementing each group with the other group's skills, and blends the boundaries between these roles.	"This blends the boundary between designer and developer." "It does a good job of covering for my lack of design skills." "(Misty is) raising the bar for developers of the app as far as the look and feel." "If designers use Misty to 'generate some code', I can use it immediately."	P1–4, P6, P12–14

Misty stimulates developers' creativity, pointing out new design directions outside of established patterns.

However, serendipity can also be a double-edged sword. Although some of the blends initially have an "amazing wow factor" (P10), when carefully scrutinizing the blend there are often small details that were initially overlooked, such as the "separation between items and not apply the button design style to one of the buttons" (P10). The randomness, which "can be good in exploration" (P7), is "also bad when you already know where to go" (P7) or "have a specific idea in mind" (P3). For such scenarios, having a "more deterministic way of working with the tool would be preferred" (P3). This experience made P3 "impressed by some things and yet underwhelmed by others."

To address these trade-offs, a future improvement could give a knob to the developer that allows them to indicate their preference for serendipity. Earlier in the design process they may set this preference higher, and lower it later in the design process when they know more precisely what design they want.

6.2.5 Refining iteratively and visually. While Misty helps in initial explorations, participants needed more support for iterative visual refinement after the initial exploratory phases (P1, P2, P8–10). Participants often found themselves wanting to make precise

adjustments directly over the output, without drag-and-drop of another example. Participants wanted the ability to "add text prompt to the output, like I could when I was adding parts of a design to full styles" (P8). This comes from a demand for pixel-perfect UI implementations in professional contexts, which current models are not always able to produce [73, 97]. From many UI developers, they are expected to produce pixel perfect implementations, especially by designers (P3). As a result, when seeing issues in the blending outputs' visual details (e.g. element sizing, alignment, and color values), developers naturally wanted to fix them with direct prompt iterations over the visually rendered UI. This is currently only supported by direct editing in our code editor. However, developers also recognized that they are fine with manually tweaking these details. While the blending outputs were not pixel perfect, developers generally thought they were "good enough output for me to build upon" (P4), then, "the minute interaction of copying over font or changing some spacing, that's something I can do way quicker by myself" (P14). Yet, developers can still benefit from direct iterations on the visual outputs in Misty, without necessarily involving more examples.

6.2.6 Blending boundaries between designers and developers. P2, P4, and P13 considered both designers and developers as potential user groups that can benefit from using Misty. The system showed potential in blending designer and developer activities in UI prototyping, shifting collaboration and handoff points in the full prototyping workflow. This is noted by eight participants in our study. P1, P4, and P14 mentioned the shifting handoff points between designers and developers, where traditionally they only receive design artifacts that contain no code implementation. If designers use Misty to generate some code, developers can directly take the code and immediately use it in their implementation (P1). Misty can also help developers to incorporate stakeholder feedback faster, enhancing cross-function collaboration (P12) and reducing the friction with designers (P3, P13).

At a higher level, Misty also complements designers and developers with each others' skills. P2 noted that Misty "does not replace me, but does a good job of covering for my lack of design skills." Misty helps to raise the bar for design for developers' UI implementations. Participants imagined designers using Misty may not have to rely on engineers and vice versa (P11). In the future, tools like Misty will continuously lower the boundary of entry for both UI design and development, changing the separated dynamics and goals for these two roles.

7 Discussion

Implications for responsible AI in conceptual blending tools. Artists and creators have expressed concern over content ownership as generative AI models advance [30, 75]. Conceptual blending, while prevalent in creative tasks, is essentially borrowing content and styles from existing examples that users do not necessarily own. Conceptual blending tools have to be responsibly deployed with careful ethical considerations around ownership, permissions, and their use in blending derivatives.

UI/UX design and development pose unique challenges for these ethical considerations in conceptual blending. Since software users do not adapt quickly to new interface and interaction patterns, designers often need to incorporate established patterns that users are already familiar with [61]. As a result, usability—instead of novelty or creativity—is often prioritized in UI/UX prototyping. For this reason, the boundary of unethical appropriation in UI/UX design is often ambiguous. Moreover, designers and developers open-source component libraries for others to reuse and re-purpose.

Misty currently does not have safety guardrails around what inputs the system can accept and what outputs the system can generate. As a result, Misty can generate inappropriate user interfaces if such blends are requested by the developer. There are continuing conversations on how to balance developer freedom around flexible intent specification and when (and whether) to deny the generation of certain user interfaces.

These are only some challenges that the generative UI community is grappling with that would need to be resolved in this evolving landscape of Generative AI technology. It would be valuable to prioritize responsible AI and copyright ethics to improve the design of Misty. Insights from this design approach, and its influence on user interaction with the tool would contribute to HCI and Responsible AI.

Augmenting refinement with bespoke widgets. In Misty, we demonstrated the feasibility and utility of "semantic diff" using a simple toggle widget Figure 1. However, there exists an expressive yet underexplored design space of bespoke widgets that make semantically meaningful adjustments beyond direct attribute edits. For example, a *layout density slider* can tweak screen element layouts from "sparse" to "dense", triggering simultaneous updates of CSS spacing attributes without breaking the harmony of the layout. A *color scheme grid* can generate the entire page's color scheme—instead of single color pickers—along axes such as "monochrome" to "multicolored" and "dark" to "light" themes. We can also imagine a screen *content verbosity slider* from "concise" to "verbose", together updating multiple pieces of text on screen.

To successfully materialize these bespoke widgets, Misty would need to incorporate additional context, such as system states and user intentions. Previous work has also discussed the potential value of serializing and saving these bespoke widgets for reuse [22].

Nevertheless, introducing these bespoke widgets to future tools similar to Misty can support more semantic, iterative refinement that improves developer productivity.

Expanding conceptual blending in Misty. Our implementation of Misty is based on simple UI screens with limited interactivity. During our user study, participants expressed interest in expanding Misty to conceptual blending of interactive prototypes. Interactive UI prototypes contain many dimensions that are relevant for conceptual blending: user interactions, animations, and screen states. Tools like CoCapture [19] have made progress towards blending interactivity, and Keyframer [84] has investigated the use of Generative AI for authoring animations. However, blending interactivity continues to pose design challenges. Blending static screenshots and sketches are relatively simple, yet it is not obvious how to specify sections and aspects—for example, of a video—for blending. Design principles from research in programming by demonstration can provide a useful starting point [25].

We also plan to expand Misty to support more "unusual blends", such as blending of non-UI media types—pictures, videos, and even

music—into UI screens. These conceptual blends might be able to borrow inspirations based on aesthetic or mood. For example, an audio clip containing somber tones could result in the generation of a user interface with dark and muted colors. Similarly, a photograph from a science fiction film might result in a user interface that mimics the affordances appropriate to that sci-fi genre. As other forms of art are rich in metaphor and figurative language, it would be amusing to consider how Generative AI can meaningfully interpret such unusual blends to construct surprising user interfaces.

8 Limitations

8.1 User Study Limitations

Our study participants were frontend developers recruited from a single US-based technology company. Other roles, such as UI/UX designers, may have different experiences with conceptual blending using Misty than developers. US-centric participants may also limit our understanding of how different culture design preferences might affect usage patterns and UI outputs. The first-use study format and controlled task environment may not always reflect how Misty would be used in real-world scenarios. A longitudinal study of Misty being used in the wild can be a valuable future work to answer these questions.

8.2 System and Technical Limitations

Misty works with static, single-screen mobile UI. It does not support the conceptual blending of interactive prototypes and UI animations. Understanding multi-screen, interactive UI prototypes with AI has been an active research area [52, 96, 101]. UI motion design protocols such as Lottie Files [51] also make conceptual blending of UI animations easier. We expect to build Misty on top of these work to extend its blending capabilities.

Misty only supports accessibility incidentally [6, 76, 81, 94]. Misty can incorporate accessibility design best practices [6]—for example, by including post-hoc checking mechanisms to ensure accessibility—such as the ones in DesignChecker [39]. Adopting an ability-based design approach, Misty can also generate blending outputs based on user abilities, to serve diverse users' needs.

Misty also does not consider cross-cultural design [10, 36, 41, 64] in conceptual blending. Research has shown that users from different cultures have diverse preferences for AI capabilities, control, and connections [32]. Cultural contexts also influence how AI models and users influence each other's decision making [47]. Misty could adopt more culturally-aware AI models in generating blending outputs [66, 78]. Co-designing and testing Misty with users from different cultures can also help us see how their preferences and desired outputs may differ [34].

8.3 Responsible AI Considerations in UI Prototyping

Practitioners and researchers have been incorporating Responsible AI and AI ethics into technical systems [2, 3, 71, 86, 87, 95]. Systems and toolkits have been proposed in HCI research to support Responsible AI in design and development workflows [15, 58, 89, 93].

The boundary of "ethical" and "unethical copying" in UI/UX is fluid. Much of today's UI design and development builds upon established design patterns to prioritize usability over creativity [52, 82]. Also, open-source communities promote re-using and re-purposing pre-built UI components [7, 8, 69, 102].

Before AI adoption in design tools, we already witnessed increased homogenization of website design [33]. AI-enabled prototyping workflows, like Misty, can amplify this trend [4]. Automated prototyping tools raise questions about authorship and intellectual property. They risk copyright infringement and can spread biased design choices. Guidelines and ethical oversight can ensure these tools support creative freedom and minimize harm.

In Misty, we evaluated the experience of conceptual blending to understand its impact on UI prototyping workflows. Misty does not currently have safeguards. Design opportunities exist for Misty to prioritize Responsible AI considerations. For example, researchers can visualize an AI model's conceptual blending outputs, to help users understand the output distributions. Combined with visual analytics [58, 93], this could help users discover and understand AI model biases. Furthermore, in-situ support similar to Farsight [90] could increase users' Responsible AI awareness in prototyping and adopting blending outputs. Future research in this direction can lead to research insights for conceptual blending in UI prototyping.

9 Conclusion

We built Misty, a system that embodies the conceptual blending and enables designers and developers to blend concepts from UI examples into their own work. We operationalized conceptual blending within Misty through a multi-modal AI model—postulating that these models have some capability to interpret and apply conceptual blending. Although imperfect and occasionally uneven in its generation, these models nevertheless demonstrated a remarkable ability to apply blends and even produce surprising and serendipitous results. Participants in our exploratory first-use study were encouraged by Misty's capabilities. Creativity is a kaleidoscope, and Misty demonstrates the potential for tools that blur the boundaries between developers and designers.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. *GPT-4 technical report*. Technical Report 2023-03-14. OpenAI. https://openai.com/research/gpt-4

[2] Sanna J Ali, Angèle Christin, Andrew Smart, and Riitta Katila. 2023. Walking the walk of AI ethics: Organizational challenges and the individualization of risk among ethics entrepreneurs. In *Fairness, Accountability, and Transparency*. 217–226.

[3] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for Human-AI Interaction. In *CHI*. 1–13.

[4] Barrett R Anderson, Jash Hemant Shah, and Max Kreminski. 2024. Homogenization effects of large language models on human creative ideation. In *Creativity & Cognition*. 413–425.

[5] Tyler Angert, Miroslav Suzara, Jenny Han, Christopher Pondoc, and Hariharan Subramonyam. 2023. Spellburst: A node-based interface for exploratory creative coding with natural language prompts. In *UIST*. 1–22.

[6] Apple Inc. 2024. *Accessibility - Apple Human Interface Guidelines*. Apple Developer. https://developer.apple.com/design/human-interface-guidelines/accessibility

[7] Apple Inc. 2024. *Apple design resources*. Apple Inc. https://developer.apple.com/design/resources/

[8] Paula M Bach and Michael Twidale. 2010. Social participation in open source: what it means for designers. *Interactions* 17, 3 (2010), 70–74.

[9] Tony Beltramelli. 2018. pix2code: Generating code from a graphical user interface screenshot. In *CHI*. 1–6.

[10] Paula Bourges-Waldegg and Stephen AR Scrivener. 1998. Meaning, the central issue in cross-cultural HCI design. *Interacting with Computers* 9, 3 (1998), 287–309.

[11] John Brooke. 1996. SUS: A "quick and dirty" usability scale. In *Usability Evaluation in Industry*. Taylor & Francis, London, 189–194.

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

[13] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. VINS: Visual search for mobile user interface design. In *CHI*. 1–14.

[14] Bill Buxton. 2010. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann.

[15] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. FairVis: Visual analytics for discovering intersectional bias in machine learning. In *Visual Analytics Science and Technology (VAST)*. IEEE, 46–56.

[16] Francisco Câmara Pereira. 2007. *Creativity and Artificial Intelligence: A conceptual blending approach*. Mouton de Gruyter.

[17] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design search and knowledge discovery through auto-created GUI component gallery. *ACM Human-Computer Interaction* 3, CSCW (2019), 1–22.

[18] Xi Chen, Yutong Feng, Mengting Chen, Yiyang Wang, Shilong Zhang, Yu Liu, Yujun Shen, and Hengshuang Zhao. 2024. Zero-shot image editing with reference imitation. *arXiv preprint arXiv:2406.07547* (2024).

[19] Yan Chen, Sang Won Lee, and Steve Oney. 2021. CoCapture: Effectively communicating UI behaviors on existing websites by demonstrating and remixing. In *CHI*. 1–14.

[20] Chin-Yi Cheng, Ruiqi Gao, Forrest Huang, and Yang Li. 2024. CoLay: Controllable layout generation through multi-conditional latent diffusion. *arXiv preprint arXiv:2405.13045* (2024).

[21] Chin-Yi Cheng, Forrest Huang, Gang Li, and Yang Li. 2023. PLay: Parametrically conditioned layout generation using latent diffusion. In *ICML*. JMLR.org, Article 216, 23 pages.

[22] Ruijia Cheng, Titus Barik, Alan Leung, Fred Hohman, and Jeffrey Nichols. 2024. BISCUIT: Scaffolding LLM-generated code with ephemeral UIs in computational notebooks. *VL/HCC* (2024).

[23] DaEun Choi, Sumin Hong, Jeongeon Park, John Joon Young Chung, and Juho Kim. 2024. CreativeConnect: Supporting reference recombination for graphic design ideation with Generative AI. In *CHI*. 1–25.

[24] John Joon Young Chung and Eytan Adar. 2023. PromptPaint: Steering text-to-image generation through paint medium-like interactions. In *UIST*. 1–17.

[25] Allen Cypher and Daniel Conrad Halbert. 1993. *Watch What I Do: Programming By Demonstration*. MIT Press.

[26] Steven P Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L Schwartz, and Scott R Klemmer. 2010. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *TOCHI* 17, 4 (2010), 1–24.

[27] K Anders Ericsson and Herbert A Simon. 1980. Verbal reports as data. *Psychological Review* 87, 3 (1980), 215.

[28] Gilles Fauconnier and Mark Turner. 2003. Conceptual blending, form and meaning. *Recherches en communication* 19 (2003), 57–86.

[29] Sidong Feng, Minmin Jiang, Tingting Zhou, Yankun Zhen, and Chunyang Chen. 2022. Auto-icon+: An automated end-to-end code generation tool for icon designs in UI development. *Interactive Intelligent Systems* 12, 4 (2022), 1–26.

[30] Catherine Flick and Kyle Worrall. 2022. The ethics of creative AI. In *The Language of Creative AI: Practices, Aesthetics and Structures*. Springer, 73–91.

[31] Brad Frost. 2016. *Atomic Design*. Brad Frost Pittsburgh.

[32] Xiao Ge, Chunchen Xu, Daigo Misaki, Hazel Rose Markus, and Jeanne L Tsai. 2024. How culture shapes what people want from AI. In *CHI*. 1–15.

[33] Samuel Goree, Bardia Doosti, David Crandall, and Norman Makoto Su. 2021. Investigating the homogenization of web design: A mixed-methods approach. In *CHI*. 1–14.

[34] Lorna Heaton. 2002. Designing Work: Situating design objects in cultural context. *Journal of Design Research* 2, 2 (2002), 47–75.

[35] Scarlett R Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P Bailey. 2009. Getting inspired! Understanding how and why examples are used in creative design practice. In *CHI*. 87–96.

[36] Geert Hofstede. 2009. Cultural Dimensions. (2009).

[37] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *CHI*. 159–166.

[38] Forrest Huang, John F Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based user interface retrieval. In *CHI*. 1–10.

[39] Mina Huh and Amy Pavel. 2024. DesignChecker: Visual design support for blind and low vision web developers. In *UIST*. 1–19.

[40] James Wayne Hunt and M Douglas MacIlroy. 1976. *An algorithm for differential file comparison*. Bell Laboratories Murray Hill.

[41] Wendy A Kellogg and John C Thomas. 1993. Cross-cultural perspectives on Human-Computer Interaction: A report on the CHI'92 workshop. *ACM SIGCHI Bulletin* 25, 2 (1993), 40–45.

[42] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the web with natural language. In *CHI*. 1–17.

[43] Tiffany Knearem, Mohammed Khwaja, Yuling Gao, Frank Bentley, and Clara E Kliman-Silver. 2023. Exploring the future of design tooling: The role of artificial intelligence in tools for user experience professionals. In *CHI*. 1–6.

[44] Ranjitha Kumar, Jerry O Talton, Salman Ahmad, and Scott R Klemmer. 2011. Bricolage: Example-based retargeting for web design. In *CHI*. 2197–2206.

[45] George Lakoff. 2014. Mapping the brain's metaphor circuitry: Metaphorical thought in everyday reason. *Frontiers in Human Neuroscience* 8 (2014), 958.

[46] Triet HM Le, Hao Chen, and Muhammad Ali Babar. 2020. Deep learning for source code modeling and generation: Models, applications, and challenges. *ACM Computing Surveys (CSUR)* 53, 3 (2020), 1–38.

[47] Kyootai Lee and Kailash Joshi. 2020. Understanding the role of cultural context and user interaction in artificial intelligence based systems. , 171–175 pages.

[48] Amanda Li, Jason Wu, and Jeffrey P Bigham. 2023. Using LLMs to customize the UI of webpages. In *UIST*. 1–3.

[49] Jie Li, Hancheng Cao, Laura Lin, Youyang Hou, Ruihao Zhu, and Abdallah El Ali. 2024. User experience design professionals' perceptions of generative artificial intelligence. In *CHI*. 1–18.

[50] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: May the source be with you! *arXiv preprint arXiv:2305.06161* (2023).

[51] LottieFiles. 2024. Download free lightweight animations for website & apps. https://lottiefiles.com/. Accessed: 2024-12-10.

[52] Yuwen Lu, Ziang Tong, Qinyi Zhao, Yewon Oh, Bryan Wang, and Toby Jia-Jun Li. 2024. Flowy: Supporting UX design decisions through AI-driven pattern annotation in multi-screen user flows. *arXiv preprint arXiv:2406.16177* (2024).

[53] Yuwen Lu, Ziang Tong, Qinyi Zhao, Chengzhi Zhang, and Toby Jia-Jun Li. 2023. UI layout generation with LLMs guided by UI grammar. *arXiv preprint arXiv:2310.15455* (2023).

[54] Yuwen Lu, Yuewen Yang, Qinyi Zhao, Chengzhi Zhang, and Toby Jia-Jun Li. 2024. AI assistance for UX: A literature review through human-centered AI. *arXiv preprint arXiv:2402.06089* (2024).

[55] Yuwen Lu, Chengzhi Zhang, Iris Zhang, and Toby Jia-Jun Li. 2022. Bridging the gap between UX practitioners' work practices and AI-enabled design support tools. In *CHI Extended Abstracts*. 1–7.

[56] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2024. DirectGPT: A direct manipulation interface to interact with Large Language Models. In *CHI*. 1–16.

[57] Mohammad Amin Mozaffari, Xinyuan Zhang, Jinghui Cheng, and Jin LC Guo. 2022. GANSpiration: Balancing targeted and serendipitous inspiration in user interface design with style-based generative adversarial network. In *CHI*. 1–15.

[58] David Munechika, Zijie J Wang, Jack Reidy, Josh Rubin, Krishna Gade, Krishnaram Kenthapadi, and Duen Horng Chau. 2022. Visual auditor: Interactive visualization for detection and summarization of model biases. In *IEEE Visualization and Visual Analytics (VIS)*. IEEE, 45–49.

[59] Tuan Anh Nguyen and Christoph Csallner. 2015. Reverse engineering mobile application user interfaces with REMAUI (T). In *Automated Software Engineering (ASE)*. IEEE, 248–259.

[60] Jeffrey Nichols, Brad A Myers, and Brandon Rothrock. 2006. UNIFORM: Automatically generating consistent remote control user interfaces. In *CHI*. 611–620.

[61] Don Norman. 2013. *The Design of Everyday Things*. Basic Books.

[62] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. Designscape: Design with interactive layout suggestions. In *CHI*. 1221–1224.

[63] Xiaohan Peng, Janin Koch, and Wendy E Mackay. 2024. DesignPrompt: Using multimodal interaction for design exploration with Generative AI. In *DIS*. 804–818.

[64] Tom Plocher, Pei-Luen Patrick Rau, Yee-Yin Choong, and Zhi Guo. 2021. Cross-cultural design. *Handbook of human factors and ergonomics* (2021), 252–279.

[65] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R Klemmer. 2011. d. tour: Style-based exploration of design example galleries. In *UIST*. 165–174.

[66] Stephen Cory Robinson. 2020. Trust, transparency, and openness: How inclusion of cultural values shapes nordic national public policy strategies for Artificial Intelligence (AI). *Technology in Society* 63 (2020), 101421.

[67] Pelsi Santika and Syafryadin Syafryadin. 2023. An analysis of figurative language in song lyrics of the album "Midnights" by Taylor Swift. *Wiralodra English Journal (WEJ)* 7, 1 (2023), 14–28.

[68] Arvind Satyanarayan and Graham M Jones. 2024. Intelligence as agency: Evaluating the capacity of Generative AI to empower or constrain human action. (2024).

[69] Shadcn. 2024. *Shadcn UI—Build your component library.* Shadcn. https://ui.shadcn.com/

[70] Hijung V Shin, Jeremy Warner, Björn Hartmann, Celso Gomes, Holger Winnemöller, and Wilmot Li. 2021. Multi-level correspondence via graph kernels for editing vector graphics designs. In *Graphics Interface*. 97–107.

[71] Ben Shneiderman. 2020. Bridging the gap between ethics and practice: Guidelines for reliable, safe, and trustworthy Human-Centered AI systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 10, 4 (2020), 1–31.

[72] Ben Shneiderman and Pattie Maes. 1997. Direct manipulation vs. interface agents. *Interactions* 4, 6 (1997), 42–61.

[73] Chenglei Si, Yanzhe Zhang, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2Code: How far are we from automating front-end engineering? *NAACL* (2025).

[74] Arvind Srinivasan and Joel Chan. 2024. Improving selection of analogical inspirations through chunking and recombination. In *Creativity & Cognition*. 374–397.

[75] Ramya Srinivasan and Kanji Uchino. 2021. The role of arts in shaping AI ethics. In *AAAI Workshop on Reframing Diversity in AI: Representation, Inclusion, and Power*.

[76] Constantine Stephanidis, Demosthenes Akoumianakis, Michael Sfyrakis, and Alexandros Paramythis. 1998. Universal accessibility in HCI: Process-oriented design guidelines and tool requirements. In *ERCIM Workshop on User Interfaces for All*. 19–21.

[77] Hari Subramonyam, Roy Pea, Christopher Pondoc, Maneesh Agrawala, and Colleen Seifert. 2024. Bridging the gulf of envisioning: Cognitive challenges in prompt based interactions with LLMs. In *CHI*. Article 1039, 19 pages.

[78] Shoko Suzuki. 2023. We need a culturally aware approach to AI. *Nature Human Behaviour* 7, 11 (2023), 1816–1817.

[79] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Amy J Ko. 2018. Rewire: Interface design assistance from examples. In *CHI*. 1–12.

[80] Amanda Swearngin, Chenglong Wang, Alannah Oleson, James Fogarty, and Amy J Ko. 2020. Scout: Rapid exploration of interface layout alternatives through high-level design constraints. In *CHI*. 1–13.

[81] Amanda Swearngin, Jason Wu, Xiaoyi Zhang, Esteban Gomez, Jen Coughenour, Rachel Stukenborg, Bhavya Garg, Greg Hughes, Adriana Hilliard, Jeffrey P Bigham, et al. 2024. Towards automated accessibility report generation for mobile apps. *TOCHI* 31, 4 (2024), 1–44.

[82] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. 2012. Learning design patterns with Bayesian grammar induction. In *UIST*. 63–74.

[83] Xiaoyu Tong, Rochelle Choenni, Martha Lewis, and Ekaterina Shutova. 2024. Metaphor understanding challenge dataset for LLMs. arXiv:2403.11810 [cs.CL] https://arxiv.org/abs/2403.11810

[84] Tiffany Tseng, Ruijia Cheng, and Jeffrey Nichols. 2024. Keyframer: Empowering animation design using Large Language Models. *arXiv preprint arXiv:2402.06071* (2024).

[85] Priyan Vaithilingam, Elena L Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically synthesized UI widgets for visualization editing. In *CHI*. 1–17.

[86] Rama Adithya Varanasi and Nitesh Goyal. 2023. "It is currently hodgepodge": Examining AI/ML practitioners' challenges during co-production of responsible AI values. In *CHI*. 1–17.

[87] Qiaosi Wang, Michael Madaio, Shaun Kane, Shivani Kapania, Michael Terry, and Lauren Wilcox. 2023. Designing Responsible AI: Adaptations of UX practice to meet responsible AI challenges. In *CHI*. 1–16.

[88] Sitong Wang, Savvas Petridis, Taeahn Kwon, Xiaojuan Ma, and Lydia B Chilton. 2023. PopBlends: Strategies for conceptual blending with Large Language Models. In *CHI* (Hamburg, Germany) *(CHI '23)*. Article 435, 19 pages.

[89] Zijie J Wang, Alex Kale, Harsha Nori, Peter Stella, Mark E Nunnally, Duen Horng Chau, Mihaela Vorvoreanu, Jennifer Wortman Vaughan, and Rich Caruana. 2022. Interpretability, then what? Editing machine learning models to reflect human knowledge and values. In *SIGKDD*. 4132–4142.

[90] Zijie J Wang, Chinmay Kulkarni, Lauren Wilcox, Michael Terry, and Michael Madaio. 2024. Farsight: Fostering Responsible AI awareness during AI application prototyping. In *CHI*. 1–40.

[91] Jeremy Warner, Kyu Won Kim, and Bjoern Hartmann. 2023. Interactive flexible style transfer for vector graphics. In *UIST*. 1–14.

[92] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Neurips* 35 (2022), 24824–24837.

[93] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The What-if Tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2019), 56–65.

[94] Jacob O Wobbrock, Shaun K Kane, Krzysztof Z Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-based design: Concept, principles and examples. *ACM Transactions on Accessible Computing (TACCESS)* 3, 3 (2011), 1–27.

[95] Austin P Wright, Zijie J Wang, Haekyu Park, Grace Guo, Fabian Sperrle, Mennatallah El-Assady, Alex Endert, Daniel Keim, and Duen Horng Chau. 2020. A comparative analysis of industry Human-AI Interaction guidelines. *arXiv preprint arXiv:2010.11761* (2020).

[96] Jason Wu, Rebecca Krosnick, Eldon Schoop, Amanda Swearngin, Jeffrey P Bigham, and Jeffrey Nichols. 2023. Never-ending learning of user interfaces. In *UIST*. 1–13.

[97] Jason Wu, Eldon Schoop, Alan Leung, Titus Barik, Jeffrey Bigham, and Jeffrey Nichols. 2024. UICoder: Finetuning large language models to generate user interface code through automated feedback. In *NACCL*. 7511–7525.

[98] Jason Wu, Amanda Swearngin, Xiaoyi Zhang, Jeffrey Nichols, and Jeffrey P Bigham. 2023. Screen correspondence: mapping interchangeable elements between UIs. *arXiv preprint arXiv:2301.08372* (2023).

[99] Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696* (2017).

[100] Lawrence M Zbikowski. 2018. Conceptual blending, creativity, and music. *Musicae Scientiae* 22, 1 (2018), 6–23.

[101] Zhuohao Jerry Zhang, Eldon Schoop, Jeffrey Nichols, Anuj Mahajan, and Amanda Swearngin. 2025. From interaction to impact: Towards safer AI agents through understanding and evaluating UI operation impacts. *IUI* (2025).

[102] Roshanak Zilouchian Moghaddam, Brian Bailey, and Wai-Tat Fu. 2012. Consensus building in open source user interface design discussions. In *CHI*. 1491–1500.

A Appendix

A.1 Interview Questions For User Study

Here, we provide a detailed list of questions asked to participants during the user study. Please refer to Section 5 for a comprehensive description of the user study procedures.

A.1.1 Step 1: Understanding Current Practices on Finding Inspirations.

(1) How do you use inspirations in your work? If so, how do you find and use them?
(2) Can you show me some examples that you learned a lot from or are very inspired by?
(3) How did you incorporate these examples into your work

A.1.2 Step 3: Follow-Up Interview Questions. Please note that given it is an semi-structured interview, we sometimes adjusted the questions based on the participant's questionnaire response and question answers.

(1) What are scenarios you would like to use Misty?
(2) How will it influence your current workflow?
(3) What are scenarios you would not like to use Misty? Why?
(4) What do you think about the ability to blend in examples of different fidelities?
(5) If you do not only blend in examples, what are some other ways you might use conceptual blending?
(6) What are some of the ways Misty can be improved? What other features do you desire?
(7) How to improve Misty further to make it more useful?
(8) Any concerns using Misty?
(9) Misty's features support design iteration, but are directly implemented using frontend code. As a developer, how do you feel? Will this influence your work? Will this impact designer-developer handoff?

A.2 Usability Questionnaire For User Study

Each question is accompanied by a 1-5 likert scale.

(1) The system is generally easy to use.
(2) The system's interaction is intuitive.
(3) The system supports me well in expressing my intents.
(4) The system's outputs generally follow my expectations.
(5) I am satisfied with the outcomes I get at the end.
(6) The system can help me to iterate on UI designs faster.
(7) The system helps me to explore more alternative design options than I usually would.
(8) I think that I would like to use a system like Misty in my work frequently.

A.3 Generative AI Prompts

Here, we provide two main types of prompts we used to generate conceptual blending outputs in Misty.

A.3.1 Global Blending. Global blending generally involves the work-in-progress code and a reference image (separately uploaded).

```
Here is my react and tailwind code:

        ${work-in-progress code}

Help me create a new component page that blends
the content of the above code with the visual
style, layout, and appearance of the reference
image. Change only the source code corresponding
to this, and no other sections.

Preserve the content in the UI of the code,
follow the layout and visual style of the
reference image, optionally add more content
where the original code's content cannot fill in
all fields in the reference image's layout. Do
not use content from the reference image, just
use its layout and visual style.

A few rules:

- First, explain in concise language the design
of the reference screenshot. Use it as a basis of
 your generation.
- Make sure all text is legible on the background
- Briefly summarize the differences between the
reference image and the code, summarize them into
 a few categories of changes you want to make.
Base your later generation of categorizedChanges
based on these categories.
- only use tailwind, react, and react icons.
Follow the current code structure, do not include
 any import or export statements, just use a
simple component definition () => {};
- when adding icons, pick from this list of
lucide react icons, do not use others: ${list of
available lucide icon names}
- there are a few stock photos for use under the
folder /stock/, they are named after their
orientation, like landscape0.jpg, landscape1.jpg,
 portrait0.jpg, etc. Do not use any other images.
 Do not use placeholder image paths.
- Summarize the code changes in your response,
use the format "categorizedChanges:" followed by
a list of changes. Be very concise in your
explanations. For example, "Color change: section
 titles, from green to purple"; "Layout change:
adapted the layout for [add the feature
description of the changed code piece]".


Return result in the below format, make sure you
use json:

{
    "designExplanation": // explain the design of
     the screenshot image, focus on layout and
    color, be really concise, less than 30 words
    "differences": // briefly summarize the
    differences between the reference image and
    the code, focus on layout orientation,
    spacing, color theme, font, etc.
    updatedCode: \`() => {}\`   // return the
    whole component for the entire screen, with
    the updates;
    // a list of objects listing the changes made
    , use the tailwind classes to indicate the
    changes
    categorizedChanges: [
        {
```

```
            category: "",   // summarize the
            category of the below changes, group
            changes together semantically, e.g. "
            Color: Changed light to dark theme",
            "Layout: Increased spacing between
            elements", "Visual details: Increased
             corner roundedness", "Image:
            Decreased image size", "Font: Changed
             font appearance", etc.
            changes: [{
                type: "color",
                before": // the tailwind class
                before the change,
                "after": // the tailwind class
                after the change
            }]
        }
    ]
}

here is a good example of the changes field:
categorizedChanges: [
    {
        category: "Color: Changed light to dark
        theme",
        changes: [{
            type: "color",
            before: "bg-black",
            after: "bg-white"
        }, {
            type: "color",
            before: "text-white",
            after: "text-gray-900"
        }, {
            type: "border",
            before: "", // you can use empty
            before field to indicate addition of
            new classes
            after: "border-2 border-gray-300/90"
        }, ...] // add as many as appropriate,
    },
    {
        category: "Font: Changed font appearance
        ",
        changes: [{
            type: "color",
            before: "bg-black",
            after: "bg-white"
        }, {
            type: "font",
            before: "text-sm",
            after: "text-lg"
        }, ...] // add as many as appropriate,
    },

    ]
```

A.3.2 Drag-and-drop Blending. The drag-and-drop blending prompt differ from the global blending prompt mainly in two ways. First, it provides the desired aspects users want to blend in (e.g. color, layout, content). Second, it adds the specific target code that was dropped on.

```
% [breaklines,fontsize=\footnotesize,breaksymbol={}]{text
}

Here is my react and tailwind code:

        ${work-in-progress code}

blend the ${user specified blending aspect} of
the reference image into ${targetCodeDropped}.
Change sections of the source code corresponding
to this, as well as sections that are of similar
layout or screen position to this. For example,
don't just apply to one element in a list, but
apply to all list elements with similar layouts.
Sometimes the specific code piece does not
correspond to parts of the source code, because
it's rendered HTML based on the source React code
. In that case, you need to identify the original
 code pieces from the source and modify them.

A few rules:


- First, explain in concise language the layout
of the reference screenshot. Use it as a basis of
 your generation.
- Make sure all text is legible on the background
.
- Briefly summarize the differences between the
reference image and the code, summarize them into
 a few categories of changes you want to make.
Pay attention to ${blendMode.filter(mode => mode
!== "Customize").join(" ")}. Base your later
generation of categorizedChanges based on these
categories.
- Never directly pulls content from the reference
 to update the source code. For blending color
and layout, preserve all original content in the
UI for source code, only change/add the original
content when it's really necessary for following
a layout. When you blend in the addition mode,
generate content based on the context of the
source code.
- Do not use list and .map functions to represent
 lists. Just generate HTML elements for each of
the list items.
- only use tailwind, react, and react icons.
Follow the current code structure, do not include
 any import or export statements, just use a
simple component definition () => {};
- when adding icons, pick from this list of
lucide react icons, do not use others: ${list of
avaialble lucide icons}

- there are a few stock photos for use under the
folder /stock/, they are named after their
orientation, like landscape0.jpg, landscape1.jpg,
 portrait0.jpg, etc. Do not use any other images.
 Do not use placeholder image paths.
- Summarize the code changes in your response,
use the format "categorizedChanges:" followed by
a list of changes. Be very concise in your
explanations. For example, "Color change: section
 titles, from green to purple"; "Layout change:
adapted the layout for [add the feature
description of the changed code piece]".
- When creating a bottom navigation bar, use "
absolute bottom-0" instead of "fixed bottom-0".
- Try to make colors and styles consistent and
harmonious with the rest of the component.


Return result in the below format, make sure you
use json:
```

```
{
    "designExplanation": // explain the design of
     the screenshot image, focus on layout and
    color, be really concise, less than 30 words
    "differences": // briefly summarize the
    differences between the reference image and
    the code, focus on layout orientation,
    spacing, color theme, font, etc.
    updatedCode: \`() => {}\`   // return the
    whole component for the entire screen, with
    the updates;
    // a list of objects listing the changes made
    , use the tailwind classes to indicate the
    changes
    categorizedChanges: [
        {
            category: "",   // summarize the
            category of the below changes, group
            changes together semantically, e.g. "
            Color: Changed light to dark theme",
            "Layout: Increased spacing between
            elements", "Visual details: Increased
             corner roundedness", "Image:
            Decreased image size", "Font: Changed
             font appearance", etc.
            changes: [{
                type: "color",
                before": // the tailwind class
                before the change,
                "after": // the tailwind class
                after the change
            }]
        }
    ]
}

here is a good example of the changes field:
categorizedChanges: [
    {
        category: "Color: Changed light to dark
        theme",
        changes: [{
            type: "color",
            before: "bg-black",
            after: "bg-white"
        }, {
            type: "color",
            before: "text-white",
            after: "text-gray-900"
        }, {
            type: "border",
            before: "", // you can use empty
            before field to indicate addition of
            new classes
            after: "border-2 border-gray-300/90"
        }, ...] // add as many as appropriate,
    },
    {
        category: "Font: Changed font appearance
        ",
        changes: [{
            type: "color",
            before: "bg-black",
            after: "bg-white"
        }, {
            type: "font",
            before: "text-sm",
            after: "text-lg"
        }, ...] // add as many as appropriate,
    },
]
```