CANVAS: A Benchmark for Vision-Language Models on Tool-Based User Interface Designs

Daeheon Jeong[1]*, Seoyeon Byun[2]*, Kihoon Son[1], Dae Hyun Kim[3], Juho Kim[1]
[1]KAIST
[2]Korea University
[3]Yonsei University
{daeheon.jeong, kihoon.son, juhokim}@kaist.ac.kr, byunhwan8832@korea.ac.kr, dhkim16@yonsei.ac.kr

Abstract

User interface (UI) design is an iterative process in which designers progressively refine their work with design software such as Figma or Sketch. Recent advances in vision language models (VLMs) with tool invocation suggest these models can operate design software to edit a UI design through iteration. Understanding and enhancing this capacity is important, as it highlights VLMs' potential to collaborate with designers within conventional software. However, as no existing benchmark evaluates tool-based design performance, the capacity remains unknown. To address this, we introduce CANVAS, a benchmark for VLMs on tool-based user interface design. Our benchmark contains 598 tool-based design tasks paired with ground-truth references sampled from 3.3K mobile UI designs across 30 function-based categories (e.g., onboarding, messaging). In each task, a VLM updates the design step-by-step through context-based tool invocations (e.g., create a rectangle as a button background), linked to design software. Specifically, CANVAS incorporates two task types: (i) *design replication* evaluates the ability to reproduce a whole UI screen; (ii) *design modification* evaluates the ability to modify a specific part of an existing screen. Results suggest that leading models exhibit more strategic tool invocations, improving design quality. Furthermore, we identify common error patterns models exhibit, guiding future work in enhancing tool-based design capabilities.

Introduction

User interface (UI) design is a nonlinear process in which designers experiment with ideas through back-and-forth actions and observations (Goldschmidt 1991; Ferreira, Noble, and Biddle 2007). In this process, designers continuously adjust, group, and arrange each design component, such as buttons and text, using design software (e.g., Figma and Sketch). Working with design software is essential for designers, as they provide familiar controls over design components for efficient iterations (Resnick et al. 2005; Stolterman and Pierce 2012; Son et al. 2024).

Recent vision-language models (VLMs) have demonstrated agentic tool invocation capabilities; a model can complete tasks by invoking tools over multiple turns (Niu et al. 2024). This capability suggests that VLMs could create and edit UI designs by interacting with design software using tool invocations (Fig. 1A). Yet, existing benchmarks solely focus on *code-based* capabilities for the implementation stage, such as generating HTML and CSS (Li et al. 2022; Si et al. 2025; Li, Zhang, and Yang 2025). The earlier *tool-based* design stage, where designers iteratively refine interfaces using design software, remains underexplored. Understanding VLMs' tool-based capabilities is crucial for enabling collaboration during this design phase (Lu et al. 2024; Choi et al. 2024).

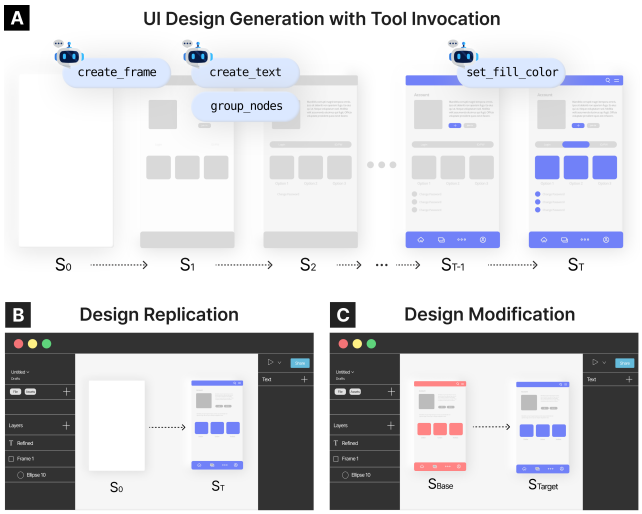

Figure 1: Overview. CANVAS evaluates a VLM's capability to (A) generate a UI design with tool invocations in two tasks: (B) design replication and (C) design modification.

To address this gap, we introduce CANVAS, a benchmark for evaluating VLMs' performance in tool-based UI design. The benchmark comprises 598 UI design tasks selected to evaluate the model's performance in two task scenarios: (i) *design replication* (Fig. 1B), where the model generates a design that replicates a given UI image, and (ii) *design modification* (Fig. 1C), where the model modifies an existing human-made design by applying a sequence of edits such as resizing a component. In each task, a VLM invokes context-based tool commands (e.g., *"create a rectan-*

*These authors contributed equally.

gle as a button background") to interact with the design software (Figma 2024), and iteratively construct a complete UI design. Each task is paired with ground-truth designs sampled from 3,327 human-crafted mobile UI designs across 30 functions (e.g., onboarding, messaging) collected from an online design community (Figma 2025a).

To evaluate model performance, CANVAS measures the similarity between generated and ground-truth designs across three hierarchical levels of visual perception, modeling how humans progressively interpret visual information: features, patterns, and objects. The evaluation includes structural similarity at the feature level (SSIM) (Wang et al. 2004), compositional similarity at the pattern level via saliency maps (Jiang et al. 2023), and semantic interpretation at the object level using BLIP captions (Li et al. 2023). To capture fine-grained editing performance, we additionally evaluate component-level attributes (Si et al. 2025).

In our experiments, we evaluate state-of-the-art VLMs capable of tool invocation using the CANVAS. Our analysis reveals two key findings: (i) in replication tasks, model performance is closely linked to diverse and strategic tool usage; and (ii) in modification tasks, precise tool selection is critical, as a single incorrect action can cause large metric score shifts due to the granularity of the task. Through the two task configurations, CANVAS captures the model's capability to operate the design strategically, while precisely determining accurate tools. Furthermore, we identify the limitations of current VLMs in tool-based design through error analysis and discuss directions for future improvement. In summary, our contributions are:

- We introduce CANVAS, the first benchmark to evaluate VLMs' ability to perform tool-based UI generation in an interactive, multi-turn design environment.
- We construct a dataset of 598 tool-driven design tasks, comprising replication and modification tasks, sampled from 3,327 UIs across 30 categories.
- We conduct a comprehensive evaluation of five state-of-the-art VLMs and discuss key insights into their tool use behaviors through quantitative and qualitative analysis.

Related Work

UI Design Generation

Research on UI design with generative models follows two approaches: code-based generation, which generates the code that renders a design, and image-based generation, which synthesizes the design image directly. Beltramelli's pix2code provides an early proof-of-concept for code-based UI design generation, translating UI screenshots into code using neural network models (Beltramelli 2018). Subsequent research advances the models to condition on more abstract representations, such as sketches, wireframes, and text prompts (Moran et al. 2020; Kim et al. 2022; Li, Zhang, and Yang 2025). Recent work has increased instruction compliance by incorporating human and self-generated feedback into a generation process (Zhou et al. 2025; Gui et al. 2025b), training on the layout structure (Tang et al. 2024), and setting incremental steps (Wan et al. 2025). Another line of research explored directly generating UI designs as images, including generative adversarial networks (Li et al. 2019; Zhao et al. 2021) and diffusion models (Cheng et al. 2023; Garg, Jiang, and Oulasvirta 2025). These approaches are unconstrained by code-specific syntax and render design and layout images through a denoising process. Nevertheless, the generated result remains less editable than designs created with design software, restricting its practical applicability; the problem persists in code-based generation (Yuan et al. 2025). Contrary to previous approaches, CANVAS evaluates tool-based design generation: models produce designs with tool invocations in design software, creating directly editable outputs.

UI Design Datasets and Benchmark

Existing datasets and benchmarks on UI generation primarily focus on code-based generation. A common dataset structure pairs UI screenshots with corresponding source code (e.g., HTML, XML) to support code-based design generation (Si et al. 2025; Yun et al. 2024). The datasets have increased the scale, starting from RICO on mobile UI design (Deka et al. 2017) to large-scale real-world datasets, such as WebCode2M (Gui et al. 2025a) and WebSight (Laurençon, Tronchon, and Sanh 2024). These datasets enable training UI generation models and benchmarking by comparing generated images and source code with ground-truth designs. Evaluation metrics include pixel-level similarity, SSIM for structural comparison, CLIP for semantic alignment, and FID (Zhao et al. 2021; Xiao et al. 2025; Li, Zhang, and Yang 2025; Gui et al. 2025b). Studies also utilize the source code to measure component-level attributes (e.g., text, position, color) (Si et al. 2025; Gui et al. 2025b) and structural relationships (Gui et al. 2025a). Still, existing metrics have limited applicability to tool-based UI design, as design software structures data representation around designer-centric concepts such as masks and layers. CANVAS addresses these limitations by proposing metrics adapted to the tool-based design context.

Benchmark Design

Task Design

To reflect real-world user scenarios, CANVAS consists of two types of user interface (UI) design tasks: (i) *design replication* and (ii) *design modification*. In *design replication*, the model completes a design based on a reference image and a build instruction, which provides the task context and the overall design goal. In *design modification*, the model refines an existing design based on a reference image and an edit instruction, which describes the target component and the expected changes in plain language without numeric values. These task settings reflect the designers' practical expectations towards AI assistance, ranging from automating repetitive tasks to assisting with micro-level design edits (Lu et al. 2022; Li et al. 2024).

CANVAS represents a UI design as a state composed of components (e.g., buttons, text) and their attributes (e.g., position, width, color), mirroring how design software organizes design components. In each task, a UI design is for-

malized as a state s made up of n component–attribute pairs.

$$s = \{ (c_i, a_{ij}, v) \mid i = 1, \ldots, n, \ j = 1, \ldots, m_i \}.$$

where each component c_i is associated with a set of attribute-value pairs (a_{ij}, v). The benchmark evaluates the model-generated design $\hat{s}$ by comparing it with a ground-truth design s_{GT}, using these attribute values and visual characteristics.

Design Replication The replication task measures a VLM's ability to map out a sequence of tool invocations to reconstruct a target UI design s_{GT}. Starting from an empty canvas $s_0 = \varnothing$, the model iteratively adds or edits components, producing a design trajectory

$$s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_t.$$

The process stops when the model determines $s_t \approx s_{\text{GT}}$ and ends tool invocations.

Design Modification The design modification task (Fig. 2) measures the VLM's ability to apply a fine-grained tool invocation over turns to complete specified changes. Starting from a pre-existing design state s_{old}, the model is instructed to modify the design into a specified target state s_{GT} based on (i) a task instruction and (ii) a ground truth image. The model needs to conduct a list of changes $\Delta = [\ldots]$ to reach the target design state

$$s_{old} \xrightarrow{\Delta} s_{GT}$$

The design modification task consists of three sub-tasks:

- *Attribute Update* (Fig. 2B-(A)): Modify attributes a in different components c to target values v' in a design. The attributes include fill color, size, text content, corner radius, and position.

$$\Delta_{\text{attr}} = \{ (c_k, a_k, v'_k) \mid k = 1, \ldots, K \}.$$

- *Component Insertion* (Fig. 2B-(B)): Create and insert a list of new component-attribute pairs into a design.

$$\Delta_{\text{add}} = \{ (c_k, a_k, v_k) \mid k = 1, \ldots, K \}.$$

- *Mode Change* (Fig. 2B-(C)): Change a list of colors to a target value to convert the design between light and dark themes.

$$\Delta_{\text{col}} = \{ (\text{rgb}_k^{\text{old}}, \ \text{rgb}_k^{\text{new}}) \mid k = 1, \ldots, K \}.$$

These operations cover the primitive modification patterns in UI design workflows: property adjustments, component additions, and systematic style changes.

Dataset Creation

We collected UI designs from an online community and refined them through a designer review.

Data Source Our dataset comprises UI designs from public projects on the Figma Community platform (Figma 2025a), a widely used online repository. We selected Figma as a data source for its popularity and standardized format. [1]

[1]All designs are licensed under Creative Commons (CC BY 4.0), and the dataset includes a link to the original project.

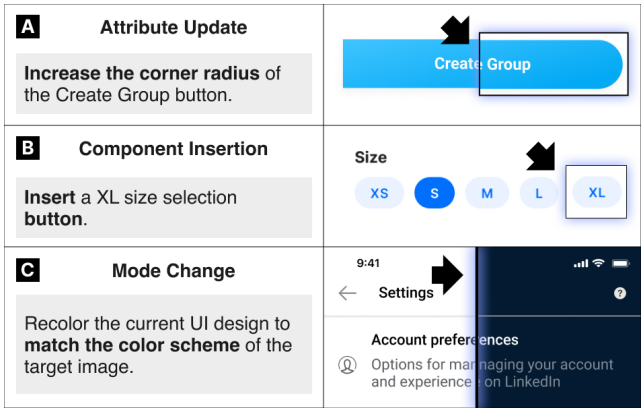

Figure 2: Design Modification Tasks. This task measures a model's capacity to perform targeted edits: (A) adjusting component attributes, (B) inserting new components, or (C) switching the overall color scheme.

Data Collection We created the dataset through selection, categorization, sampling, and annotation to ensure UI type diversity (see Appendix for details).

- (i) *Data Selection*. We manually selected projects with editable UI designs, excluding non-UI resources, e.g., prototyping kits. Selection stopped when additional designs showed minimal variation. Each design was exported as SVG, PNG, and JSON via Figma's REST API.
- (ii) *Data Sampling*. The designs were categorized into 30 UI types by GPT-4.1-Mini (OpenAI 2025a) with zero-shot prompting; one of the authors defined the types, adapting Mobbin's categorization with GPT-4.5 (Mobbin Ltd. 2025; OpenAI 2025b). The replication set ($n = 298$) was sampled using stratified sampling from the original pool ($n = 3,327$), covering all UI categories (10 per category, except one with 8). The modification set ($n = 300$) was manually sampled from the same pool based on task-relevant attributes (e.g., round borders).
- (iii) *Data Annotation*. For attribute update and component insertion tasks, we generated instructions using GPT-4.1-Mini based on visual differences between manually created "base" and "target" states. These states were manually created by editing the original design in Figma (e.g., deleting a component).

Data Refinement To improve dataset quality, we reviewed and refined the dataset through designer review of designs and annotations. (see Appendix for details).

- (i) *Data Analysis*. We cleaned the designs through automated pre-processing (e.g., placeholder replacement, font standardization) and defined common issues, including occlusions, illustrations, and visual clutter (misleading layout composition).
- (ii) *Manual Revision*. Four experienced designers revised 598 designs by fixing issues, correcting flawed instructions from GPT-4.1-Mini, and annotating required tools. This process resolved occlusions in 33.4% of the designs, illustration errors in 15.5%, and clutter in 29.0%.

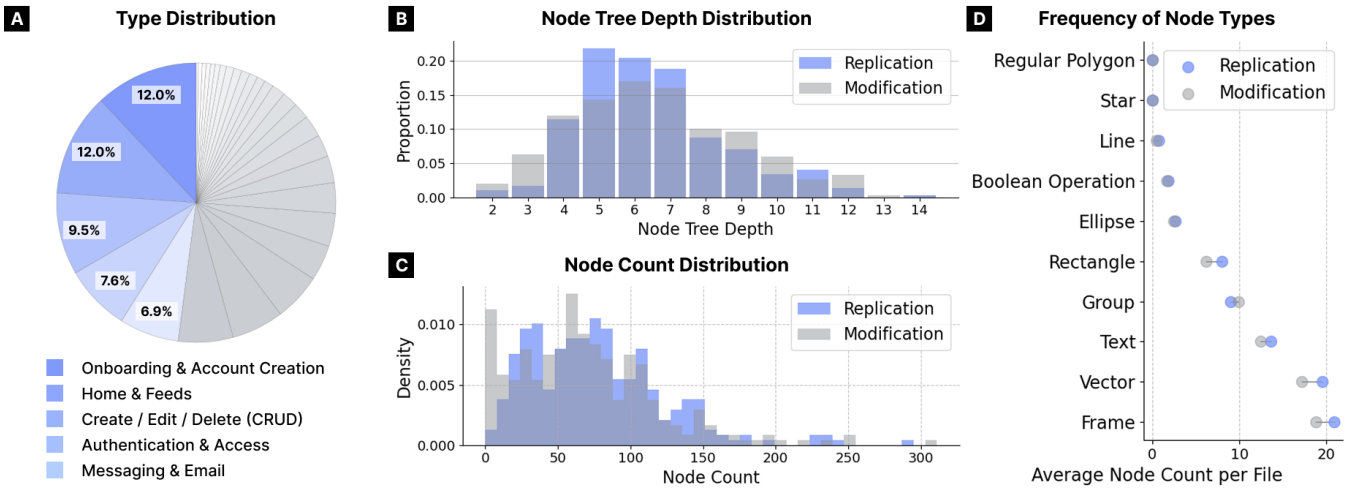

Figure 3: CANVAS Data Statistics: (A) Frequency of the five most frequent UI design types, with other categories grouped (see Appendix for full distribution). (B) The distribution of node tree depth is similar across the replication and modification sets with a Gaussian-like pattern. (C) The node count distribution is also similar across both sets. (D) The skewed frequency of node types per design indicates common patterns in component usage.

Data Statistics

Overview CANVAS contains 598 design tasks collected from 3,327 mobile UI designs across 30 categories collected through stratified sampling (Fig.3A). Each design includes an image and a JSON file representing Figma's node structure, where components (e.g., vector, text, rectangle) are organized in hierarchical trees with attributes such as position, size, and color (see Appendix for an example). Vector, text, and rectangle nodes are most common, while group and frame nodes also appear frequently (Fig.3D). Through the data refinement process, we remove outliers with an excessive number of nodes, reducing variance within the dataset.

Structural Characteristics We analyzed the structural properties of our benchmark tasks to ensure they offer an appropriate level of complexity for evaluating tool-based design capabilities. First, the node tree depth is slightly shallower in both replication ($M = 6.48,\ SD = 2.04$) and modification ($M = 6.62,\ SD = 2.40$) tasks than in the source dataset ($M = 7.06,\ SD = 2.32$) (Fig. 3B); the node count is also lower (replication: $M = 76.30$, modification: $M = 69.11$, source: $M = 126.25$; Fig. 3C), achieving a more efficient representation of the design while preserving core structure. Despite this reduction, the normalized *Shannon entropy* of the node types remains comparable, with $J = 0.763$ ($K = 10$) for replication, $J = 0.760$ ($K = 10$) for modification, and $J = 0.726$ ($K = 13$) for source dataset, retaining component type diversity in the design. The distribution of component types remains consistent across tasks, with nodes such as `frame`, `text`, and `vector` most frequent (Fig. 3D). These patterns suggest that our benchmark tasks simplify structural complexity while maintaining component-level information, enabling consistent evaluation of model performance.

Evaluation Metrics

Our evaluation framework measures perceptual and semantic alignment between generated and reference designs based on human visual processing. The framework decomposes visual similarity into three hierarchical levels that approximate the bottom-up stages of human visual perception: *features* (low-level properties such as edges and colors), *patterns* (mid-level compositions such as shapes and groupings), and *semantic objects* (high-level interpretations such as buttons or input fields) (Ware 2010). This hierarchy reflects how humans sequentially extract meaning from a design, progressively assembling basic visual features into recognizable patterns and eventually inferring functional design elements. Additionally, to capture model performance on component-level attribute editing (position, color, text), we include component-wise similarity, which evaluates how accurately individual UI elements are reproduced (see Appendix for implementation).

Structural Similarity Index Measure (*Feature-level*): SSIM compares local image statistics (mean, variance, and covariance) to assess how well low-level features such as size and shape are preserved (Wang et al. 2004).

Saliency Similarity (*Pattern-level*): Saliency similarity measures predicted human attention patterns within a design. We compute histogram intersection between normalized saliency maps of the ground truth and generated design using UEyes (Jiang et al. 2023), trained on eye-tracking data from UI screens.

BLIP Caption Similarity (*Object-level*): We generate captions for both ground-truth and generated designs using BLIP-2 and compute cosine similarity between their embeddings using SentenceTransformer (Reimers and Gurevych 2019), approximating semantic-level similarity.

Model	Replication				Modification			
	SSIM	Saliency	BLIP	Comp. Wise	SSIM	Saliency	BLIP	Comp. Wise
GPT-4o	0.739 (±0.172)	0.478 (±0.136)	0.495 (±0.250)	0.671 (±0.087)	0.843 (Δ0.136)	0.845 ($-\Delta$0.009)	0.740 ($-\Delta$0.021)	0.943 (Δ0.015)
GPT-4.1	0.767 (±0.129)	0.612 (±0.137)	**0.655 (±0.251)**	**0.716 (±0.075)**	**0.890** (Δ0.183)	**0.861** (Δ0.007)	**0.806** (Δ0.044)	**0.951** (Δ0.024)
Claude-3.5-Sonnet	0.725 (±0.180)	0.483 (±0.151)	0.518 (±0.272)	0.666 (±0.089)	0.816 (Δ0.109)	0.858 (Δ0.004)	0.775 (Δ0.013)	0.946 (Δ0.018)
Gemini-2.5-Flash	0.736 (±0.184)	0.619 (±0.149)	0.571 (±0.270)	0.702 (±0.100)	0.874 (Δ0.167)	0.857 (Δ0.003)	0.784 (Δ0.023)	0.948 (Δ0.020)
Gemini-2.5-Pro	**0.774 (±0.117)**	**0.630 (±0.162)**	0.620 (±0.273)	0.694 (±0.094)	0.867 (Δ0.159)	0.851 ($-\Delta$0.003)	0.804 (Δ0.043)	0.935 (Δ0.007)

± indicates the standard deviation; Δ indicates the average score increase from the base design; ■ marks the best score in each column.

Table 1: CANVAS scores on replication and modification tasks. Each score indicates the average similarity scores between completed designs and the ground truth. In replication, GPT-4.1 and Gemini-2.5-Pro exhibit leading performance in the replication task, while GPT-4.1 consistently exhibits robust performance in the modification task.

Component-wise Similarity Following Design2Code (Si et al. 2025) and related work (Li, Zhang, and Yang 2025), we perform one-to-one component matching using the Hungarian algorithm (IoU for visual elements, text-position similarity for text), then compute similarity across four attributes: component match rate, position (Euclidean distance), text content (F1 score), and fill color (RGB distance).

Metric Aggregation For both tasks, we evaluate the similarity between the generated design s_t and the reference design s_{GT}, reporting the final score as the average similarity values across all benchmark cases. In the modification task, we measure how much the model's edits improve similarity to the ground truth s_{GT} by comparing scores before and after editing.

Experiments

Configuration

Pipeline CANVAS implements a tool-based UI design generation pipeline based on the Model Context Protocol (MCP)[2]. Models have access to 50 predefined tools for design operations, including creation, deletion, layout adjustment, styling, and content modification, which are relayed to Figma via MCP (See Appendix for details). Models initiate each task by receiving the target screen UI image and task instructions, employing predefined tools; for modification tasks specifically, the pipeline initializes existing designs on the Figma canvas from a structured JSON representation. In each task, models operate within an agentic loop based on the ReAct framework (Yao et al. 2023), which involves cycles of thought, action (tool invocation), and observation.

Model Setup We evaluate five state-of-the-art vision-language models (VLMs) with visual understanding and tool-use capabilities, including GPT-4o (OpenAI 2024), GPT-4.1 (OpenAI 2025), Claude-3.5-Sonnet (Anthropic 2024), Gemini-2.5-Flash and Pro (Comanici et al. 2025). All models are instructed using standardized instructions with the temperature level at 0 for reproducibility (See Appendix for the instructions). We excluded open-source models due to their high failure rates in multi-turn tool invocation. These models commonly terminated after one or two turns, resulting in blank or incomplete outputs.

[2]Architecture building upon https://github.com/grab/cursor-talk-to-figma-mcp, MIT License

Main Results

Table 1 shows average similarity scores between completed designs and ground truth across the replication and modification tasks. The score with (±) represents the standard deviation in the replication task, while the score with (Δ) indicates score changes due to model actions from the initial state s_{old} to the target state s_{GT} in the modification task.

Replication As shown in Table 1, Gemini-2.5-Pro scored highest in SSIM and saliency, indicating strength in feature-to-pattern-level similarity (contours and shape compositions). GPT-4.1 achieved the best scores for BLIP and component-wise similarity, indicating strength in replicating design semantics and preserving component attributes (position, color, text)

Modification GPT-4.1 achieved the highest scores across all four metrics. Importantly, several models demonstrate small or negative Δ values, indicating score stagnation or degradation due to model actions. We explore these results in the analysis section.

Analysis

To understand performance differences across models, we analyzed tool invocation histories at each turn. The histories directly capture model actions and their influence on overall scores. Our analysis reveals two primary observations: (i) models achieving higher scores in the replication task demonstrate diverse tool invocations over increased turns; and (ii) models performing well in the modification task exhibit precise, targeted tool invocations within fewer turns.

Replication task demands diverse invocation of tools. We identify that in the replication task, high-scoring models exhibit more diverse tool patterns. Figure 4 shows average tool invocation frequency and tool type diversity by model on each task. The higher-performing models, namely GPT-4.1 and Gemini-2.5-Pro, display increased tool diversity, indicating more strategic behaviors. For example, we observe a modular approach, where a model creates a component (e.g., a buy button) once and propagates it across the design using the `copy_node` action, saving turns. In contrast, lower-performing models, such as GPT-4o and Claude-3.5-Sonnet, complete the design by simply creating components in order. These observations suggest that intrinsic rewards encouraging exploration could help models learn strategic, diverse tool patterns.

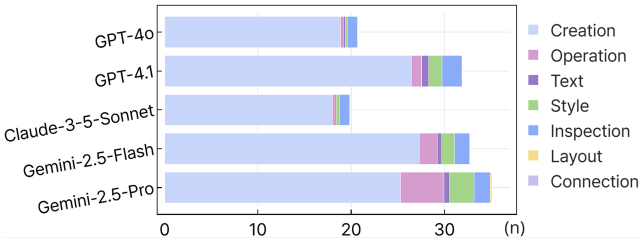

Figure 4: Tool Invocation Frequency. Average tool invocations per task $x\,axis$ (n). Colored blocks show tool types (e.g., creation includes `create_rectangle`). Higher-performing models exhibit greater tool diversity.

Modification task requires precise selection of tools. We posit that the modification task poses a unique challenge for models due to its high granularity. It requires precise tool invocations, where slight inaccuracies lead to non-uniform impacts across similarity metrics. For instance, adding a line break may minimally affect textual content but can cause a large shift in a saliency map. This phenomenon is reflected in the small or negative Δ values in Table 1 (Modification); due to non-uniform shifts, the score changes converge towards zero on average.

For a more in-depth understanding, we introduce Pos@K, a metric counting the number of cases where the number (K) of similarity scores increases after an edit. For instance, A Pos@3 indicates the number of cases where the model edit introduced a positive increase in *three* metrics and a decrease in the remaining metrics. Table 2 shows the distribution of Pos@K values for each model in the modification Task 1, *attribute update*, in ratio. Notably, in the table, models that underperformed elsewhere — Claude-3.5-Sonnet and Gemini-2.5-Flash — yielded a higher proportion of Pos@4 than other models.

Model	Pos@4	Pos@3	Pos@2	Pos@1	Pos@0	F
GPT-4o	24.0%	**41.0%**	18.0%	13.0%	2.0%	2.0%
GPT-4.1	25.0%	31.0%	**23.0%**	14.0%	6.0%	1.0%
Claude-3.5-Sonnet	30.0%	38.0%	17.0%	10.0%	4.0%	1.0%
Gemini-2.5-Flash	**31.0%**	37.0%	20.0%	9.0%	3.0%	0.0%
Gemini-2.5-Pro	27.0%	36.0%	12.0%	**16.0%**	**8.0%**	1.0%

: the highest value in each column. F: the ratio of failed cases.

Table 2: Distribution of Pos@K cases in ratio for each model in Task 1: Attribute Update.

To investigate this performance inversion, we compared tools invoked by models against human-annotated tools for each task. Three designers annotated the required tools for each of the 300 modification cases, serving as reference annotations (see Appendix for details). As shown in Table 3, Gemini-2.5-Pro and GPT-4.1 exhibited lower tool precision but higher diversity compared to Claude-3.5-Sonnet and Gemini-2.5-Flash. This suggests that excessive tool diversity impaired performance in this task. The finding generalizes across all modification tasks: treating Pos@K as an ordinal score, we found a positive correlation between tool precision and the Pos@K score ($\rho = 0.149, p < 0.01$) and a negative correlation between tool diversity and Pos@K ($\rho = -0.365, p < 0.01$). These results indicate that precise tool selection is critical for the modification task. To improve precision, methods that teach accurate skills, such as imitation learning, are necessary.

Model	Tool Precision	Tool Recall	Tool Diversity
GPT-4o	0.4195	0.9967	4.3100
GPT-4.1	0.5434	1.0000	5.4433
Claude-3.5-Sonnet	0.5677	0.9900	3.6300
Gemini-2.5-Flash	0.5943	0.9767	3.4067
Gemini-2.5-Pro	0.5659	0.9633	4.1400

Tool Precision and Recall: normalized intersection with human-annotated tools. Tool Diversity: average unique tools per case.

Table 3: Comparison of tool invocation statistics per model against human-annotated required tool data.

Our metrics align with design experts. A study with human annotators reveals that our metrics closely align with human preference. Table 4 demonstrates that three metrics, saliency, BLIP, and component-wise similarity, are statistically significant predictors of pairwise human judgments on design similarity.

Following the methodology of Design2Code (Si et al. 2025), we collected pairwise human preference data from design experts on Prolific. We used 100 design cases, chosen from our replication results via stratified sampling (weighted by UI type and complexity). Each pair received "win," "lose," or "tie" labels from three annotators, with final outcomes determined by majority vote, excluding cases without a majority. Subsequently, we trained a logistic regression model to predict the binary preference (win=1, lose=0) using the difference in our metric scores between two designs in a pair as the independent variable. On a 50/50 training/test split of the 363 non-tie samples, the model achieved 75% prediction accuracy.

	coef	std err	z	p
SSIM	-0.0500	0.2241	-0.223	0.82336
Saliency	0.9808	0.2758	3.557	0.00038
BLIP	0.7323	0.2357	3.106	0.00189
Comp. Wise	0.5836	0.2806	2.080	0.03506

Table 4: Logistic regression on human pairwise preferences using similarity metrics as features.

Ablation Study

We conducted an ablation study to isolate the effects of multi-turn iteration and tool-based interaction. (i) The baseline setting is tool-based multi-turn, which is identical to the setting in the CANVAS benchmark. To control the effect of the turns, we include the (ii) tool-based single-turn condition, where all tool invocations are generated at once without intermediate feedback. This enables a direct comparison against our third condition, (iii) a code-based single-turn adapted from prior work (Si et al. 2025), thereby isolating the impact of using tools.

Method	Model	Replication		
		SSIM	Saliency	BLIP
Code (Single-Turn)	GPT-4.1	0.773 (±0.110)	0.649 (±0.129)	**0.689 (±0.231)**
	Gemini-2.5-Pro	0.771 (±0.108)	0.696 (±0.120)	0.685 (±0.233)
Tool (Single-Turn)	GPT-4.1	0.704 (±0.219)	0.510 (±0.156)	0.534 (±0.281)
	Gemini-2.5-Pro	**0.775 (±0.118)**	**0.699 (±0.121)**	0.660 (±0.259)
Tool (Agent)	GPT-4.1	0.767 (±0.129)	0.612 (±0.137)	0.655 (±0.251)
	Gemini-2.5-Pro	0.774 (±0.117)	0.630 (±0.162)	0.620 (±0.273)

: the *best overall* in each column.

Table 5: Ablation study results for each generation method.

Table 5 shows that Gemini-2.5-Pro achieves its highest scores in the single-turn tool-based setting, while GPT-4.1 performs better under the multi-turn condition. The single-turn and multi-turn settings induce distinct model behaviors: the multi-turn setting encourages strategic, non-linear operations (e.g., layer reordering), while the single-turn setting favors more uniform, linear action sequences. This divergence explains the performance results in Table 5: the riskier strategic operations (e.g., copying incorrectly designed buttons) can degrade output quality, leading to lower scores in some design cases. Nevertheless, we propose that this multi-turn, agentic configuration holds important potential for complex, collaborative design tasks.

Error Cases

We conducted an error analysis to identify common failure cases of current models and their underlying causes.

Geometric Operation Errors Models frequently fail to control the geometric properties of UI elements, leading to errors in count, directionality, and spatial arrangement. For instance, in the map user interface shown in Fig. 5, the model produces (A) an incorrect number of location markers and (B) paths with incoherent directions.

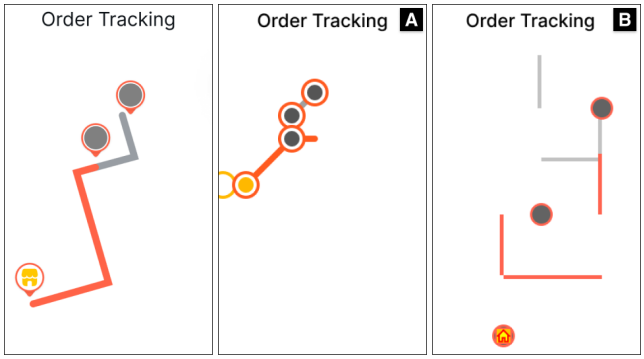

Figure 5: Error case 1. The models (A) miscount the markers and (B) draw irregular lines (see Appendix for full image).

Layout Operation Errors Models reveal erroneous reasoning about screen auto-layout. In Figma, this feature automatically realigns and resizes child elements when their parent resizes; similar mechanisms appear in other editors (Figma 2025b; Sketch 2025). Because any modification to the layout parameters can propagate through an entire hierarchy, the task demands prediction of changes without dictating their values through commands. As shown in Fig. 6, the models' adjustment to auto-layout disintegrates the selector button and its placeholder (A) and forces the placeholder below the viewport (B).

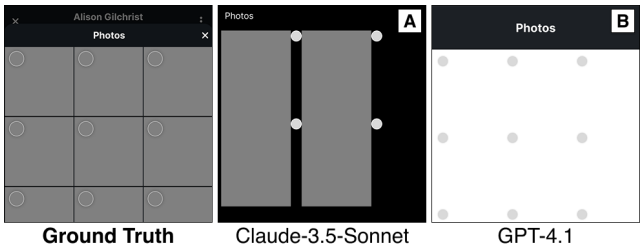

Figure 6: Error case 2. Updates to auto-layout settings trigger (A) disintegration between selector and placeholder or (B) push the placeholder downwards beyond the screen (see Appendix for full image).

Text Operation Errors Models commonly fail to infer appropriate dimensions from text attributes (e.g., font, size, and spacing), leading to narrowly sized text components and subsequent text overflow. Figure 7 illustrates two typical failures: (A) the price label extends beyond its bounding box, breaking visual alignment, and (B) the button label overruns its frame, blending into the background.

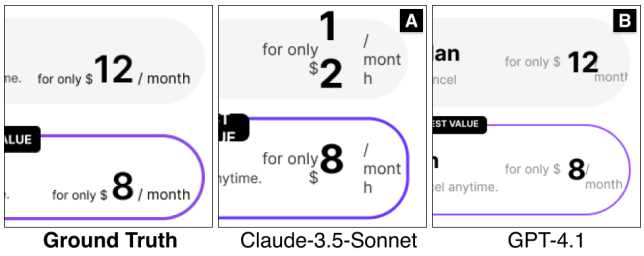

Figure 7: Error case 3. Models (A, B) fail to assign sufficient span to the text component and trigger text overflow (see Appendix for full image).

Conclusion

We introduce CANVAS, the first benchmark designed to evaluate vision-language models (VLMs) on tool-based UI design tasks. Our findings suggest that current state-of-the-art models exhibit promising capabilities in replicating and modifying interface designs through tool invocation with design software. Particularly, high-performing models exhibit more diverse tool use and strategic behaviors. Nonetheless, adverse tool selection and design errors highlight the limitations in the implementation framework and the model capacity. By presenting an initial evaluation of VLMs performing design tasks within conventional software, CANVAS offers valuable insights toward achieving human-aligned design automation with VLMs.

Acknowledgements

We thank Chulwoo Kim, May Jorella Lazaro, and Ji Soo Yi from the CX Insight Team (MX Division), Samsung Electronics for insights on UI design workflows, and members of the KAIST Interaction Lab (KIXLAB) for their constructive feedback. This work was supported by the CX Insight Team (MX Division), Samsung Electronics, and by an IITP grant funded by the Korean government (MSIT) (No. RS-2024-00443251, Accurate, Safe, Multimodal, & Multilingual Personalized AI Tutors).

References

Anthropic. 2024. Introducing Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. Accessed: 2025-05-15.

Beltramelli, T. 2018. pix2code: Generating Code from a Graphical User Interface Screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '18. New York, NY, USA: Association for Computing Machinery. ISBN 9781450358972.

Cheng, C.-Y.; Huang, F.; Li, G.; and Li, Y. 2023. PLay: parametrically conditioned layout generation using latent diffusion. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Choi, D.; Hong, S.; Park, J.; Chung, J. J. Y.; and Kim, J. 2024. CreativeConnect: Supporting Reference Recombination for Graphic Design Ideation with Generative AI. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24. New York, NY, USA: Association for Computing Machinery. ISBN 9798400703300.

Comanici, G.; Bieber, E.; Schaekermann, M.; Pasupat, I.; Sachdeva, N.; Dhillon, I.; Blistein, M.; Ram, O.; Zhang, D.; Rosen, E.; et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; and Kumar, R. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, 845–854. New York, NY, USA: Association for Computing Machinery. ISBN 9781450349819.

Ferreira, J.; Noble, J.; and Biddle, R. 2007. Agile Development Iterations and UI Design. In *Agile 2007 (AGILE 2007)*, 50–58.

Figma. 2024. Figma: The Collaborative Interface Design Tool. https://www.figma.com/. Accessed: 2025-05-13.

Figma. 2025a. Figma Community — Explore Free Design Files, Plugins & More. https://www.figma.com/community. Accessed: 2025-05-13.

Figma. 2025b. Guide to auto layout. https://help.figma.com/hc/en-us/articles/360040451373-Guide-to-auto-layout. Accessed 28-07-2025.

Garg, A.; Jiang, Y.; and Oulasvirta, A. 2025. Controllable GUI Exploration. *CoRR*, abs/2502.03330.

Goldschmidt, G. 1991. The dialectics of sketching. *Creativity Research Journal*, 4(2): 123–143.

Gui, Y.; Li, Z.; Wan, Y.; Shi, Y.; Zhang, H.; Chen, B.; Su, Y.; Chen, D.; Wu, S.; Zhou, X.; Jiang, W.; Jin, H.; and Zhang, X. 2025a. WebCode2M: A Real-World Dataset for Code Generation from Webpage Designs. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, 1834–1845. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712746.

Gui, Y.; Wan, Y.; Li, Z.; Zhang, Z.; Chen, D.; Zhang, H.; Su, Y.; Chen, B.; Zhou, X.; Jiang, W.; and Zhang, X. 2025b. UICopilot: Automating UI Synthesis via Hierarchical Code Generation from Webpage Designs. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, 1846–1855. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712746.

Jiang, Y.; Leiva, L. A.; Rezazadegan Tavakoli, H.; R. B. Houssel, P.; Kylmälä, J.; and Oulasvirta, A. 2023. UEyes: Understanding Visual Saliency across User Interface Types. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23. New York, NY, USA: Association for Computing Machinery. ISBN 9781450394215.

Kim, T. S.; Choi, D.; Choi, Y.; and Kim, J. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22. New York, NY, USA: Association for Computing Machinery. ISBN 9781450391573.

Laurençon, H.; Tronchon, L.; and Sanh, V. 2024. Unlocking the conversion of Web Screenshots into HTML Code with the WebSight Dataset. arXiv:2403.09029.

Li, G.; Baechler, G.; Tragut, M.; and Li, Y. 2022. Learning to Denoise Raw Mobile UI Layouts for Improving Datasets at Scale. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22. New York, NY, USA: Association for Computing Machinery. ISBN 9781450391573.

Li, J.; Cao, H.; Lin, L.; Hou, Y.; Zhu, R.; and El Ali, A. 2024. User Experience Design Professionals' Perceptions of Generative Artificial Intelligence. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24. New York, NY, USA: Association for Computing Machinery. ISBN 9798400703300.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 19730–19742. PMLR.

Li, J.; Yang, J.; Hertzmann, A.; Zhang, J.; and Xu, T. 2019. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Li, R.; Zhang, Y.; and Yang, D. 2025. Sketch2Code: Evaluating Vision-Language Models for Interactive Web Design

Prototyping. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 3921–3955. Albuquerque, New Mexico: Association for Computational Linguistics. ISBN 979-8-89176-189-6.

Lu, Y.; Yang, Y.; Zhao, Q.; Zhang, C.; and Li, T. J.-J. 2024. AI Assistance for UX: A Literature Review Through Human-Centered AI. arXiv:2402.06089.

Lu, Y.; Zhang, C.; Zhang, I.; and Li, T. J.-J. 2022. Bridging the Gap Between UX Practitioners' Work Practices and AI-Enabled Design Support Tools. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22. New York, NY, USA: Association for Computing Machinery. ISBN 9781450391566.

Mobbin Ltd. 2025. Mobbin: UI & UX Design Inspiration for Mobile & Web Apps. https://mobbin.com/. Accessed: 2025-05-13.

Moran, K.; Bernal-Cárdenas, C.; Curcio, M.; Bonett, R.; and Poshyvanyk, D. 2020. Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. *IEEE Transactions on Software Engineering*, 46(2): 196–221.

Niu, R.; Li, J.; Wang, S.; Fu, Y.; Hu, X.; Leng, X.; Kong, H.; Chang, Y.; and Wang, Q. 2024. ScreenAgent: a vision language model-driven computer control agent. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24. ISBN 978-1-956792-04-1.

OpenAI. 2024. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*. Accessed: 2025-05-15.

OpenAI. 2025a. GPT-4.1 Mini. https://openai.com/index/gpt-4-1/. Accessed: 2025-05-13.

OpenAI. 2025b. GPT-4.5: OpenAI's Large Language Model. https://openai.com/index/gpt-4-5. Accessed: 2025-05-15.

OpenAI. 2025. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/. Accessed: 2025-05-15.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992. Hong Kong, China: Association for Computational Linguistics.

Resnick, M.; Myers, B.; Nakakoji, K.; Shneiderman, B.; Pausch, R.; Selker, T.; and Eisenberg, M. 2005. Design principles for tools to support creative thinking.

Si, C.; Zhang, Y.; Li, R.; Yang, Z.; Liu, R.; and Yang, D. 2025. Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 3956–3974. Albuquerque, New Mexico: Association for Computational Linguistics. ISBN 979-8-89176-189-6.

Sketch. 2025. Stack Layout — sketch.com. https://www.sketch.com/docs/designing/stack-layout/. Accessed 28-07-2025.

Son, K.; Choi, D.; Kim, T. S.; and Kim, J. 2024. Demystifying Tacit Knowledge in Graphic Design: Characteristics, Instances, Approaches, and Guidelines. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24. New York, NY, USA: Association for Computing Machinery. ISBN 9798400703300.

Stolterman, E.; and Pierce, J. 2012. Design tools in practice: studying the designer-tool relationship in interaction design. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, 25–28. New York, NY, USA: Association for Computing Machinery. ISBN 9781450312103.

Tang, Z.; Wu, C.; Li, J.; and Duan, N. 2024. LayoutNUWA: Revealing the Hidden Layout Expertise of Large Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Wan, Y.; Wang, C.; Dong, Y.; Wang, W.; Li, S.; Huo, Y.; and Lyu, M. 2025. Divide-and-Conquer: Generating UI Code from Screenshots. *Proc. ACM Softw. Eng.*, 2(FSE).

Wang, Z.; Bovik, A.; Sheikh, H.; and Simoncelli, E. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612.

Ware, C. 2010. *Visual thinking for design*. Elsevier.

Xiao, J.; Wan, Y.; Huo, Y.; Wang, Z.; Xu, X.; Wang, W.; Xu, Z.; Wang, Y.; and Lyu, M. R. 2025. Interaction2Code: Benchmarking MLLM-based Interactive Webpage Code Generation from Interactive Prototyping. arXiv:2411.03292.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yuan, M.; Chen, J.; Hu, Y.; Feng, S.; Xie, M.; Mohammadi, G.; Xing, Z.; and Quigley, A. J. 2025. Towards Human-AI Synergy in UI Design: Supporting Iterative Generation with LLMs. *ACM Trans. Comput.-Hum. Interact.* Just Accepted.

Yun, S.; Lin, H.; Thushara, R.; Bhat, M. Q.; Wang, Y.; Jiang, Z.; Deng, M.; Wang, J.; Tao, T.; Li, J.; Li, H.; Nakov, P.; Baldwin, T.; Liu, Z.; Xing, E. P.; Liang, X.; and Shen, Z. 2024. Web2Code: A Large-scale Webpage-to-Code Dataset and Evaluation Framework for Multimodal LLMs. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 112134–112157. Curran Associates, Inc.

Zhao, T.; Chen, C.; Liu, Y.; and Zhu, X. 2021. GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 748–760.

Zhou, T.; Zhao, Y.; Hou, X.; Sun, X.; Chen, K.; and Wang, H. 2025. DeclarUI: Bridging Design and Development with Automated Declarative UI Code Generation. *Proc. ACM Softw. Eng.*, 2(FSE).