OpenGenUI: Enhancing Generative UI in MLLMs via Autonomous Exploration and Agentic Reinforcement Learning

Anonymous ACL submission

Abstract

Generating user interfaces from instructions, recently termed Generative UI, has emerged as an important capability of MLLMs. However, this task is still limited by the high cost of paired image–text supervision and the difficulty of designing reliable rewards for interactive generation. We propose OpenGenUI, a three-stage training framework that turns the open web into a multimodal training resource for Generative UI in MLLMs. Webpages are collected at web scale through autonomous exploration over diverse real-world websites, and their visual appearance is reconstructed into clean, self-contained image–HTML supervision for cold-start SFT. Image-to-code reinforcement learning then uses visual similarity rewards computed from rendered outputs, enabling optimization for visual fidelity without requiring source code targets. Both stages can scale naturally with the open web. For open-ended text-to-code generation, we further introduce *Verifiable Interaction Reward* (VIR), which formulates reward computation as agentic browser evaluation. A generated webpage is interacted with through web actions, and state changes between adjacent states are extracted as evidence of interactive behavior. Experiments show that OpenGenUI improves generation quality across both reconstruction and open-ended settings.

1 Introduction

Recent MLLMs are increasingly used to produce not only text but also rendered visual and interactive interfaces (OpenAI, 2026; GLM-5-Team, 2026), recently termed *Generative UI* (Leviathan et al., 2026), where a model generates complete webpages from natural language instructions (Google, 2026; Gui et al., 2025c).

Progress on this task faces two practical challenges. The first is the high cost of paired image–text supervision. The scaling of LLMs has been driven in large part by the abundance of text on the web, yet the same web, which is natively multimodal, has not been comparably exploited for vision language training. A natural idea is to crawl raw HTML together with rendered screenshots from the open web, but most modern websites are compiled from JavaScript frameworks such as Vue and React, producing markup that does not resemble the clean, author-readable code a generative model should emit (Sun et al., 2025; Lu et al., 2025c; Liu et al., 2024). A second route is to synthesize image–code pairs with a stronger model, but such purely synthetic corpora tend to concentrate on the layouts and stylistic patterns of the generator's own priors and lack the visual diversity of real pages. Curating supervision-ready image–HTML pairs that scale to the open web without manual annotation thus remains an open problem for training Generative UI models.

The second challenge is the difficulty of designing reliable rewards for interactive generation. When webpages are produced from text prompts alone, neither a reference screenshot nor ground-truth code is available, and the reward must reflect how the generated page actually behaves under user interaction (Chen et al., 2025; Guo et al., 2025; Xiao et al., 2025). Consequently, the reward mechanism must inherently possess the capabilities of multimodal understanding, user interaction simulation, and runtime execution feedback (Li et al., 2025b; Tao et al., 2023). Specifically, it must not only comprehend the complex joint representation of visual layouts and underlying code, but also actively simulate diverse user behavior sequences while accurately capturing the resultant state transitions from code execution. Designing a reward with these capabilities is exceptionally challenging, as it must accurately capture dynamic page behaviors while remaining reproducible and scalable for large-scale RL training.

To address these challenges, we introduce Open-

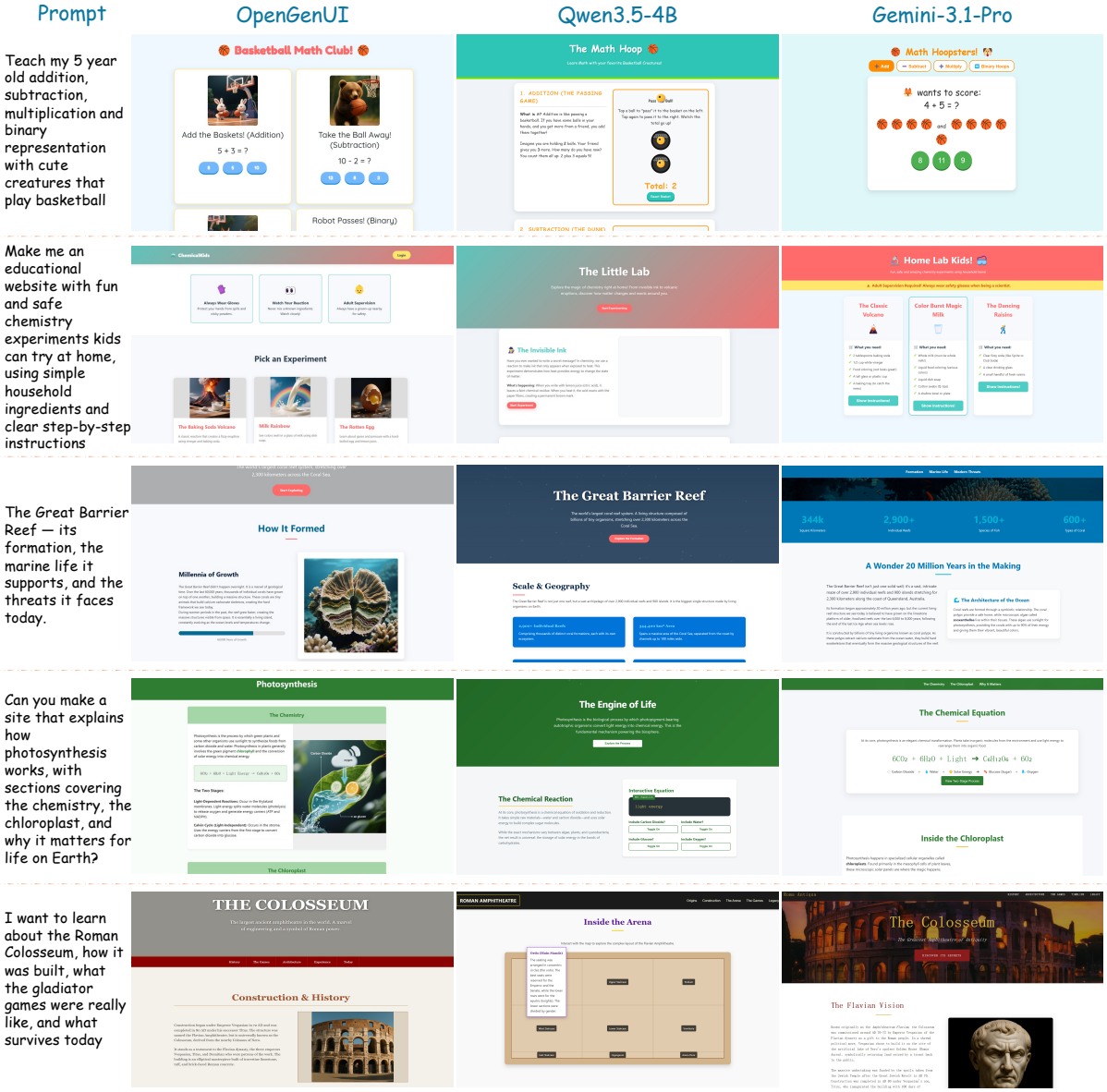

Figure 1: Qualitative comparison on open-ended prompts. OpenGenUI generates visually richer and more coherent webpages than the Qwen3.5-4B base backbone, and approaches the quality of Gemini-3.1-Pro.

GenUI, a progressive three-stage training framework powered by a scalable data reconstruction pipeline. Stage-1 utilizes this pipeline to collect open-web pages via autonomous browser exploration and reconstruct them into clean image–HTML pairs for cold-start SFT. Stage-2 applies image-to-code GRPO driven entirely by rendered visual similarity rewards, enabling scalable RL optimization by relying solely on visual feedback. Stage-3 introduces *Verifiable Interaction Reward* (VIR) for open-ended text-to-code generation, where a browser agent interacts with the generated page and scores its interactive behaviors.

As shown in Figure 1, OpenGenUI is validated on a compact multimodal backbone (Qwen3.5-4B (Team, 2026)). On open-ended prompts, the resulting 4B model is rated comparably to Gemini-3.1-Pro in a human study, and VIR produces more interaction-aware webpages than reward designs based on a single rendered state. The main contributions are summarized as follows:

- A scalable supervision pipeline for Generative UI that turns autonomously crawled webpages into clean image–HTML pairs through visual reconstruction, and uses rendered visual similarity as a reinforcement learning reward for both image-to-code cold-start training and the visual-reward optimization stage.
- *Verifiable Interaction Reward* (VIR) for open-ended text-to-code generation, which formulates reward computation as agentic browser

evaluation and uses pixel-level state changes induced by primitive web actions as evidence of interactive behavior beyond what a single rendered state can reveal.

- Experiments on compact MLLMs show that OpenGenUI improves generation quality on both reconstruction and open-ended settings, and that VIR produces more interaction-aware webpages than reward designs that score the generated page only from a single static rendered observation.

2 Related Work

2.1 Multimodal Large Language Models

Multimodal large language models (MLLMs) have substantially advanced visual reasoning, document understanding, and code generation (OpenAI, 2026; Bytedance, 2026; GLM-5-Team, 2026). Proprietary systems such as Gemini (Google, 2026) and Claude (Anthropic, 2026) have demonstrated strong multimodal interaction and tool-use capabilities in practical applications. Recent open-weight foundation models (DeepSeek-AI, 2026; Team, 2026) further show strong coding abilities. In parallel, GUI-centric models have been developed for interface understanding and operation (Awal et al., 2025; Li et al., 2025a; Lu et al., 2025b). CogAgent (Hong et al., 2024), OmniParser (Wan et al., 2024), and UI-TARS (Qin et al., 2025) show strong performance in screen parsing, element grounding, and autonomous GUI navigation (Yang et al., 2025a; Lù et al., 2025; Lu et al., 2025a). However, existing GUI models focus primarily on perceiving or operating interfaces. They do not inherently address the inverse, highly structured task of UI generation.

2.2 Web Interface Generation

Transitioning from perception to generation, MLLM-based web interface synthesis now represents a frontier in front-end automation. Early work targets layout reconstruction from screenshots using curated pairs that establish foundational benchmarks (Beltramelli, 2018; Si et al., 2025). To support training at scale, subsequent efforts construct large-scale corpora with diverse layout metadata and visual instruction pairs (Yun et al., 2024; Laurençon et al., 2024; Gui et al., 2025a). More recent benchmarks broaden the setting toward visual diversity, structural fidelity, responsive design, and functional consistency (Lai et al., 2025; Lin et al., 2025; Wang et al., 2025; Zhang et al., 2025; Chen et al., 2025; Awal et al., 2025), and further extend the task to interactive prototyping and end-to-end functional websites (Xiao et al., 2025; Tao et al., 2023). On the modeling side, representative approaches include layout-structured synthesis (Gui et al., 2025b; Wu et al., 2025b; Gui et al., 2025c), agent-based pipelines with planning subagents (Jiang et al., 2025), and iterative refinement that uses visual feedback to reduce rendering discrepancies (Yang et al., 2025b; Li et al., 2025b; Yang et al., 2025a). Although reinforcement learning frameworks excel in text reasoning (Shao et al., 2024; Zheng et al., 2025; Yu et al., 2026), their expansion to Generative UI remains constrained by interaction-aware reward signals.

3 Method

3.1 Overview

OpenGenUI is a three-stage framework. Stage-1 performs cold-start SFT, Stage-2 introduces GRPO driven by a rendered visual-similarity reward, and Stage-3 extends this policy optimization with an agentic reward derived from browser-agent interaction (Zhuge et al., 2024). The overall framework is shown in Figure 2.

3.2 Autonomous Web Exploration

A challenge for training Generative UI models is paired image–code supervision. Such data is expensive to obtain at scale through manual annotation, and raw HTML crawled from the open web is largely unusable as direct supervision. A common alternative is to synthesize image–code pairs with a stronger model from a fixed set of seed prompts or templates. However, such purely synthetic corpora tend to collapse onto a narrow band of layouts, color schemes, and component patterns characteristic of the generator's own priors, which limits the diversity of visual surfaces a downstream model is exposed to. The open web, in contrast, exhibits the layout, stylistic, and content diversity that synthetic data struggles to reproduce, and pairs each rendered page with reusable image, icon, and logo assets. An autonomous browser agent can therefore harvest paired visual and asset signals at scale, yielding the exploration pipeline shown in Figure 3.

Inputs. The exploration pipeline is driven by Insta-150k-v3 (Trabucco et al., 2025), which provides a large pool of webpage URLs together with per-URL exploration tasks (e.g., navigation goals such

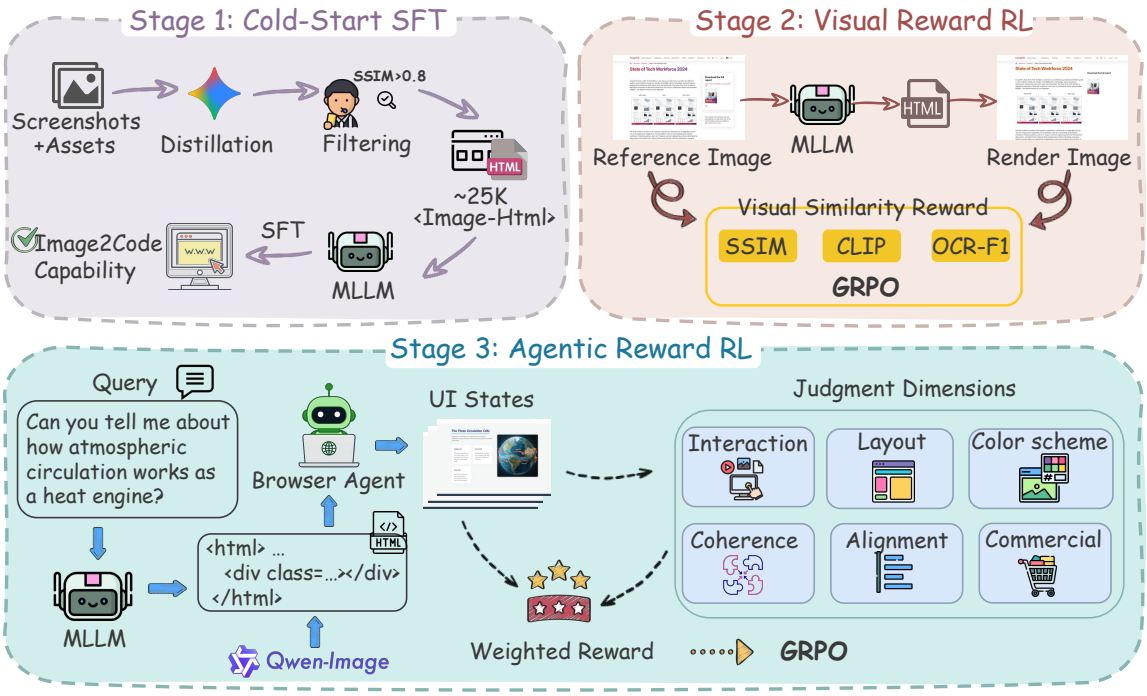

Figure 2: Overview of OpenGenUI. Stage-1 constructs clean image–HTML supervision from autonomously explored webpages and applies cold-start SFT. Stage-2 aligns image-to-code generation with rendered visual similarity rewards. Stage-3 optimizes open-ended text-to-code generation with the proposed Verifiable Interaction Reward, which uses browser-induced state transitions as interaction evidence.

as *"Click on the 'How you can help' link"*), specifying both *where* the agent goes and *what* it does on each page.

Browser Agent. For each input, a browser agent operates inside a sandboxed environment with a virtual display, performing a small set of primitive web actions (Navigate, Loading, Click, Capture, and Download) and following the per-URL task to reach exploration-worthy page states. At each visited state, the agent captures full-page screenshots and downloads the page's image, logo, and icon assets.

Outputs. For each successfully explored URL, the pipeline produces multiple per-step screenshots together with the page's asset grid. Screenshots provide the visual targets for Stage-1 reconstruction, and assets provide reusable visual content that grounds the reconstructed pages in real imagery rather than placeholders. Because the pipeline operates on rendered pages and reusable assets rather than on framework-compiled source markup, both data curation in Stage-1 and reward computation in Stage-2 can scale with the volume of crawled web.

3.3 Stage-1: Cold-Start SFT

Stage-1 turns the screenshots and assets collected in Section 3.2 into supervision-ready image–HTML pairs and uses them for cold-start SFT, equipping

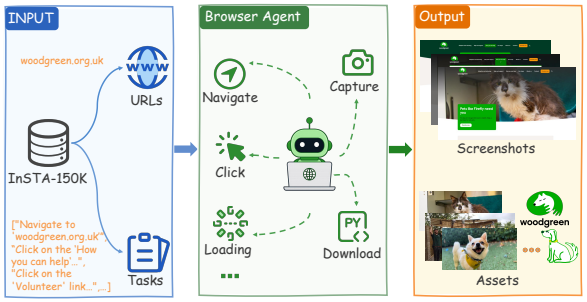

Figure 3: Autonomous web exploration. A browser agent is driven by URLs and exploration tasks from Insta-150k-v3, executes actions, and produces per-step screenshots together with the page's assets.

the policy with a basic image-to-code capability before either reinforcement learning stage is applied.

Visual-to-Code Distillation. Each captured screenshot, paired with its asset grid, is provided to a vision–language model that is prompted to reconstruct a self-contained Tailwind CSS HTML page reproducing the screenshot. The reconstructed HTML is clean, framework-agnostic, and paired by construction with the source screenshot.

Visual Fidelity Filtering. Reconstruction does not always succeed. Rendering may fail, or the rendered output may diverge from the original screenshot. Each reconstructed page is therefore re-rendered in a headless browser and compared to the source screenshot with SSIM, retaining only pairs

satisfying SSIM > 0.8. This filter removes both rendering failures and low-fidelity reconstructions.

SFT. The curated corpus is used to fine-tune the base MLLM with LoRA on the image-to-code task by minimizing the cross-entropy loss over reconstructed HTML, providing an initialization for the subsequent reinforcement learning stages.

3.4 Stage-2: Visual Reward RL

We first formalize the reward used in the reinforcement learning stages. Given a natural-language prompt x, the policy π_θ generates an HTML page y. A rendering function $\mathcal{R}(y)$ executes y in a headless browser under a predefined action policy and produces an ordered sequence of interaction states $\mathcal{R}(y) = (s_0, s_1, \ldots, s_T)$, where each s_t is the rendered screenshot after the t-th browser-agent action. An evaluator $\mathcal{E}$ extracts verifiable evidence from $\mathcal{R}(y)$, and a multimodal judge $\mathcal{J}$ consumes this evidence to produce a scalar reward,

$$r(y \mid x) = \mathcal{J}\big(\mathcal{E}(\mathcal{R}(y)),\ x\big). \tag{1}$$

Static rewards correspond to reducing $\mathcal{R}(y)$ to a single screenshot s_0, which is the case in Stage-2. Stage-3 (Section 3.5) replaces $\mathcal{R}$ with a full browser-agent interaction trace executed against the page in a real browser environment.

Stage-2 shifts the optimization target from token reproduction to visual fidelity, evaluated directly over the rendered output. Token-level imitation fails to capture visual equivalence, since structurally different HTML can render identically while trivial syntactic deviations incur disproportionate penalties. Because the reward is computed from rendered pages rather than canonical source-code targets, any image–HTML pair from Stage-1 can be reused directly as a reward source.

Visual Similarity Reward. For each generated HTML y, the code is rendered into a screenshot and compared against the reference image along three axes. SSIM captures structural, luminance, and contrast fidelity at the pixel level; CLIP similarity (Radford et al., 2021) measures semantic correspondence through pretrained image embeddings; and OCR-F1 measures textual fidelity by penalizing hallucinated or missing visible content. These three components are linearly combined into the final visual reward signal used during training,

$$r_{\text{vis}}(y) = \sum_{k \in \mathcal{K}} \lambda_k \cdot \mathcal{S}_k\big(\mathcal{R}(y),\ s^\star\big), \tag{2}$$

where $\mathcal{K} = \{\text{SSIM}, \text{CLIP}, \text{OCR}\}$, $s^\star$ is the reference screenshot, and the weights are set to $\lambda_{\text{SSIM}} = \lambda_{\text{CLIP}} = \lambda_{\text{OCR}} = 1/3$, treating structural, semantic, and textual fidelity as equally important for visual equivalence.

GRPO Optimization. Stage-2 is initialized from the Stage-1 checkpoint and optimized with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) using r_{vis} as the reward signal. For each prompt x, G completions $\{y_i\}_{i=1}^{G}$ are sampled from $\pi_{\theta_{\text{old}}}$ and rendered to obtain rewards $\{r_i\}$. The group-standardized advantages A_i then drive the clipped surrogate objective with a KL penalty against a fixed reference policy π_{ref}:

$$\begin{aligned} J_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim P(X),\, \{y_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)} \\ \Bigg[\frac{1}{G} \sum_{i=1}^{G} \min\Bigg(\frac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\text{old}}}(y_i \mid x)} A_i, \\ \text{clip}\Big(\tfrac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\text{old}}}(y_i \mid x)},\ 1-\varepsilon,\ 1+\varepsilon \Big) A_i \Bigg) \\ - \beta\, D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big) \Bigg]. \end{aligned} \tag{3}$$

The reward r_{vis} aligns the policy with the rendered appearance of a reference screenshot, but it relies on such a reference and cannot reveal how generated controls behave under interaction. Stage-3 therefore replaces the static reference with a browser-agent interaction trace.

3.5 Stage-3: Agentic Reward RL

Stage-3 specializes the abstract reward in Eq. (1) to open-ended text-to-code generation through an agentic evaluator that interacts with each generated webpage in a real browser environment. The resulting *Verifiable Interaction Reward* (VIR) targets behavior under interaction rather than appearance in a single rendered state. To support visual content, the policy emits `<img>` tags whose `alt` attributes serve as image prompts, which are rendered into actual images by an external text-to-image model (Qwen-Image (Wu et al., 2025a)) before the page is passed to the evaluator.

Agentic Evidence Collection. For a generated HTML page y, a browser agent executes a fixed action policy covering five primitive actions of modern webpages (page initialization, animation settling, click, drag, and scroll), yielding the screenshot trajectory $\mathcal{R}(y) = (s_0, s_1, \ldots, s_T)$ with $T = 4$. To make state changes explicit to the judge, a normalized pixel difference is computed between

adjacent browser states,

$$\Delta_t = \phi(s_t, s_{t-1}) \in [0,1], \quad t = 1, \ldots, T, \quad (4)$$

where ϕ denotes the normalized pixel-wise difference between two screenshots. Each Δ_t thus serves as direct, browser-observed evidence of interactive behavior. A distilled Markdown outline $o(y)$ of the DOM structure complements the visual signals with a compact semantic skeleton that summarizes headings, sections, and visible textual structure without exposing raw source code. The final evidence package passed to the multimodal judge is

$$\mathcal{E}(\mathcal{R}(y)) = \big(\{s_t\}_{t=0}^{T},\ \{\Delta_t\}_{t=1}^{T},\ o(y)\big). \quad (5)$$

Reward Computation. Given the agentic evidence $\mathcal{E}(\mathcal{R}(y))$ and the user prompt x, the multimodal judge $\mathcal{J}$ produces integer scores $J_d \in [0, 10]$ over six dimensions $\mathcal{D} = \{\text{inter}, \text{layout}, \text{color}, \text{coher}, \text{align}, \text{comm}\}$, corresponding to interaction effectiveness, layout quality, color harmony, element coherence, prompt alignment, and commercial quality. Each judge score is then capped by an upper bound C_d derived from the agentic evidence and the structural completeness of y,

$$\tilde{J}_d = \min\big(J_d,\ C_d(\mathcal{E}(\mathcal{R}(y)), y)\big), \quad (6)$$

so that pages without observable interactive response or with insufficient structural content cannot receive high scores (detailed cap rules in Appendix A). The aligned scores are aggregated under a topology-adaptive weighting $w(y)$ derived from $\mathcal{E}(\mathcal{R}(y))$ and $o(y)$,

$$w(y) = \begin{cases} w^{\text{narr}}, & \text{if } y \text{ is a scroll narrative,} \\ w^{\text{def}}, & \text{otherwise,} \end{cases} \quad (7)$$

which acknowledges that long-form scroll narratives call for a different evaluation profile than application-like pages with active controls. The default profile emphasizes interaction effectiveness and layout quality, while the narrative profile shifts weight toward prompt alignment and commercial quality. The aligned scores and adaptive weights yield the final scalar reward,

$$r_{\text{VIR}}(y \mid x) = \sum_{d \in \mathcal{D}} w_d(y) \cdot \frac{\tilde{J}_d}{10}. \quad (8)$$

Stage-3 is initialized from the Stage-2 checkpoint and optimized with the GRPO objective of Eq. (3), replacing r_{vis} with r_{VIR}.

4 Experiments

4.1 Experimental Setup

Data. The image-to-code stages use the corpus of approximately 25K image–HTML pairs curated in Section 3.2 from roughly 10,000 source URLs, with multiple states explored per URL. This corpus is sufficient for cold-start SFT and visual-reward RL on a 4B backbone. Many Stage-3 text-to-code prompts target factual topics whose content cannot be inferred from the prompt alone. To keep training reproducible without an external search service, the matching English Wikipedia page is retrieved offline and prepended to the prompt as background knowledge. The same context is given to both the policy and the reward judge, so that the reward reflects how the policy organizes the given knowledge rather than its parametric recall.

Training Configuration. Stage-1 fine-tunes the backbone with LoRA SFT for 3 epochs at batch size 256 and learning rate 3×10^{-5}, with a sequence length of 8,192 tokens to accommodate full HTML pages. Stages-2 and -3 share an identical GRPO setup. Each step uses a prompt batch of 64 with $G = 16$ rollouts per prompt, sampling temperature 1.0, learning rate 1×10^{-6}, KL coefficient $\beta = 0.02$, and completion length 8,192. The two stages differ only in the reward signal: r_{vis} (Section 3.4) for Stage-2 and r_{VIR} (Section 3.5) for Stage-3. The full three-stage schedule consumes approximately 840 A100 GPU-hours.

4.2 Reconstruction Evaluation

Setup. This evaluation isolates the image-to-code stages of the pipeline (Stages-1 and -2) from the open-ended Stage-3. Each generated HTML is re-rendered and compared against a reference screenshot. Image-to-code performance is measured by SSIM, CLIP, and OCR-F1 between rendered output and reference screenshot on Qwen3.5-4B (Table 1).

Stage-wise gains. Stage-1 SFT lifts SSIM ($0.51 \rightarrow 0.60$), CLIP ($0.58 \rightarrow 0.66$), and OCR-F1 ($0.29 \rightarrow 0.33$), confirming that autonomously curated supervision provides an effective foundation. Adding Stage-2 visual-reward RL further pushes CLIP to **0.78** and OCR-F1 to **0.61**, moving optimization beyond token-level imitation toward visual equivalence. Stage-2 alone reaches **0.68** SSIM but lags on OCR-F1 (0.53), which shows that cold-start supervision is required before visual reward takes full effect. The OCR-F1 trajectory ($0.29 \rightarrow 0.33 \rightarrow 0.61$ across Base, Stage-1, and the full configuration)

Configuration	SSIM↑	CLIP↑	OCR-F1↑
GPT-5	0.62	0.68	0.25
Gemini-3.1-Pro	0.53	0.63	0.30
Base	0.51	0.58	0.29
Stage-1 only	0.60	0.66	0.33
Stage-2 only	**0.68**	0.74	0.53
Stage-1 + Stage-2	0.66	**0.78**	**0.61**

Table 1: Image-to-code reconstruction on the held-out evaluation set: OPENGENUI stage ablations on the Qwen3.5-4B backbone. Best per column in **bold**.

illustrates this further. Stage-1 alone provides limited gain on faithful text rendering, while the OCR-F1 component of r_{vis} in Eq. (2) directly optimizes on-screen text fidelity during Stage-2. The full configuration matches or surpasses proprietary baselines (GPT-5, Gemini-3.1-Pro) on three metrics.

4.3 Human Evaluation

GSB preference. The main comparison is conducted on 50 held-out prompts by five annotators. Two baselines, the Qwen3.5-4B base and Gemini-3.1-Pro, are each compared against the full OPENGENUI (Stage-1 & Stage-2 & Stage-3) via pairwise GSB judgments (Likert $\alpha = 0.63$, GSB $\alpha = 0.33$, protocol in Appendix C). Against the Qwen3.5-4B base backbone (Table 2), annotators prefer OPENGENUI in 51.2% of judgments, the base in 10.4%, and rate the two as comparable in the remaining 38.4% (GSB$\Delta = -40.8$). This indicates a clear but not saturated improvement from the training pipeline. Against Gemini-3.1-Pro, the two systems receive comparable preference. 62.4% of judgments are rated as Same, with Gemini preferred in 19.6% and OPENGENUI in 18.0% (GSB$\Delta = +1.6$). The net gap is small relative to the inter-annotator agreement on GSB labels ($\alpha = 0.33$), suggesting that the 4B model approaches but does not match a proprietary system roughly an order of magnitude larger.

Likert diagnostics. OPENGENUI lifts the overall Likert score from 3.16 to 3.87 over the base (Table 3). The largest gain comes from interaction effectiveness ($2.85 \rightarrow 4.12$). Compared with Gemini-3.1-Pro, the overall Likert scores are close (3.87 vs. 3.99). OPENGENUI trails on visual quality (3.92 vs. 4.15) and layout (3.78 vs. 4.08), and leads on interaction effectiveness (4.12 vs. 3.57). The interaction-effectiveness gap is consistent with the design of VIR and is not captured by single-state automatic metrics.

Baseline	Good	Same	Bad	GSBΔ
Qwen3.5-4B (Base)	10.4%	38.4%	51.2%	-40.8
Gemini-3.1-Pro	19.6%	62.4%	18.0%	$+1.6$

Table 2: Pairwise GSB against OPENGENUI. *Good*/*Bad* mean the baseline/OPENGENUI is preferred. Negative GSBΔ favors OPENGENUI.

Model	Vis.	Lay.	Prompt	Inter.	Over.
Qwen3.5-4B	3.20	3.20	3.30	2.85	3.16
Gemini-3.1-Pro	**4.15**	**4.08**	**4.01**	<u>3.57</u>	**3.99**
OPENGENUI	<u>3.92</u>	<u>3.78</u>	<u>3.84</u>	**4.12**	<u>3.87</u>

Table 3: Likert scores (1–5, higher is better). Best per column in **bold**, second-best <u>underlined</u>. Protocol in Appendix C.

4.4 Case Study

Figure 1 compares OPENGENUI, the Qwen3.5-4B base, and Gemini-3.1-Pro on five representative prompts. OPENGENUI consistently produces multi-section pages with themed visuals and prompt-relevant content, while the base backbone emits sparse, placeholder layouts even on prompts requiring richer content. The training stages thus appear to change *what* the compact backbone learns to emit, not merely how aggressively it spends its decoding budget.

Compared with Gemini-3.1-Pro, OPENGENUI reaches comparable visual quality and layout on structured-information prompts (e.g., Great Barrier Reef). On these prompts both systems produce coherent multi-section pages with appropriate hierarchy and themed imagery. Gemini retains an edge on long-form typographic styling, while OPENGENUI matches or exceeds it on interaction-heavy prompts (e.g., the basketball math tutor) by producing dedicated interactive sections per concept rather than a single static block. This pattern matches the largest Likert gain observed in our human study.

4.5 Ablation Study

Necessity of Stage-1. Applying Stage-2 GRPO directly to the base model without cold-start SFT yields an SSIM of 0.68. This is marginally higher than the full Stage-1 & Stage-2 configuration (0.66) but lags on CLIP (0.74 vs. 0.78) and OCR-F1 (0.53 vs. 0.61, Table 1). SSIM is dominated by low-level pixel statistics and can be recovered by the visual-similarity reward alone. CLIP and OCR-F1 require both ingredients. Cold-start SFT establishes a code distribution from which GRPO rollouts can render meaningful pages, and the visual reward then

Baseline (vs. VIR)	Good	Same	Bad	GSBΔ
Stage-1 + Stage-2	9.6%	34.0%	56.4%	-46.8
Single-state Judge	13.2%	42.8%	44.0%	-30.8
Agentic Expl. Judge	21.2%	44.0%	34.8%	-13.6

Table 4: Pairwise GSB of VIR against earlier reward variants. Good/Bad mean the baseline variant/VIR is preferred. Negative GSBΔ favors VIR.

Configuration	SSIM↑	CLIP↑	OCR-F1↑
Base	0.16	0.59	0.07
Stage-1 only	0.55	0.64	0.29
Stage-2 only	0.60	0.73	0.51
Stage-1 + Stage-2	**0.69**	**0.78**	**0.60**

Table 5: Ablation on Qwen3-VL-4B-Instruct (same procedure as Table 1). Best per column in **bold**.

shapes them toward semantic correspondence.

Reward Design Variants. Two earlier Stage-3 reward designs are evaluated under the same setup as VIR (details in Appendix B). The *Single-state Judge* feeds one screenshot plus a code outline to a multimodal judge, without browser interaction or pixel-level evidence. The *Agentic Exploration Judge* lets a multimodal agent autonomously choose browser actions via tool calls. All variants are trained from the same Stage-2 checkpoint, and are compared against the full OpenGenUI model on the same 50 open-ended prompts and by the same five annotators as in Section 4.3 (full protocol in Appendix C). Table 4 reports GSB of VIR against each variant. VIR is preferred over the Stage-1 & Stage-2 checkpoint with GSB$\Delta = -46.8$, confirming Stage-3's contribution beyond visual-only RL. The gap remains in favor of VIR against Single-state Judge (-30.8) and Agentic Exploration Judge (-13.6). This suggests that browser-grounded evidence drives most of VIR's advantage, and that a fixed action policy with pixel-level state differences is more discriminative than free-form agent-driven exploration.

Cross-Backbone Generalization. To check that the image-to-code training schedule is not specific to a single backbone, the Stage-1 cold-start SFT and the Stage-2 visual-reward GRPO are re-run on Qwen3-VL-4B-Instruct under identical data, hyperparameters, and reward. Results are reported in Table 5, evaluated on the same image-to-code set as Table 1. Despite a substantially lower starting point (SSIM 0.16, OCR-F1 0.07), largely caused by the base model hallucinating malformed, unrenderable HTML, Stage-1 & Stage-2 brings all three metrics to a level comparable with the Qwen3.5-4B Stage-1 & Stage-2 configuration. The relative gain pattern across the four configurations matches that of Table 1, indicating that the Stage-1 and Stage-2 gains stem from the training pipeline rather than any particular backbone choice.

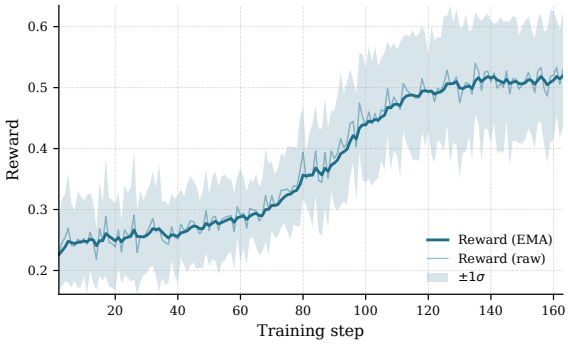

Figure 4: Stage-3 GRPO training reward. The solid curve is the EMA-smoothed reward, and the shaded region denotes $\mu_t \pm \sigma_t$, where σ_t is the rollout-level reward standard deviation within each GRPO update.

4.6 Training Dynamics

Figure 4 shows the GRPO training reward in Stage-3. The average reward rises from approximately 0.25 to 0.52, with the transition concentrated between steps 70 and 100. This pattern suggests that the policy discovers a new behavior mode rather than drifting smoothly along an existing one. The rollout-level reward standard deviation remains non-zero throughout training, which preserves a discriminative within-group signal for GRPO. The KL divergence with respect to the reference policy grows to about 0.06, reflecting a moderate but non-trivial movement of probability mass away from the policy at the start of Stage-3. The two RL stages also differ in reward latency: Stage-2's rule-based reward takes about 1 minute per rollout, while Stage-3's agentic browser-interaction reward takes about 10 minutes.

5 Conclusion

We present OpenGenUI, a three-stage training framework for Generative UI in MLLMs. The framework includes a scalable data exploration pipeline that turns autonomously crawled webpages into clean image–HTML supervision, a visual-similarity reward for image-to-code generation, and VIR that grounds open-ended text-to-code generation in browser-induced state transitions. Together, these stages allow MLLMs to acquire Generative UI capabilities from static reconstruction, rendered visual alignment, and executable interaction feedback without manual annotation.

6 Limitations

To focus on code generation, this work uses Wikipedia retrieval as a proxy for external search. This controlled setting isolates the effect of the proposed reward, but does not capture the noise of real search engines. Future work will extend OPENGENUI to an end-to-end setting with live search integration.

7 Ethical Considerations

Our work utilizes publicly available datasets (Insta-150k-v3) and open-source models (Qwen3.5-4B, Qwen3-VL-4B-Instruct, Qwen-Image). The research does not involve human subjects, private data, or create applications with immediate potential for harm.

References

Anthropic. 2026. Claude-Design. https://www.anthropic.com/news/claude-design-anthropic-labs.

Rabiul Awal, Mahsa Massoud, Aarash Feizi, Zichao Li, Suyuchen Wang, Christopher Pal, Aishwarya Agrawal, David Vazquez, Siva Reddy, Juan A Rodriguez, and 1 others. 2025. Webmmu: A benchmark for multimodal multilingual website understanding and code generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25129–25156.

Tony Beltramelli. 2018. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI symposium on engineering interactive computing systems*, pages 1–6.

Bytedance. 2026. Seed2.0. https://seed.bytedance.com/en/seed2.

Yang Chen, Minghao Liu, Yufan Shen, Yunwen Li, Tianyuan Huang, Xinyu Fang, Tianyu Zheng, Wenxuan Huang, Cheng Yang, Daocheng Fu, and 1 others. 2025. Iwr-bench: Can lvlms reconstruct interactive webpage from a user interaction video? *arXiv preprint arXiv:2509.24709*.

DeepSeek-AI. 2026. DeepSeek$_V$4..

GLM-5-Team. 2026. Glm-5: from vibe coding to agentic engineering. *Preprint*, arXiv:2602.15763.

Google. 2026. Gemini-3.1-pro. https://deepmind.google/models/gemini/pro/.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Bohua Chen, Yi Su, Dongping Chen, Siyuan Wu, Xing Zhou, and 1 others. 2025a. Webcode2m: A real-world dataset for code generation from webpage designs. In *Proceedings of the ACM on Web Conference 2025*, pages 1834–1845.

Yi Gui, Zhen Li, Zhongyi Zhang, Guohao Wang, Tianpeng Lv, Gaoyang Jiang, Yi Liu, Dongping Chen, Yao Wan, Hongyu Zhang, and 1 others. 2025b. Latcoder: Converting webpage design to code with layout-as-thought. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 721–732.

Yi Gui, Yao Wan, Zhen Li, Zhongyi Zhang, Dongping Chen, Hongyu Zhang, Yi Su, Bohua Chen, Xing Zhou, Wenbin Jiang, and 1 others. 2025c. Uicopilot: Automating ui synthesis via hierarchical code generation from webpage designs. In *Proceedings of the ACM on Web Conference 2025*, pages 1846–1855.

Hongcheng Guo, Wei Zhang, Junhao Chen, Yaonan Gu, Jian Yang, Junjia Du, Shaosheng Cao, Binyuan Hui, Tianyu Liu, Jianxin Ma, and 1 others. 2025. Iw-bench: Evaluating large multimodal models for converting image-to-web. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6449–6466.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14281–14290.

Yilei Jiang, Yaozhi Zheng, Yuxuan Wan, Jiaming Han, Qunzhong Wang, Michael R Lyu, and Xiangyu Yue. 2025. Screencoder: Advancing visual-to-code generation for front-end automation via modular multimodal agents. *arXiv preprint arXiv:2507.22827*.

Peichao Lai, Jinhui Zhuang, Kexuan Zhang, Ningchang Xiong, Shengjie Wang, Yanwei Xu, Chong Chen, Yilei Wang, and Bin Cui. 2025. Webrenderbench: Enhancing web interface generation through layout-style consistency and reinforcement learning. *arXiv preprint arXiv:2510.04097*.

Hugo Laurençon, Léo Tronchon, and Victor Sanh. 2024. Unlocking the conversion of web screenshots into html code with the websight dataset. *arXiv preprint arXiv:2403.09029*.

Yaniv Leviathan, Dani Valevski, Matan Kalman, Danny Lumen, Eyal Segalis, Eyal Molad, Shlomi Pasternak, Vishnu Natchu, Valerie Nygaard, Srinivasan, Venkatachary, James Manyika, and Yossi

Matias. 2026. Generative ui: Llms are effective ui generators. *Preprint*, arXiv:2604.09577.

Chunyang Li, Yilun Zheng, Xinting Huang, Tianqing Fang, Jiahao Xu, Lihui Chen, Yangqiu Song, and Han Hu. 2025a. Webdevjudge: Evaluating (m) llms as critiques for web development quality. *arXiv preprint arXiv:2510.18560*.

Yuhang Li, Chenchen Zhang, Ruilin Lv, Ao Liu, Ken Deng, Yuanxing Zhang, Jiaheng Liu, Wiggin Zhou, and Bo Zhou. 2025b. Relook: Vision-grounded rl with a multimodal llm critic for agentic web coding. *arXiv preprint arXiv:2510.11498*.

Zhiyu Lin, Zhengda Zhou, Zhiyuan Zhao, Tianrui Wan, Yilun Ma, Junyu Gao, and Xuelong Li. 2025. Webuibench: a comprehensive benchmark for evaluating multimodal large language models in webui-to-code. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15780–15797.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J Pal, and Siva Reddy. 2025. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*.

Yuxuan Lu, Ting-Yao Hsu, Hansu Gu, Limeng Cui, Yaochen Xie, William Headden, Bingsheng Yao, Akash Veeragouni, Jiapeng Liu, Sreyashi Nag, and 1 others. 2025a. Agenta/b: Automated and scalable web a/btesting with interactive llm agents. *arXiv preprint arXiv:2504.09723*.

Yuxuan Lu, Bingsheng Yao, Hansu Gu, Jing Huang, Zheshen Jessie Wang, Yang Li, Jiri Gesi, Qi He, Toby Jia-Jun Li, and Dakuo Wang. 2025b. Uxagent: An llm agent-based usability testing framework for web design. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Zimu Lu, Yunqiao Yang, Houxing Ren, Haotian Hou, Han Xiao, Ke Wang, Weikang Shi, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2025c. Webgen-bench: Evaluating llms on generating interactive and functional websites from scratch. *arXiv preprint arXiv:2505.03733*.

OpenAI. 2026. GPT-5-5. https://openai.com/zh-Hans-CN/index/introducing-gpt-5-5/.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *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)*, pages 3956–3974.

Haoyu Sun, Huichen Will Wang, Jiawei Gu, Linjie Li, and Yu Cheng. 2025. Fullfront: Benchmarking mllms across the full front-end engineering workflow. *arXiv preprint arXiv:2505.17399*.

Heyi Tao, Sethuraman TV, Michal Shlapentokh-Rothman, and Derek Hoiem. 2023. Webwise: Web interface control and sequential exploration with large language models. *arXiv preprint arXiv:2310.16042*.

Qwen Team. 2026. Qwen3.5-omni technical report. *Preprint*, arXiv:2604.15804.

Brandon Trabucco, Gunnar Sigurdsson, Robinson Piramuthu, and Ruslan Salakhutdinov. 2025. Insta: Towards internet-scale training for agents. *arXiv preprint arXiv:2502.06776*.

Jianqiang Wan, Sibo Song, Wenwen Yu, Yuliang Liu, Wenqing Cheng, Fei Huang, Xiang Bai, Cong Yao, and Zhibo Yang. 2024. Omniparser: A unified framework for text spotting key information extraction and table recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15641–15653.

Kuang-Da Wang, Zhao Wang, Yotaro Shimose, Wei-Yao Wang, and Shingo Takamatsu. 2025. Webgen-v bench: Structured representation for enhancing visual design in llm-based web generation and evaluation. *arXiv preprint arXiv:2510.15306*.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, and 1 others. 2025a. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*.

Fan Wu, Cuiyun Gao, Shuqing Li, Xin-Cheng Wen, and Qing Liao. 2025b. Mllm-based ui2code automation guided by ui layout information. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):1123–1145.

Jingyu Xiao, Yuxuan Wan, Yintong Huo, Zixin Wang, Xinyi Xu, Wenxuan Wang, Zhiyao Xu, Yuhang Wang, and Michael R Lyu. 2025. Interaction2code: Benchmarking mllm-based interactive webpage code generation from interactive prototyping. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 241–253. IEEE.

Hao Yang, Weijie Qiu, Ru Zhang, Zhou Fang, Ruichao Mao, Xiaoyu Lin, Maji Huang, Zhaosong Huang, Teng Guo, Shuoyang Liu, and 1 others. 2025a. Ui-ug: A unified mllm for ui understanding and generation. *arXiv preprint arXiv:2509.24361*.

Zhen Yang, Wenyi Hong, Mingde Xu, Xinyue Fan, Weihan Wang, Jiele Cheng, Xiaotao Gu, and Jie Tang. 2025b. Ui2codeˆ n: A visual language model for test-time scalable interactive ui-to-code generation. *arXiv preprint arXiv:2511.08195*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Q Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, and 1 others. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. *Advances in neural information processing systems*, 37:112134–112157.

Chenchen Zhang, Yuhang Li, Can Xu, Jiaheng Liu, Ao Liu, Changzhi Zhou, Ken Deng, Dengpeng Wu, Guanhua Huang, Kejiao Li, and 1 others. 2025. Artifactsbench: Bridging the visual-interactive gap in llm code generation evaluation. *arXiv preprint arXiv:2507.04952*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, and 1 others. 2024. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*.

A VIR Implementation Details

Action policy. The browser agent executes a fixed five-step trajectory on each generated page. The five steps are page initialization, animation settling, click on the first interactive button, drag on the first range slider, and scroll to mid-page. Each adjacent state pair yields one normalized pixel difference Δ_t, which is used both as evidence in the judge prompt and as input to the programmatic constraints described below.

Programmatic constraints. The upper bounds C_d in Eq. (6) are implemented as three constraints applied to the raw judge scores before aggregation. The concrete cap values are summarized in Table 6. **Constraint A** caps the interaction score by the maximum click/drag pixel difference $\Delta^\star = \max(\Delta_{\text{click}}, \Delta_{\text{drag}})$, so that pages without observable response cannot receive high interaction scores. **Constraint B** caps the layout, color, coherence, and commercial dimensions when the HTML is structurally incomplete, measured jointly by byte size and section count. **Constraint C** couples richness and interaction. When richness is low and the interaction score is also low, the alignment and commercial dimensions are further capped, preventing thin pages from being scored as polished products. The three constraints together anchor the reward to interactions that are observable in the rendered trajectory, embedded in a structurally complete page, and consistent with the DOM outline $o(y)$ given to the judge.

Weight profiles. The two profiles in Eq. (7) are listed in Table 7. The narrative profile applies when three conditions hold simultaneously: the page has no observable click or drag response (neither Δ_{click} nor Δ_{drag} exceeds a small threshold), the DOM

Condition	Cap
Constraint A: J_{inter} *by* $\Delta^\star$	
$\Delta^\star < 0.02$	≤ 2
$\Delta^\star \in [0.02, 0.10)$	≤ 4
$\Delta^\star \in [0.10, 0.20)$	≤ 6
$\Delta^\star \in [0.20, 0.40)$	≤ 8
Constraint B: layout/color/coher/comm by $\lvert y \rvert$	
$\lvert y \rvert < 5\,\text{KB}$	≤ 5
$\lvert y \rvert \in [5, 15)\,\text{KB}$	≤ 7
$\lvert y \rvert \in [15, 50)\,\text{KB}$	≤ 9
Constraint C: $J_{\text{comm}}/J_{\text{align}}$ *when* $J_{\text{inter}} \leq 2$	
richness < 3	$1/2$
richness $\in [3, 6)$	$4/6$
richness ≥ 6, text $< 1{,}000$ chars	$4/6$
richness ≥ 6, text $\geq 1{,}000$ chars	$6/8$

Table 6: Programmatic constraints applied to raw judge scores. Richness denotes the sum of section count and capped heading count from the DOM outline.

Dimension	w^{def}	w^{narr}
Interaction effectiveness	0.30	0.10
Layout quality	0.20	0.20
Color harmony	0.15	0.13
Element coherence	0.15	0.12
Prompt alignment	0.10	0.25
Commercial quality	0.10	0.20

Table 7: Default and narrative weight profiles used in the topology-adaptive aggregation of Eq. (7).

outline is structurally rich (richness ≥ 6, defined as in Table 6), and the visible-text length is substantial. Otherwise the default profile applies. This routing prevents long-form scroll narratives from being dominated by the interaction dimension.

B Reward Design Variants Used in Stage-3 Ablation

Single-state Judge. The Single-state Judge renders each generated page in a headless browser and produces a small set of screenshots that approximate the page in its initial state. The screenshots include a full-page screenshot, viewport screenshots, and a mobile-viewport screenshot. They are paired with a distilled HTML outline and submitted to a multimodal judge in a single call. The judge produces scores along four dimensions, namely commercial quality, visual diversity, aesthetics, and interactivity. To stabilize the resulting reward distribution under GRPO, a sigmoid transformation is applied to the averaged score. This design is fully deterministic and low-latency. However, interaction effectiveness is inferred from CSS-driven affordances rather than from observed state changes, which limits its ability to evaluate interactive behavior beyond hover and toggle effects.

Agentic Exploration Judge. The Agentic Exploration Judge replaces the fixed action policy of the browser agent with a multimodal agent that decides its own actions through tool calls. The agent can issue browser actions, inspect the rendered page, and read a truncated HTML outline before producing scores along five dimensions. Compared with the Single-state Judge, this variant evaluates a richer set of interactions and is closer in spirit to user-driven testing. However, the exploration trajectories vary across rollouts of the same page, and tool-call parsing introduces additional failure modes during large-scale training. These two effects increase the variance of the reward signal and lead to less stable GRPO optimization. These observations motivate the design of VIR, which keeps interaction-based evaluation while replacing autonomous exploration with a controlled browser-agent policy combined with pixel-grounded evidence and programmatic constraints.

C Human Evaluation Protocol

Annotators. Five annotators are recruited for the evaluation. All hold at least an undergraduate degree in computer science or a closely related field, and have prior experience in front-end web development or web user interface design. None of the annotators is involved in the model design, training, or reward implementation of OpenGenUI. None has access to the configuration identity behind each generated webpage during annotation. All annotators are paid above the local hourly minimum wage.

Evaluation samples. A subset of 50 prompts is sampled uniformly at random (without replacement, fixed seed) from the held-out evaluation set, covering diverse webpage topics such as educational content, product descriptions, scientific explanations, historical and cultural overviews, and tool-style interfaces. The same 50 prompts and the same five annotators are used in both human evaluations reported in the paper. The *main comparison* in Section 4.3 compares two baselines (the Qwen3.5-4B base and Gemini-3.1-Pro) against the full OpenGenUI model via pairwise GSB judgments. The *reward-variant comparison* in Section 4.5 compares OpenGenUI against three alternatives that share the same Stage-2 checkpoint,

namely the Stage-2 checkpoint with no Stage-3, the Single-state Judge variant, and the Agentic Exploration Judge variant. For every prompt, all candidate webpages are rendered in a unified browser environment to ensure consistent rendering and interaction conditions. The configuration identity is anonymized, and the order of pages is randomized per prompt.

Procedure and aggregation. Annotators evaluate one prompt at a time, with the candidate webpages presented side by side in randomized order. Each annotator rates every page independently along five dimensions on a 1–5 Likert scale, and is required to operate the page with mouse and keyboard before scoring interaction effectiveness. For each (prompt, configuration, dimension) triple, the final Likert score is the average of the five ratings. For GSB judgments, results are reported as the percentage of pairwise votes across all annotators (5 annotators $\times$ 50 prompts = 250 judgments per condition), with $\Delta = \text{Good\%} - \text{Bad\%}$. The five Likert dimensions are designed for user-facing holistic evaluation and do not correspond one-to-one with the six VIR reward dimensions used during training. The training side splits visual quality into color harmony and element coherence for finer reward shaping.

Annotation guidelines. The annotation guidelines distributed to human annotators are attached on the following pages.

Annotation Guidelines for Web Page Generation Quality Evaluation

1. Task Introduction

This evaluation aims to assess how well different MLLMs generate web pages from natural language instructions. For each question, you will see:

- **A user prompt:** a description of the web page the user wants to obtain;
- **Multiple web pages:** each generated by a different (anonymized) MLLM from this prompt;
- **A rating form:** where you score each page in five dimensions.

Your job is to browse and use these pages as a real user would, and to give independent, honest ratings based on your actual experience.

2. Things to Know Before You Start

- All web pages are loaded in the same browser environment with the same viewport size.
- The pages appear in random order for each question. The mapping from page index to a specific MLLM therefore changes from question to question.
- The MLLM identity behind each page is fully anonymous during annotation, and is only revealed after the entire task is completed.

3. Core Principles

Please follow these four principles throughout the task:

- **Principle 1 — Independent judgment:** Do not discuss specific questions or pages with other annotators. Independent scoring is essential for the credibility of the evaluation.
- **Principle 2 — Anonymous scoring:** Do not try to guess which MLLM produced a given page. Score only based on what you see.
- **Principle 3 — Score against the prompt:** Always interpret the page in the context of the prompt. A visually polished page that does not address the prompt should not receive a high score on prompt alignment. Likewise, a page whose content matches the prompt but looks rough should not receive a high score on visual quality.
- **Principle 4 — Do not inspect the source code:** Score only from the rendered page and your interaction with it. Do not open developer tools or read the underlying HTML/CSS.

4. Rating Scale

All five dimensions share the same 1–5 Likert scale:

Score	Level	Meaning
5	Excellent	Close to release-grade quality on this dimension; almost no issues.
4	Good	Generally good, with only minor flaws that do not affect use.
3	Acceptable	Usable but with clear weaknesses; an ordinary user would consider it "okay but not great".
2	Poor	Has obvious defects that noticeably hurt the user experience.
1	Unacceptable	Essentially unusable, for example broken rendering, blank content, or no working buttons.

Reminder: Please use the full 1–5 range. Do not default to 3 just to avoid extreme scores. If a page is truly excellent or truly bad, give it 5 or 1.

5. The Five Dimensions

Dimension 1 — Visual Quality

Focus: The overall aesthetic impression, including color scheme, typography, spacing, image rendering, and the consistency of the visual style.

Score	Typical Signs
5	Harmonious colors, appropriate fonts, correctly rendered images and icons, consistent and professional style.

3	Generally acceptable, but with issues such as inconsistent fonts, mismatched colors, or uneven spacing.
1	Broken images, severe color clashes, unreadable text, or clearly damaged rendering.

Dimension 2 — Layout

Focus: The structural organization of the page, including section partitioning, alignment, use of grids or columns, and visual hierarchy.

Score	Typical Signs
5	Clear sections, balanced composition, distinct hierarchy, consistent alignment.
3	The structure is recognizable, but some areas feel cluttered, the hierarchy is weak, or there are local misalignments.
1	Overlapping or stacked elements, no discernible structure, content overflowing its container.

Dimension 3 — Prompt Alignment

Focus: Whether the page actually responds to what the prompt asks for.

Score	Typical Signs
5	The page clearly conveys the requested topic, includes the key information asked for, and matches the intended use case.
3	The page partially addresses the prompt, but some requested elements are missing or only mentioned superficially.
1	The page drifts off-topic, or the content is largely unrelated to the prompt.

Dimension 4 — Interaction Effectiveness

Focus: Whether interactive elements on the page actually work, including buttons, toggles, sliders, dropdown menus, animations, and scroll-driven effects.

Score	Typical Signs
5	Interactive elements respond consistently and meaningfully; clicking, dragging, or scrolling produces visible and reasonable state changes.
3	Some interactive elements work, while others are decorative or do nothing when clicked.
1	Elements look clickable but none respond; or the prompt expects interaction but the page is entirely static.

Required: Before scoring this dimension, you must actually use the mouse and keyboard to operate the page. This includes clicking buttons, opening dropdowns, dragging sliders, scrolling through any animated sections, hovering over elements, and trying out any forms.

Dimension 5 — Overall Preference

Focus: Your overall impression of the page as a finished product. This is a holistic judgment. Take the previous four dimensions into account, but do not compute a numerical average. A useful question to ask: " Among these candidate pages, which one would I, as a user, most want to keep browsing or using?"

6. Special Cases

- **6.1 Page fails to load:** If a page does not load or shows up blank, check the "rendering failure" option in the form. The page will be excluded from the statistics for the relevant dimensions rather than being treated as a 1.
- **6.2 Factual content you are unsure about:** If the prompt involves facts you are not certain about (for example details of a historical event), score the page on whether it is clearly expressed and internally consistent, rather than deducting points based on factual knowledge you are not confident in.

- **6.3 Long pages:** For long pages (such as long-form articles), scroll to the bottom at least once before scoring, so that you have actually seen the full content.

7. Quality Control

To preserve the integrity of the rating distribution, pages that fail to load are not treated as 1-point ratings. If a page does not render, mark it with the "rendering failure" flag in the form; such cases are counted separately and excluded from the statistics for the relevant dimensions.

8. Summary of the Procedure

For each question, follow these steps:

1. Read the prompt. Make sure you understand what the user is asking for.
2. Go through the pages one by one.
    - Scroll through the full content.
    - Actively interact with every visible interactive element.
3. Score each page on the five dimensions: visual quality, layout, prompt alignment, interaction effectiveness, and overall preference.
4. Submit and move on to the next question.