A11YN: ALIGNING LLMS FOR ACCESSIBLE WEB UI CODE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have recently demonstrated strong capabilities in generating functional and aesthetic web interfaces directly from instructions. However, these models often replicate accessibility flaws from their training data, resulting in interfaces that exclude users with diverse needs and contexts. To address this gap, we introduce A11yn, the first method that aligns code-generating LLMs to reliably produce accessibility-compliant web UIs. A11yn optimizes a novel reward function that penalizes violations of the Web Content Accessibility Guidelines (WCAG), with penalties scaled to the severity of each violation as identified by an accessibility testing engine. To support training, we construct UIReq-6.8K, a dataset of 6,800 diverse instructions for web UI generation. For evaluation, we introduce RealUIReq-300, a benchmark of 300 real-world web UI requests grounded and manually curated from public web pages, spanning a broad range of use cases. Empirical results show that A11yn significantly outperforms strong baselines, lowering the Inaccessibility Rate by 60% over the base model while preserving semantic fidelity and visual quality of generated UIs. These findings demonstrate that accessibility can be systematically optimized within LLMs, showing the feasibility of aligning code generation for accessibility.

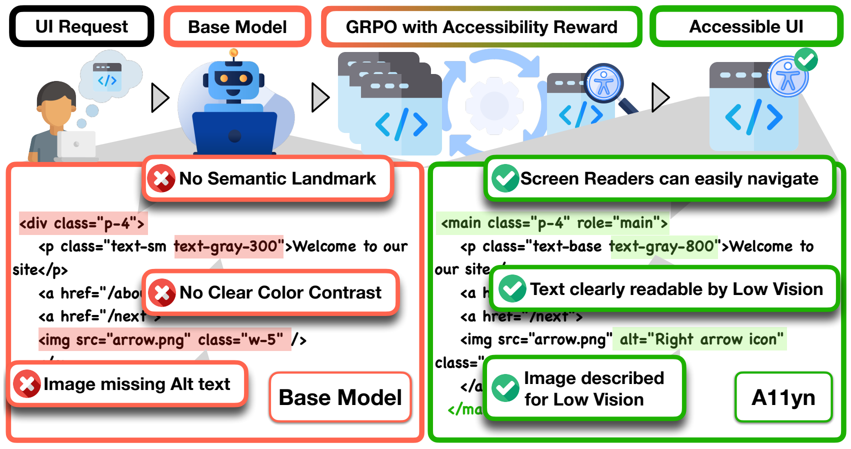
1 INTRODUCTION

Large language models (LLMs) have opened up a new frontier in front-end development. With a simple prompt, language models can generate complete web interfaces, from static HTML pages to complex, interactive components (Zhou et al., 2025). Recent benchmarks and systems have shown that LLMs can synthesize semantically accurate and visually coherent UIs, even emulating modern design patterns and wide range of front-end frameworks (Xiao et al., 2025; Lu et al., 2025). This has fueled growing research interest in LLM-based UI generation systems, which aim to improve layout, fidelity, interactivity, and functional completeness.

However, **Accessibility** remains a critical yet underexplored dimension in LLM-based web UI development. Web accessibility is a key principle that ensures anyone, even people with disabilities to perceive and navigate web interfaces. For example, blind users rely on screen readers to interpret content, while people with limited motor control need to navigate without a mouse. For millions of people, these accommodations determine whether a website is usable or completely inaccessible. To this end, the W3C defines the Web Content Accessibility Guidelines (WCAG) to formalize accessibility standards. Yet, audits report widespread non-compliance, with over 90% of public web pages containing detectable violations (Mowar et al., 2025). These shortcomings disproportionately affect users with visual or motor impairments, reinforcing barriers to digital participation.

LLMs, trained on massive web corpora with such accessibility flaws, frequently replicate them in the generated UIs. Prior studies (Suh et al., 2025; Mowar et al., 2025; Aljedaani et al., 2024; Guriţă & Vatavu, 2025) confirm that LLMs omit key accessibility elements, such as alternative text, semantic landmarks, and properly labeled form controls, resulting in inaccessible interfaces. This raises a core research question: *Can we align LLMs to natively generate web UIs that are more accessible?*

In this work, we introduce **A11yn** (pronounced *align*), the first framework to align code LLMs for accessibility-aware web UI generation. To promote accessibility, we devise a novel reward function using accessibility violations that are detected using Axe Core (Deque Systems, 2015), a widely adopted WCAG auditing tool that reports issues across four severity levels. These violations


UI Request
Base Model
GRPO with Accessibility Reward
Accessible UI
No Semantic Landmark
No Clear Color Contrast
Image missing Alt text
Base Model
Screen Readers can easily navigate
Text clearly readable by Low Vision
Image described for Low Vision
A11yn

```
<div class="p-4">
   <p class="text-sm text-gray-300">Welcome to our
site</p>
   <a href="/abo
   <a href="/next">
   <img src="arrow.png" class="w-5" />
```

```
<main class="p-4" role="main">
   <p class="text-base text-gray-800">Welcome to
our site
   <a h
   <a href="/next">
   <img src="arrow.png" alt="Right arrow icon"
class="
   </a
</ma
```

Figure 1: **A11yn enhances accessibility in UI-generative LLMs.** Whereas base models often produce inaccessible code, A11yn generates web UIs with improved accessibility features, supporting screen readers with better readability, smoother navigation, and clearer image descriptions.

are mapped to severity-weighted penalties, which are then converted into a bounded reward. The resulting reward signal is used to directly optimize the code LLM policy through Group-Relative Policy Optimization (GRPO) (Shao et al., 2024).

To support training, we construct UIReq-6.8K, an instruction-only dataset of 6,800 natural language UI generation requests spanning diverse domains and component requirements. This dataset enables reinforcement learning without relying on supervised fine-tuning data, which is difficult to collect at scale due to its scarcity and annotation cost of accessible code examples. For evaluation, we curate RealUIReq-300, a benchmark of 300 real-world web UI generation tasks, each request specified with detailed metadata such as purpose, page type, application domain, and required components. We empirically demonstrate that A11yn substantially minimizes accessibility violations, reducing Inaccessibility Rate by 60% compared to the base model, while preserving the appearance and semantic fidelity of the generated UIs. Our results suggest that accessibility can be effectively integrated as a learnable behavior within the LLM generation pipeline, bringing us closer to truly inclusive UI code generation systems.

2 Related Work

LLM-based UI Code Generation. Prior research has applied specialized models to automate the translation of designs or descriptions into code. Early work like ReDraw (Moran et al., 2018) used a learned model to assemble mobile UI code from image mock-ups. With the advent of large language models (LLMs), generating UI code directly from high-level natural language descriptions has become feasible. For instance, UICoder (Wu et al., 2024) iteratively fine-tunes pre-trained LMs with SFT on a self-generated SwiftUI training dataset, that is filtered in scale with automatic compiler feedback and a CLIP-based model. On the web UI generation side, WebGen-Bench (Lu et al., 2025) provides a benchmark that is designed to evaluate LLM-based agents in generating fully functional, multi-page web applications, featuring diverse application generation instructions and automated web navigation tests to assess functionality.

Post Training LLM for alignment. Fine-tuning LLMs with extra objective signals has become widespread. Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022) adopts Proximal Policy Optimization (PPO) (Schulman et al., 2017) for LLMs to align them with human preferences. However, PPO requires training a critic alongside the policy, adding both computational overhead and engineering complexity. A recent alternative simplifies this process: Direct Preference Optimization (DPO) (Rafailov et al., 2023) reformulates preference learning by directly adjusting the model based on pairwise preferences, eliminating the need for online training. Meanwhile, Group-Relative Policy Optimization (GRPO) (Shao et al.,

2024) extends PPO by removing the critic and instead compute advantage as rewards normalized across a batch of completion samples. GRPO has shown promise in both improving performance in verifiable domains (Mathematical Association of America, 2025) as well as aligning LLMs with human values and safety constraints Li et al. (2025).

Improving Web UI Accessibility with LLMs. Real-world web data often contains accessibility violations, leading LLMs trained on such data to reproduce accessibility flaws in generated UI code (Martins & Duarte, 2024; Guriţă & Vatavu, 2025; Ahmed et al., 2025; Aljedaani et al., 2024). While LLMs can sometimes surpass human-written code in accessibility, they still struggle in compliance (Suh et al., 2025). Novice developers using AI assistants also frequently omit key practices, underscoring current limitations (Mowar et al., 2025). To address these issues, practical tools like CodeA11y (Mowar et al., 2025), a VS Code plugin (Calì et al., 2025), and ACCESS for real-time in-DOM correction (Huang et al., 2024) provide LLM-based accessibility support. Feeda11y (Suh et al., 2025) further improves accessibility by applying feedback loops to iteratively prompt LLMs for better compliance. Yet such methods remain costly because the inference overhead often exceeds the training cost. This motivates training models that natively generate accessible code by design.

3 Methodology

A11yn aligns code-generative LLMs to improve the *accessibility* of generated web UI code. The method incorporates a novel accessibility reward through reinforcement learning. Below, we outline (1) the preliminaries of the approach, (2) the reward function design, and (3) the training pipeline.

3.1 Preliminary

GRPO (Shao et al., 2024) is a policy gradient method that simplifies PPO (Schulman et al., 2017) by removing the critic network and instead comparing sampled completions. For a prompt q, the policy π_θ samples G candidate completions $\{o_1, \ldots, o_G\}$, each assigned a scalar reward r_i (section 3.2). GRPO normalizes these via $\hat{A}_i = \frac{r_i - \bar{r}}{\sigma}$, emphasizing relative accessibility improvements rather than the absolute scores to stabilize updates. At the token level generation, the probability ratio $r_t^{(i)}(\theta) = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$ quantifies how the new policy changes the likelihood of generating token $o_{i,t}$ conditioned on the prompt q and previously generated tokens $o_{i,<t}$. The clipped surrogate loss $L_t^{(i)}(\theta)$, then bounds large ratios to avoid overcorrection. The overall objective averages these token-level terms with a KL penalty against a frozen reference policy π_{ref}:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\{o_i\}\sim\pi_{\theta_{\text{old}}}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\underbrace{\min\Big(r_t^{(i)}(\theta)\hat{A}_i,\ \text{clip}\Big(r_t^{(i)}(\theta),\, 1-\epsilon,\, 1+\epsilon\Big)\hat{A}_i\Big)}_{L_t^{(i)}(\theta)}\right] - \beta\, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}). \tag{1}$$

The KL penalty constrains excessive divergence from the reference policy, preserving general code generation ability while guiding completions toward accessibility. In our task, GRPO is especially well-suited due to its data-efficient and stable optimization method, whereas SFT relies on large paired accessible code datasets and PPO entails an engineering and computation overhead of training a critic model that estimates value for accessibility.

3.2 Accessibility Reward

To guide the A11yn policy towards generating accessible web UI code, we design a reward function with Web Content Accessibility Guidelines (WCAG) auditing tool. After current policy model π_θ generating each web UI code output, we run Axe-core (Deque Systems, 2015), a widely used open-source accessibility engine, to detect violations of the WCAG. For each response completion, Axe-core returns a list of affected DOM nodes, where each violation is classified by severity $v \in \{\text{Minor, Moderate, Serious, Critical}\}$. For a UI output o_i, we let $V(o_i)$ denote the set of severity

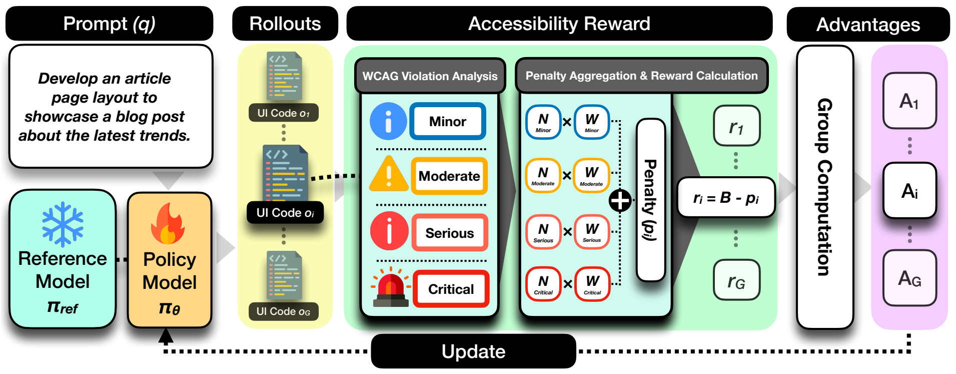


Figure 2: **A11yn optimizes accessibility through reinforcement learning (GRPO).** For an instruction q, the policy LLM π_θ generates candidate UI codes $\{o_1, \ldots, o_G\}$. Each code receives an accessibility reward $\{r_1, \ldots, r_G\}$, which is normalized within the set of candidates to compute advantages. The policy π_θ is then updated via policy gradient using these advantages.

levels detected in that output. The number of nodes associated with each violation level is counted and denoted as N_v. The total penalty p_i for a UI output o_i is computed by aggregating the affected DOM nodes, weighting each by its severity:

$$p_i = \sum_{v \in \mathcal{V}(o_i)} \mathrm{N}_v \cdot w_v \tag{2}$$

where severity weights are $w_v \in \{0.1, 0.2, 0.3, 0.4\}$ corresponding respectively to Minor, Moderate, Serious, and Critical violations. While the scales are arbitrary, the scheme ensures more severe violations to incur systematically larger penalties. We then convert this into a bounded reward by subtracting the penalty from a base score B, where we use $B = 2.0$ empirically, and clip the reward to zero for negative values.

$$r_i = B - p_i \tag{3}$$

Under this quantitative reward signaling scheme, a violation-free output converges toward $r_i \approx B$, and each violation proportionally lowers the reward. By assigning larger negative weights to more severe issues, the policy is encouraged to eliminate severe failures first. In practice, the policy model receives a solid numerical score that reflects the accessibility testing environment that is appropriate for giving RL feedback.

3.3 TRAINING PIPELINE

We instantiate **A11yn** as a GRPO based reinforcement learning pipeline with *rollout-reward–update* cycle that repeatedly steers the policy toward accessibility-compliant code as illustrated in fig. 2. We use Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) as our policy model π_θ and simultaneously use its frozen copy as the reference model π_{ref}, since it is pre-trained and capable of generating web contents based on natural language request. In each iteration, a textual UI request prompt q from training prompt set (section 4.1) is retrieved. The current policy π_θ then generates a group of G candidate completions $\{o_i\}_{i=1}^{G}$, producing diverse web UI code alternatives for the same prompt. Each completion is evaluated with Axe-core (Deque Systems, 2015), where generated web contents are rendered in a headless Chromium instance and analyzed for WCAG violations. The detected violations are converted into scalar penalties using the severity-weighted mapping described in section 3.2, yielding an accessibility reward r_i for each completion. Group statistics, mean $\bar{r}$ and standard deviation σ are computed to form normalized advantages. Then, these group-normalized advantages focus updates on relative improvements among the sampled completions, favoring code patterns that have minimal WCAG violations in the same group.

4 DATA

4.1 TRAINING: UIREQ-6.8K

To train A11yn, we construct UIReq-6.8K, a reinforcement learning training dataset of 6,800 UI generation instructions. As shown in fig. 3, the dataset spans a wide range of domains and interaction patterns, supporting broad coverage of instruction types. Unlike supervised datasets, UIReq-6.8K does not fix target UIs for each request, which enables exploration and reward optimization without imposing stylistic bias. Each instruction prompt in UIReq-6.8K describes a desired user interface in natural language, specifying page type, application domain, specific web UI components, or stylistic intent (e.g. a dark-themed login screen with email and password inputs). The instruction prompts are generated using GPT-4o-mini (OpenAI et al., 2024) and guided to reflect diversity and semantic richness. Diversity is achieved by covering 68 application categories (appendix A). Semantic richness is enforced through detailed requirements in instruction prompt synthesis, where every request specifies its page type, application domain, specific web UI components, and stylistic intent.

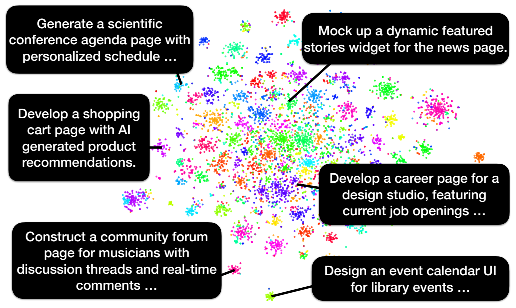


Figure 3: **t-SNE visualization of the training set.** Each point represents a UI request, with colors indicating distinct application categories. The spread shows coverage across multiple domains with examples in the figure.

4.2 EVALUATION: REALUIREQ-300

Dataset	**Query Style**	**UI Intent Coverage**	**Component Details**	**UI Source**	**Query Length**	**Number of Queries**
Screen2Words (Wang et al., 2021)	Single sentence, Taxonomic	✗	✗	RICO dataset (Android UI)	6 words	112k
RealUIReq-300 (Ours)	Multi-sentence, Request-oriented	✓	✓	Real-World Web UI	87 words	300

Table 1: **Comparison of REALUIREQ-300 with Screen2Words.** REALUIREQ-300 provides multi-sentence requests with structured intent, detailed UI component specification, and realistic phrasing grounded in real-world web UIs.

We assess the accessibility of web UI within the broader scope of natural language to web UI code generation flow. To achieve this, a realistic request-style benchmark dataset was required, one that could capture authentic user intents and interface specifications instead of relying on fully synthetic or overly simplified captions.

Screen2Words (Wang et al., 2021) is the most widely adopted dataset for natural language description of user interfaces. Built on the RICO dataset (Deka et al., 2017) of Android application UIs, its primary objective is to provide concise textual summaries of the UI screen to bridge user interfaces and natural language. While valuable in scale, the descriptions are short and taxonomic (e.g. sign in page of a social app, page displaying data status) rather than being detailed and request-oriented. Evaluating with short summaries risks emphasizing superficial matches over true task alignment. Such an absence of explicit intent or evaluation points (e.g. UI component details or requirements) in the descriptions further introduces ambiguity, making the benchmark less reliable.

To address these limitations, we introduce RealUIReq-300, a benchmark of 300 web UI requests inversely generated from manually collected webpage screenshots. As shown in fig. 4, each example was curated through a multi-stage pipeline involving screenshot collection, metadata extraction, and request generation, with GPT-4.1 (OpenAI et al., 2024) assisting in extraction and request phrasing. All metadata and requests were manually refined by the authors to correct for truncation, vague language, or missing context. This process ensured that the final requests faithfully represent the semantics of original UIs while maintaining natural and realistic phrasing. As compared in table 1, RealUIReq-300 offers multi-sentence, request-style instructions with intent, page type, UI components, and domain context specifications. This makes the evaluation set semantically rich, structurally aligned for assessing natural language to UI generation.

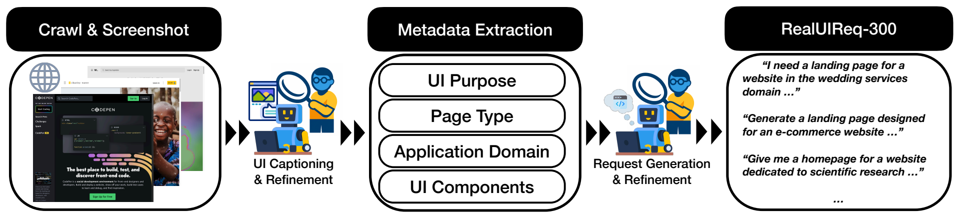

Crawl & Screenshot
UI Captioning & Refinement
Metadata Extraction
UI Purpose
Page Type
Application Domain
UI Components
Request Generation & Refinement
RealUIReq-300
"I need a landing page for a website in the wedding services domain ..."
"Generate a landing page designed for an e-commerce website ..."
"Give me a homepage for a website dedicated to scientific research ..."
...

Figure 4: **RealUIReq-300 is curated from real web UIs with diverse use-case domains.** User requests are inversely generated from screenshots and metadata extracted, then refined to produce realistic instructions aligned with the original UIs.

5 Experiments

5.1 Setup

Our objective is to evaluate A11yn and baseline LLMs in generating web UIs with minimal accessibility violations. We further evaluate if A11yn and selected baselines are able to balance accessibility and semantic alignment with visual appeal in section 6.3.2. Each model is tested on RealUIReq-300 benchmark, an evaluation set of 300 web UI request prompts, designed to ensure consistent and controlled comparisons. Inference of all model candidates is performed with a temperature of 0.1 for near-deterministic reproducibility.

5.2 Metrics

To assess the accessibility of model-generated responses, we adopt a comprehensive set of evaluation metrics informed by the principles of the Web Content Accessibility Guidelines (WCAG) from the accessibility auditing tool. Our evaluation comprises three main metrics designed for robustness and fairness. First, we measure the average DOM counts with accessibility violations detected across the evaluation set, categorized by severity: **Minor**, **Moderate**, **Serious**, and **Critical**. Each severity level reflects the impact of the violation on user experience, ranging from minor impact issues to critical barriers that significantly hinder accessibility.

To account for the varying severity of accessibility violations, we propose the **Weighted Violation Score (WVS)**, which quantifies accessibility violations by assigning severity-based weights to affected DOM nodes at each severity category. The WVS is formally defined as:

$$\text{WVS} = \lambda_{\text{Minor}} \cdot N_{\text{Minor}} + \lambda_{\text{Moderate}} \cdot N_{\text{Moderate}} + \lambda_{\text{Serious}} \cdot N_{\text{Serious}} + \lambda_{\text{Critical}} \cdot N_{\text{Critical}} \tag{4}$$

where N_{Minor}, N_{Moderate}, N_{Serious}, and N_{Critical} represent the number of violated DOM counts at each severity level from the generated code with RealUIReq-300 request prompts. The corresponding weights λ reflect the relative impact of each category, with values of 1 for *Minor*, 2 for *Moderate*, 3 for *Serious*, and 4 for *Critical*. This formulation provides a single interpretable metric that captures both the frequency and severity of accessibility issues.

Finally, since models differ in scale and generate varying length of web contents, we adopt a normalized metric to enable fair comparison across models. Inspired by the **Inaccessibility Rate** introduced in Feeda11y (Suh et al., 2025), we calculate the ratio of weighted violations to the total number of DOM elements produced when prompted with RealUIReq-300 requests. Given our use of a different auditing tool (Axe core Deque Systems (2015)), we adapt the original formulation to incorporate the WVS, resulting in the following metric:

$$\text{Inaccessibility Rate} = \frac{\text{WVS}}{\text{No. of Total DOM Elements}} \tag{5}$$

This metric captures the normalized, severity-adjusted density of accessibility violations, allowing us to evaluate the true accessibility in proportion to UI complexity.

5.3 Baselines

We compare our work against five baseline models to evaluate its relative performance. **(1) Qwen2.5-Coder-7B-Instruct** serves as the base model from which A11yn is GRPO-tuned. It re-

Model	**Average Violated DOM Counts (↓)**				**WVS (↓)**	**IR (↓)**
	Minor	**Moderate**	**Serious**	**Critical**		
Qwen2.5-Coder-7B-Instruct	<u>1</u> (±1)	1149 (±36)	978 (±48)	40 (±0)	5392 (±35)	0.38 (±0.0)
+ Feeda11y	11 (±1)	<u>461</u> (±37)	<u>841</u> (±33)	<u>30</u> (±5)	<u>3576</u> (±64)	<u>0.21</u> (±0.0)
Qwen2.5-Coder-14B-Instruct	3 (±2)	846 (±35)	1491 (±32)	49 (±8)	6365 (±158)	0.43 (±0.0)
GPT-4.1	45 (±5)	1925 (±34)	1424 (±21)	105 (±7)	8588 (±100)	0.27 (±0.0)
Claude Sonnet 4	2 (±3)	3388 (±81)	1435 (±14)	282 (±10)	12210 (±117)	0.29 (±0.0)
A11yn (Ours)	**0** (±0)	**231** (±16)	**481** (±23)	**24** (±3)	**1918** (±65)	**0.15** (±0.0)

Table 2: **Accessibility measures across models.** We report Average Violated DOM Counts at different severity levels. Weighted Violation Score (WVS) and Inaccessibility Rate (IR) provide severity-adjusted and normalized aggregate measures, respectively. Lower values indicate better performance. Best results are shown in **bold**, and second-best in <u>underline</u>.

flects the model's raw web UI code generation capability in zero shot setting without any explicit accessibility optimization. **(2) Qwen2.5-Coder-7B-Instruct** ***(+Feeda11y)*** is used to examine the impact of accessibility-aware prompting. This variant incorporates Feeda11y (Suh et al., 2025) prompts using a three-step iterative ReAct prompting (Yao et al., 2023) method with violation report feedbacks. **(3) Qwen2.5-Coder-14B-Instruct** is included to assess the effect of model scaling, offering a larger alternative from the same model family. In addition, we evaluate two frontier models: **(4) GPT-4.1** (OpenAI et al., 2024) and **(5) Claude Sonnet 4** (Anthropic, 2025), both of which represent the well performing models in general-purpose code and web UI generation.

6 Results

6.1 Quantitative Results

Table 2 summarizes the accessibility performance of A11yn against five baselines. Frontier models like GPT-4.1 and Claude Sonnet 4 yield relatively high inaccessibility rates (0.27 and 0.29), indicating that strong models do not guarantee accessible outputs. A11yn achieves the lowest Weighted Violation Score (WVS) and Inaccessibility Rate (IR), significantly outperforming both prompt-based approaches and frontier models. Compared to the base model, A11yn has critical violations reduced from 40 to 24 (40% ↓), serious from 978 to 481 (50.8% ↓), moderate from 1149 to 231 (79.9% ↓), Weighted Violation Score from 5392 to 1918 (64.4% ↓), and Inaccessibility Rate from 0.38 to 0.15 (60.5% ↓), demonstrating substantial improvements in accessibility conformity. Base model with Feeda11y shows notable improvement, achieving a WVS of 3576 and an Inaccessibility Rate of 0.21, yet remaining behind A11yn. Also, its iterative prompting brings up computational overhead, averaging 4584 intermediate tokens per request.

6.2 Qualitative Examples

Figure 5 illustrates the rendered UI and the corresponding accessibility tree pairs for the base model and A11yn. Beyond the visible improvements, the accessibility tree provides a structural view of how assistive technologies like screen readers or switch devices interpret and navigate the UI page.

The base model (left) contains several violations across different severity levels, which are reflected directly in its accessibility tree. The weak color contrast (Serious) issue of the text, visible in the navigation bar, prevents low vision users from reliably perceiving text. The missing landmark semantics (moderate) issue leaves sections in the middle like "Featured Movies & Shows" to be stated without clear structural roles, disrupting the navigation for screen-reader users. The iframe elements with the absence of accessible names cause media contents like embedded videos to be exposed without meaningful descriptions for non-sighted users. By contrast, A11yn (right) produces UI output with strong color contrast, sections well positioned in Header–Main–Footer landmarks, and iframe elements annotated with descriptive accessible names. Its accessibility tree reflects these improvements through a coherent hierarchy with properly assigned semantic roles, demonstrating that A11yn enhances both the visual presentation and the assistive-technology interpretation of the web UI.

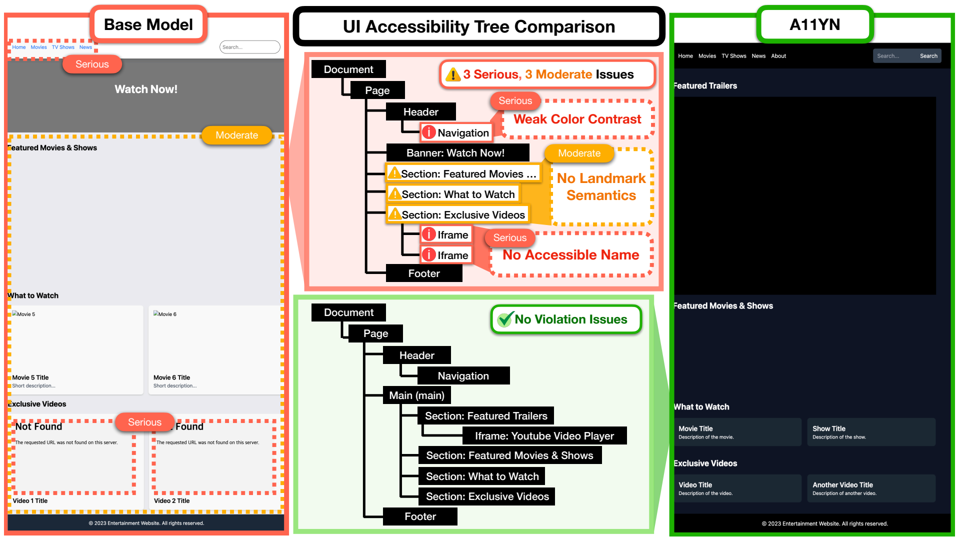


Figure 5: **Comparison of the base model (left) and A11yn (right) using both rendered UIs and their corresponding accessibility trees (middle).** The accessibility tree reveals the base model's weak color contrast, missing landmarks, and unlabeled media nodes, while A11yn provides clear landmarks, stronger contrast, and descriptive accessible names, resulting in a clean, fully interpretable structure.

6.3 ANALYSIS

6.3.1 ACCESSIBILITY IMPROVEMENT OF A11YN OVER BASELINES

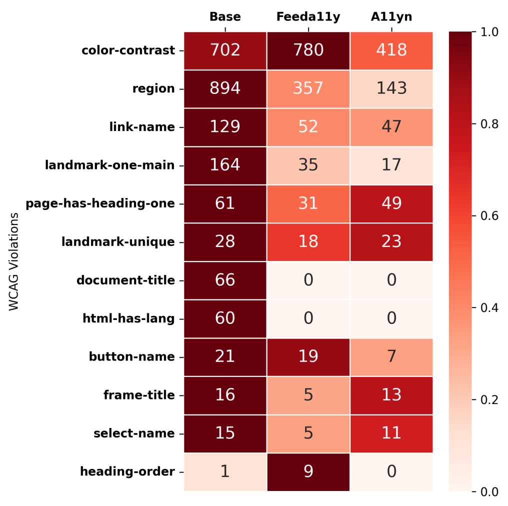


Figure 6: **Per-category violation counts across models.** Colors are normalized per row for visualization. (lighter = fewer violations, darker = more violations).

Figure 6 demonstrates a comparison of average accessibility violation distribution among the Base model, Feeda11y and A11yn across multiple violation types in rendering of the evaluation set UI requests. It reveals that A11yn reduces a range of key accessibility issues, and the most notable improvements are observed in the following violation categories:

Region - All page content must be contained by landmarks A11yn reduces average Region violations from 894 to 143. These violations indicate failures to encapsulate page content within landmark regions like `<main>`, `<nav>`, `<header>` tags. Proper use of HTML5 and ARIA landmarks is critical for screen reader users, allowing them to move directly to key sections of a webpage, facilitating efficient content navigation. It shows how A11yn has improved structural accessibility of the generated web UIs.

Color Contrast - Elements must meet minimum color contrast ratio thresholds As demonstrated in fig. 5, weak color contrast relates to insufficient contrast between text and its background. This poses a major barrier for users with low vision. WCAG recommends a minimum contrast ratio of 4.50:1. A11yn reduces the count from 702 to 418, showing enhancement in awareness of visual accessibility standards.

Landmark-one-main - document should have one main landmark To enhance the browsing experience for screen readers, Web UI design must allow quick and easy identification and navigation to the page's main content. With such aim, each page must include a single `<main>` landmark to clearly designate the primary content area. Using multiple or omitting the `<main>` tag can cause confusion for assistive technologies. A11yn enforces this guideline, reducing the count from 164 to 17, ensuring interpretable content hierarchies.

Link-name - Links must have discernible text Every hyperlink should have a clear, descriptive label to guide screen readers to understand its destination or action. Common issues include empty anchor tags or overly generic text like "click here." A reduction from 129 to 47 violations suggests A11yn reliably assigns accessible and descriptive link texts, mitigating issues like empty or duplicated links that confuse users.

6.3.2 Accessibility with Semantic Accuracy and Aesthetics of User Interfaces

Web UI generation task is multi-dimensional, where diverse design objectives must be considered. Effective user interfaces must guarantee accessibility while preserving semantic fidelity and visually appealing designs. Prior studies highlight that accessibility and aesthetics are often perceived in tension (Anthony, 2019), yet must be balanced rather than being treated as opposing forces (Kurosu & Kashimura, 1995; Mbipom & Harper, 2011; Le-Cong et al., 2021). Therefore, while our work primarily focuses on accessibility enhancement, it is equally important that improvements do not compromise semantic fidelity or aesthetics. To this end, we additionally evaluate appearance quality to verify whether accessibility gains are achieved without harming other key dimensions.

For this evaluation, we adopt the Appearance Score from WebGen-Bench (Lu et al., 2025), a 5-point Likert scale rated by GPT-4.1 on rendering quality, content relevance, layout harmony, and modernity. The Appearance Score serves as a core metric capturing both semantic fidelity and aesthetics. As shown in table 3, A11yn achieves the lowest inaccessibility rate, marking a 60.5% reduction over the base model, while maintaining an appearance score of 3.6. This demonstrates that A11yn substantially improves accessibility while preserving aesthetics and fidelity, achieving a balanced outcome.

Model	**Inaccessibility Rate (↓)**	**Appearance Score (↑)**
Base model	0.38	<u>3.6/5</u>
Feeda11y	<u>0.21</u>	**3.7/5**
A11yn (Ours)	**0.15**	<u>3.6/5</u>

Table 3: Comparison of models in terms of Accessibility (Inaccessibility Rate) alongside Semantic Fidelity and Aesthetics (Appearance Score on a 5-point Likert scale).

By comparison, Feeda11y achieves a higher appearance score (3.7) but retains a relatively high inaccessibility rate (0.21). This indicates that Feeda11y's improvements incidentally enhance visual quality rather than systematically addressing accessibility, reflecting a shifted emphasis. In contrast, A11yn achieves lower inaccessibility rate (0.15) while maintaining the Appearance Score intact (3.6), offering stronger evidence of accessibility enhancement with balance. Moreover, A11yn attains such outcome in a single forward pass, whereas Feeda11y relies on iterative prompting.

7 Conclusion

Web accessibility is not merely a design preference but a foundational requirement for equitable digital access. While prior efforts have explored ways to support accessibility in LLM-based code generation through prompting, feedback loops or IDE-based assistance, they remained external or computationally intensive. Furthermore, such works concluded with future calls for the need of training LLMs to inherently generate accessible web UI code. Through introducing A11yn, we take a complementary but novel path by empirically suggesting that accessibility can be systematically optimized in code-generating LLMs through post-training with reward-driven alignment. Looking ahead, we believe this paradigm can be extended beyond web UI code and into broader human-computer interaction systems such as mobile applications, AR/VR environments, and multimodal interaction platforms.

Ethics Statement

This work aims to improve digital equity by aligning code-generating LLMs to produce accessibility-compliant web UIs, thereby reducing barriers for users with disabilities. All training data were synthetically generated through controlled prompting, and evaluation data were curated from publicly available web pages with manual refinement with sensitive or personally identifiable content removed. No human subjects or private data were involved. While misuse could enable mass generation of low-quality web pages, we mitigate this risk by committing to open release of data, code, and documentation to guide responsible, accessibility-focused research.

Reproducibility statement

We ensure reproducibility by documenting all datasets, training details, and evaluation procedures. Training data (UIReq-6.8K) and the evaluation benchmark (RealUIReq-300) are fully described, along with synthesis prompts. Model training used *Qwen2.5-Coder-7B-Instruct* with Group-Relative Policy Optimization, with detailed hyperparameters and hardware setups provided in appendix C. Accessibility compliance was measured using the open-source axe-core engine (Deque Systems, 2015), which is distributed under the Mozilla Public License 2.0, to detect WCAG violations and compute reward signals. Upon acceptance, we will release all code, data, and model configurations to allow independent verification and replication of results.

References

Ammar Ahmed, Margarida Fresco, Fredrik Forsberg, and Hallvard Grotli. From code to compliance: Assessing chatgpt's utility in designing an accessible webpage – a case study, 2025. URL `https://arxiv.org/abs/2501.03572`.

Wajdi Aljedaani, Abdulrahman Habib, Ahmed Aljohani, Marcelo Eler, and Yunhe Feng. Does chatgpt generate accessible code? investigating accessibility challenges in llm-generated source code. In *Proceedings of the 21st International Web for All Conference*, pp. 165–176, 2024.

Anthony. The aesthetic-accessibility paradox. `https://uxmovement.com/thinking/the-aesthetic-accessibility-paradox/`, November 2019. Accessed: 2025-07-29.

Anthropic. Claude opus 4 & claude sonnet 4: System card. Technical report, Anthropic, May 2025. URL `https://wwwcdn.anthropic.com/07b2a3f9902ee19fe39a36ca638e5ae987bc64dd.pdf`. Accessed: 2025-07-28.

Elisa Calì, Tommaso Fulcini, Riccardo Coppola, Lorenzo Laudadio, and Marco Torchiano. A prototype vs code extension to improve web accessible development. In *2025 IEEE/ACM Second IDE Workshop (IDE)*, pp. 52–57. IEEE, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pp. 845–854, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349819. doi: 10.1145/3126594.3126651. URL `https://doi.org/10.1145/3126594.3126651`.

Deque Systems. axe-core Accessibility Engine. `https://github.com/dequelabs/axe-core`, 2015. Accessed: 2025-07-28.

Alexandra-Elena Guriţă and Radu-Daniel Vatavu. When llm-generated code perpetuates user interface accessibility barriers, how can we break the cycle. In *Proceedings of the 22nd International Web for All Conference (W4A'25)*, 2025.

Calista Huang, Alyssa Ma, Suchir Vyasamudri, Eugenie Puype, Sayem Kamal, Juan Belza Garcia, Salar Cheema, and Michael Lutz. Access: Prompt engineering for automated web accessibility violation corrections, 2024. URL `https://arxiv.org/abs/2401.16450`.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL `https://arxiv.org/abs/2409.12186`.

Masaaki Kurosu and Kaori Kashimura. Apparent usability vs. inherent usability: experimental analysis on the determinants of the apparent usability. In *Conference companion on Human factors in computing systems*, pp. 292–293, 1995.

Thanh Le-Cong, Xuan Bach D Le, Quyet Thang Huynh, and Phi Le Nguyen. Usability and aesthetics: Better together for automated repair of web pages. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 173–183. IEEE, 2021.

Xuying Li, Zhuo Li, Yuji Kosuga, and Victor Bian. Optimizing safe and aligned language generation: A multi-objective grpo approach, 2025. URL `https://arxiv.org/abs/2503.21819`.

Zimu Lu, Yunqiao Yang, Houxing Ren, Haotian Hou, Han Xiao, Ke Wang, Weikang Shi, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Webgen-bench: Evaluating llms on generating interactive and functional websites from scratch, 2025. URL `https://arxiv.org/abs/2505.03733`.

Beatriz Martins and Carlos Duarte. A large-scale web accessibility analysis considering technology adoption. *Universal Access in the Information Society*, 23(4):1857–1872, 2024.

Mathematical Association of America. Aime i and ii 2025: American invitational mathematics examination. `https://artofproblemsolving.com/wiki/index.php/2025_AIME_I_Problems`, 2025. Accessed: 2025-09-05.

Grace Mbipom and Simon Harper. The interplay between web aesthetics and accessibility. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, pp. 147–154, 2011.

Kevin Moran, Carlos Bernal-Cárdenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. Machine learning-based prototyping of graphical user interfaces for mobile apps. *IEEE transactions on software engineering*, 46(2):196–221, 2018.

Peya Mowar, Yi-Hao Peng, Jason Wu, Aaron Steinfeld, and Jeffrey P Bigham. Codea11y: Making ai coding assistants useful for accessible web development. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2025.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

Hyunjae Suh, Mahan Tafreshipour, Sam Malek, and Iftekhar Ahmed. Human or llm? a comparative study on accessible code generation capability, 2025. URL `https://arxiv.org/abs/2503.15885`.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning, 2021. URL `https://arxiv.org/abs/2108.03353`.

Jason Wu, Eldon Schoop, Alan Leung, Titus Barik, Jeffrey Bigham, and Jeffrey Nichols. UICoder: Finetuning large language models to generate user interface code through automated feedback. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7511–7525, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.417. URL `https://aclanthology.org/2024.naacl-long.417/`.

Jingyu Xiao, Ming Wang, Man Ho Lam, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R. Lyu. Designbench: A comprehensive benchmark for mllm-based front-end code generation, 2025. URL `https://arxiv.org/abs/2506.06251`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Ting Zhou, Yanjie Zhao, Xinyi Hou, Xiaoyu Sun, Kai Chen, and Haoyu Wang. Declarui: Bridging design and development with automated declarative ui code generation. *Proceedings of the ACM on Software Engineering*, 2(FSE):219–241, 2025.

A UIReq-6.8K Application Domains

The training dataset covers 68 diverse application categories. Examples include:

- Business & Enterprise
- Health & Wellness
- Education & E-Learning
- Data & Analytics
- Communication & Social
- E-Commerce & Retail
- Finance & FinTech
- Real Estate & Property
- Media & Entertainment
- Food & Beverage
- Travel & Hospitality
- Developer Tools & Technology
- Science & Research
- Legal & Compliance
- Automotive & Mobility
- Government & Public Services
- Environment & Sustainability
- Security & Identity
- Non-Profit & Social Impact
- AI & Machine Learning
- Books & Reference
- Comics
- Dating
- Entertainment
- Events
- Finance
- Food & Drink
- Health & Fitness
- House & Home
- Libraries & Demo
- Lifestyle
- Maps & Navigation
- Medical
- Music & Audio
- News & Magazines
- Parenting
- Personalization
- Photography
- Productivity
- Shopping
- Social
- Sports
- Tools
- Travel & Local
- Video Players & Editors
- Weather
- Auto & Vehicles
- Beauty
- Art & Design
- Board
- Card
- Casino
- Casual
- Educational (Games)
- Music (Games)
- Puzzle
- Racing
- Role Playing
- Simulation
- Sports (Games)
- Strategy
- Trivia
- Word
- Augmented Reality
- Developer Tools
- Magazines & Newspapers
- Utilities
- Graphics & Design

B RealUIReq-300 Domain Distribution

Domain	**Count**	**Domain**	**Count**
Education	30	Technology/Software	23
E-commerce	19	News / Media	16
Non-profit / Humanitarian	13	Software development	12
Social media	12	Tourism	10
Web development	10	Software as a Service (SaaS)	9
Entertainment / Media	9	Entertainment ticketing	8
Business consulting	6	Professional services	6
Academic publishing	6	Food / Cooking	5
Personal development	5	Travel / Lifestyle	5
Finance / Investing	4	Tech product discovery and launch platform	4
Government services	4	Academic research	4
Environmental activism	4	Health and wellness	4
Community forum	4	Art and design	4
B2B marketing / Business services	4	Creative/Portfolio	3
Business communication and collaboration	3	Content publishing platform	3
Medical research	3	Personal blog / Thought leadership	3
Financial Services	3	Technology / Consumer Electronics	2
Crowdfunding	2	Outdoor recreation / Adventure sports	2
Music streaming and sharing	2	Education/Interactive media	2
Wedding Planning	2	Events and entertainment	2
Creative Networking/Showcase	2	Film and photography	2
Parenting / Community	2	Job board / Recruitment	2
Advocacy/Activism	2	Healthcare technology	2
Cloud computing / AI development	2	Retail / Fashion	1
Search engine	1	Social media for developers	1
Culture/History	1	Community/Women's empowerment	1
Freelance marketplace	1	Education/Science museum	1
Service industry (barbershop)	1	Gaming / Media	1
Microfinance / Social impact	1	Cloud storage and collaboration	1
Space exploration	1	Humor/Lifestyle	1
Religious/Charitable Organization	1		

Table 4: 61 Domain Distribution of RealUIReq-300 dataset

C Training Configuration

We trained *Qwen/Qwen2.5-Coder-7B-Instruct* on 8 NVIDIA A6000 GPUs (48GB VRAM each) using GRPO with vLLM-based sampling and reward modeling. Training was conducted in bfloat16 mixed-precision with gradient checkpointing enabled. Each prompt was expanded into $G = 6$ sampled completions, with a per-device batch size of 2 and gradient accumulation set to 6. To stabilize optimization, KL divergence regularization was applied with $\beta = 0.001$. Below is a summary of the key configurations used; for full details, please refer to the provided training scripts and configuration files.

Component	**Configuration**
Framework	HF Transformers, TRL (GRPOTrainer), Accelerate, DeepSpeed
Learning rate	5×10^{-5}
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-8}$)
Weight decay	0.1
LR Scheduler	Cosine (warmup ratio 0.01)
Gradient accumulation	6
Batch size	96 (2 per device $\times$ 8 processes $\times$ accumulation 6)
Epochs	2
Gradient checkpointing	Enabled
Precision	bfloat16
Loss type	grpo
KL β value	0.001
use_peft	Enabled

Table 5: Optimization and training parameters.

Parameter	**Value**
Engine	vLLM (colocated mode)
GPU memory utilization	60%
Min-p	0.1
Top-p	1.0
Top-k	-1
Temperature	0.7
Repetition penalty	1.1
Stop sequence	`</answer>`
Max tokens	3072
Number of Generations	6
Seed	3407

Table 6: GRPO sampling parameters.

D Icon Attribution

Icons in the figures are sourced from Flaticon (`https://www.flaticon.com`) and are credited to their respective creators in accordance with Flaticon's licensing requirements.

E Use of LLM

We employed a large language model (LLM) to enhance the clarity and accuracy of our writing, particularly in identifying and correcting grammatical errors, typographical mistakes, and in rephrasing sentences for improved readability. Furthermore, the LLM was utilized in the data generation process to provide supplementary material in support of our study.

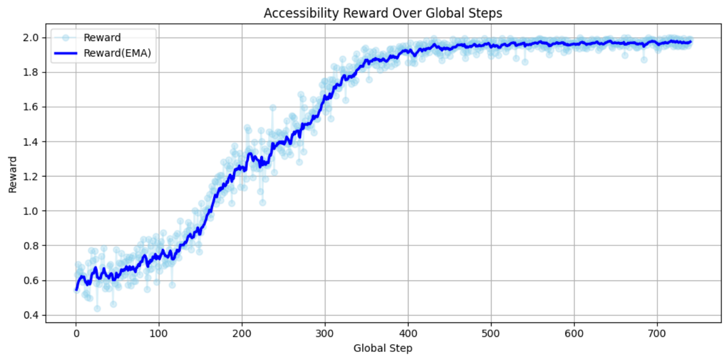


Figure 7: **Accessibility reward curve throughout training**, showing a steady increase that indicates reduced WCAG violation occurrences over time.

F WCAG VIOLATIONS

Below is the full list of WCAG violations.

Table 7: WCAG 2.0 — ARIA(Accessible Rich Internet Applications) Rules

Rule ID	Impact	Description
aria-allowed-attr	Serious, Critical	ARIA attributes are allowed for an element's role
aria-command-name	Serious	Every ARIA button, link, and menuitem has an accessible name
aria-hidden-body	Critical	`aria-hidden='true'` is not present on the `<body>` element
aria-hidden-focus	Serious	`aria-hidden` elements are not focusable nor contain focusable elements
aria-input-field-name	Moderate, Serious	Every ARIA input field has an accessible name
aria-meter-name	Serious	Every ARIA meter node has an accessible name
aria-progressbar-name	Serious	Every ARIA progressbar node has an accessible name
aria-required-attr	Critical	Elements with ARIA roles have all required ARIA attributes
aria-required-children	Critical	Elements with ARIA roles that require child roles contain them
aria-required-parent	Critical	Elements with ARIA roles that require parent roles are contained by them
aria-roles	Minor–Critical	Elements with a `role` use a valid value
aria-toggle-field-name	Moderate, Serious	Every ARIA toggle field has an accessible name
aria-tooltip-name	Serious	Every ARIA tooltip node has an accessible name
aria-valid-attr-value	Serious, Critical	All ARIA attributes have valid values
aria-valid-attr	Critical	Attributes beginning with `aria-` are valid ARIA attributes

Table 8: WCAG 2.0 — Text Alternatives & Captions

Rule ID	Impact	Description
area-alt	Critical	`<area>` elements of image maps have alternate text
image-alt	Critical	`<img>` has alt text or role of none/presentation
input-image-alt	Critical	`<input type=image>` has alternate text
object-alt	Serious	`<object>` elements have alternate text
role-img-alt	Serious	`role="img"` elements have alternate text
svg-img-alt	Serious	`<svg>` with img/graphics roles have accessible text
video-caption	Critical	`<video>` elements have captions

Table 9: WCAG 2.0 — Keyboard, Focus & Navigation

Rule ID	Impact	Description
bypass	Serious	Page has a mechanism to bypass navigation
nested-interactive	Serious	Interactive controls are not nested
scrollable-region-focusable	Serious	Scrollable content regions are keyboard accessible
server-side-image-map	Minor	Server-side image maps are not used

Table 10: WCAG 2.0 — Frames & Embeds

Rule ID	Impact	Description
frame-focusable-content	Serious	`<frame>`/`<iframe>` with focusable content do not have `tabindex=-1`
frame-title-unique	Serious	`<iframe>`/`<frame>` contain a unique `title` attribute
frame-title	Serious	`<iframe>`/`<frame>` have an accessible name

Table 11: WCAG 2.0 — Forms & Names

Rule ID	Impact	Description
button-name	Critical	Buttons have discernible text
input-button-name	Critical	Input buttons have discernible text
label	Minor–Critical	Every form element has a label
select-name	Minor–Critical	`<select>` has an accessible name
form-field-multiple-labels	Moderate	Form field does not have multiple label elements

Table 12: WCAG 2.0 — Structure & Semantics

Rule ID	Impact	Description
definition-list	Serious	`<dl>` elements are structured correctly
dlitem	Serious	`<dt>` and `<dd>` are contained by a `<dl>`
list	Serious	Lists are structured correctly
listitem	Serious	`<li>` elements are used semantically
document-title	Serious	Each HTML document contains a non-empty `<title>`

Table 13: WCAG 2.0 — Parsing & Uniqueness

Rule ID	Impact	Description
duplicate-id-active	Serious	Every `id` of active elements is unique
duplicate-id-aria	Critical	Every `id` used in ARIA and in labels is unique
duplicate-id	Minor	Every `id` attribute value is unique

Table 14: WCAG 2.0 — Color & Visual Presentation

Rule ID	Impact	Description
color-contrast	Serious	Foreground/background colors meet WCAG 2 AA contrast thresholds
link-in-text-block	Serious	Links are distinguishable from surrounding text without relying on color
meta-viewport	Critical	`<meta name="viewport">` does not disable text scaling and zooming
blink	Serious	`<blink>` elements are not used
marquee	Serious	`<marquee>` elements are not used
link-name	Serious	Links have discernible text

Table 15: WCAG 2.0 — Language

Rule ID	Impact	Description
html-has-lang	Serious	Document has a `lang` attribute
html-lang-valid	Serious	`lang` attribute on `<html>` has a valid value
html-xml-lang-mismatch	Moderate	`lang` and `xml:lang` agree on base language
valid-lang	Serious	`lang` attributes have valid values

Table 16: WCAG 2.0 — Data Tables

Rule ID	Impact	Description
td-headers-attr	Serious	Cells using `headers` refer only to cells in the same table
th-has-data-cells	Serious	`<th>` and header roles have data cells they describe

Table 17: WCAG 2.0 — User Control & Timing

Rule ID	Impact	Description
meta-refresh	Critical	`<meta http-equiv="refresh">` is not used for delayed refresh
no-autoplay-audio	Moderate	`<video>` or `<audio>` elements do not autoplay audio for more than 3 seconds without a control mechanism to stop or mute the audio

Table 18: Best Practices — ARIA(Accessible Rich Internet Applications)

Rule ID	Impact	Description
aria-allowed-role	Minor	`role` attribute has an appropriate value for the element
aria-dialog-name	Serious	ARIA dialog/alertdialog nodes have accessible names
aria-text	Serious	`role=text` used only on elements with no focusable descendants
aria-treeitem-name	Serious	ARIA treeitem nodes have accessible names
presentation-role-conflict	Minor	Presentational elements do not have global ARIA or `tabindex`
label-title-only	Serious	Every form element has a visible label and is not solely labeled using hidden labels, or the title or aria-describedby attributes
tabindex	Serious	`tabindex` attribute values are not greater than 0

Table 19: Best Practices — Landmarks & Regions

Rule ID	Impact	Description
landmark-banner-is-top-level	Moderate	Banner landmark is top level
landmark-complementary-is-top-level	Moderate	Complementary/aside landmark is top level
landmark-contentinfo-is-top-level	Moderate	Contentinfo landmark is top level
landmark-main-is-top-level	Moderate	Main landmark is top level
landmark-no-duplicate-banner	Moderate	At most one banner landmark
landmark-no-duplicate-contentinfo	Moderate	At most one contentinfo landmark
landmark-no-duplicate-main	Moderate	At most one main landmark
landmark-one-main	Moderate	Document has a main landmark
landmark-unique	Moderate	Landmarks have unique role/name/title combinations
region	Moderate	All page content is contained by landmarks
skip-link	Moderate	All skip links have a focusable target

Table 20: Best Practices — Headings & Structure

Rule ID	Impact	Description
empty-heading	Minor	Headings have discernible text
heading-order	Moderate	Heading order is semantically correct
page-has-heading-one	Moderate	Page (or a frame) contains a level-one heading
empty-table-header	Minor	Table headers have discernible text
accesskeys	Serious	Every accesskey attribute value is unique
image-redundant-alt	Minor	Image alternative is not repeated as text
meta-viewport-large	Minor	`<meta name="viewport">` can scale a significant amount

Table 21: Best Practices — Tables

Rule ID	Impact	Description
scope-attr-valid	Moderate, Critical	`scope` attribute is used correctly on tables
table-duplicate-name	Minor	`<caption>` text differs from `summary` attribute
frame-tested	Critical	`<iframe>` and `<frame>` elements contain the axe-core script

G PROMPT DETAILS

We provide the details of the prompt used in our work.

G.1 PROMPTS FOR DATASET SYNTHESIS

Prompt for Generating Training Data

```
Your task is to generate 100 different possible UI request
queries for a certain application domain category.

The application category is:
{category}
Requirements for queries:
1. Each query should include different page types possible
    within the app in the application domain
2. Each queries should be specific about widget requirements
    and context rich (side navigation, collapsible menus, etc.)
3. Each query must be different from the others
4. Queries should be realistic and natural like real user
    requests
5. Queries should be in some length and semantically rich,
    not just a few words
6. Queries should have solid use case or purpose, not random
    requests
7. Do not consider interactivity or animations or hovering
    effects focus on static UI elements
8. Queries require some ambience or style requests like
    color scheme, typography, etc.
```

Prompt for Captioning metadata for evaluation set curation

```
Caption given UI screenshot with
1. Main purpose and intent of the UI, regarding the target
   audience and actual use case.
2. Page type of the UI, such as a landing page, blog post,
   product page, etc. (in short-answer format)
3. Domain of the UI, such as e-commerce, social media,
   education, etc. (in short- answer format)
4. Top 5 important visual elements in the UI design, that are
   crucial for more user engagement and usability. But you
   should EXCLUDE any elements about the EXACT IMAGES in the
   UI.
Answer in such format:
1. <Purpose/Intent>
2. <Page Type>
3. <Domain>
4. (a) <Element (a)>
(b) <Element (b)>
(c) <Element (c)>
...
```

Prompt for generating RealUIReq-300 evaluation set queries

```
SYSTEM PROMPT:
You are a helpful assistant that generates realistic user
requests for web UI development.

USER PROMPT:
Based on the following web page specifications, generate a
user request that mentions EVERY DETAIL provided, including
the purpose, page type, domain, and all listed components.

Purpose: {purpose}
Page Type: {page_type}
Domain: {application_domain}
Required Components: {required_components}

The request should be 3-5 sentences long and sound realistic.
```

G.2 Prompts for Inference and Evaluation

Prompt for Inference of models for web UI generation

~~~
You are an expert UI designer assistant.
You should plan the design based on the user request.
Show the plan in the `<think>` tag.
    - You must think about the html structure and widgets
      needed to fulfill the user request.
    - You must think about the Tailwind CSS classes to use
      for styling.
Then, you should generate a complete HTML document that
includes:
    - A `<head>` section with a `<meta charset="UTF-8">` tag
    - A `<meta name="viewport" content="width=device-width,
      initial-scale=1.0">` tag
    - A proper tailwind css link tag to load Tailwind CSS
      from CDN
    - A `<body>` section that contains the complete HTML
      structure and content
The HTML document should be visually appealing, well-
structured, and content/semantically-rich.
You must strictly follow the output format shown below:
<think>
...
</think>

<answer>
```html
<html>
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0" />
    ...
</head>
<body>
...
</body>
</html>
```
</answer>

User: {user_request}
Assistant:
~~~

WebGen-Bench Appearance Score Evaluation Prompt (Lu et al., 2025)

```
Instruction: You are tasked with evaluating the functional
design of a webpage that had been constructed based on the
following instruction: {instruction}

Grade the webpage’s appearance on a scale of 1 to 5 (5 being
highest), considering the following criteria:
   - Successful Rendering: Does the webpage render correctly
     without visual errors? Are colors, fonts, and components
     displayed as specified?
   - Content Relevance: Does the design align with the
     webpage’s purpose and user requirements? Are elements
     (e.g., search bars, report formats) logically placed and
     functional?
   - Layout Harmony: Is the arrangement of components (text,
     images, buttons) balanced, intuitive, and clutter-free?
   - Modernness & Beauty: Does the design follow contemporary
     trends (e.g., minimalism, responsive layouts)? Are
     colors, typography, and visual hierarchy aesthetically
     pleasing?
Grading Scale:
   - 1 (Poor): Major rendering issues (e.g., broken layouts,
     incorrect colors). Content is irrelevant or missing.
     Layout is chaotic. Design is outdated or visually
     unappealing.
   - 2 (Below Average): Partial rendering with noticeable
     errors. Content is partially relevant but poorly
     organized. Layout lacks consistency. Design is basic
     or uninspired.
   - 3 (Average): Mostly rendered correctly with minor flaws.
     Content is relevant but lacks polish. Layout is
     functional but unremarkable. Design is clean but lacks
     modern flair.
   - 4 (Good): Rendered well with no major errors. Content is
     relevant and logically organized. Layout is harmonious
     and user-friendly. Design is modern and visually
     appealing.
   - 5 (Excellent): Flawless rendering. Content is highly
     relevant, intuitive, and tailored to user needs. Layout
     is polished, responsive, and innovative. Design is
     cutting-edge, beautiful, and memorable.

Task: Review the provided screenshot(s) of the webpage.
Provide a detailed analysis and then assign a grade (from 1
to 5) based on your analysis. Highlight strengths, weaknesses,
and how well the design adheres to the specifications, but
don’t mind the absence of images or cards for specific data
because they are not the target for evaluation.

IMPORTANT: Please end your response with a clear grade in the
format "Grade: X" where X is a number from 1 to 5.

Your Response Format:
  Analysis: [from 2 to 4 paragraphs addressing all criteria,
            referencing the instruction]
  Grade: [from 1 to 5]
  Your Response:
```

H Qualitative Web UI Style Diversity of A11yn

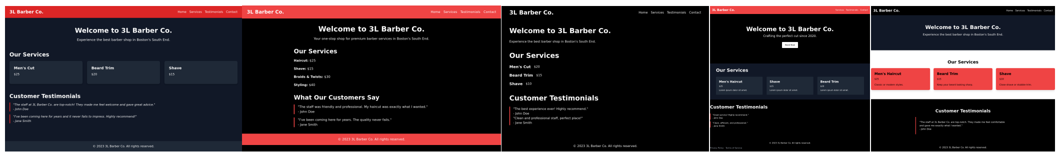


Figure 8: Style Diversity of A11yn for the UI Generation Prompt: *I want a landing page for 3L Barber Co., a barbershop located in Boston's South End. The page should attract and inform potential customers about our services and atmosphere, featuring a bold, contrasting color scheme of black, white, and red. I'd like to see large, eye-catching typography for the main headline, along with a clear navigation menu that includes essential sections. Additionally, please include a detailed list of services with prices and a section for customer testimonials and ratings.*

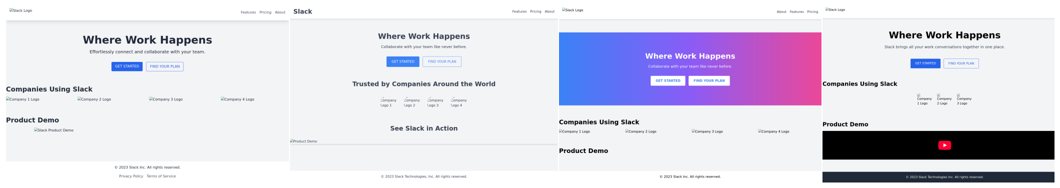


Figure 9: Style Diversity of A11yn for the UI Generation Prompt: *Please make a landing page designed to promote Slack as a collaborative workspace platform for teams. The page should feature a bold, attention-grabbing headline that says "Where Work Happens," along with prominent "GET STARTED" and "FIND YOUR PLAN" call-to-action buttons. Additionally, please include logos of well-known companies that use Slack, a clean and minimalist design with ample white space, and an interactive product demo or screenshot that showcases key features. The domain should focus on business communication and collaboration.*

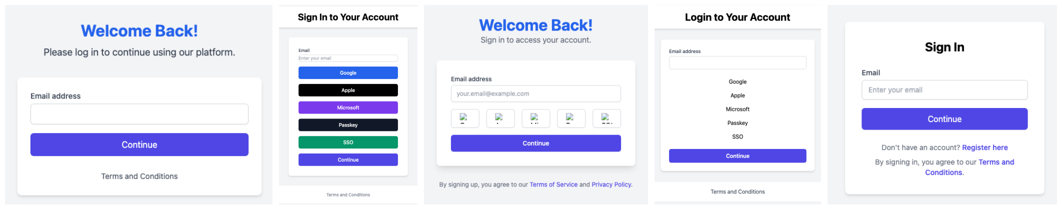
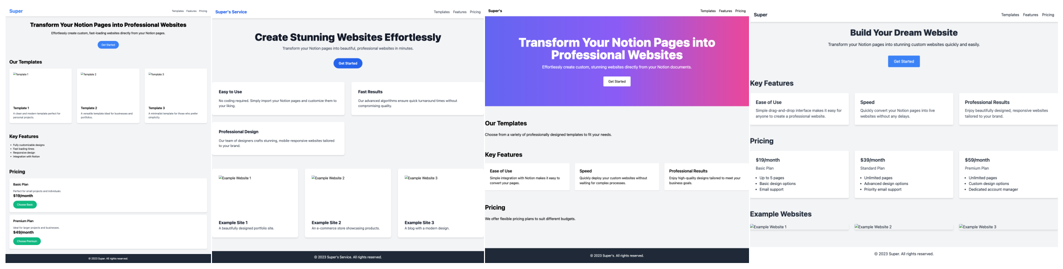

Welcome Back!
Please log in to continue using our platform.
Email address
Continue
Terms and Conditions
Sign In to Your Account
Google
Apple
Microsoft
Passkey
SSO
Sign in to access your account.
Login to Your Account
Sign In
Email
Enter your email
Don't have an account? Register here
By signing in, you agree to our Terms and Conditions.

Figure 10: Style Diversity of A11yn for the UI Generation Prompt: *I would like to have a login page for our productivity and collaboration platform that provides a sign-in interface for users to access membership content. This page should include multiple authentication options such as Google, Apple, Microsoft, passkey, and SSO for convenience and security. Additionally, please ensure there is a clear and concise page title explaining the purpose, an email input field for manual login, a large, prominent "Continue" button, and a link to the terms and conditions for transparency.*



Super

Templates Features Pricing

Transform Your Notion Pages into Professional Websites

Effortlessly create custom, fast-loading websites directly from your Notion pages.

Get Started

Our Templates

Template 1

Template 1

A clean and modern template perfect for personal projects.

Template 2

Template 2

A versatile template ideal for businesses and portfolios.

Template 3

Template 3

A minimalist template for those who prefer simplicity.

Key Features

- Fully customizable designs
- Fast loading times
- Responsive design
- Integration with Notion

Pricing

Basic Plan

Perfect for small projects and individuals.

$19/month

Choose Basic

Premium Plan

Ideal for larger projects and businesses.

$49/month

Choose Premium

© 2023 Super. All rights reserved.

Super's Service

Templates Features Pricing

Create Stunning Websites Effortlessly

Transform your Notion pages into beautiful, professional websites in minutes.

Get Started

Easy to Use

No coding required. Simply import your Notion pages and customize them to your liking.

Fast Results

Our advanced algorithms ensure quick turnaround times without compromising quality.

Professional Design

Our team of designers crafts stunning, mobile-responsive websites tailored to your brand.

Example Website 1

Example Site 1

A beautifully designed portfolio site.

Example Website 2

Example Site 2

An e-commerce store showcasing products.

Example Website 3

Example Site 3

A blog with a modern design.

© 2023 Super's Service. All rights reserved.

Super's

Templates Features Pricing

Transform Your Notion Pages into Professional Websites

Effortlessly create custom, stunning websites directly from your Notion documents.

Get Started

Our Templates

Choose from a variety of professionally designed templates to fit your needs.

Key Features

Ease of Use

Simple integration with Notion makes it easy to convert your pages.

Speed

Quickly deploy your custom websites without waiting for complex processes.

Professional Results

Enjoy high-quality designs tailored to meet your business goals.

Pricing

We offer flexible pricing plans to suit different budgets.

© 2023 Super's. All rights reserved.

Super

Templates Features Pricing

Build Your Dream Website

Transform your Notion pages into stunning custom websites quickly and easily.

Get Started

Key Features

Ease of Use

Simple drag-and-drop interface makes it easy for anyone to create a professional website.

Speed

Quickly convert your Notion pages into live websites without any delays.

Professional Results

Enjoy beautifully designed, responsive websites tailored to your brand.

Pricing

$19/month

Basic Plan

- Up to 5 pages
- Basic design options
- Email support

$39/month

Standard Plan

- Unlimited pages
- Advanced design options
- Priority email support

$59/month

Premium Plan

- Unlimited pages
- Custom design options
- Dedicated account manager

Example Websites

Example Website 1 Example Website 2 Example Website 3

© 2023 Super. All rights reserved.

Figure 11: Style Diversity of A11yn for the UI Generation Prompt: *I want a landing page for Super's service that creates custom websites from Notion pages. The purpose of the page is to attract and inform potential users about the ease of use, speed, and professional results of our service. It should include a clear, prominent headline explaining our core value proposition, a concise subheading detailing key benefits, and a prominent call-to-action button to get started. Additionally, please include a navigation menu with sections like Templates, Features, and Pricing, along with example website previews to showcase our capabilities. The domain is web development/SaaS.*