

A11yn: Aligning LLMs for Web Accessibility-Aware UI Generation

Janghan Yoon¹ Jaegwan Cho¹ Junhyeok Kim¹ Jiwan Chung¹
Jaehyun Jeon¹ Seungwon Lim¹ Youngjae Yu²
Yonsei University¹ Seoul National University²
jeffrobot99@yonsei.ac.kr

Abstract

Large language models can generate visually coherent web UIs from natural language requests, but they frequently violate Web Content Accessibility Guidelines (WCAG), excluding users with diverse needs and contexts. We address this by introducing A11yn, a post-training framework for web accessibility-aware web UI generation. A11yn converts off-the-shelf WCAG audits into a verifiable web accessibility reward, enabling accessibility alignment without data from dense human annotations. To support training and evaluation, we release UIReq-6.8K, a dataset of 6,800 diverse UI generation instructions, and RealUIReq-300, a benchmark of 300 realistic web UI generation tasks. Experimental results show that A11yn achieves the strongest web accessibility compliance among baselines, reducing the inaccessibility rate by 87.5% and 58.1% under two independent WCAG auditors relative to the base model. Unlike prompting-based post-hoc accessibility-correction baselines, A11yn achieves these gains in a single-pass generation setting while maintaining comparable semantic fidelity and visual quality. The code and dataset are available at <https://github.com/jeffrobot/A11yn>.

1 Introduction

Large Language Models (LLMs) can generate web interfaces from natural language requests, and prior UI-generation work has primarily emphasized design fidelity and visual coherence (Xiao et al., 2025; Zhou et al., 2025; Lu et al., 2025b). However, *Web Accessibility* is often missing from the interface-quality objectives used in LLM-based web UI generation. Web accessibility is a fundamental concept that concerns whether interfaces can be perceived and operated by users with diverse abilities; for example, whether content is interpretable by screen readers for blind users or navigable via keyboard-only interaction for users with motor impairments. The World Wide Web Consortium (W3C) formalizes web accessibility requirements through the Web Content Accessibility Guidelines (WCAG) (World Wide Web Consortium, 2024). WCAG is a technical standard developed through a rigorous international, multi-stakeholder process involving accessibility experts, people with disabilities, and organizations, providing a shared and actionable baseline for web accessibility. Consistent with this standards-based framing, web accessibility auditors are widely used for reproducible web UI evaluation through violation detection (Alsaeedi, 2020; Pool, 2023; Fischer et al., 2025).

Despite mature WCAG standards and auditing tools, basic web accessibility failures remain pervasive: large-scale audits report detectable WCAG violations on over 90% of public web pages (Mowar et al., 2025). LLMs trained on such web-scale data reproduce these defects in generated UIs. Recent studies consistently show missing alternative text, weak semantic landmarks, and improperly labeled controls in LLM-generated code (Suh et al., 2025; Aljedaani et al., 2024; Guriță & Vatavu, 2025). FeedA11y (Suh et al., 2025) demonstrates that prompting LLMs with web accessibility feedback can mitigate some violations, but it introduces unwanted violations during refinement, underscoring the limits of post-hoc prompting alone. At the application level, Mowar et al. (2025) further demonstrates that web accessibility-unaware coding assistants propagate and amplify omissions among novice

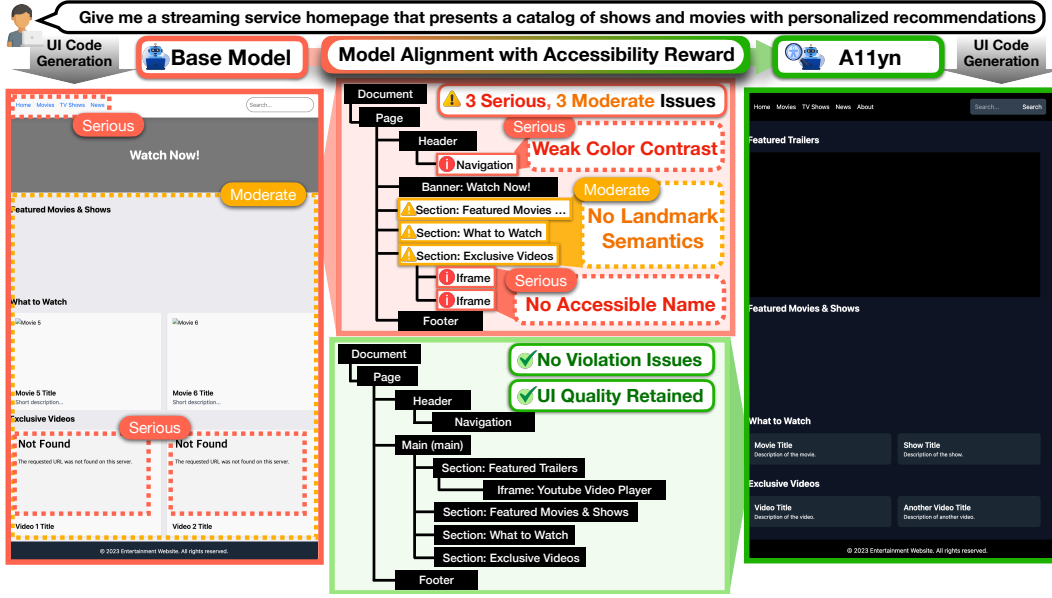


Figure 1: **Illustrative example of A11yn.** Compared with the base model, A11yn uses training-time alignment with an auditor-derived accessibility reward to generate a web UI with minimal WCAG violations in a single pass with comparable interface visual quality.

developers. These findings motivate a core research question: *Can we train LLMs to natively generate web UIs with minimal accessibility violations?*

To address this gap, we introduce **A11yn** (pronounced *align*), the first post-training framework for aligning code-generating LLMs toward web accessibility-aware UI generation. The prompting analysis (Appendix J) shows that explicit accessibility constraints improve accessibility but remain suboptimal, motivating behavioral alignment beyond prompt-only or post-hoc repair. A11yn therefore converts WCAG violation reports from Axe-core (Deque Systems, 2015) into verifiable rewards for GRPO (Shao et al., 2024), enabling optimization and internalization of web accessibility objectives for each request.

Our contributions are threefold: (1) We formulate accessibility-aware UI generation as an auditor-guided reinforcement learning task and introduce a web-accessibility reward that converts off-the-shelf audit reports into a learnable signal. (2) We release two complementary resources: UIReq-6.8K for instruction-only RL training and RealUIReq-300 for realistic request-style evaluation. (3) We empirically show that A11yn improves accessibility in native generation and outperforms baselines, with gains validated across independent WCAG auditors while maintaining comparable semantic fidelity and visual quality.

2 Related Work

LLM-based UI Code Generation. With the advent of LLMs, generating UI code directly from high-level natural language descriptions has become feasible. Existing approaches have primarily focused on visual semantics, design, and functionality. UICoder (Wu et al., 2024) improves generation quality via automated feedback during fine-tuning, WebGen-Bench (Lu et al., 2025b) benchmarks aesthetic and functional multi-page web application generation, and WebGen-Agent (Lu et al., 2025a) enhances website generation from the perspective of visual design and functionality through post-training of LLMs. However, these methods do not explicitly optimize LLMs for web accessibility compliance.

Web Accessibility Assessment with Auditors Web accessibility evaluation with human participants is indispensable, but difficult to scale because of recruitment constraints and context-dependent variability (Mack et al., 2021). Consequently, a broad body of HCI

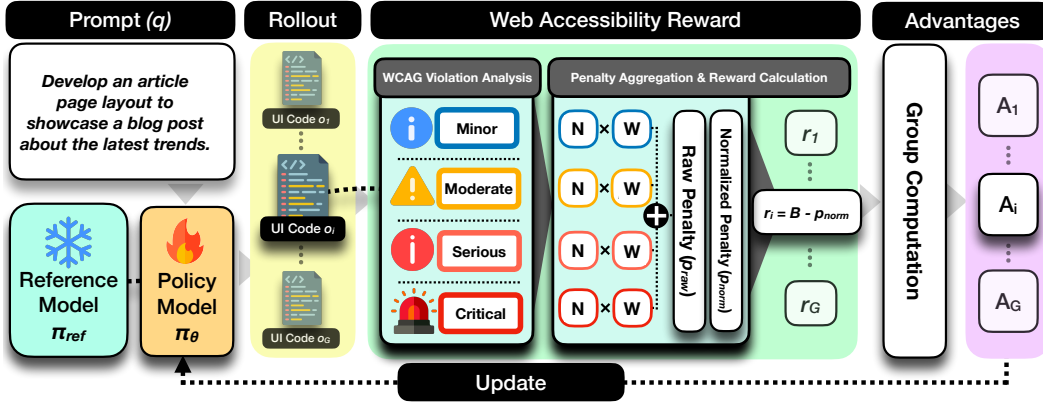


Figure 2: **A11yn aligns UI-generative LLMs using Web-Accessibility Reward.** For an instruction q , the policy LLM π_θ generates candidate UI codes $\{o_1, \dots, o_G\}$. Off-the-shelf violation feedback from a WCAG-based accessibility auditor is converted into a penalty-aware, objectively verifiable reward signal $\{r_1, \dots, r_G\}$ for each candidate. Rewards are normalized within the candidate set to compute advantages, which guide GRPO updates of π_θ toward outputs that prioritize web accessibility improvements.

and accessibility research adopts WCAG-based auditors as a reproducible and objective compliance baseline, and operationalizes accessibility improvement as reducing detectable barriers, i.e., WCAG violations reported by those auditors (Asok & Rekha, 2025; Ismail & Kuppusamy, 2022; Hong et al., 2007; Deque Systems, 2015; Gay & Li, 2010; Suh et al., 2025; Alsaeedi, 2020; Pool, 2023; Fischer et al., 2025). Furthermore, cross-validation across multiple tools is recommended to improve assessment robustness (Suh et al., 2025; Alsaeedi, 2020; Pool, 2023; Fischer et al., 2025). Following the established protocol, we use automated auditors as our primary evaluation method for WCAG compliance: we train and evaluate with Axe-core and report independent cross-auditor validation with AChecker.

LLM-based Accessibility Enhancement for Web UIs. Although web accessibility is fundamental for inclusive interfaces, it remains under-optimized in LLM-based web UI generation (Martins & Duarte, 2024; Guriță & Vatavu, 2025; Ahmed et al., 2025; Aljedaani et al., 2024). This gap has practical consequences: CodeA11y shows that accessibility-unaware coding assistants can propagate and amplify inaccessible patterns for novice developers (Mowar et al., 2025), and recent studies find that many generated UIs still fail baseline WCAG-aligned requirements (Suh et al., 2025; Mowar et al., 2025). Prior efforts, including FeedA11y (Suh et al., 2025), a VS Code plugin (Cali et al., 2025), and real-time in-DOM correction (Huang et al., 2024), improve accessibility via iterative feedback or post-hoc correction. However, new violations are introduced during refinement, motivating training-time methods that prioritize accessibility during single-pass generation (Suh et al., 2025).

3 Methodology

A11yn incorporates a novel, verifiable web accessibility reward for optimizing the LLM policy for web UI code generation. This section presents (1) the preliminary background of our approach, (2) web accessibility reward design, and (3) the overall training pipeline.

3.1 Preliminary Background

We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a policy gradient method that optimizes policies by comparing multiple sampled completions for the same prompt. Given a UI request q , the policy samples a group of candidates $\{o_1, \dots, o_G\}$, each assigned a scalar reward r_i . Rewards are normalized within the group to compute relative advantages, emphasizing comparative improvements rather than absolute scores. GRPO

updates the policy using a clipped surrogate objective with KL regularization against a frozen reference model:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q, \{o_i\} \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} L_t^{(i)}(\theta) \right] - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}). \quad (1)$$

Unlike PPO (Schulman et al., 2017) that relies on critic-estimated absolute returns, GRPO is critic-free and uses within-prompt candidate comparisons. This makes GRPO a better match for accessibility optimization, where rewards are most informative as relative improvements among alternative UIs for the same request and are more stable under sparse, prompt-dependent signals.

3.2 Web-Accessibility Reward

To train the LLM for accessible web UI generation, we design a reward that steers the model to be aware of WCAG accessibility violations. The objective is to reward outputs with minimal violations while avoiding strong incentives toward trivially simpler interfaces, which can degrade UI quality and the model’s generation capability throughout training.

Raw Penalty. After the policy LLM π_{θ} generates each web UI output, we run Axe-core (Deque Systems, 2015), a widely used open-source accessibility engine, to detect WCAG violations. For each completion, Axe-core reports affected DOM nodes together with severity levels $v \in \{\text{Minor, Moderate, Serious, Critical}\}$. For output o_i , $\mathcal{V}(o_i)$ denotes the set of detected severities and $N_v(o_i)$ is the number of affected nodes at severity v . We define the raw penalty as the severity-weighted sum of violated nodes:

$$P_{\text{raw}}^{(i)} = \sum_{v \in \mathcal{V}(o_i)} N_v(o_i) \cdot w_v, \quad (2)$$

where w_v is the penalty weight assigned to severity level v . The ablation study in Section 6.2.3 supports the choice of severity-dependent rather than uniform weighting.

DOM-Size Normalization. Raw penalty provides a violation-aware learning signal that encourages the policy to simply prefer samples with minimal accessibility failures. However, as larger DOMs naturally expose more opportunities for violations, optimizing raw penalty alone can bias the policy toward shorter, simpler outputs. This is especially problematic in GRPO, where candidates sampled for the same prompt are ranked within each rollout group despite potential differences in DOM size. We therefore normalize the raw penalty P_{raw} by DOM size as P_{norm} , where D_i is the number of rendered DOM elements in output o_i . This yields a size-aware accessibility penalty that enables fairer comparison among candidates with different interface sizes. We examine the empirical effect of this normalization against raw penalty in Section 6.2.3 and Figure 7. Finally, we define the reward by subtracting the normalized penalty from the base score B , which we set to 1.0.

$$R_i = B - P_{\text{norm}}^{(i)}, \quad \text{where} \quad P_{\text{norm}}^{(i)} = \frac{P_{\text{raw}}^{(i)}}{D_i}. \quad (3)$$

Under this quantitative reward signaling scheme, a violation-free output converges toward 1.0 as illustrated in Figure 8 in the appendix. In practice, the policy model receives a comparable numerical score that is appropriate for giving RL feedback.

3.3 Training Pipeline

As illustrated in Figure 2, we instantiate A11yn as a GRPO-based *rollout-reward-update* pipeline. We use Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) as the policy model π_{θ} and a frozen copy as the reference model π_{ref} . At each iteration, we sample a UI prompt q from UIReq-6.8K (section 4.1) and generate a group of G candidate completions $\{o_i\}_{i=1}^G$.

Each completion is rendered, audited by Axe-core (Deque Systems, 2015), and converted to reward r_i using the reward defined in section 3.2. GRPO then normalizes rewards within the group, computes relative advantages, and updates the policy toward lower-violation outputs. Because rewards are produced online through auditor calls, the pipeline scales without manually annotated accessibility labels. At inference time, the trained policy generates accessibility-aware UI code in a single pass.

4 Data

4.1 Training: UIReq-6.8K

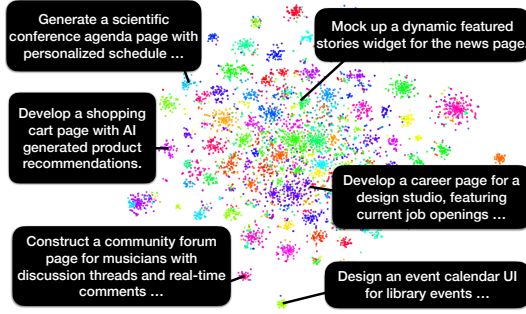


Figure 3: **t-SNE visualization of UIReq-6.8K demonstrating dataset diversity.** Each point corresponds to a UI request, colored by application category, highlighting broad coverage across domains and prompt interaction types.

To train A11yn, we construct UIReq-6.8K, a reinforcement learning training dataset of 6,800 UI generation instructions. As shown in Figure 3, the dataset spans a wide range of domains and interaction patterns, supporting broad coverage of instructions. Unlike supervised datasets, UIReq-6.8K does not fix target UIs for each request, which enables exploration and reward optimization.

Each instruction prompt in UIReq-6.8K describes a desired user interface in natural language, specifying page type, application domain, specific web UI components, and 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 covered by 68 application categories (section A). Semantic richness is enforced

through detailed requirements in instruction prompt synthesis (section H), where every request specifies its page type, application domain, specific UI components, and intent.

4.2 Evaluation: RealUIReq-300

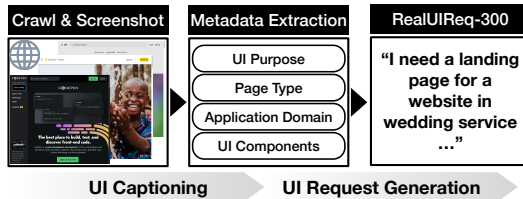


Figure 4: **RealUIReq-300 is curated from real web UIs with diverse use-cases.** UI requests are reverse-engineered from screenshots and extracted metadata, then refined into realistic instructions aligned with the original UIs.

generated. RealUIReq-300 reduces the evaluation ambiguity through detailed, objective, real-page-grounded requests that specify expected components, enabling accessibility, semantic fidelity, and appearance to be compared under intended UI request scope. As illustrated in Figure 4, each example is meticulously synthesized through a multi-stage, human-in-the-loop pipeline (section D) including screenshot collection, metadata extraction, request drafting, and human filtering/refinement. GPT-4.1 (OpenAI et al., 2024) is

We evaluate web accessibility within a natural-language-to-web-UI code generation task, which requires request-style prompts with concrete interface details. A natural reference is Screen2Words (Wang et al., 2021). However, Screen2Words was introduced for UI summarization. It captures UI semantics in a narrow descriptive form: short summaries of the screen, rather than actionable requests specifying constraints.

Accessibility metrics can be applied to such open-ended prompts, but the audit scores can be confounded by variation in accessibility-relevant DOM components

used as an assistive role for metadata extraction and drafting candidate requests, while human annotators review and refine the metadata and requests (Appendix D). Accordingly, RealUIReq-300 serves as our primary benchmark, while Screen2Words remains a useful contextual reference.

Dataset	Query Style	UI Intent Coverage	Component Details	UI Source	Query Length
Screen2Words (Wang et al., 2021)	Single phrase, Summary-style	✗	✗	RICO dataset (Android UI)	6 words
RealUIReq-300 (Ours)	Multi-sentence, Request-oriented	✓	✓	Real-world Web UI	87 words

Table 1: **Contextual comparison of RealUIReq-300 with Screen2Words.** RealUIReq-300 provides multi-sentence, request-oriented instructions with structured intent and explicit component requirements grounded in real-world web UIs.

5 Experimental Setup

We evaluate A11yn and baselines by measuring the accessibility violations in their generated web UIs. In addition, we assess whether A11yn can balance accessibility with semantic alignment and visual appeal in section 6.2.2. Each model is tested on the RealUIReq-300 benchmark (section 4.2), an evaluation set of 300 web UI request prompts designed to ensure consistent and controlled comparisons. We use greedy decoding for all models to balance reproducibility with limited exploration that mitigates repetitive failure modes.

5.1 Evaluation Metrics

We adopt a comprehensive set of evaluation metrics informed by WCAG principles and accessibility-auditor outputs (Deque Systems, 2015; Gay & Li, 2010). Our evaluation comprises four main metrics designed for robustness and fairness. First, we report the total number of affected-node instances across the evaluation set, categorized by Axe-core severity as *Minor*, *Moderate*, *Serious*, and *Critical*. Each instance denotes a violated DOM node reported under a specific rule, and each severity level reflects the impact of the violation, ranging from minor to critical barriers that largely hinder accessibility. Second, we use **Weighted Violation Score (WVS)**, which summarizes accessibility violations by applying severity-specific weights to affected DOM nodes. The WVS is formally defined as:

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

where N_{Minor} , N_{Moderate} , N_{Serious} , and N_{Critical} represent the number of DOM elements with violations at each severity level for generated code on RealUIReq-300 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 issues. Third, we report the mean **Lighthouse Accessibility Score** (Google Chrome Developers, 2025), a practitioner-facing measure that aggregates automated accessibility audits from Axe-core using impact-based weights; higher values indicate better accessibility. Finally, following Sullivan & Matson (2000) and subsequent accessibility literature (Alshayban et al., 2020; Suh et al., 2025), we use **Inaccessibility Rate (IR)** as a conventional web accessibility measure. We adopt this normalized metric to fairly compare models that generate web content with varying lengths. Specifically, we measure IR_{Axe} by calculating the ratio of WVS to the total number of DOM elements produced. In addition, as auditors may differ in WCAG rule coverage and detection heuristics, we include $\text{IR}_{\text{AChecker}}$ from Suh et al. (2025) for robustness, which measures inaccessibility rate based on AChecker detection heuristics.

Model	Violated DOM Counts (↓)				WVS (↓)	Lighthouse (↑)	IR (↓)	
	Minor	Moderate	Serious	Critical			IR _{Axe}	IR _{AChecker}
Frontier Proprietary Models								
GPT-5.1	0	163	1118	61	3924	92.69	0.13	0.35
Claude Sonnet 4.6	7	3044	4885	144	21326	87.09	0.45	0.57
Gemini 3 Pro	1	1089	908	57	5131	90.51	0.24	0.51
Base Model & Adapted Model Variants								
Qwen2.5-Coder-14B-Inst	3	842	1652	35	6783	73.02	0.48	0.68
Qwen2.5-Coder-7B-Inst (Base)	7	951	794	47	4479	90.88	0.40	0.62
+ FeedA11y	26	1195	970	33	5458	93.65	0.32	0.54
+ DPO (Axe-pref.)	6	1042	793	41	4633	90.18	0.40	0.64
+ Best-of-N Sampling (N=8)	0	352	266	12	1550	95.43	0.14	0.40
A11yn (Ours)	0	74	138	25	662	97.68	0.05	0.26

Table 2: **Web accessibility measures across models.** We report Violated DOM Counts at different severity levels, Weighted Violation Score (WVS), mean Lighthouse Accessibility Score, and Inaccessibility Rate (IR). Higher is better for Lighthouse; lower is better for other metrics. **Bold** and underlined values indicate the best and second-best values in each column, respectively.

$$\text{IR}_{\text{Axe}} = \frac{\text{WVS}}{\text{No. of Total DOM Elements}} \quad \text{IR}_{\text{AChecker}} = \frac{\text{No. of AChecker Violations}}{\text{No. of AChecker Flagged Elements}} \quad (5)$$

These metrics capture the density of accessibility violations, enabling fair assessment and robust ranking of model candidates in web accessibility compliance.

5.2 Baselines

We compare A11yn with eight baselines: (1) Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) is the backbone from which A11yn is trained, reflecting the model’s raw web UI generation capability without any accessibility optimization. (2) FeedA11y (Suh et al., 2025) is a direct prior effort aimed at improving web accessibility in LLM-generated UIs; here, we incorporate its prompts via a three-step iterative ReAct prompting (Yao et al., 2023) loop with violation feedback from Axe-core. Implementation details are in Appendix F. (3) Qwen2.5-Coder-14B-Inst is included to assess the effect of model scaling with further analysis in Appendix J. (4) The DPO (Rafailov et al., 2023) baseline is trained on auditor-labeled preference pairs: for each prompt, two completions are audited, and the one with lower IR is assigned as the preferred sample. (5) The Best-of-N baseline is an inference-time method, where we sample N candidates for each evaluation prompt and select the one with the lowest IR for final evaluation. We report $N = 8$ for fair comparison. We also report frontier models: (6) GPT-5.1 (Singh et al., 2025), (7) Claude Sonnet 4.6 (Anthropic, 2025), and (8) Gemini 3 Pro (Team et al., 2023).

6 Results

6.1 Quantitative Results

Table 2 summarizes overall web accessibility performance of model candidates. **A11yn** achieves the strongest performance, with the lowest WVS (662), IR_{Axe} (0.05), and IR_{AChecker} (0.26), as well as the highest mean Lighthouse score (97.68). Relative to the base model, A11yn reduces moderate/serious/critical violations by 92.2%, 82.6%, and 46.8%, respectively. Aggregate metrics show similarly strong gains: WVS drops 85.2%, IR_{Axe} drops 87.5%, and IR_{AChecker} drops 58.1%. The underperformance of the 14B variant under default prompting (IR_{Axe}: 0.48 vs. 0.40; Lighthouse: 73.02 vs. 90.88) shows that scale alone does not guarantee better accessibility. However, larger models benefit more consistently when accessibility requirements are explicitly elicited through prompting (Appendix J). As our primary direct baseline, FeedA11y improves IR and critical counts relative to the base model,

but gains are uneven with additional moderate and minor violations. FeedA11y also incurs notable inference overhead due to iterative prompting, averaging 4584 intermediate tokens per request. The DPO baseline remains close to the base model, suggesting that preference-pair optimization with the same auditor signal is less effective. Although Best-of-N yields fewer Critical affected nodes, A11yn remains stronger on aggregate metrics, showing broad accessibility gains. This residual gap is consistent with the sparsity of Critical violations in the initial training distribution (Appendix C.1). We further extend Best-of-N up to N=20 (Appendix K), and find that larger N yields diminishing returns without closing the gap to A11yn under either auditor, suggesting that the enhancement is more than inference-time search and consistent with RL-induced policy improvement. Among frontier references, GPT-5.1 performs best, followed by Gemini 3 Pro, and Claude Sonnet 4.6.

6.2 Analysis

6.2.1 WCAG Rule-Level Diagnostics

	Base	FeedA11y	A11yn	
Low text contrast	681	781 (+100)	85 (-596)	Reduced Violations
Unlandmarked content	718	367 (-351)	11 (-707)	
Nested main landmark	0	353 (+353)	0 (+0)	
Links without names	92	134 (+42)	37 (-55)	
Ambiguous landmarks	35	192 (+157)	25 (-10)	
Missing or multiple main	132	42 (-90)	1 (-131)	
Multiple main landmarks	0	152 (+152)	0 (+0)	
Missing main heading	64	18 (-46)	37 (-27)	
Buttons without labels	29	15 (-14)	9 (-20)	
Frames without titles	17	19 (+2)	12 (-5)	
Selects without labels	15	12 (-3)	10 (-5)	
Nested footer landmark	0	16 (+16)	0 (+0)	
Nested header landmark	0	15 (+15)	0 (+0)	
Broken heading order	1	13 (+12)	0 (-1)	
Multiple header landmarks	0	14 (+14)	0 (+0)	
Missing page title	2	11 (+9)	0 (-2)	
Missing page language	1	11 (+10)	0 (-1)	
Repeated alt text	7	4 (-3)	0 (-7)	
Nested sidebar landmark	0	11 (+11)	0 (+0)	
Missing input labels	2	0 (-2)	5 (+3)	Increased Violations

Figure 5: **WCAG rule-level diagnostics across models.** Cells show per-rule violations of Base/FeedA11y/A11yn; parentheses show change vs. Base; colors denote violation increase or reduction (Appendix L).

While Table 2 shows that FeedA11y and A11yn improve web accessibility relative to the base model, Figure 5 provides a diagnostic view of WCAG rule analysis. Specifically, we compare FeedA11y and A11yn against the base model across key WCAG violation rules. Each cell shows the absolute violation count for a rule, with the parenthesized value and color indicating its change from the base model.

Although both methods reduce existing violation types of the base model, FeedA11y shows the inadvertent introduction of additional violations such as color-contrast and landmark-structure violations. Consistent with the finding of Suh et al. (2025), post-hoc prompting remains prone to unintended regressions.

In contrast, A11yn directly addresses the failure mode flagged by FeedA11y by delivering consistent reductions across a broad range of rule categories, including major reductions in color contrast, region, and landmark-one-main. These results indicate that internalizing accessibility through training mitigates the instability of post-hoc prompting while retaining the underlying UI generation capability of the base model.

6.2.2 Preserving Semantics and Aesthetics

Web UI generation is inherently multi-dimensional, requiring harmony among accessibility, semantic fidelity, and visual quality. Although accessibility and aesthetics are often framed as conflicting (Anthony, 2019), prior work suggests they should be jointly optimized (Kurosu & Kashimura, 1995; Mbipom & Harper, 2011; Le-Cong et al., 2021). Accordingly, we evaluate whether A11yn’s accessibility gains preserve semantic fidelity and visual appeal.

We adopt Appearance Score from WebGen-Bench (Lu et al., 2025b) with its rubric-based evaluation prompt (Appendix H.2). GPT-5.1 assigns a 5-point Likert rating by jointly assessing successful rendering, content relevance, layout harmony, and modernness/beauty.

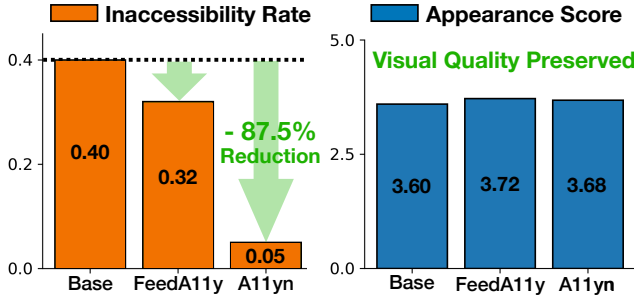


Figure 6: **Comparison of models in accessibility and visual quality.** We report Inaccessibility Rate (↓) and Appearance Score (↑, 5-point Likert scale).

Given the high correlation with human judgments reported in Lu et al. (2025b), we treat it as a practical proxy for joint semantic-fidelity and aesthetics alignment. As shown in Figure 6, A11yn maintains appearance quality comparable to FeedA11y (3.68 vs. 3.72) while reducing inaccessibility much more strongly (0.05 vs. 0.32; 87.5% lower than the base model at 0.40). Notably, A11yn achieves this in a single forward pass, whereas FeedA11y depends on iterative prompting.

We additionally conduct a pairwise human evaluation with five evaluators. A11yn is preferred in 39.3% of judgments, 39.3% rate the outputs as similar, and 21.4% prefer the base model, supporting that A11yn preserves visual and semantic quality (Appendix E).

6.2.3 Ablation Study

Weighted vs Uniform Penalty

To evaluate the impact of severity weighting, we compare two trained A11yn variants: a *weighted* reward that penalizes WCAG violations by severity and a *uniform* reward that assigns equal penalties. The weighted variant uses coefficients (0.1, 0.2, 0.3, 0.4) for Minor, Moderate, Serious, and Critical violations, while the uniform variant uses (0.25, 0.25, 0.25, 0.25). As shown in Table 3, severity-weighted rewards reduce accessibility violations more effectively than uniform weighting.

Scheme	Violated DOM Counts (↓)				Inaccessibility Rate (↓)	
	Minor	Moderate	Serious	Critical	IR _{Axe}	IR _{Checker}
Base	7	951	794	47	0.40	0.62
Uniform	3	126	201	22	0.07	0.31
Weighted	0	74	138	25	0.05	0.26

Table 3: **Ablation of reward weighting strategies.** *Weighted* penalties more effectively reduce WCAG violations than *uniform* weighting.

Effect of DOM-Size Normalization Figure 7 compares two A11yn variants trained with different penalty formulations: a raw penalty (P_{raw}), and a DOM-normalized penalty, (P_{norm}). Both variants substantially improve accessibility over the base model, reducing IR_{Axe} from 0.40 to 0.05. However, A11yn trained with P_{raw} exhibits clear reward-hacking behavior, collapsing to much shorter outputs (average DOM/page: 15.20 vs. 37.02 for the base model), which coincides with a sharp drop in appearance score (3.60 to 2.41). In contrast, P_{norm} preserves interface complexity while maintaining the same accessibility gain, producing substantially larger pages and a higher appearance score (3.68). These results show that DOM-size normalization prevents output size collapse and enables accessibility improvements without sacrificing UI richness or visual quality.

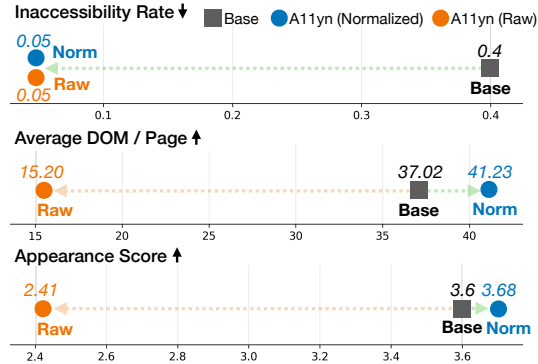


Figure 7: **Comparison of A11yn variants under Raw vs. Normalized Penalties.** Although A11yn trained under P_{norm} and P_{raw} reach similarly low inaccessibility, A11yn with a normalized penalty (P_{norm}) maintains UI richness and appearance.

7 Conclusion

Web accessibility is essential yet still under-optimized in LLM-based web UI generation. We present **A11yn**, an auditor-guided RL framework that aligns code-generating LLMs with a penalty-aware, DOM-normalized WCAG reward derived from off-the-shelf accessibility audits. Enabling accessibility signals into the training objective, A11yn prioritizes web-accessible behavior at generation time.

Across our experiments, A11yn improves web accessibility compliance while preserving semantic fidelity and visual quality, with enhancements validated by Axe-core, AChecker, and Lighthouse. Relative to prompting approaches such as FeedA11y, A11yn internalizes web accessibility requirements into the policy and strengthens single-pass generation. More broadly, our findings provide evidence that verifier-checkable standards can be operationalized as practical alignment objectives for software-generation models, particularly in accessibility-aware web UI generation.

Ethics Statement

This work aims to improve digital equity by aligning code-generating LLMs to produce accessibility-aware web UIs, thereby complying with WCAG and helping reduce 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; sensitive or personally identifiable content was removed. Our human evaluation involved five recruited evaluators who reviewed only generated UI screenshots and request metadata; no sensitive, personally identifiable, or private user data were collected. While misuse could enable mass generation of low-quality web pages, we mitigate this risk by committing to the 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 setup provided in Section C. Accessibility compliance was measured using the open-source Axe-core engine distributed under the Mozilla Public License 2.0 and AChecker (Deque Systems, 2015; Gay & Li, 2010), distributed under the GNU General Public License version 2.0 (GPLv2) to detect WCAG violations and compute reward signals. We release the code and data at <https://github.com/jeffrobot/A11yn>.

Use of LLMs

For transparency, we disclose the following uses of LLMs in this work: (1) **Data generation and curation support**: LLM prompting was used to synthesize UIReq-6.8K requests and draft metadata/query candidates for RealUIReq-300, followed by author review and manual refinement; (2) **LLM-based evaluation**: consistent with the protocol of Lu et al. (2025b), MLLM was used to assign WebGen-Bench Appearance Scores.

Acknowledgments

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University), No. RS-2026-25522885, Development of a World Foundation Model for Training and Development of Physical AI Systems, No. RS-2026-25512061, Development of Large

Action Model-Based Autonomous Digital Twin Operation Technology for Proactive Problem Solving), the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00354218, No. RS-2026-25561904), and the Technology Innovation Program (RS-2025-25456760, Development of a humanoid robot specialized in chemical processes based on AI foundation model) funded by the Ministry of Trade, Industry and Resources (MOTIR, Korea). We express special thanks to KAIT GPU project. The ICT at Seoul National University provides research facilities for this study.

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.
- Abdullah Alsaedi. Comparing web accessibility evaluation tools and evaluating the accessibility of webpages: proposed frameworks. *Information*, 11(1):40, 2020.
- Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. Accessibility issues in android apps: state of affairs, sentiments, and ways forward. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, ICSE ’20*, pp. 1323–1334, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371216. doi: 10.1145/3377811.3380392. URL <https://doi.org/10.1145/3377811.3380392>.
- 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://www-cdn.anthropic.com/07b2a3f9902ee19fe39a36ca638e5ae987bc64dd.pdf>. Accessed: 2025-07-28.
- AR Arya Asok and RV Rekha. Digital inclusion in higher education: A web content accessibility evaluation of best asian university library websites. *The Journal of Academic Librarianship*, 51(5):103120, 2025.
- 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.
- Deque Systems. axe-core Accessibility Engine. <https://github.com/dequelabs/axe-core>, 2015. Accessed: 2025-07-28.
- Thomas Fischer, Björn Lundell, and Jonas Gamalielsson. Coverage of web accessibility guidelines provided by automated checking tools. *Universal Access in the Information Society*, 24(4):3615–3637, 2025.
- Greg Gay and Cindy Qi Li. Achecker: open, interactive, customizable, web accessibility checking. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, W4A ’10, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300452. doi: 10.1145/1805986.1806019. URL <https://doi.org/10.1145/1805986.1806019>.
- Google Chrome Developers. Lighthouse accessibility score, 2025. URL <https://developer.chrome.com/docs/lighthouse/accessibility/scoring>. Accessed July 27, 2026.
- Alexandra-Elena Guriță 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.

- Soongoo Hong, Pairin Katerattanakul, and Dae-hyung Lee. Evaluating government website accessibility: Software tool vs human experts. *Management Research News*, 31(1):27–40, 2007.
- 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, Jiayi 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>.
- Abid Ismail and KS Kuppasamy. Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites. *Journal of King Saud University-Computer and Information Sciences*, 34(3):901–911, 2022.
- 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.
- Zimu Lu, Houxing Ren, Yunqiao Yang, Ke Wang, Zhuofan Zong, Juntong Pan, Mingjie Zhan, and Hongsheng Li. Webgen-agent: Enhancing interactive website generation with multi-level feedback and step-level reinforcement learning. *arXiv preprint arXiv:2509.22644*, 2025a.
- 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, 2025b. URL <https://arxiv.org/abs/2505.03733>.
- Kelly Mack, Emma McDonnell, Dhruv Jain, Lucy Lu Wang, Jon E. Froehlich, and Leah Findlater. What do we mean by “accessibility research”? a literature survey of accessibility papers in chi and assets from 1994 to 2019. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–18, 2021.
- 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.
- 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.
- 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, Florenzia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report, 2024. URL <https://arxiv.org/abs/2303.08774>.
- Jonathan Robert Pool. Accessibility metatesting: comparing nine testing tools. In *Proceedings of the 20th International Web for All Conference*, pp. 1–4, 2023.
- 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>.
- Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.
- 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>.
- Terry Sullivan and Rebecca Matson. Barriers to use: usability and content accessibility on the web’s most popular sites. In *Proceedings on the 2000 Conference on Universal Usability*, CUU ’00, pp. 139–144, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581133146. doi: 10.1145/355460.355549. URL <https://doi.org/10.1145/355460.355549>.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- 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>.
- World Wide Web Consortium. Web content accessibility guidelines (wcag) 2.2, December 2024. URL <https://www.w3.org/TR/2024/REC-WCAG22-20241212/>.
- 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:

Training Set Application Domains

- 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

61 Domain Distribution of RealUIReq-300 Dataset

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	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 discovery platforms 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 Collaboration 3	Content publishing 3
Medical research 3	Personal blog 3
Financial Services 3	Consumer electronics 2
Crowdfunding 2	Adventure / Outdoor sports 2
Music streaming 2	Education / Interactive media 2
Wedding planning 2	Events / Entertainment 2
Creative networking 2	Film / Photography 2
Parenting / Community 2	Job board / Recruitment 2
Advocacy / Activism 2	Healthcare tech 2
Cloud computing / AI 2	Retail / Fashion 1
Search engine 1	Dev social 1
Culture / History 1	Women empowerment 1
Freelance marketplace 1	Science museum 1
Service industry 1	Gaming / Media 1
Microfinance 1	Cloud storage 1
Space exploration 1	Humor / Lifestyle 1
Religious / Charitable Org 1	

C Training Configuration

We trained *Qwen/Qwen2.5-Coder-7B-Instruct* on 4 NVIDIA RTX PRO 6000 Blackwell GPUs (96GB 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 $G = 8$ sampled completions, with a per-device batch size of 2 and gradient accumulation set to 12. To stabilize optimization, KL divergence regularization was applied with $\beta = 0.001$. Below is a summary of the key configurations used.

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	12
Batch size	96
Epochs	1
Gradient checkpointing	Enabled
Precision	bfloat16
Loss type	GRPO
KL β	0.001
PEFT	Enabled
PEFT Defaults	LoRA r=16, alpha=32, dropout=0.05

Table 4: 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 completion length	3072
Max prompt length	1024
Number of generations	8
Seed	3407

Table 5: GRPO sampling parameters.

C.1 Training-Time Severity Distribution

We audit base-model generations on all UIReq-6.8K training prompts with Axe-core.

Severity	Affected Nodes
Minor	121
Moderate	17,581
Serious	12,138
Critical	672

Table 6: **Initial training-time severity distribution.** Axe-core affected-node counts from base-model generations on UIReq-6.8K before A11yn optimization.

Critical rules are sparse in the base model’s training-time violation profile. This helps explain why A11yn reduces Critical affected nodes from 47 to 25 but trails Best-of-N at 12, especially for button-name and select-name, which appear only 135 and 155 times in the training audit. Thus, A11yn’s strength is broad aggregate violation reduction, where targeted prompt augmentation on training set may further improve some underrepresented rules.

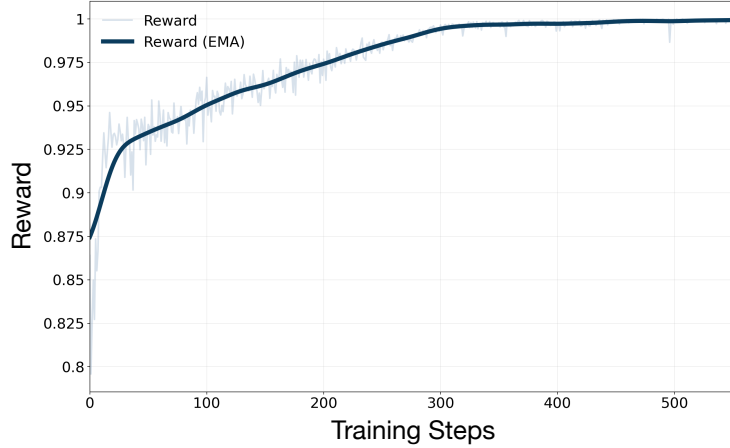


Figure 8: **Web Accessibility Reward curve throughout training.** The curve shows a steady increase indicating reduced WCAG violation occurrences over time.

D Data Annotation of RealUIReq-300

The construction of RealUIReq-300 involved human-in-the-loop interventions to ensure semantic accuracy and realism of evaluation requests. Human contributors consisted of the paper authors. Human involvement occurred at three stages:

- **UI Captioning Validation.** LLM-generated metadata extracted from webpage screenshots (e.g., UI purpose, page type, domain, and key components) was manually reviewed to verify that notable and functionally important UI elements were correctly captured. Among the set, 14% (43/300) had low-resolution or unclear screenshots, for which the authors manually visited the corresponding webpages and rewrote the prospective UI request with explicit specifications (UI purpose, page type, domain, key components).
- **UI Request Draft Editing.** Initial request drafts were edited to improve clarity, specificity, and naturalness, resolving vague phrasing and ensuring alignment with the underlying UI content. For this process, tone biases (e.g., “Make a page ...”) were spotted, and the annotators manually diversified the tone for all 300 requests while preserving the original specifications, so that the final benchmark reflects human-reviewed tone rather than a model’s preferences.
- **Consistency and Coherence Checking.** Final requests were reviewed to ensure consistency and semantic coherence between the request texts and the corresponding real-world UIs, including alignment with the intended UI purpose, structure, and static web UI scope.

E Human Evaluation of Visual and Semantic Quality

We conduct a pairwise human evaluation to assess whether A11yn’s accessibility improvements preserve the visual and semantic quality of generated UIs relative to the base model. Five recruited evaluators provide a total of 1,330 judgments. For each judgment, an evaluator is shown the user-request metadata and two rendered UIs, one generated by the base model and one generated by A11yn. The left–right position of the two UIs is randomized. Evaluators select one of five responses: strongly prefer the left UI, weakly prefer the left UI, similar, weakly prefer the right UI, or strongly prefer the right UI. We aggregate the responses after resolving the randomized positions to the corresponding model identities.

As shown in Table 7, the most common response is that the two outputs are similar (39.3%). A11yn is preferred in another 39.3% of all judgments, compared with 21.4% for the base

Human Judgment	Count	Rate
Strong preference for A11yn	182	13.7%
Weak preference for A11yn	340	25.6%
Similar	523	39.3%
Weak preference for Base	196	14.7%
Strong preference for Base	89	6.7%

Table 7: **Pairwise human evaluation of A11yn and the base model.** Counts and rates are aggregated over 1,330 judgments after resolving randomized left-right positions.

model. Among non-tie judgments, 64.7% favor A11yn. Overall, the results indicate that accessibility alignment does not reduce perceived visual or semantic quality and may improve it for some requests.

F FeedA11y Baseline Implementation Details

We implement the FeedA11y baseline by following its three-stage loop: *Generation*, *Evaluation*, and *ReAct Refinement*. Each stage is run by an independent agent, all backed by Qwen2.5-Coder-7B-Instruct to keep the backbone consistent with our primary controlled comparisons. During refinement, we retain the original FeedA11y editing constraints (e.g., “fix only where necessary” and “preserve structure/functionality”) to avoid unnecessary rewrites.

Stage-wise procedure. Given a user request, the baseline first generates an initial HTML/CSS candidate (Generation). The candidate is then audited to produce accessibility findings (Evaluation). The ReAct agent consumes the code and the violation report, proposes targeted edits, and outputs a refined version (ReAct Refinement). This evaluation-refinement cycle is repeated with strict stopping criteria.

Stopping criteria. To prevent over-editing and infinite refinement loops, we use three deterministic stopping rules:

1. terminate immediately when no accessibility violations are detected;
2. stop early when refined code is identical to the previous iteration (code convergence);
3. cap refinement at a maximum of three rounds.

Differences from the original FeedA11y setup. Our implementation differs from the original FeedA11y in two aspects:

1. **Prompt instantiation.** The original formulation is summary-to-code reconstruction, whereas our benchmark is request-to-code generation. We therefore adapt prompt wording to the request-style task while preserving the same three-stage control flow.
2. **Auditing signal in Evaluation.** The original FeedA11y implementation generates accessibility reports via an LLM. In our pipeline, Evaluation uses auditor-grounded Axe-core output (e.g., rule IDs and descriptions) as the refinement signal for fairness.

This design keeps the overall FeedA11y workflow unchanged while aligning the auditing interface with our auditor-based evaluation protocol. It reduces confounds from model-dependent violation reporting and supports fairer comparison under a shared WCAG auditing signal.

G Limitations and Future Work

Reliance on Accessibility Audits. Following prior accessibility-technology studies, we operationalize web accessibility using automated auditors (Axe-core and AChecker) rather than direct human-centered assessment. Fully capturing the diverse, context-dependent accessibility needs of users through human evaluation remains an open challenge in HCI and is left for future work.

Scope of HTML-based Web UIs. The current study focuses on HTML-based web UI generation, which covers a broad and practical class of real-world interfaces. Nevertheless, extending accessibility-aligned optimization to additional frontend frameworks and UI technologies remains an important direction as accessibility requirements may vary across frontend platforms.

Targeting of Static UI. Our analysis considers static UI renderings derived from screenshots and does not explicitly model interactive behaviors. As some accessibility challenges arise from runtime interactions, future work may explore dynamic and interactive web interfaces, including accessibility considerations under UI state changes and user interactions.

H Prompt Details

We provide the details of the prompts used.

H.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.

H.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 `` section with a `` tag
  - A `` tag
  - A proper tailwind css link tag to load Tailwind CSS from CDN
  - A `` 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., 2025b)

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.

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:

I Additional Evaluation Statistics

We report complementary diagnostics beyond aggregate metrics in Table 2. Specifically, we provide UI domain-level accessibility outcomes (Table 8), page-level pass counts per auditor (Table 9), and per-request pairwise outcomes (Table 10).

I.1 Normalized Domain Coverage

Table 8 follows the exact 61 normalized domain bins and counts from the RealUIReq-300 domain-distribution chart. For each method and domain, we report: (1) **Count**, the number

of pages in that domain that fail at least one auditor; (2) $IR_{AChecker}$; and (3) IR_{Axe} . Lower is better for all metrics. Boldface marks the lowest value within each row (Count, $IR_{AChecker}$, IR_{Axe}); ties are all bolded.

| Normalized Domain Family | N | A11yn | | | Base | | | FeedA11y | | | Best-of-8 | | |
|-----------------------------------|----|-------|-----------------|------------|-------|-----------------|------------|----------|-----------------|------------|-----------|-----------------|------------|
| | | Count | $IR_{AChecker}$ | IR_{Axe} | Count | $IR_{AChecker}$ | IR_{Axe} | Count | $IR_{AChecker}$ | IR_{Axe} | Count | $IR_{AChecker}$ | IR_{Axe} |
| Education | 30 | 17 | 0.27 | 0.06 | 30 | 0.64 | 0.39 | 29 | 0.55 | 0.38 | 21 | 0.41 | 0.15 |
| E-commerce | 19 | 10 | 0.31 | 0.08 | 18 | 0.56 | 0.39 | 19 | 0.68 | 0.45 | 16 | 0.21 | 0.11 |
| Non-profit / Humanitarian | 13 | 5 | 0.21 | 0.07 | 13 | 0.70 | 0.65 | 13 | 0.45 | 0.28 | 10 | 0.45 | 0.16 |
| Social media | 12 | 7 | 0.24 | 0.04 | 12 | 0.67 | 0.43 | 12 | 0.60 | 0.37 | 12 | 0.41 | 0.16 |
| Web development | 10 | 3 | 0.46 | 0.05 | 10 | 0.63 | 0.31 | 9 | 0.56 | 0.22 | 9 | 0.45 | 0.13 |
| Entertainment / Media | 9 | 3 | 0.14 | 0.02 | 9 | 0.58 | 0.26 | 9 | 0.47 | 0.31 | 5 | 0.36 | 0.09 |
| Business consulting | 6 | 2 | 0.08 | 0.03 | 6 | 0.56 | 0.34 | 6 | 0.41 | 0.18 | 4 | 0.41 | 0.13 |
| Academic publishing | 6 | 4 | 0.30 | 0.06 | 5 | 0.59 | 0.37 | 5 | 0.49 | 0.15 | 5 | 0.41 | 0.13 |
| Personal development | 5 | 1 | 0.23 | 0.00 | 4 | 0.45 | 0.59 | 4 | 0.32 | 0.19 | 2 | 0.20 | 0.04 |
| Finance / Investing | 4 | 4 | 0.62 | 0.12 | 4 | 0.52 | 0.25 | 4 | 0.58 | 0.29 | 3 | 0.15 | 0.07 |
| Government services | 4 | 2 | 0.33 | 0.08 | 4 | 0.79 | 0.65 | 3 | 0.53 | 0.18 | 2 | 0.13 | 0.04 |
| Environmental activism | 4 | 3 | 0.40 | 0.11 | 3 | 0.67 | 0.34 | 3 | 0.53 | 0.68 | 3 | 0.48 | 0.22 |
| Community forum | 4 | 3 | 0.33 | 0.05 | 4 | 0.48 | 0.28 | 4 | 0.65 | 0.38 | 1 | 0.10 | 0.03 |
| B2B marketing / Business services | 4 | 1 | 0.21 | 0.00 | 4 | 0.68 | 0.34 | 3 | 0.50 | 0.21 | 3 | 0.48 | 0.14 |
| Business Collaboration | 3 | 1 | 0.43 | 0.00 | 3 | 0.68 | 0.87 | 3 | 0.47 | 0.32 | 2 | 0.50 | 0.24 |
| Medical research | 3 | 0 | 0.00 | 0.00 | 3 | 0.77 | 0.36 | 3 | 0.45 | 0.36 | 3 | 0.20 | 0.10 |
| Financial Services | 3 | 1 | 0.06 | 0.03 | 3 | 0.65 | 0.35 | 3 | 0.60 | 0.28 | 1 | 0.36 | 0.07 |
| Crowdfunding | 2 | 1 | 0.06 | 0.03 | 2 | 0.63 | 0.34 | 2 | 0.54 | 0.31 | 2 | 0.72 | 0.23 |
| Music streaming | 2 | 2 | 0.22 | 0.12 | 2 | 0.35 | 0.20 | 2 | 0.61 | 0.29 | 2 | 0.21 | 0.12 |
| Wedding planning | 2 | 2 | 0.46 | 0.10 | 2 | 0.57 | 0.43 | 2 | 0.55 | 0.32 | 2 | 0.59 | 0.21 |
| Creative networking | 2 | 1 | 0.25 | 0.03 | 2 | 0.44 | 0.14 | 2 | 0.67 | 0.44 | 1 | 0.17 | 0.02 |
| Parenting / Community | 2 | 1 | 0.40 | 0.07 | 2 | 0.22 | 0.17 | 2 | 0.33 | 0.10 | 1 | 0.20 | 0.06 |
| Advocacy / Activism | 2 | 2 | 0.50 | 0.32 | 2 | 0.82 | 0.50 | 2 | 0.77 | 0.55 | 2 | 0.40 | 0.33 |
| Cloud computing / AI | 2 | 1 | 0.33 | 0.00 | 2 | 0.42 | 0.39 | 2 | 0.18 | 0.10 | 2 | 0.50 | 0.20 |
| Search engine | 1 | 0 | 0.00 | 0.00 | 1 | 0.40 | 0.15 | 1 | 0.50 | 0.25 | 1 | 0.33 | 0.11 |
| Culture / History | 1 | 1 | 0.75 | 0.20 | 1 | 0.88 | 0.24 | 1 | 0.50 | 0.27 | 1 | 0.50 | 0.24 |
| Freelance marketplace | 1 | 0 | 0.00 | 0.00 | 1 | 0.73 | 0.26 | 1 | 0.43 | 0.33 | 1 | 0.25 | 0.07 |
| Service industry | 1 | 0 | 0.00 | 0.00 | 1 | 0.64 | 0.60 | 1 | 0.61 | 0.77 | 0 | 0.00 | 0.00 |
| Microfinance | 1 | 0 | 0.00 | 0.00 | 1 | 0.50 | 0.75 | 1 | 0.64 | 0.68 | 0 | 0.00 | 0.00 |
| Space exploration | 1 | 0 | 0.00 | 0.00 | 1 | 0.33 | 0.17 | 1 | 0.88 | 1.33 | 0 | 0.00 | 0.00 |
| Religious / Charitable Org | 1 | 0 | 0.00 | 0.00 | 1 | 0.46 | 0.71 | 1 | 0.33 | 0.12 | 1 | 0.50 | 0.33 |
| Technology / Software | 23 | 11 | 0.27 | 0.03 | 22 | 0.62 | 0.31 | 23 | 0.52 | 0.31 | 20 | 0.49 | 0.10 |
| News / Media | 16 | 13 | 0.50 | 0.10 | 16 | 0.66 | 0.41 | 16 | 0.70 | 0.42 | 15 | 0.41 | 0.17 |
| Software development | 12 | 8 | 0.36 | 0.05 | 12 | 0.66 | 0.37 | 12 | 0.61 | 0.33 | 10 | 0.47 | 0.13 |
| Tourism | 10 | 6 | 0.23 | 0.05 | 10 | 0.68 | 0.58 | 10 | 0.52 | 0.37 | 10 | 0.56 | 0.23 |
| SaaS | 9 | 3 | 0.11 | 0.02 | 9 | 0.61 | 0.49 | 8 | 0.33 | 0.21 | 7 | 0.43 | 0.15 |
| Entertainment ticketing | 8 | 5 | 0.29 | 0.07 | 8 | 0.61 | 0.40 | 8 | 0.56 | 0.44 | 6 | 0.33 | 0.12 |
| Professional services | 6 | 2 | 0.24 | 0.09 | 6 | 0.56 | 0.34 | 5 | 0.35 | 0.15 | 5 | 0.54 | 0.36 |
| Food / Cooking | 5 | 1 | 0.10 | 0.01 | 4 | 0.65 | 0.43 | 4 | 0.23 | 0.09 | 4 | 0.14 | 0.02 |
| Travel / Lifestyle | 5 | 2 | 0.26 | 0.04 | 4 | 0.55 | 0.42 | 5 | 0.48 | 0.35 | 5 | 0.47 | 0.12 |
| Tech discovery platforms | 4 | 2 | 0.27 | 0.13 | 4 | 0.72 | 0.50 | 4 | 0.62 | 0.30 | 3 | 0.53 | 0.15 |
| Academic research | 4 | 3 | 0.41 | 0.06 | 4 | 0.68 | 0.38 | 4 | 0.61 | 0.30 | 4 | 0.50 | 0.16 |
| Health and wellness | 4 | 2 | 0.31 | 0.02 | 4 | 0.71 | 0.48 | 4 | 0.46 | 0.36 | 3 | 0.50 | 0.26 |
| Art and design | 4 | 1 | 0.03 | 0.00 | 3 | 0.50 | 0.25 | 4 | 0.59 | 0.37 | 3 | 0.22 | 0.05 |
| Creative / Portfolio | 3 | 1 | 0.25 | 0.02 | 3 | 0.64 | 0.66 | 3 | 0.60 | 0.49 | 2 | 0.08 | 0.05 |
| Content publishing | 3 | 1 | 0.00 | 0.02 | 3 | 0.63 | 0.36 | 3 | 0.51 | 0.30 | 3 | 0.27 | 0.13 |
| Personal blog | 3 | 2 | 0.44 | 0.02 | 1 | 0.43 | 0.08 | 3 | 0.62 | 0.29 | 1 | 0.33 | 0.04 |
| Consumer electronics | 2 | 1 | 0.33 | 0.06 | 2 | 0.61 | 0.31 | 1 | 0.60 | 0.18 | 2 | 0.11 | 0.14 |
| Adventure / Outdoor sports | 2 | 0 | 0.00 | 0.00 | 2 | 0.53 | 0.80 | 2 | 0.43 | 0.30 | 2 | 0.48 | 0.29 |
| Education / Interactive media | 2 | 1 | 0.14 | 0.02 | 2 | 0.43 | 0.33 | 2 | 0.50 | 0.18 | 1 | 0.25 | 0.04 |
| Film / Photography | 2 | 2 | 0.63 | 0.11 | 2 | 0.79 | 0.42 | 2 | 0.60 | 0.33 | 1 | 0.17 | 0.04 |
| Job board / Recruitment | 2 | 0 | 0.00 | 0.00 | 1 | 0.30 | 0.18 | 2 | 0.42 | 0.43 | 2 | 0.43 | 0.20 |
| Healthcare tech | 2 | 0 | 0.00 | 0.00 | 2 | 0.72 | 0.38 | 2 | 0.50 | 0.45 | 2 | 0.53 | 0.21 |
| Retail / Fashion | 1 | 0 | 0.00 | 0.00 | 1 | 0.82 | 1.08 | 1 | 0.72 | 0.92 | 1 | 0.53 | 0.36 |
| Dev social | 1 | 1 | 0.50 | 0.05 | 1 | 0.11 | 0.27 | 1 | 0.58 | 0.36 | 1 | 0.40 | 0.04 |
| Women empowerment | 1 | 0 | 0.00 | 0.00 | 0 | 0.00 | 0.00 | 1 | 0.13 | 0.06 | 1 | 0.33 | 0.19 |
| Science museum | 1 | 0 | 0.00 | 0.00 | 1 | 0.33 | 0.47 | 1 | 0.18 | 0.04 | 1 | 0.43 | 0.15 |
| Gaming / Media | 1 | 1 | 0.50 | 0.09 | 1 | 0.10 | 0.12 | 1 | 0.29 | 0.14 | 1 | 0.20 | 0.04 |
| Cloud storage | 1 | 0 | 0.00 | 0.00 | 1 | 0.83 | 1.84 | 1 | 0.29 | 0.21 | 1 | 0.62 | 0.38 |
| Humor / Lifestyle | 1 | 1 | 0.57 | 0.33 | 1 | 0.71 | 0.32 | 1 | 0.50 | 0.32 | 1 | 0.63 | 0.33 |
| Events / Entertainment | 2 | 1 | 0.60 | 0.10 | 2 | 0.69 | 0.39 | 2 | 0.49 | 0.21 | 2 | 0.45 | 0.25 |

Table 8: Domain-level accessibility performance on RealUIReq-300 (61 normalized domains). For each method, columns report (i) Count: pages in that domain that fail at least one auditor, (ii) $IR_{AChecker}$, and (iii) IR_{Axe} . Bold marks the lowest value within each row for each metric block; ties are bolded. Lower is better.

I.2 Page-level Pass Rates and Pairwise Win Rates

To complement the domain-level results in Table 8, we provide two prompt-level summaries on the same RealUIReq-300 benchmark. Table 9 reports how many of the 300 generated pages have zero violations under Axe, AChecker, and both auditors jointly. Table 10 reports per-request head-to-head outcomes (W-L-T) between A11yn and each baseline, where a win indicates a lower page-level inaccessibility rate and ties are resolved by the weighted violation score.

| Method | Pages w/o
Axe Violations | Pages w/o
AChecker Violations | Pages w/o Both
Violations |
|-----------|-----------------------------|----------------------------------|------------------------------|
| Base | 18/300 | 19/300 | 12/300 |
| Best-of-8 | 79/300 | 86/300 | 62/300 |
| FeedA11y | 16/300 | 17/300 | 11/300 |
| A11yn | 177/300 | 166/300 | 151/300 |

Table 9: **Page-level pass counts on RealUIReq-300.** Each value is the number of pages out of 300 with zero violations under the specified auditor; *Pages w/o Both Violations* requires passing both Axe and AChecker simultaneously.

| Comparison | W-L-T | Win Rate Excluding Ties |
|--------------------|-----------|-------------------------|
| A11yn vs Base | 270-14-16 | 95.1% |
| A11yn vs FeedA11y | 273-18-9 | 93.8% |
| A11yn vs Best-of-8 | 190-52-58 | 78.5% |

Table 10: **Per-request pairwise outcomes (A11yn vs. each baseline) on RealUIReq-300.** W-L-T denotes wins/losses/ties across the same 300 prompts. A win means a lower page-level inaccessibility rate for that prompt.

J Model Scaling with and without Accessibility-Aware Prompting

To distinguish model capacity from the elicitation of accessibility objectives, we evaluate the Qwen2.5-Coder family under both the default UI-generation prompt and an accessibility-aware system prompt. Under default prompting, accessibility performance is non-monotonic. With accessibility-aware prompting, larger models generally obtain lower IR_{Axe} and higher Lighthouse scores. These findings suggest that scale provides greater latent capacity to follow accessibility constraints, but such capacity is not reliably activated by default UI-generation instructions. Nevertheless, the prompted 32B model remains below A11yn (IR_{Axe} : 0.05, $IR_{AChecker}$: 0.26, Lighthouse: 97.68), supporting the benefit of our training-time alignment.

| Model | Default Prompt | | | Accessibility-Aware Prompt | | |
|-------------------|-----------------------|----------------------------|-----------------------|----------------------------|----------------------------|-----------------------|
| | $IR_{Axe} \downarrow$ | $IR_{AChecker} \downarrow$ | Lighthouse $\uparrow$ | $IR_{Axe} \downarrow$ | $IR_{AChecker} \downarrow$ | Lighthouse $\uparrow$ |
| Qwen2.5-Coder-3B | 0.56 | 0.67 | 71.91 | 0.43 | 0.62 | 80.35 |
| Qwen2.5-Coder-7B | 0.40 | 0.62 | 90.88 | 0.29 | 0.57 | 90.11 |
| Qwen2.5-Coder-14B | 0.48 | 0.68 | 73.02 | 0.25 | 0.60 | 92.65 |
| Qwen2.5-Coder-32B | 0.42 | 0.68 | 81.07 | 0.18 | 0.49 | 93.65 |

Table 11: **Model scaling under default and accessibility-aware prompting.** Explicitly eliciting accessibility requirements in the system prompt makes the scaling trend more consistent. Bold values mark the best result within each prompt setting.

K Extended Best-of-N Scaling up to N=20

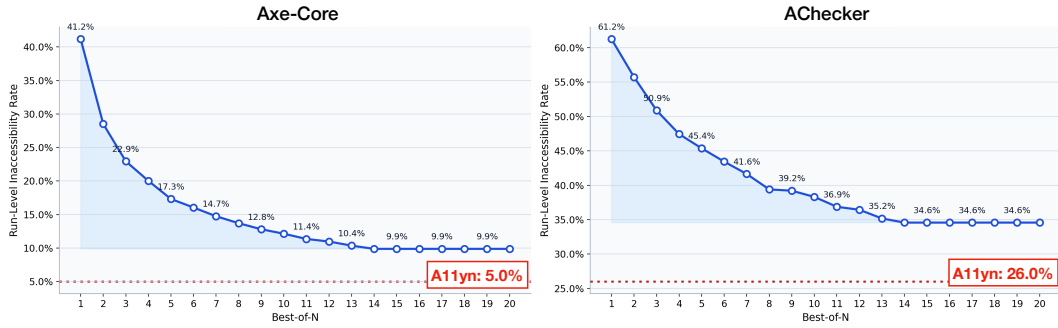


Figure 9: Extended Best-of-N Scaling under Axe-core and AChecker Evaluation. As N increases, Best-of-N initially improves accessibility under both auditors, but the gains diminish and plateau within the tested range. Under Axe-core, the run-level inaccessibility rate decreases from 41.2% at N=1 to 9.9% at N=14 and remains at 9.9% through N=20; under AChecker, it decreases from 61.2% to 34.6% at N=14 and likewise remains at 34.6% through N=20. In both cases, the Inaccessibility Rate remains above A11yn at 5.0% and 26.0%, respectively, suggesting that A11yn’s improvement is not reducible to test-time sampling and selection alone, and is instead consistent with RL-induced policy improvement.

L WCAG Rule ID Mapping

| Plot Label | Rule ID | Appendix Description |
|---|-------------------------------------|--|
| Low text contrast
Unlandmarked content | color-contrast
region | Elements do not meet minimum color contrast.
Page content is not fully contained within landmark regions. |
| Nested main landmark | landmark-main-is-top-level | The main landmark is nested inside another landmark. |
| Links without names | link-name | Links do not have discernible accessible text. |
| Ambiguous landmarks | landmark-unique | Landmarks do not have unique role or label combinations. |
| Missing or multiple main | landmark-one-main | The page does not have exactly one main landmark. |
| Multiple main landmarks | landmark-no-duplicate-main | The page has more than one main landmark. |
| Missing main heading | page-has-heading-one | The page does not contain a level-one heading (H1). |
| Buttons without labels | button-name | Buttons do not have discernible accessible text. |
| Frames without titles | frame-title | Frames or iframes do not have an accessible name. |
| Selects without labels | select-name | Select elements do not have an accessible name. |
| Nested footer landmark | landmark-contentinfo-is-top-level | The contentinfo landmark is nested inside another landmark. |
| Nested header landmark | landmark-banner-is-top-level | The banner landmark is nested inside another landmark. |
| Broken heading order | heading-order | Heading levels increase by more than one level at a time. |
| Multiple header landmarks | landmark-no-duplicate-banner | The page has more than one banner landmark. |
| Missing page title | document-title | The document is missing a meaningful title element. |
| Missing page language | html-has-lang | The page is missing a valid language attribute on the html element. |
| Repeated alt text | image-redundant-alt | Image alt text repeats adjacent visible text. |
| Nested sidebar landmark | landmark-complementary-is-top-level | The complementary landmark is nested inside another landmark. |
| Missing input labels | label | Form controls are missing programmatically associated labels. |

Table 12: WCAG rule mapping used in rule-level diagnostics.