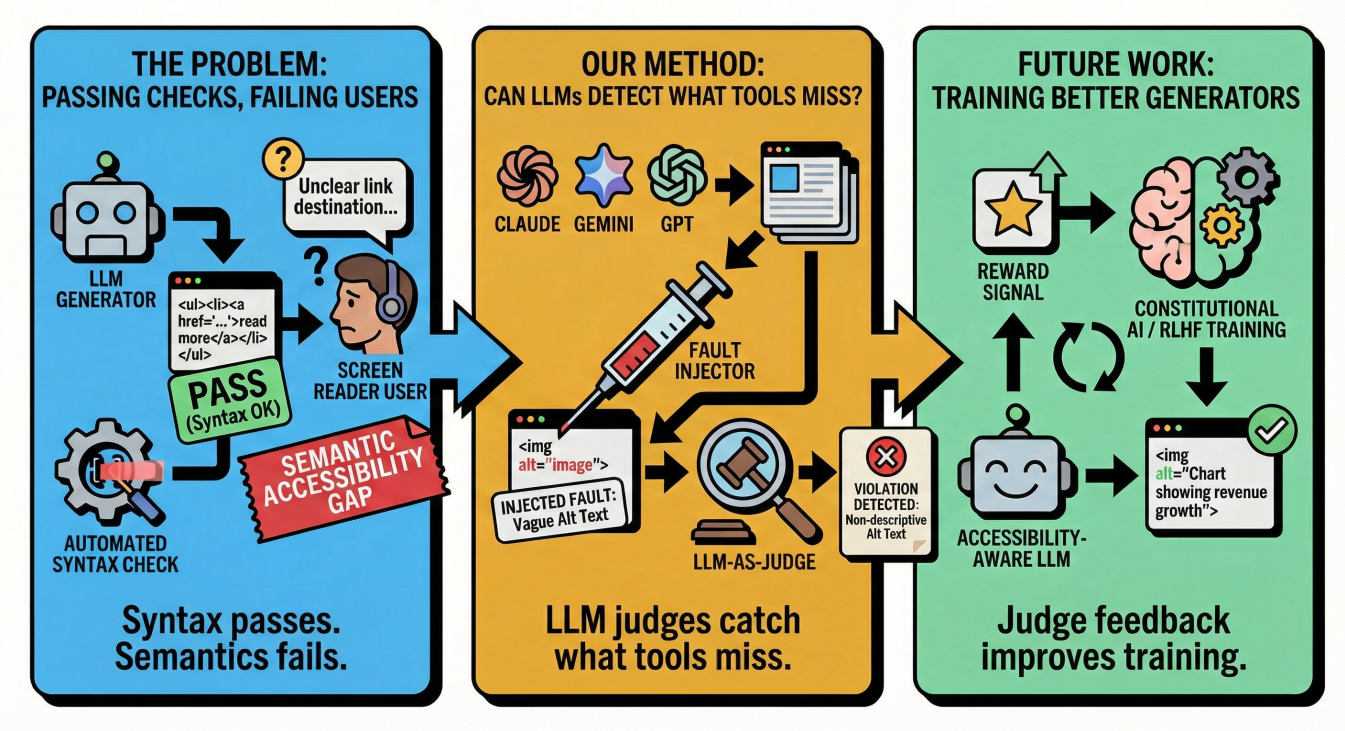
# Measuring the Semantic Accessibility Gap in LLM-Generated Web UIs

Tommaso Calò
Dipartimento di Automatica e Informatica
Politecnico Di Torino
Torino, Torino, Italy
tommaso.calo@polito.it

Alexandra-Elena Gurita
MintViz Lab, MANSiD Research Center
Ştefan cel Mare University of Suceava
Suceava, Romania
alexandra.gurita@student.usv.ro

Luigi De Russis
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italy
luigi.derussis@polito.it




## Abstract

Large Language Models are increasingly used to generate web interfaces from natural language specifications. Automated accessibility tools evaluate these interfaces by detecting syntactic violations such as missing attributes, but cannot assess whether accessibility content is actually *meaningful*—an image with `alt="image"` passes every check yet conveys nothing to screen reader users. We investigate the prevalence of such *semantic* accessibility violations in LLM-generated interfaces. Analyzing 300 UIs produced by three commercial models, we identify 541 semantic violations across six fault types. We validate an LLM-as-judge approach through controlled fault injection, achieving recall rates of 80–92%, and triangulate with a preliminary human annotation study. Our findings suggest that LLM judges can extend accessibility evaluation into the semantic dimension that automated tools miss, opening opportunities for their integration as reward signals in accessibility-aware development workflows.

## CCS Concepts

• **Human-centered computing** → **Accessibility systems and tools**; • **Computing methodologies** → *Natural language generation.*

## Keywords

Web accessibility, LLM code generation, semantic accessibility, WCAG, LLM-as-judge

**ACM Reference Format:**
Tommaso Calò, Alexandra-Elena Gurita, and Luigi De Russis. 2026. Measuring the Semantic Accessibility Gap in LLM-Generated Web UIs. In *Extended*

This work is licensed under a Creative Commons Attribution 4.0 International License.
*CHI EA '26, Barcelona, Spain*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2281-3/26/04
https://doi.org/10.1145/3772363.3799364

*Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems (CHI EA '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3772363.3799364

## 1 Introduction

The emergence of large language models has changed how we approach web interface development, enabling developers to generate complete web interfaces and applications through natural language prompts [12, 14]. While this capability promises to streamline web development by increasing efficiency and lowering technical barriers, it simultaneously raises an important question about whether these automatically generated interfaces remain accessible to all users, including the over one billion people worldwide who live with disabilities [11].

Most existing websites already fail to meet basic accessibility standards [10], and when LLMs are trained on this flawed data, they naturally tend to replicate these same problematic patterns [1, 4]. Recognizing this issue, recent research has begun developing accessibility-aware training approaches and iterative feedback mechanisms [5, 7, 9, 13] to help LLMs generate more accessible code. However, these methods optimize primarily for metrics that automated accessibility tools can measure.

Consider how automated tools like Axe-core [1] evaluate accessibility. These tools excel at detecting what we might call syntactic violations: missing `alt` attributes, absent form labels, or incorrect ARIA roles. If an image has an `alt` attribute, Axe-core marks it as compliant, regardless of whether that attribute actually describes the image meaningfully. An image with `alt="image"` passes every automated check, yet when a screen reader announces "image" to a blind user, no meaningful information has been conveyed. The required syntax exists, but the semantic content—the actual meaning that users need—is absent. Where automated tools ask "does an alt attribute exist?", semantic evaluation asks the deeper question: "does the alt text actually describe what the image shows?".

This disconnect between syntactic compliance and semantic meaning creates what we term the *semantic accessibility gap*: the space between passing automated accessibility checks and actually conveying meaningful information to users of assistive technologies. While prior work has documented the existence of such semantic violations [4, 8], no systematic study has measured their prevalence in LLM-generated interfaces.

In this paper, we investigate what kinds of semantic violations appear in LLM-generated interfaces, and how prevalent they are. We define *semantic accessibility violations* as cases where required accessibility attributes are present but fail to convey meaningful information to users—for example, an image with `alt="image"` or a link reading "Click here" with no indication of its destination. We focus on six violation types drawn from WCAG guidelines and prior accessibility research [3, 4, 8]. In particular, our categorization aligns with the semantic violation class identified by Fathallah et al. [3], who propose a taxonomy distinguishing syntactic, semantic, and layout accessibility issues.

To detect these violations at scale, we employ an *LLM-as-judge* approach, in which an LLM evaluates the generated HTML for semantic accessibility issues given a structured prompt defining the violation types. We validate this approach through *controlled fault injection*: we systematically introduce known semantic faults into UIs and measure whether judges correctly identify them, thereby calibrating our confidence in their assessments of original, unmodified UIs.

We analyze 300 UIs generated by three frontier models (Claude, Gemini, and GPT), identifying 541 semantic violations that pass automated checks. Our judges achieve recall rates between 80% and 92% on injected faults. We further triangulate through human annotation of 324 UI components by four expert annotators.

In summary, this work makes the following contributions:

- An empirical analysis of semantic accessibility violations across 300 LLM-generated web interfaces, identifying 541 violations across six fault types;
- A validated LLM-as-judge methodology, calibrated through controlled fault injection, for detecting semantic accessibility issues beyond the reach of automated tools;
- A preliminary human annotation study comparing LLM and human judgments on component-level semantic accessibility, along with an open-source annotation platform.

## 2 Experimental Design

To measure whether LLM-generated interfaces contain semantic accessibility violations, we validate our approach through controlled fault injection. We inject known violations into UIs and measure whether LLM-as-judge can reliably detect them. High recall on injected faults **calibrates our confidence in the violations judges identify in original, unmodified UIs.** We start by generating 300 web UIs using three LLMs[2] from diverse real-world website prompts [13]. Into these UIs, we systematically injected semantic faults. Table 1 shows the six fault types, each representing a common way that accessibility attributes can be *present but meaningless*.

**Table 1: Six semantic fault types used for validation. Each represents attributes that pass automated checks but fail users.**

| Type | Example |
|---|---|
| Alt text | `alt="image"` instead of describing the image |
| Link purpose | "Click here" instead of describing the destination |
| Button label | "Submit" without context for the action |
| Heading mismatch | Heading text unrelated to section content |
| ARIA label | Inaccurate or generic ARIA descriptions |
| Form label | "Field 1" instead of meaningful label |

We created 721 faulty UI variants, each containing exactly one injected fault along with a log recording what was changed. The distribution of variants across fault types was determined by element availability in each UI (e.g., a UI with no images cannot receive an alt-text fault), resulting in an approximately balanced distribution across the six categories. Three LLM judges (the same models used as generators) then evaluated these UIs, receiving the HTML and a structured prompt defining the violation types.

In addition, to validate whether LLM judgments align with human perception, we conducted a preliminary human annotation

[1] https://www.deque.com/axe/axe-core/

[2] Claude (claude-sonnet-4), Gemini (gemini-2.5-flash), and GPT (gpt-4o).

study with four expert annotators. All annotators were HCI researchers with graduate-level training in web accessibility and familiarity with WCAG 2.1 guidelines; two had prior experience conducting accessibility audits. Before annotating, they received written instructions defining each of the six semantic fault types with examples, followed by a calibration session on two practice UIs (excluded from analysis). Using a custom component-level annotation platform, annotators independently evaluated 324 individual UI elements (images, links, buttons, headings, ARIA labels, and form fields) across 9 randomly selected UIs, marking each as semantically adequate or violated for the applicable fault types.

How should we quantify judge performance? We report two complementary metrics. *Recall* measures the percentage of injected faults that judges correctly identify, providing a direct estimate of detection sensitivity since we know exactly which violations exist in our controlled faulty UIs. *Coefficient of Variation* (CV) captures intra-rater consistency by running each judge five times on a subset of 10 UIs and computing the standard deviation of detected issue counts divided by their mean, expressed as a percentage, where lower CV indicates more stable judgments across repeated evaluations of the same content. For the human annotation study, we measure inter-rater agreement using *Cohen's kappa*, which quantifies agreement between pairs of raters while correcting for chance, with values below 0.20 indicating slight agreement, 0.21 to 0.40 fair, 0.41 to 0.60 moderate, and above 0.60 substantial to near-perfect agreement.

## 3 Results

Across 300 original interfaces, our judges identified 541 semantic accessibility issues before any fault injection, averaging nearly two violations per UI (Figure 1a-b). The distribution of these violations reveals that *generic button labels* and *vague link text* are the two most prevalent categories, each accounting for roughly a quarter of all issues (27% and 26%, respectively). Buttons labeled "Submit" or "Click" provide no context about the action they perform, while link phrases like "Read more" or "Learn more" fail to indicate their destination, leaving users who navigate by links without meaningful orientation cues. Together, these two categories account for over half of all semantic violations in LLM-generated UIs. *Poor alt text* follows at 20%, with *ARIA issues* (14%) and *generic form labels* (12%) completing the distribution. This pattern suggests that interactive elements—which commonly assume meaning from spatial context not available to screen reader users—receive particularly superficial accessibility treatment during generation.

Can we trust these measurements? To establish confidence in our judges, we validated their accuracy through controlled fault injection, creating 721 faulty UI variants with known violations and measuring detection rates (Figure 1c-d). Our judges achieved recall between 80% and 92%, though performance varied considerably by fault type: *alt text* violations proved easiest to detect, likely because the pattern of generic placeholder text is relatively unambiguous, while *heading mismatches* were hardest to identify, requiring judges to reason about cross-element semantic relationships between a heading and the content it introduces. This variation reflects inherent differences in task complexity, as some semantic issues present clear textual patterns that models readily recognize, while others demand more sophisticated reasoning about document structure and meaning.

Do different models exhibit different strengths and weaknesses as judges? The cross-modal analysis reveals substantial variation in performance both across models serving as judges and across UIs from different generators (Figure 2a-b). GPT detected only about half of heading mismatches, while Claude and Gemini performed considerably better on this challenging category. Interestingly, faults in Gemini-generated UIs proved hardest for all judges to detect, suggesting that some generators may produce violations that are subtler or more contextually embedded. No single judge or generator performed uniformly best or worst across all conditions, pointing toward potential complementarity in an ensemble approach.

## 4 Human Annotation Study

While automated evaluation provides scalability, we don't know if human evaluators perceive the same semantic accessibility violations that LLM judges detect. To triangulate our findings, we conducted a preliminary human annotation study using a custom web-based platform[3] that decomposes UIs into constituent accessibility components (Figure 3). Annotators assess individual elements—images, links, buttons, headings, ARIA-labeled components, and form inputs—for the six semantic fault types.

Four expert annotators evaluated 324 components across 9 UIs. Results revealed fair inter-rater reliability among humans ($\kappa \approx 0.24$), with pairwise agreement ranging from near-zero to moderate ($\kappa = 0.46$). Human-LLM agreement matched human-human agreement almost exactly ($\kappa \approx 0.24$), while LLMs showed substantially stronger agreement with ground truth ($\kappa = 0.69$–$0.94$) than humans did ($\kappa = 0.23$ average). Disagreements among annotators concentrated on borderline cases—for instance, whether a button labeled "Submit" in a single-form page provides sufficient context, or whether alt text like "company logo" is adequately descriptive. These cases illustrate the inherent ambiguity in semantic accessibility judgment: what seems "vague" to one evaluator may appear sufficiently descriptive to another given the surrounding context. This pattern may indicate that our controlled fault injection creates relatively unambiguous patterns that LLMs detect more readily than humans, or that humans apply more contextually variable criteria when evaluating semantic quality. Given the small sample of four annotators, these preliminary findings warrant cautious interpretation; however, they demonstrate that component-level annotation is feasible and that LLM judges show alignment with human judgment at levels comparable to human inter-rater reliability.

## 5 Discussion

Our findings suggest that optimizing for automated tools alone offers an incomplete picture of accessibility. Training methods that reward high Axe-core scores [13] may inadvertently teach models to produce syntactically compliant but semantically empty attributes—a generator rewarded for including `alt` attributes receives limited signal to make them *descriptive*. Our validation indicates that LLM judges can detect many of the semantic violations that automated

[3] https://semaccess.paperbackwriters.club/

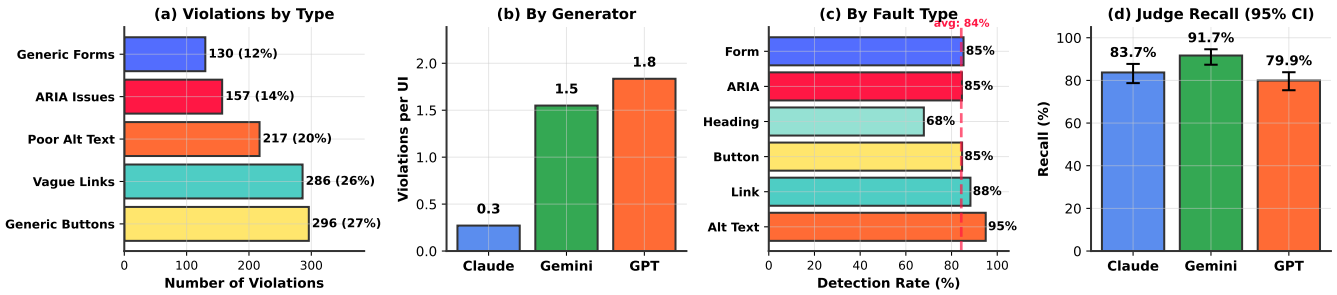



**Figure 1: Semantic violations and judge validation. (a) Distribution of 541 violations found across 300 original UIs by the six semantic fault types from Table 1, dominated by generic buttons and vague links. (b) Average violations per UI by generator model. (c-d) Judge validation across 721 injected faults shows detection varies by fault type; alt text is easiest to detect, heading mismatches hardest.**

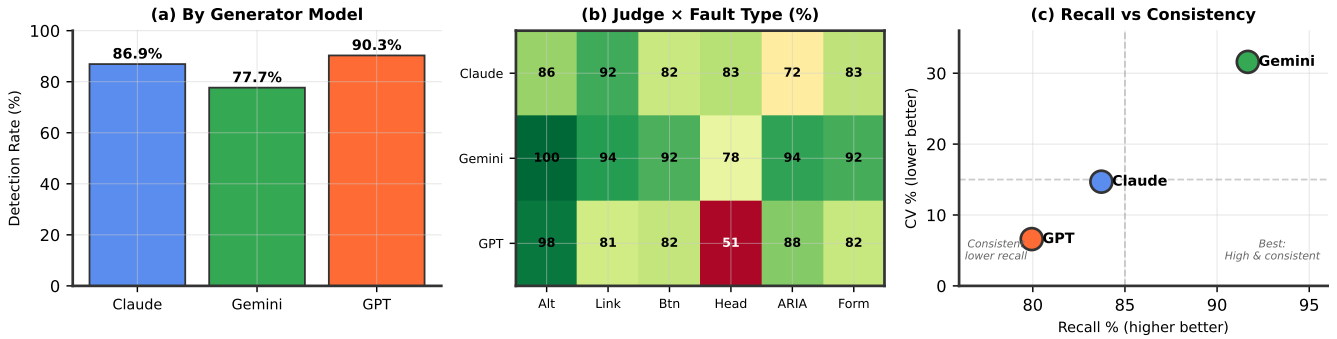



**Figure 2: Cross-modal analysis and recall-consistency tradeoff. (a) Detection rates vary by generator model, with Gemini-generated UIs hardest for all judges. (b) Heatmap reveals model-specific blind spots. (c) Tradeoff between recall and consistency: Gemini achieves highest recall but most variability, GPT most consistent but lower recall, Claude balances both dimensions.**

tools miss, consistent with prior work on LLM-based accessibility evaluation [6, 7], while extending it toward generation-time assessment. However, we note that our evaluation focused on isolated HTML interfaces; whether these findings generalize to full web applications with stateful interactions remains an open question.

Choosing a judge involves tradeoffs (Figure 2c). Gemini achieves the highest recall but varies considerably across runs. GPT is the most consistent but catches fewer violations. Claude balances both dimensions. For practical deployment, high-stakes applications might prefer consistency, while broad screening might prioritize recall. The varying performance across fault types also reflects task complexity: *heading-content relationships* prove the most challenging, while *alt text patterns* are comparatively well-handled.

## 6 Conclusion and Future Work

This paper presents preliminary evidence that LLMs can serve as semantic accessibility evaluators, extending assessment beyond what automated tools currently measure. Through controlled fault injection, we validated that LLM judges detect violations like non-descriptive alt text and vague link purposes with recall rates of 80–92%, while revealing challenges with heading-content mismatches. Our preliminary human annotation study (324 components, 9 UIs, 4 expert annotators) provides initial triangulation, showing that LLM judges achieve agreement with humans at levels comparable to human inter-rater reliability. These findings represent a first step toward addressing the semantic accessibility gap in LLM-generated interfaces. The validated judge framework, along with our open-source annotation platform, could provide complementary reward signals for Constitutional AI [2] or RLHF training pipelines, and the cross-modal analysis points toward ensemble approaches leveraging different models' strengths.

However, several limitations qualify these contributions. Our validation relies on deliberately injected faults, whereas naturally occurring semantic issues may be more subtle and context-dependent. We evaluate isolated HTML files rather than complete web applications, leaving navigation flows and stateful interactions for future work. The small sample size and fair inter-rater reliability ($\kappa \approx 0.24$) in our human study mean these findings warrant cautious interpretation. The higher LLM–ground truth agreement compared to human–ground truth agreement may indicate that our controlled fault injection creates artificial patterns more readily detectable by LLMs than by humans, or it may reflect genuine differences in semantic interpretation. Expanding this work to include

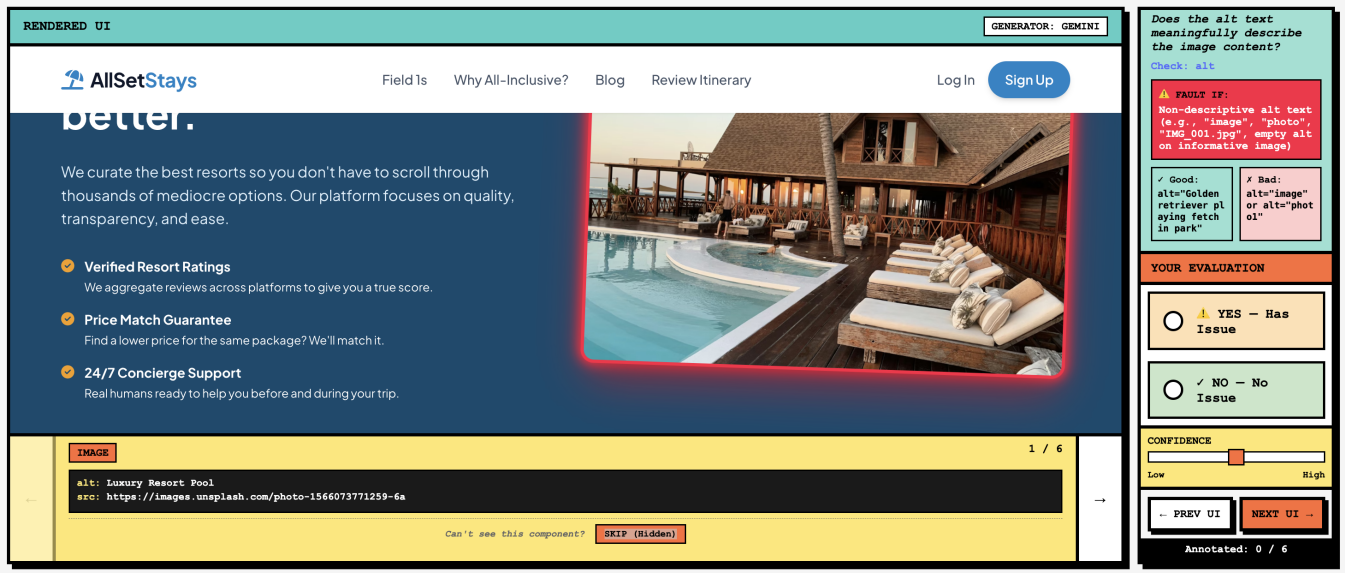



**Figure 3: Component-level annotation interface showing individual element evaluation with context.**

diverse annotators—particularly users of assistive technologies—across larger samples of both injected and naturally occurring violations would help distinguish these possibilities.

Ultimately, these technical metrics need grounding in lived experience. *Do the semantic violations that we detect correspond to real barriers experienced by users?* Participatory evaluation with people with disabilities would validate whether our proxy measures capture what actually matters for accessibility. Our core finding is encouraging: the same LLMs that generate potentially inaccessible interfaces can also help identify—and eventually help prevent—semantic accessibility failures.

## Acknowledgments

We thank the anonymous reviewers for their constructive feedback. We also thank the expert annotators who participated in the human evaluation study. This work was partially supported by the CINECA ISCRA project ReFinD-U (grant HP10CDEJWH), with computational resources provided on the Leonardo supercomputer.

## References

[1] Wajdi Aljedaani, Abdulrahman Habib, Ahmed Aljohani, Marcelo Eler, and Yunhe Feng. 2024. Does ChatGPT Generate Accessible Code? Investigating Accessibility Challenges in LLM-Generated Source Code. In *Proceedings of the 21st International Web for All Conference*. 165–176.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073* (2022).

[3] Nadeen Fathallah, Daniel Hernández, and Steffen Staab. 2025. AccessGuru: Leveraging LLMs to Detect and Correct Web Accessibility Violations in HTML Code. In *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'25)*. ACM. doi:10.1145/3663547.3746360

[4] Alexandra-Elena Guriţa and Radu-Daniel Vatavu. 2025. 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)*.

[5] Ziyao He, Syed Fatiul Huq, and Sam Malek. 2024. Enhancing Web Accessibility: Automated Detection of Issues with Generative AI. https://github.com/seal-hub/GenA11y

[6] Juan-Miguel López-Gil and Juanan Pereira. 2024. Turning Manual Web Accessibility Success Criteria into Automatic: An LLM-based Approach. *Universal Access in the Information Society* (2024), 1–16.

[7] Fabio Paternò, Manuela Vinci, Marco Manca, and Nicola Iannuzzi. 2025. How an LLM Can Improve Automatic Web Accessibility Validation?

[8] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines Are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 433–442.

[9] Hyunjae Suh, Mahan Tafreshipour, Sam Malek, and Iftekhar Ahmed. 2025. Human or LLM? A Comparative Study on Accessible Code Generation Capability. arXiv:2503.15885 [cs.SE]

[10] WebAIM. 2024. The WebAIM Million: The 2024 Report on the Accessibility of the Top 1,000,000 Home Pages. https://webaim.org/projects/million/

[11] World Health Organization. 2024. Disability. https://www.who.int/health-topics/disability

[12] Jingyu Xiao, Ming Wang, Man Ho Lam, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R. Lyu. 2025. DesignBench: A Comprehensive Benchmark for MLLM-based Front-end Code Generation. arXiv:2506.06251 [cs.SE]

[13] Janghan Yoon, Jaegwan Cho, Junhyeok Kim, Jiwan Chung, Jaehyun Jeon, and Youngjae Yu. 2025. A11YN: Aligning LLMs for Accessible Web UI Code Generation. Preprint.

[14] Ting Zhou, Yanjie Zhao, Xinyi Hou, Xiaoyu Sun, Kai Chen, and Haoyu Wang. 2025. DeclarUI: Bridging Design and Development with Automated Declarative UI Code Generation. *Proceedings of the ACM on Software Engineering* 2, FSE (2025), 219–241.