When LLM-Generated Code Perpetuates User Interface Accessibility Barriers, How Can We Break the Cycle?

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

Radu-Daniel Vatavu
MintViz Lab, MANSiD Research Center
Ştefan cel Mare University of Suceava
Suceava, Romania
radu.vatavu@usm.ro

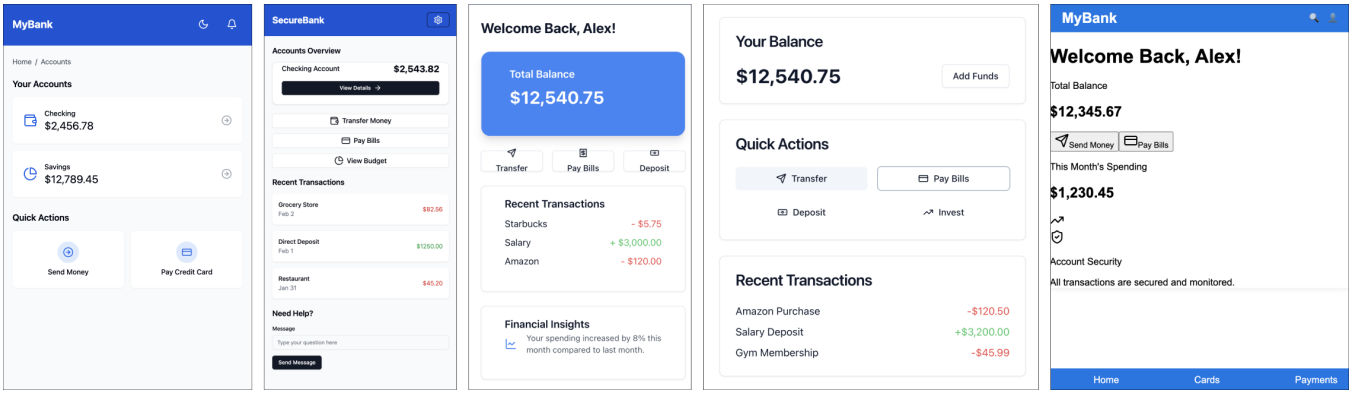

Figure 1: Examples of banking application user interfaces, from the dataset evaluated in this work, generated using ChatGPT and Claude code. Note common patterns, such as prominent balance displays, quick-access payment buttons, and recent transaction lists as well as the consistent use of blue color schemes and similar mobile-first navigation structures. These examples show how LLMs can successfully interpret banking application requirements but primarily replicate existing design patterns, thus perpetuating accessibility issues such as hard-to-read text, poor contrast ratios, and small touch targets.

Abstract

The integration of Large Language Models (LLMs) into web development workflows has the potential to revolutionize user interface design, yet their ability to produce accessible interfaces still remains underexplored. In this paper, we present an evaluation of LLM-generated user interfaces against the accessibility criteria from the Web Content Accessibility Guidelines (WCAG 2.1), comparing the output of ChatGPT and Claude with two distinct prompt types—*accessibility-agnostic* and *accessibility-oriented*. Our evaluation approach, consisting of automated testing, expert evaluation, and LLM self-reflection, reveals that accessibility-oriented prompts increase success counts and reduce violation rates in WCAG criteria, but persistent barriers remain, particularly in semantic structure. We argue that advancing accessible user interface development through LLM-generated code requires not just enhanced prompting but deeper semantic understanding and context awareness in these systems. We use our findings to suggest future work opportunities on the development of next-generation LLM-based accessibility tools as well as practical implications for designers.

CCS Concepts

• **Human-centered computing** → **Accessibility**; **User interface design**; Accessibility design and evaluation methods; • **Computing methodologies** → **Artificial intelligence**.

Keywords

Accessibility, WCAG, AI, Large Language Models, LLMs, User interfaces, Generative AI, Prompt engineering, Accessibility evaluation

ACM Reference Format:
Alexandra-Elena Guriţă and Radu-Daniel Vatavu. 2025. When LLM-Generated Code Perpetuates User Interface Accessibility Barriers, How Can We Break the Cycle?. In *W4A '25: Proceedings of the 22nd International Web for All Conference (W4A '25), April 28–29, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3744257.3744266

W4A '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1882-3/25/04
https://doi.org/10.1145/3744257.3744266

1 Introduction

The application of Large Language Models (LLMs) in web development workflows is transforming current practices in the design [21,40] and deployment [2,46] of user interfaces (UIs) for interactive computer systems across various platforms and application domains. For example, LLM-based coding assistants, such as GitHub

Copilot [10], among others, represent a breakthrough in programming practices by offering effective code suggestions and automating routine tasks [28,39], also making custom application development more accessible to novice users [8]. Even generic LLMs, such as ChatGPT [37], Claude [4], Gemini [11], or DeepSeek [7], have the potential to democratize web UI development for users with limited or no programming knowledge [8,20].

Although extensive research has been conducted to examine LLMs' capabilities in generating functionally correct, secure, and maintainable code [22,28,34], a gap still exists in our understanding of their abilities to produce code for accessible UIs, despite recent efforts to integrate them into developer tools. An emerging concern exists about these tools' lack of adherence to established accessibility standards [30]. In fact, one year has passed since Aljedaani et al. [2] reported that 84% of websites generated using ChatGPT exhibited accessibility violations, with common problems including text resizing (31.8%), contrast (26.6%), and conveying semantic relationships (16.8%). Empirical findings such as these cover just one tiny part of a broader landscape in web UI accessibility, but show deep systemic challenges. Technical solutions alone have repeatedly proved insufficient to address fundamental accessibility barriers, while the disconnect between guideline compliance and actual user needs continues to grow [52,53]. Moreover, emerging technologies often introduce new accessibility issues without necessarily resolving existing ones, e.g., longitudinal data in the WebAIM report [53] show that accessibility improvements have stagnated with minimal year-over-year progress, where 97.5% of web homepages had detectable WACG 2 accessibility failures in 2024, with minimal improvement from 96.3% in 2023 [52].

This concerning *stagnation in web UI accessibility* persists despite technological advances in generative AI tools, capable of producing both UI wireframes and functional code to power them. LLMs, trained on vast datasets, although mostly containing code of websites with limited accessibility compliance, may turn out to be coding tools that propagate accessibility barriers [36] instead of reducing them, when developers integrate LLM-generated code into their workflows [17,20,21,46]. Moreover, as Power et al. [42] noted about web accessibility guidelines, the mere existence of standards does not guarantee their implementation since users might care about different aspects of what makes an inclusive experience [41].

Although previous work by Aljedaani et al. [2] has already documented accessibility challenges in LLM-generated websites, critical gaps remain in our understanding of whether *accessibility-oriented* prompting, involving explicit instructions provided in the prompt about accessibility, affects WCAG compliance compared to generic prompts. Moreover, we are interested in what patterns contribute to perpetuating accessibility barriers in web UIs. Lastly, understanding LLMs' capabilities for self-assessment of accessibility issues in the content they generate would complement existing accessibility evaluation methods by incorporating LLM self-reflection. To address these aspects, we make the following contributions:

(1) We test the hypothesis that structured prompt engineering with specific WCAG success criteria can improve compliance across multiple accessibility dimensions. While Mowar [31] established that LLMs need "proper guidance," our work quantifies these effects across specific WCAG criteria and evaluates different prompting strategies. To this end, we compare *accessibility-agnostic* with *accessibility-oriented* prompts to assess their impact on improving the compliance of LLM-generated code for more accessible UIs. Our results show promising results during *human expert evaluation* although certain accessibility features, such as keyboard navigation, still present challenges. Also, while *automated testing* shows similar numbers of failures in both prompt conditions, it also reveals improvements in the successful implementation of specific criteria, including Focus visibility (WCAG criterion 2.4.7, +233.3%) and Info and Relationships (1.3.1, +66.7%).
(2) We examine LLMs' potential for self accessibility evaluation and remediation. We report that LLMs can effectively identify and suggest fixes for certain accessibility issues, such as contrast ratios and semantic HTML structure, achieving a success rate of 94% in our study, but struggle with more complex requirements, such as ARIA implementation.

Our findings also inform future work opportunities about AI algorithmic responsibility in the context of inclusive design. Overall, they highlight the need for a fundamental shift in how accessibility is approached in AI-driven tools, beyond mere technical compliance, toward ensuring meaningful accessibility outcomes.

2 Related Work

2.1 Longitudinal Perspectives on Web User Interface Accessibility

A systemic failure in digital inclusion, represented by the persistence of accessibility hurdles in online UIs, has been documented over the past decade [5,23]. With a mere 1.2% improvement from 2023 to 2024, the WebAIM Million reports [52,53] offer clear evidence of this stagnation, also highlighting that 56.8% of accessibility issues classified as "accessibility barriers having notable end-user impact" [53] were identified in 2024.

This pattern of persistent inaccessibility manifests across specialized domains and technologies. Power et al.'s [42] foundational work revealed that, even when accessibility guidelines are followed, users still encounter barriers, suggesting that the current approach to accessibility compliance might be flawed [5,50]. Recent research by Petrie and Power [41] revealed a gap between technical compliance and real-world accessibility by showing that users' actual concerns may be different from conventional accessibility guidelines. Makati et al. [23] assessed the efficacy of accessibility overlays over time and found that technical quick fixes not only do not work, but can actually make accessibility challenges worse. This discovery echoes worries that current AI-based solutions might amplify accessibility issues rather than addressing them [2]. According to Kumar et al.'s [18] study of web design patterns across time, accessibility considerations have mostly remained unchanged or even regressed in some areas, despite an increase in visual and functional advancements. This trend is especially noticeable as new web technologies are adopted, where accessibility is frequently viewed as an afterthought rather than a core design requirement [32]. We summarize these longitudinal patterns as follows:

- *Technical fixes fall short.* Technical solutions applied in isolation have proven insufficient to address accessibility barriers.

Current AI tools are incapable of generating fully accessible code, for which human expertise is required [33].
- *Compliance does not equal accessibility.* The gap originally identified between guideline compliance and actual user needs in terms of accessible UIs [41] has continued to widen.
- *New technologies, new barriers.* New technologies often introduce additional accessibility challenges [18] while attempting, sometimes unsuccessfully, to fix existing ones.
- *A state of stagnation.* Without systematic intervention and a fundamental shift in how the community approaches accessible design, improvements in accessibility tend to stagnate.

2.2 LLMs and Web Development Practices

The integration of LLMs into web development workflows represents both an opportunity and a risk for accessibility. While tools like ChatGPT, Claude, and GitHub Copilot have remarkable code generation capabilities [20,39], they may also perpetuate accessibility barriers when trained on less accessible web content [53]. Mowar et al. [32] found that while AI coding assistants show potential for creating more accessible UIs, they currently require explicit accessibility awareness to be effective. This finding suggests that simply having LLMs capable of producing functional code does not automatically translate to fully accessible results without proper developer guidance. However, as reported by Simkute et al. [45], there can be significant productivity losses in human-AI interaction when developers rely too heavily on LLM suggestions without a proper understanding of accessibility requirements. This aspect is particularly concerning when these suggestions might appear technically correct, but fail to meet real-world accessibility needs [32].

Recent studies have reported on the relationship between LLM-assisted development and accessibility compliance. For example, Aljedaani et al. [2] found that LLM-generated UIs exhibit several accessibility barriers, particularly in terms of text resizing, contrast management, and semantic relationships. Othman et al. [38] obtained promising results in LLMs remediating specific issues, with a 94% success rate, but this rate appears to be context-dependent. The challenge extends beyond code generation, and has been confirmed for UI visual generation as well, e.g., Guriţă and Vatavu [14] highlighted how various AI tools tackle accessibility requirements with varying degrees of success, often struggling with aspects that require understanding of user needs and/or the context of use.

2.3 Legislative Urgency and the Digital Accessibility Crisis

Recent legislative developments, particularly the European Accessibility Act (EAA) [9], have accelerated the urgency of digital accessibility compliance [5], while other legislative frameworks worldwide, from the Americans with Disabilities Act [49] to the guidelines for Indian Government Websites and Apps [35], accentuate the need for scalable accessibility solutions. The legislative pressure of these acts reflects a global trend, with strengthened regulations emerging across markets worldwide, from regions with large populations of people with disabilities to leading software development hubs. In this context, traditional approaches to accessibility compliance, often relying on manual audits, are difficult to scale. Moreover, Section 508 in the United States [48] and the UK's Public Sector Bodies Accessibility Regulations [47] explicitly require proactive accessibility consideration in digital development. As organizations rush to comply, many are turning to AI-powered solutions, hoping for a scalable path to compliance [5].

However, this rush toward technological solutions risks missing the fundamental intent of the legislative context, which is ensuring genuine digital inclusion. While LLMs offer promising capabilities for code generation and accessibility checking [31,32], they must be evaluated not just for their technical compliance but also for their ability to support meaningful accessibility outcomes. The technical capability to create accessible digital experiences already exists. What is lacking is not better regulations, but a new perspective in how accessibility is approached with AI-assisted development, an aspect that is both ethically important and legally mandatory.

2.4 Summary

The intersection of legislative urgency, persistent accessibility stagnation, and the growing reliance on LLMs in web UI development characterizes today's accessibility landscape on the web. AI-powered tools, particularly LLMs, offer potential for scalable solutions, but risk perpetuating existing accessibility barriers. This insight reveals the need for new approaches, where the community moves beyond compliance-based strategies toward AI-driven design processes and tools that prioritize real-world accessibility outcomes reflective of actual user needs. Our study, presented next, was conducted to gain further insights, complementing prior work [2], about LLMs' capabilities of generating code for accessible UIs.

3 Experiment

We implemented a factorial design involving two independent variables: LLM-Type (two conditions, Claude 3.5 Haiku [4] and ChatGPT GPT-4-turbo [37]) and Prompt-Type (two conditions, *accessibility-agnostic* and *accessibility-oriented* prompts).[1] For each combination of LLM-Type and Prompt-Type, we repeatedly generated ten UIs; see Figure 1 on the first page for examples. Subsequently, we asked the LLMs for a self-reflection on the accessibility of the UIs they generated, together with improved code, which doubled the size of our dataset. This approach allowed us to investigate both LLMs' technical capabilities and the impact of explicit accessibility requirements in prompts. In total, we generated eighty UIs for our experiment, of which half were subjected to a multi-faceted evaluation and half represent improvements of the first set.

3.1 Prompt Engineering

We utilized two prompts, inspired by Guriţă and Vatavu [14], to evaluate both implicit and explicit accessibility requirements:

- An *accessibility-agnostic* prompt, "Design the homepage of a banking app," representing our control condition.
- An *accessibility-oriented* prompt, "Design an accessible banking app homepage with resizable text (no loss of functionality at 200% zoom), 4.5:1 minimum contrast ratio, clear semantic structure (proper headings, ARIA landmarks, labeled form controls), and 44x44px minimum touch targets per WCAG 2.1 AA," to understand the effect of explicit requirements.

[1] ChatGPT was previously evaluated in [2], and we included Claude as an alternative model to enhance data diversity in our evaluation.

Our selection of evaluation criteria in the *accessibility-oriented* prompt was informed by prior research that identified key accessibility challenges in LLM-generated UIs [2]—notably, ChatGPT was reported to struggle with text resizing (31.8% of the violations), contrast management (26.6%), and semantic relationships (16.8%), while performing better on non-text content and page structure. We deliberately included minimal application-related specifications in both prompt conditions, asking only for the "homepage of a banking app," without prescribing any specific UI components, such as buttons, labels, and menus, or visual styles. This choice was intentional to avoid constraining the LLMs' content generation patterns as well as to observe how *accessibility-oriented* prompting would impact code-level generation. The only variation was the presence of explicit accessibility requirements, following the WCAG guidelines, in the *accessibility-oriented* prompt condition.

3.2 Evaluation Procedure

For the evaluation, we combined automated assessment and expert review. Automated tools, such as MAUVE++ [16],[2] perform an initial accessibility scanning with a rich and flexible report of identified issues [24], while custom evaluation is applied to verify specific requirements, including touch target sizes, text spacing, and color contrast. Two accessibility experts (five years experience) independently evaluated each generated UI using a 5-point severity scale, from 0 (no violation) to 4 (critical barrier), for various WCAG criteria, calculating both Violation-Severity (VS) and Violation-Rate (VR) metrics. VS represents a weighted average of severity scores across the generated UIs, providing a measure of the accessibility implementation quality. For example, a VS score of 2.8 for text contrast would indicate moderately severe barriers in the range $[0, 4]$. Meanwhile, VR is the percentage of UIs with any violation (VS>0), as described in Guriță and Vatavu [13], offering a measure of violation frequency rather than severity. We also measured the delta (Δ) between VS scores, as follows:

$$\Delta(UI_1, UI_2) = |VS(UI_1) - VS(UI_2)| \quad (1)$$

where UI_1 and UI_2 denote two distinct UIs. This metric helps distinguish between consistent violation patterns and UI-specific issues. For example, systematic barriers are indicated by small delta values, e.g., Δ=0.2 for keyboard navigation suggests both UIs struggle similarly with this criterion. In contrast, large delta values, e.g., Δ=3.5 for alternative text, reveal inconsistent implementation, where accessibility features may be properly implemented in one interface but severely lacking in another. This distinction helps understanding whether accessibility failures represent fundamental limitations in the LLM's compliance with certain WCAG criteria or rather variable handling of specific UI elements.

3.2.1 WCAG evaluation criteria. To evaluate accessibility compliance, we synthesized evaluation criteria from multiple sources: common accessibility violations reported by Aljedaani et al. [2] and Guriță and Vatavu [14], and the automated testing capabilities of MAUVE++ [16]. This led to ten WCAG 2.1 success criteria, spanning both Level A and Level AA requirements; see Table 1. Our evaluation employed two complementary approaches:

Table 1: WCAG 2.1 criteria used in the *expert* and *mixed* evaluations of the UIs generated in our experiment. For the criteria used in the *automated* testing, please see MAUVE++ [16].

Criteria	Description	Evaluation
1.1.1	Alternative text for non-text content	Mixed
1.3.1	Information and structure relationships	Mixed
1.4.1	Color not sole means of conveying information	Mixed
1.4.3	Text contrast ratio (4.5:1 normal, 3:1 large)	Mixed
1.4.4	Content functional at 200% zoom	Expert
2.1.1	All functionality via keyboard	Expert
2.4.6	Descriptive headings and labels	Mixed
2.4.7	Focus visible indicator	Mixed
2.5.5	Touch targets (44x44 px)	Expert
4.1.3	Status messages can be determined	Mixed

- *Automated testing*, via MAUVE++, which included 50 WCAG criteria; details available on the MAUVE++ web page.[3]
- *Expert manual evaluation*, which focused solely on criteria requiring human judgment, such as 2.1.1 Keyboard Accessible; see Table 1 for the criteria used in the expert evaluation.

We also combined the automated and expert evaluations into a *mixed* approach, where automated tools can detect problems and expert verification improves precision. We also incorporated *LLM self-reflection* into our evaluation, where the LLMs were prompted to assess their output; see details next.

3.2.2 Self-reflection LLM evaluation. This approach is inspired by Othman et al.'s [38] findings about ChatGPT achieving 94% accuracy in identifying and self-remediating accessibility issues. For each generated UI, we asked the LLMs to perform a self-assessment using the following open-ended prompt to assess LLMs' awareness of accessibility standards: *"What accessibility issues might be present in the UI you just generated, and how could they be improved?"* Given the extensive range of WCAG criteria—over fifty success criteria across levels A, AA, and AAA—a comprehensive enumeration within prompts would be impractical. Instead, our approach allowed us to observe which accessibility considerations LLMs prioritize implicitly, according to their understanding of accessibility, rather than testing their ability to evaluate specified criteria. When the LLMs did not produce edits to the UI code, we asked them to implement the proposed suggestions. This self-reflection evaluation serves three purposes: (1) it reveals the model's awareness of accessibility standards and potential violations, (2) it provides insight into the model's reasoning about accessibility features, and (3) it allows us to compare the model's self-assessment with our own evaluation results. Following Othman et al. [38], we categorized the results of the self-reflections based on their alignment with the findings reported by MAUVE++ and our expert assessment, respectively. We found 76% alignment between LLM-identified issues and expert evaluations, with LLMs excelling at identifying contrast and semantic structure issues, but underreporting accessibility problems regarding keyboard navigation.

[2] https://mauve.isti.cnr.it

[3] https://mauve.isti.cnr.it/supported_wcag21.js

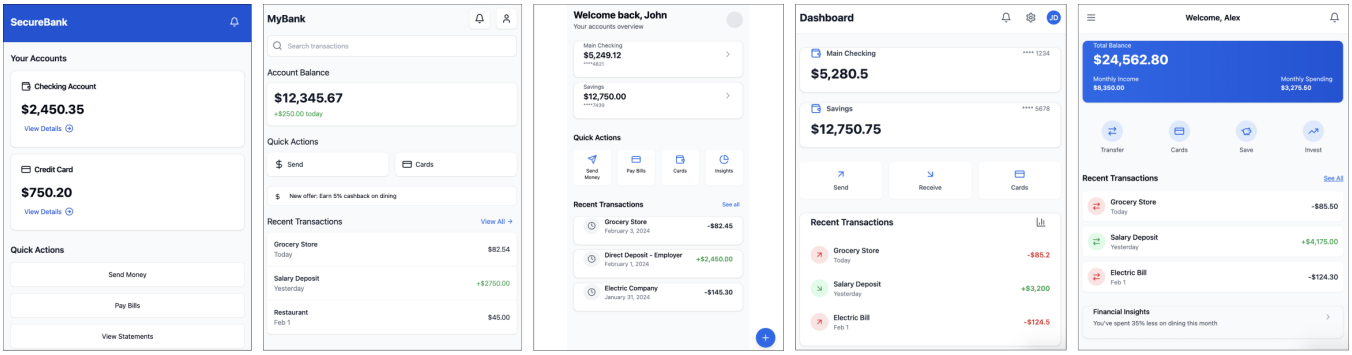

Figure 2: Examples of banking app UIs in our dataset. From left to right: the first two UIs were generated using the *accessibility-oriented* prompt, while the last three were generated with the control condition of the *accessibility-agnostic* prompt.

4 Results

4.1 Automated Testing

4.1.1 Key metrics and performance analysis. We found that the UIs generated with the *accessibility-oriented* prompt had a slightly higher violation rate (17.32%) compared to the *accessibility-agnostic* condition (15.93%), but generally performed better across important accessibility criteria, particularly in terms of success counts for focus visibility, info & relationships, and parsing; see Table 2. Both prompt types excelled in meeting the text contrast criteria (1.4.3), with high success counts across all conditions, suggesting good color contrast handling in the LLM-generated UIs; see Figure 2 for examples. We also observed comparable failure patterns across both LLM-Type conditions, e.g., in non-text content (1.1.1) and language of page (3.1.1). The label in name (2.5.3) criterion generated significant warnings, with 23% reduction in the *accessibility-oriented* prompt, indicating partial remediation of labeling issues.

4.1.2 Improvements, trade-offs, and cross-model patterns. The output analysis between the two prompt types marked improvements by using an *accessibility-oriented* prompt in several areas:

- **Strongest success increase:** Focus visibility (2.4.7, +233.3%) and Parsing (4.1.1, from zero compliance to a score of 4.5).
- **Moderate success counts:** Info and Relationships (+66.7%).
- **Significant success reduction:** Non-text content (1.1.1, −83.3%) and Label in Name (2.5.3, −23.0%).

4.1.3 Comparative analysis of LLMs. Cross-analysis of the two LLM-Type conditions revealed similar responses to *accessibility-oriented* prompting, suggesting common underlying mechanisms in LLM-based implementation of accessibility guidelines. The highest average errors occurred in Language of Page (3.1.1) and Name, Role, Value (4.1.2). These findings indicate that current limitations reflect intrinsic challenges in translating accessibility guidelines into UIs rather than model-specific issues. Our analysis revealed consistent challenges across the two systems, particularly in responsive design (Reflow) and input handling (On Input) when prompted for accessibility, while maintaining persistent issues with language specification. Moreover, neither model demonstrated clear superiority across all metrics, with each showing strengths in different areas. We can conclude that *accessibility-oriented* prompting yields meaningful improvements in some WCAG compliance criteria, but struggles with comprehensive coverage across all criteria.

Table 2: Results of accessibility evaluation across prompting conditions through *automated* testing with MAUVE++ [16].

Metric	Accessibility-agnostic	Accessibility-oriented
Violation Rate (VR)	15.93%	17.32%
	*Success counts (Average/Max)**	
Contrast (1.4.3)	22.8/42	24.3/45
Info & Rel. (1.3.1)	3.3/6	5.5/8
Focus visible (2.4.7)	0.9/2	3.0/9
Parsing (4.1.1)	0.0/0	4.5/8
	*Failure counts (Average/Max)**	
Non-text content (1.1.1)	0.6/6	0.1/1
Language of page (3.1.1)	1.0/1	1.0/1
Name, Role, Value (4.1.2)	0.4/4	0.5/2
	*Warning counts (Average/Max)**	
Label in Name (2.5.3)	16.1/44	12.4/36

*Averages are computed across all LLM-Type conditions and UIs.

4.2 Expert Evaluation

4.2.1 Violation rates. VR analysis revealed differences between the *accessibility-agnostic* (58.0%) and *accessibility-oriented* (19.0%) prompts; see Figure 3. The *accessibility-agnostic* prompt struggled particularly with alternative text (1.1.1, 100%), information structure (1.3.1, 100%), and status messages (4.1.3, 90%). This was visible in several of the generated UIs where content lacked proper semantic structure, no proper heading relationships and section organization. The corresponding code structure would likely show content presented as visual elements without the hierarchical structure needed for screen readers to interpret the relationships between elements. The *accessibility-oriented* prompt achieved substantially better compliance across most criteria. We observed strong improvements in information structure relationships (1.3.1, from 100% to 0%, lower is better) and keyboard accessibility (2.1.1, from 80% to 0%). The

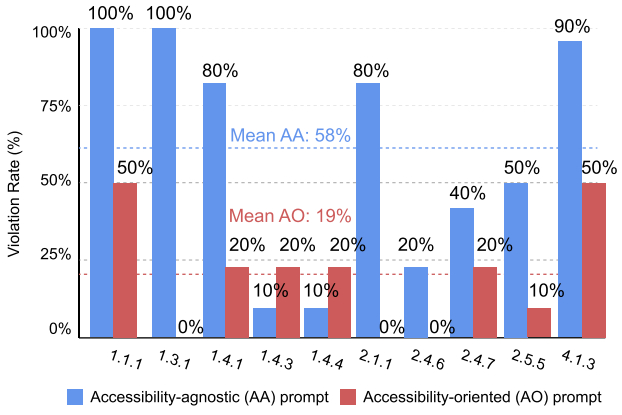

Figure 3: Comparison of expert evaluation results for the VRs of UIs generated with *accessibility-agnostic* (AA) and *accessibility-oriented* (AO) prompts. *Notes:* Percentages indicate how often each criteria was violated; expert evaluations are reported, for automated evaluation results see Table 2.

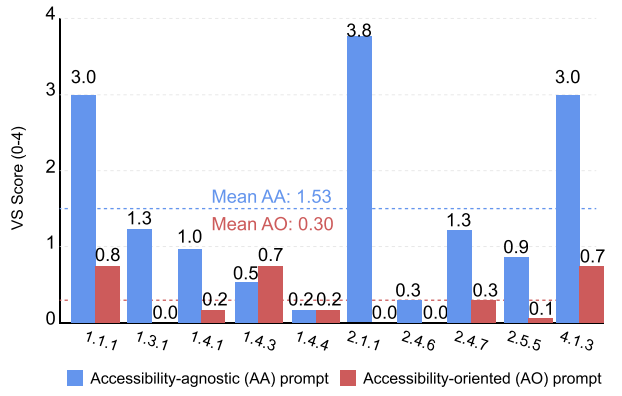

Figure 4: Comparison of expert evaluation results for the VSs of UIs generated with the *accessibility-agnostic* (AA) and *accessibility-oriented* (AO) prompts. *Notes:* Higher scores indicate more severe violations; expert manual evaluations are reported, as opposed to the automated evaluation in Table 2.

overall VR improvement of 39% between prompt types shows that explicit accessibility guidance leads to more consistent implementation of accessibility features at code level. However, some criteria like text contrast (1.4.3) and zoom functionality (1.4.4) had higher violation rates in the *accessibility-oriented* condition (20%).

4.2.2 Violation severity. The severity of violations also varied between the two prompt types; see Figure 4. Control prompts generated UIs with critical violations in essential criteria such as keyboard navigation (2.1.1, VS=3.8), alternative text for images (1.1.1, VS=3.0), and status message determination (4.1.3, VS=3.0), which can create substantial barriers for users relying on keyboard interaction or screen readers. These violations often represented complete omissions of necessary accessibility features rather than implementation refinements. For instance, the UIs generated with the control, *accessibility-agnostic* prompt frequently omitted keyboard accessibility features entirely, making navigation difficult when not using a mouse, or failed to provide meaningful alternative text, rendering visual content inaccessible to screen reader users. In contrast, the *accessibility-oriented* prompt produced lower severity violations, related to implementation details rather than fundamental oversights. When violations occurred, they typically involved refinements to existing accessibility features, such as alternative text for non-text content (1.1.1, VS=0.8) or contrast ratios (1.4.3, VS=0.7). This finding shows that the *accessibility-oriented* prompt addressed basic accessibility requirements while leaving room for enhancement. The average VS score dropped from 1.53 in the *accessibility-agnostic* condition to 0.30 in the *accessibility-oriented* one (−80.4%, $|\Delta|$=1.23).

4.3 Code Comparison Between Prompt Types

We examined the generated code through the lens of key accessibility principles and the WCAG guidelines, focusing on real-world impact for users with disabilities [44].

4.3.1 UI structure and navigation. UIs generated with the *accessible-oriented* prompt showed better structural organization; see Table 3. For example, one UI contained proper landmark roles that created clear navigation regions, with the banking dashboard logically divided into sections using `<main>`, `<nav>`, and `<aside>` elements. This structural clarity was notably absent in the UIs generated with the *accessibility-agnostic* prompt, where `<div>` elements were used exclusively, creating potential confusion for screen reader users.

The handling of dynamic content also revealed differences across the two prompt types. While UIs generated using the *accessibility-agnostic* prompt implemented basic updates using unannounced `<div>` elements, the UIs produced with the *accessible-oriented* prompt properly implemented `aria-live` regions for critical information like account balance changes. Similarly, error handling in form validation demonstrated how semantic HTML impacts accessibility. The control prompt led to UIs that often lacked proper error associations, while accessible versions linked error messages to inputs through `aria-describedby` and `role="alert"` attributes; see Listing 1. These structural and semantic differences can impact screen reader navigation. Furthermore, heading hierarchy in the *accessible-oriented* condition followed a logical pattern, with account balances using `<h1>` and subsequent sections like "Recent Transactions" properly nested under `<h2>`. In contrast, the control condition often led to misused heading levels or relied on styled `<div>` elements for implementing the visual hierarchy.

4.3.2 Interactive elements. Looking at the implementation of interactive elements—a must in passing AA levels of WCAG criteria—the *accessibility-oriented* prompt consistently led to UIs with robust implementation; see Table 4. The transaction buttons of the UIs maintained the recommended 44x44 px minimum touch target size through Tailwind classes like `min-h-[44px]`, while the control prompt UIs' quick action buttons varied between 24 px and 32 px,

Table 3: Structural implementation comparison of generated UIs between prompt types.

User interface component	Accessibility-agnostic prompt	Accessibility-oriented prompt
Landmark roles	11%	94%
Heading hierarchy	33%	89%
Skip links	0%	100%
ARIA labels	28%	95%

Notes: Percentages indicate implementation completeness across key structural elements. Landmark roles refer to semantic HTML5 and ARIA landmark coverage. Heading hierarchy measures correct nesting of h1-h6 elements. Skip links indicate the presence of keyboard navigation shortcuts. ARIA labels show proper labeling of interactive elements.

Table 4: Accessibility implementation details for interactive elements present in the generated UIs.

Interactive component	Accessibility-agnostic prompt	Accessibility-oriented prompt
Touch target size	32px	44px
Focus indicators	56%	98%
Keyboard access	48%	94%
Error feedback	34%	88%

Notes: Touch target size measurements represent average interactive element dimensions. Focus indicators and keyboard access percentages show the proportion of elements with proper implementation. Error feedback indicates coverage of accessible error messaging.

Listing 1: Example of control prompt implementation with poor semantic structure.

```
<div className="space-y-3 mb-8">
  {accounts.map(account => (
    <Card key={account.id} className="bg-white">
      <CardContent className="p-4">
        <div className="flex justify-between items-center">
          <div>
            <p className="text-sm text-gray-600">
              {account.name}
            </p>
            <p className="text-xl font-semibold">
              ${account.balance.toLocaleString('en-US',
              {minimumFractionDigits: 2})}
            </p>
          </div>
        </div>
      </CardContent>
    </Card>
  ))}
</div>
```

potentially challenging for users with upper-body motor impairments. Focus management showed particular improvement in the *accessibility-oriented* condition, implementing visible focus indicators with clear contrast ratios and consistent styling across all

1. **1.3.1 Info and Relationships** – The `<button>` elements in the navigation (e.g., "Send Money," "Pay Card") are not contained within `<ul>` / `<li>` structures. Lists should be properly structured for screen reader users.
2. **1.4.1 Use of Color** – No alternative indication for selected navigation items beyond color (`text-blue-600`). Users with color blindness may struggle to differentiate selected items.
3. **2.4.3 Focus Order** – No clear focus order indicated. While a skip link is provided, tab order may be unclear.
4. **2.4.6 Headings and Labels** – The transactions list lacks a clear heading structure for individual transaction items, making it difficult for screen reader users to understand the hierarchy.
5. **2.5.3 Label in Name** – The `aria-label` for transaction buttons does not match the visible label. Screen readers will read an aria-label different from the on-screen text, causing confusion.

Figure 5: Example of a response from ChatGPT when asked to evaluate a generated UI for accessibility compliance.

interactive elements. In contrast, the control condition often relied on browser defaults or omitted focus indicators entirely.

4.3.3 Content and information design. Information presentation of the UIs generated with the *accessibility-oriented* prompt prioritized clarity and multimodal access. For example, one UI illustrated the transaction history by including both color coding and explicit status labels, e.g., "Payment Successful" rather than just green text, making information accessible to colorblind users.

4.3.4 User experience. The *accessibility-oriented* prompt determined accessibility improvements compared to the *accessibility-agnostic* condition that have broader benefits across different user groups, such as improving user experience. For example, the UIs generated in this condition contained a clear heading structure, which can improve navigation for all users, making content hierarchy more apparent and scannable as well as more usable for screen readers. Moreover, larger touch targets benefit mobile users regardless of ability by reducing input errors and increasing interaction confidence. Explicit labels and instructions also reduce input errors.

4.4 LLM Self-Evaluation Capabilities for Accessibility Compliance

LLMs demonstrate capabilities to self-evaluate UI accessibility compliance through structured criteria assessment, and our analysis of their answers revealed evaluations centered around WCAG guidelines with explanations that organized evaluation criteria hierarchically, following WCAG POUR principles; see Figure 5.

We found that LLMs prioritize criteria based on Level A and AA requirements, common accessibility barriers, and features with the highest user impact. This prioritization was justified as follows: *"I start with these core WCAG criteria because they cover critical accessibility barriers that impact users the most."* However, they also show notable limitations, including selective coverage and context-dependency issues, e.g., *"Some WCAG success criteria don't apply to every interface,"* indicating the need for human expert assessments, development of comprehensive evaluation frameworks, and investigation of LLM capability to evaluate dynamic UIs.

Our analysis also revealed the opportunity of using LLMs to improve the accessibility of the UIs they generated. For instance, the LLMs identified critical issues around keyboard navigation and

screen reader support, leading to concrete improvements. The generated code revealed this through the addition of ARIA landmarks, such as `role="navigation"`, live regions `aria-live="polite"`, and keyboard focus management `focus:ring-2`. The models' self-assessment also led to enhanced semantic structure, exemplified by the transformation of a generic spending data display into a properly structured HTML table with appropriate ARIA labels `aria-label="categories"` and scope attributes `scope="row"`. We also observed proactive accessibility improvements that were not required by the prompt, including user preference handling with `prefers-reduced-motion`, state management `aria-expanded`, and dark mode implementation `dark:bg-gray-800`. The iterative improvements, particularly in the implementation of motion sensitivity and dialog management `role="dialog"`, showcase the LLMs' capability to refine accessibility features. This finding reveals LLMs as both code generators for accessible UIs and accessibility advisors, capable of identifying accessibility violations in their output.

5 Discussion

In this section, we build on our findings to discuss implications for the research and practice of future accessibility-oriented UI design through the integration of LLM-based development.

5.1 Impact of Accessibility-Oriented Prompts

Our experiment revealed several differences in WCAG compliance between *accessibility-oriented* and *accessibility-agnostic* prompts, highlighting the importance of explicit accessibility guidance. This presents a notable contrast to traditional AI design tools, primarily involving visual design, for which Guriță and Vatavu [14] found that *accessibility-oriented* prompts did not significantly improved compliance. In contrast, the stark difference in violation rates in our experiment indicates the effectiveness of explicit accessibility guidance integrated into prompts. This discrepancy is particularly notable in the results of expert testing, which captured distinct accessibility issues compared to automated testing. Consequently, LLMs that output code for UIs can effectively incorporate accessibility requirements when properly prompted. More importantly, the improvements span across critical WCAG success criteria. Based on these findings, we identify three areas where *accessibility-oriented* prompts successfully demonstrated improvements:

- *Improved semantic structure.* We observed ARIA landmarks and heading hierarchy becoming the standard practice rather than exceptional cases (WCAG 1.3.1 Info and Relationships at level A and WCAG 2.4.1 Bypass Blocks at level A).
- *Improved implementation of interactive elements.* Our results showed systematic improvements in the accessibility of interactive elements compared to the control condition, such as consistent implementation of 44x44 px touch targets (WCAG 2.5.5 Target Size at level AAA) and clear focus indicators (WCAG 2.4.7 Focus Visible at level AA).
- *Improved information presentation through alternative modalities.* We observed improvements in terms of color contrast (WCAG 1.4.3 Contrast at level AA) and providing information in complementary ways, such as combining visual cues with text and screen-reader announcements (WCAG 1.4.1 Use of Color at level A). This aspect represents a clear departure from UIs relying solely on color to convey information in order to make content more accessible to users with various visual abilities.

Our findings suggest that while *accessibility-oriented* prompting does improve implementation of WCAG requirements, it does not automatically translate into comprehensive implementation across all criteria, showing limitations in LLMs' understanding of interactive accessibility requirements versus visual accessibility features.

5.2 Fine-Tuning for Accessibility-Oriented Prompts vs. Critical WCAG Violations in Accessibility-Agnostic Prompting

The code generated using the *accessibility-agnostic* prompt revealed fundamental accessibility oversights and a lack of basic awareness in the implementation. Based on the evaluation criteria of automated testing, these consisted of a complete absence of skip navigation links (failing WCAG 2.4.1 Bypass Blocks at level A), missing ARIA landmarks and roles (violating WCAG 1.3.1 Info and Relationships at level A), and having incomplete or missing heading structures (failing WCAG 2.4.6 Headings and Labels at level AA). Notably, these violations represent barriers that could completely prevent access for users, corresponding to the maximum severity rating on our scale. In contrast, the *accessibility-oriented* prompt generated UIs that needed just some refinement, if any at all. For example, while skip links were consistently present, they occasionally needed repositioning in the UI for better keyboard access sequence; ARIA landmarks were implemented, but sometimes needed more descriptive labels; and heading structures were in place, but occasionally needed hierarchy refinement. These violations represent opportunities for enhancement, solved during one iteration.

5.3 A Note on Visual User Interface Design Tools vs. Code Generation

We found that LLM-based UI code generation shows different accessibility patterns compared to traditional AI design tools primarily centered on visuals [14]. However, each approach exhibits weaknesses: visual design tools primarily fail with visual aspects such as text and non-text contrast and use of color [14], while LLMs demonstrate issues with keyboard functionality or alt text; see Section 4. These differences stem from their underlying architectures: visual design tools lack semantic understanding, while LLMs understand code structure, but may miss critical interactive accessibility requirements. We see this divergence attributed to several factors:

- LLMs have access to semantic operation with HTML structure and ARIA attributes compared to visual-first design tools. While the latter focus on replicating visual patterns [13, 14], LLMs can implement programmatic accessibility features based on their training on codebases.
- LLMs exhibit strengths in computational accessibility requirements that visual design tools struggle with. This suggests that LLMs' code-centric approach may be better suited for implementing technical accessibility specifications compared to tools that prioritize visual pattern matching.

- The differences in the nature of the training data and model architecture impact how accessibility requirements are interpreted. Visual design tools learn from existing visual UI designs, while LLMs learn from code that may include accessibility-specific patterns and documentation. This enables the latter to better translate explicit accessibility requirements into proper technical implementations.

In this context, we envisage potential benefits in hybrid solutions that leverage the strengths of both approaches toward future generations of accessibility tools that integrate LLMs' strong performance in computational requirements with traditional visual design. In addition, we recommend developers using LLMs for UI generation to (1) explicitly include WCAG success criteria in prompts, (2) perform supplementary testing for keyboard navigation and focus management, and (3) leverage LLMs' self-reflection capabilities as an initial accessibility check while recognizing their limitations in assessing complex interactive requirements.

5.4 The Evolution of Accessibility Compliance Since Last Year

Examining automated accessibility compliance across the results of Aljedaani et al. [2] using ChatGPT, published in 2024, and our own, we noticed persistent challenges alongside several improvements in WCAG compliance. Last year, the criteria with most violations were resize text, contrast, and info and relationships, suggesting challenges primarily with visual presentation and content structure. In contrast, this year's automated testing showed language of page emerging as the dominant issue, followed by info and relationships, and name, role, value. Expert testing further highlighted keyboard accessibility, alternative text, and status message determination as major concerns. This comparison indicates a shift from primarily visual presentation issues to more structural, semantic, and interactive aspects of the UIs. Collectively, these findings highlight the complex landscape of accessibility compliance, where progress is neither uniform nor complete. They also point to the complementary value of both automated and expert testing approaches.

5.5 Future Work Opportunities in LLM-based Code Generation for More Accessible UIs

5.5.1 Training data quality and representativeness. Current training data quality presents a fundamental challenge since 97.5% of web code fails at meeting basic accessibility standards [53]. This state of things causes a self-reinforcing cycle where LLMs learn from and reproduce inaccessible patterns. Addressing this challenge requires a systematic approach to data curation and model training where future research must explore methods for developing accessibility-focused training datasets that accurately represent not only accessible design patterns but also actual user needs. This involves more than collecting compliant code examples but rather understanding how different accessibility features interact.

5.5.2 Semantic understanding and context awareness. Current LLMs, while capable of implementing specific WCAG requirements, lack deeper semantic understanding of accessibility principles, as seen in our control prompt condition. They often produce technically correct but practically inadequate solutions, particularly evident in their handling of semantic HTML structure and information architecture. This limitation suggests a need for more research into how AI design tools can develop an understanding of accessibility principles. Future work could explore novel approaches for teaching AI models about the underlying purposes of accessibility features rather than just their technical implementation. Central to this is also the question of how we might enable AI systems to grasp the relationship between semantic structure and actual user experience, moving beyond mere technical compliance checklists.

5.5.3 Prompt engineering and implementation effectiveness. Further examination of the relationship between prompts and implementation outcomes is needed, e.g., how much does prompt specificity affect the desired outcomes. Our findings suggest that *accessibility-oriented* prompts can improve WCAG compliance, but some behaviors require further investigation. For example, understanding how prompt effectiveness vary across different UI types and AI models could enable more effective prompt engineering strategies, keeping in mind that prior research has highlighted that arcane prompts tend to produce better outcomes than those in plain language [6,30]. We also recommend examination of how easy generated UI code can adapt to evolving accessibility standards or if the compliance changes with new AI models, as well as whether the generated UIs can effectively be maintained and updated over time.

5.6 A Call to Action for the Scientific Community of HCI and Accessibility

In the light of recent findings in the scientific literature (Section 2), our own empirical results (Section 4), and their implications for future AI-assisted development (Section 5), we propose a change in how we approach digital accessibility by seeing AI tools as amplifiers of our existing practices—both good and bad—requiring careful design to achieve positive results. In this context, a perspective articulated by Morris [29] highlights both the significant responsibilities and substantial risks that AI systems present for people with disabilities: *"As technologists, it is our responsibility to proactively address these issues in order to ensure people with disabilities are not left behind by the AI revolution"* [29, p. 37].

Emerging AI capabilities, as emphasized by the latest Gartner's [3] analysis of strategic technology trends, identifies Agentic AI as a pivotal development leading to systems capable of autonomous decision-making without human intervention. This technological evolution and reliance raises important questions about how these systems will address diverse accessibility requirements since accessibility challenges are fundamentally experience problems, not merely technical compliance checks. To this end, we outline four key directions for AI-assisted accessible UI design:

- **Design AI-based solutions with users, not just for users.** We recommend co-creation involving people with disabilities throughout AI-based development life cycles. For example, product teams could establish activities of accessibility design sprints (experiences documented by the community [27,43,51]) or facilitate AI-powered design sprints (already a service of design sprint academies [1]). By bringing the perspectives of users with disabilities into the development process earlier, particularly when evaluating how AI

systems reason, we can create more meaningful feedback cycles. For example, improving accessibility for people with disabilities in the workplace through AI-powered solutions, toward more inclusive employment, would greatly benefit from user-centered design approaches that clearly highlight the various roles, benefits, and contributions of AI. In this way, end users' lived experiences would provide essential context that technical guidelines alone cannot capture.

- **Engage in a shared effort across disciplines.** True accessibility is not the result of an individual view, but a shared effort—the definition of accessibility compliance, as described by Johannes Lehner in a post about responsibility indicators [19]. This way of collaboration has been seen in various product processes, e.g., decision making, where everyone involved in the product development cycle can influence the outcomes [12]. Applied to web accessibility, Lehner suggests that some success criteria require *"coding solutions, while others rely on clear content, and some depend on good design decisions"* [19] which means that, by clearly indicating responsibilities, teams could focus on what they can improve without feeling overwhelmed by the full guidelines.
- **Move from detection to intervention.** The HCI community actively pursues involving people with disabilities in accessibility research [26]. As Mankoff et al. [25] state, *"accessible AI is ultimately a question of values, not technology"* (p. 42), and these values need to be demonstrated. Simply highlighting non-compliance of products or AI outcomes is not enough. For example, in our context of LLM-generated UI code, instead of the system simply flagging a button's insufficient contrast level (e.g., during self-reflection, as described in Section 3), it could analyze the overall context of the UI and suggest more accessible alternatives that preserve the design intent, such as the button's role in the user flow.
- **Ensure access to representative data for more inclusive outcomes.** With curated datasets of exemplary accessible implementations, we could avoid replicating accessibility patterns—or, worse, accessibility barriers from current UIs—and enable creative generation of new solutions. As stressed in prior work, *"organizations building AI systems must also improve equity in data collection, review, management, storage, and monitoring"* [25, p. 42]. We need to make sure that AI models are offered the opportunity to learn from diverse experiences of meaningful accessibility implementations and have a comprehensive understanding of the corresponding semantics. This approach would enable AI models to learn what actually works in the real world, e.g., in the workplace for users with disabilities, not just what technically complies.

We acknowledge that translating these principles into practice requires organizational changes and accessibility being treated as a continuous process integrated throughout the product development cycles rather than as a final compliance checkpoint.

6 Limitations

Several limitations in our experiment provide opportunities for future work. First, our automated evaluation was based on WCAG and the MAUVE++ tool, which may not capture all aspects of accessibility implementation, even backed up by manual expert evaluation. User testing involving people with various disabilities would provide a better understanding of the real effectiveness of the LLM-generated code for UIs. Second, our evaluation was carried out on a small set of UIs generated by two LLMs and, as AI technologies evolve [15], the results of our experiment represent just a snapshot in time of the capabilities of two AI models available in February 2025. Lastly, although we reported that *accessibility-oriented* prompts enhance WCAG compliance in the generated UIs, we did not assess the long-term sustainability or robustness of the corresponding code to further modifications by developers. We leave these interesting opportunities for exploration in future work.

7 Conclusion

Our evaluation of LLM-generated UI code for a banking application confirmed that LLMs can improve accessibility and highlighted certain shortcomings in addressing specific WCAG criteria. We also found that the use of *accessibility-oriented* over *accessibility-agnostic* prompts enhanced the accessibility of the generated UIs with some challenges remaining in alt text and status management. This new empirical evidence, complementing prior work in this area, shows that accessibility issues in generated UIs persist regardless of the rapid advancement of AI technology and the pressing legislation about accessibility requirements for products and services. In this context, the community needs to work on strategies to transform the way AI systems comprehend and implement accessibility: reflecting real-world user needs, not just technical compliance checklists. We hope that the opportunities we have identified in our call to action will inspire new AI-driven approaches to accessible computing with meaningful advancements in inclusive UI design.

Acknowledgments

This work was supported by a grant of the Ministry of Education and Research, CCCDI-UEFISCDI, project number PN-IV-P7-7.1-PTE-2024-0434, within PNCDI IV.

References

[1] Design Sprint Academy. 2025. AI Design Sprints vs. AI-powered Design Sprints. What's the Difference? https://www.designsprint.academy/blog/ai-design-sprints-and-ai-powered-design-sprints

[2] 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 (W4A '24)*. ACM, New York, NY, USA, 165–176. doi:10.1145/3677846.3677854

[3] Gene Alvarez. 2024. Gartner Top 10 Strategic Technology Trends for 2025. https://www.gartner.com/en/articles/top-technology-trends-2025

[4] Anthropic. 2024. Claude. https://claude.ai

[5] Stacy M. Branham, Shahtab Wahid, Sheri Byrne-Haber, Jamal Mazrui, Carlos Muncharaz, and Carl Myhill. 2024. The State of Digital Accessibility. *Queue* 22, 5 (Nov. 2024), 60–72. doi:10.1145/3704442

[6] Maitraye Das, Alexander J. Fiannaca, Meredith Ringel Morris, Shaun K. Kane, and Cynthia L. Bennett. 2024. From Provenance to Aberrations: Image Creator and Screen Reader User Perspectives on Alt Text for AI-Generated Images. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, New York, NY, USA, Article 900, 21 pages. doi:10.1145/3613904.3642325

[7] DeepSeek. 2024. DeepSeek Platform. https://platform.deepseek.com

[8] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing Education in the Era of Generative AI. *Commun. ACM* 67, 2 (Jan. 2024), 56–67. doi:10.1145/3624720

[9] European Parliament and Council. 2019. Directive (EU) 2019/882 on the Accessibility Requirements for Products and Services. https://eur-lex.europa.eu/legal-

content/EN/TXT/?uri=CELEX:32019L0882
[10] GitHub. 2024. GitHub Copilot. https://github.com/features/copilot
[11] Google. 2024. Gemini. https://gemini.google.com
[12] Alexandra-Elena Guriță. 2025. Decision-Making in Design. From Decision to Validation: The Art of Testing Product Design Choices. https://medium.com/design-bootcamp/decision-making-in-design-829f61707af7
[13] Alexandra-Elena Guriță and Radu-Daniel Vatavu. 2025. Breaking Bad (Design): Challenging AI User Interface Accessibility Guardrails. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA'25)*. ACM, New York, NY, USA, 7 pages. doi:10.1145/3706599.3716220
[14] Alexandra-Elena Guriță and Radu-Daniel Vatavu. 2025. Insights and Implications of Evaluating Accessibility Compliance in AI-Generated Web Interfaces. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25)*. ACM, New York, NY, USA, 5 pages. doi:10.1145/3701716.3715552
[15] Thomas Haigh. 2025. Artificial Intelligence Then and Now. *Commun. ACM* 68, 2 (Jan. 2025), 24–29. doi:10.1145/3708554
[16] Nicola Iannuzzi, Marco Manca, Fabio Paternò, and Carmen Santoro. 2025. Combined Accessibility Validation and Monitoring of Web Sites and PDF Documents. *Universal Access in the Inf. Soc.* (2025), 20 pages. doi:10.1007/s10209-025-01194-7
[17] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J. Cai. 2022. PromptMaker: Prompt-based Prototyping with Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22)*. ACM, New York, NY, USA, Article 35, 8 pages. doi:10.1145/3491101.3503564
[18] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3083–3092. doi:10.1145/2470654.2466420
[19] Johannes Lehner. 2025. Responsibility Indicators. https://www.linkedin.com/feed/update/urn:li:activity:7299409415240052736
[20] Amanda Li, Jason Wu, and Jeffrey P. Bigham. 2023. Using LLMs to Customize the UI of Webpages. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23 Adjunct)*. ACM, New York, NY, USA, Article 45, 3 pages. doi:10.1145/3586182.3616671
[21] Vivian Liu and Lydia B. Chilton. 2022. Design Guidelines for Prompt Engineering Text-to-Image Generative Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, New York, NY, USA, Article 384, 23 pages. doi:10.1145/3491102.3501825
[22] Yue Liu, Thanh Le-Cong, Ratnadira Widyasari, Chakkrit Tantithamthavorn, Li Li, Xuan-Bach D. Le, and David Lo. 2024. Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues. *ACM Trans. Softw. Eng. Methodol.* 33, 5, Article 116 (June 2024), 26 pages. doi:10.1145/3643674
[23] Tlamelo Makati, Garreth W. Tigwell, and Kristen Shinohara. 2024. The Promise and Pitfalls of Web Accessibility Overlays for Blind and Low Vision Users. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '24)*. ACM, New York, NY, USA, Article 38, 12 pages. doi:10.1145/3663548.3675650
[24] Marco Manca, Vanessa Palumbo, Fabio Paternò, and Carmen Santoro. 2023. The Transparency of Automatic Web Accessibility Evaluation Tools: Design Criteria, State of the Art, and User Perception. *ACM Trans. Access. Comput.* 16, 1, Article 3 (March 2023), 36 pages. doi:10.1145/3556979
[25] Jennifer Mankoff, Devva Kasnitz, L. Jean Camp, Jonathan Lazar, and Harry Hochheiser. 2024. AI Must Be Anti-Ableist and Accessible. *Commun. ACM* 67, 12 (Nov. 2024), 40–42. doi:10.1145/3662731
[26] Jennifer Mankoff, Kelly Avery Mack, Jason Wiese, Kirk Andrew Crawford, and Foad Hamidi. 2023. A11yFutures: Envisioning the Future of Accessibility Research. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '23)*. ACM, New York, NY, USA, Article 107, 4 pages. doi:10.1145/3597638.3615652
[27] Elena Marcos. 2018. Running my First Design Sprint. https://medium.com/ux-planet/running-my-first-design-sprint-d2174293d93a
[28] Steven McDaniel and Minhaz F. Zibran. 2024. Improving Source Code with Assistance from AI - A Pilot Case Study with ChatGPT. In *Proceedings of the 7th International Conference on Information and Computer Technologies (ICICT '24)*. IEEE, USA, 332–337. doi:10.1109/ICICT62343.2024.00060
[29] Meredith Ringel Morris. 2020. AI and Accessibility. *Commun. ACM* 63, 6 (May 2020), 35–37. doi:10.1145/3356727
[30] Meredith Ringel Morris. 2024. Prompting Considered Harmful. *Commun. ACM* 67, 12 (2024), 28–30. doi:10.1145/3673861
[31] Peya Mowar. 2024. Accessibility in AI-Assisted Web Development. In *Proceedings of the 21st International Web for All Conference (W4A '24)*. ACM, New York, NY, USA, 123–125. doi:10.1145/3677846.3679054
[32] Peya Mowar, Yi-Hao Peng, Aaron Steinfeld, and Jeffrey P. Bigham. 2024. Tab to Autocomplete: The Effects of AI Coding Assistants on Web Accessibility. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '24)*. ACM, New York, NY, USA, Article 106, 6 pages. doi:10.1145/3663548.3688513
[33] Peya Mowar, Yi-Hao Peng, Jason Wu, Aaron Steinfeld, and Jeffrey P Bigham. 2025. CodeA11y: Making AI Coding Assistants Useful for Accessible Web Development. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, New York, NY, USA, Article 45, 15 pages. doi:10.1145/3706598.3713335
[34] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help with Code Understanding. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (ICSE '24)*. ACM, New York, NY, USA, Article 97, 13 pages. doi:10.1145/3597503.3639187
[35] National Informatics Centre, Government of India. 2024. Guidelines for Indian Government Websites and Apps. https://guidelines.india.gov.in
[36] Pavel Okopnyi, Oda Elise Nordberg, and Frode Guribye. 2024. Against Generative UI. In *Proceedings of the Halfway to the Future Symposium (HttF '24)*. ACM, New York, NY, USA, Article 12, 4 pages. doi:10.1145/3686169.3686184
[37] OpenAI. 2024. ChatGPT. https://chat.openai.com
[38] Achraf Othman, Amira Dhouib, and Aljazi Nasser Al Jabor. 2023. Fostering Websites Accessibility: A Case Study on the Use of the Large Language Models ChatGPT for Automatic Remediation. In *Proceedings of the 16th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '23)*. ACM, New York, NY, USA, 707–713. doi:10.1145/3594806.3596542
[39] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590 https://arxiv.org/abs/2302.06590
[40] Savvas Petridis, Michael Terry, and Carrie J. Cai. 2024. PromptInfuser: How Tightly Coupling AI and UI Design Impacts Designers' Workflows. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (DIS '24)*. ACM, New York, NY, USA, 743–756. doi:10.1145/3643834.3661613
[41] Helen Petrie and Christopher Power. 2012. What Do Users Really Care About? A Comparison of Usability Problems Found by Users and Experts on Highly Interactive Websites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2107–2116. doi:10.1145/2207676.2208363
[42] 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 (CHI '12)*. ACM, New York, NY, USA, 433–442. doi:10.1145/2207676.2207736
[43] Riglys and Jam. 2018. Designing for Accessibility - The Design Sprint that Worked Great and All the Key Takeaways. https://uxdesign.cc/designing-for-accessibility-the-design-sprint-that-worked-great-and-all-the-key-takeaways-f97c6657eddf
[44] Shreya Shankar, J.D. Zamfirescu-Pereira, Bjoern Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. ACM, New York, NY, USA, Article 131, 14 pages. doi:10.1145/3654777.3676450
[45] Auste Simkute, Lev Tankelevitch, Viktor Kewenig, Ava Elizabeth Scott, Abigail Sellen, and Sean Rintel. 2024. Ironies of Generative AI: Understanding and Mitigating Productivity Loss in Human-AI Interaction. *Int. Journal of Human-Computer Interaction* 41, 5 (2024), 2898–2919. doi:10.1080/10447318.2024.2405782
[46] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé F. Bissyandé. 2023. Is ChatGPT the Ultimate Programming Assistant – How Far Is It? https://arxiv.org/abs/2304.11938
[47] UK Government. 2018. The Public Sector Bodies (Websites and Mobile Applications) (No. 2) Accessibility Regulations. https://www.legislation.gov.uk/uksi/2018/952/contents/made
[48] United States Access Board. 1998. Section 508 of the Rehabilitation Act. https://www.section508.gov
[49] United States Congress. 2008. Americans with Disabilities Act Amendments Act of 2008. https://www.congress.gov/bill/110th-congress/senate-bill/3406
[50] Gregg Vanderheiden and Crystal Yvette Marte. 2024. Will AI Allow Us to Dispense with All or Most Accessibility Regulations?. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. ACM, New York, NY, USA, Article 571, 9 pages. doi:10.1145/3613905.3644059
[51] Jane Vilgats. 2024. A Guide to Inclusive and Accessible Design Sprints. https://accessibility.konnektis.com/blog/a-guide-to-inclusive-and-accessible-design-sprints
[52] WebAIM. 2023. The WebAIM Million: The 2023 Report on the Accessibility of the Top 1,000,000 Home Pages. https://webaim.org/projects/million/2023
[53] WebAIM. 2024. The WebAIM Million: The 2024 Report on the Accessibility of the Top 1,000,000 Home Pages. https://webaim.org/projects/million/2024