Accessible Web Content Generation Using LLMs: An Empirical Study on Prompting Strategies and Template-Guided Remediation

Guillermo Vera-Amaro, and José Rafael Rojano-Cáceres

***Abstract*—Web accessibility remains a persistent challenge, particularly for visually impaired users who rely on screen readers. This study investigates the potential of large language models (LLMs) to remediate accessibility issues only through structured prompt engineering without accessibility specialization. We evaluate GPT-4o and Gemini 2.0 Flash across 20 variants from two websites using different input formats (HTML, Markdown) and template-guided strategies. Outputs were assessed with automated tools, token efficiency metrics, and manual evaluations with experts and blind users. Results show an average Lighthouse score of 93.25, with WAVE errors reduced by 92.85%. Usability evaluation yielded an average success rate of 95.83% on completed tasks, with accuracy values reaching up to 0.86. GPT-4o demonstrated greater token efficiency, while Gemini produced more visually dynamic outputs. Certain violations persisted, confirming the need for human-in-the-loop validation. Overall, findings suggest that effectively guided LLMs can streamline remediation and foster more inclusive web experiences.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9994

***Index Terms*—Web accessibility; Large Language Models; Web scraping; Prompt engineering; GPT, Gemini; Blind users.**

I. Introduction

WEB accessibility remains a persistent and critical challenge in modern web development. Despite increased awareness and regulatory frameworks such as the Web Content Accessibility Guidelines (WCAG) [1], according to the 2025 WebAIM analysis, 95% of the top one million websites exhibit accessibility errors [2]. Notably, while websites with fewer errors are slowly improving, those with more issues have deteriorated over time. These issues impact users who rely on assistive technologies such as screen readers to navigate and access content online [3]. Globally, over 2.2 billion people live with some form of visual impairment, including approximately 39 million individuals who are blind [4]. These figures highlight the gap between existing standards and current development practices.

In parallel, the rise of large language models (LLMs) has introduced new possibilities for code generation, raising hopes that artificial intelligence could contribute to solving long-standing issues in software engineering [5], including accessibility [6]. When guided by well-crafted prompts, these models can address violations such as missing alternative text, unlabeled form elements, and incorrect semantic hierarchy [7]. However, recent findings suggest that LLM-generated code does not consistently comply with accessibility standards by default, often replicating common design flaws found in the training data [8]. Moreover, while automated evaluations have commonly been used in prior studies [9], recent research highlights the importance of combining tool-based assessments with expert review to obtain a more holistic understanding of accessibility performance [10].

In this context, the present study introduces an applied approach to improving web accessibility through generative AI. The method is grounded in a modular workflow that transforms inaccessible content into a born-accessible alternative [11], combining web scraping, content adjustment, prompt engineering, template-based generation, and accessibility evaluation. The main contributions of this study are:

- Evaluation of LLM-based remediation for visual accessibility issues.
- Comparison of prompt strategies with and without templates across HTML and Markdown inputs.
- Empirical assessment through automated tools, expert reviews, and blind-user evaluations.
- Analysis of token usage and cost in remediation tasks.
- A replicable pipeline to support future research in generative AI and accessibility.

The paper is organized as follows. Section II introduces key concepts. Section III reviews related work. Section IV describes the research design. Section V outlines the experimental setup. Section VI presents the findings. Section VII discusses the implications, and Section VIII presents conclusions.

II. Background and Motivation

Web accessibility refers to the design of websites that are usable by individuals with disabilities [12], particularly those with visual impairments. WCAG tools can detect syntactic issues such as missing alt text [13], but deeper semantic accessibility—such as interpreting page hierarchy or ensuring correct use of landmarks—often requires human evaluation with assistive technologies like screen readers [14]. True accessibility goes beyond conformance checks, requiring a combined approach of automated analysis and user-centered design [15]. To enable such approaches, web scraping provides a way to programmatically extract structured content from websites [16]. This process has become particularly relevant for accessibility research, as it enables isolating page components to be remediated or tested in structured workflows [17]. Once extracted, this content can be processed through generative artificial intelligence, which has shown the ability to synthesize

The associate editor coordinating the review of this manuscript and approving it for publication was Ruth Aguilar (*Corresponding author: Guillermo Vera-Amaro*).

This work is supported by SECIHTI under scholarship CVU: 1351657.

Guillermo Vera-Amaro, and J. R. Rojano-Cáceres are with School of Statistics and Informatics, Universidad Veracruzana, Xalapa, Veracruz, México (e-mails: gvera@uv.mx, and rrojano@uv.mx).

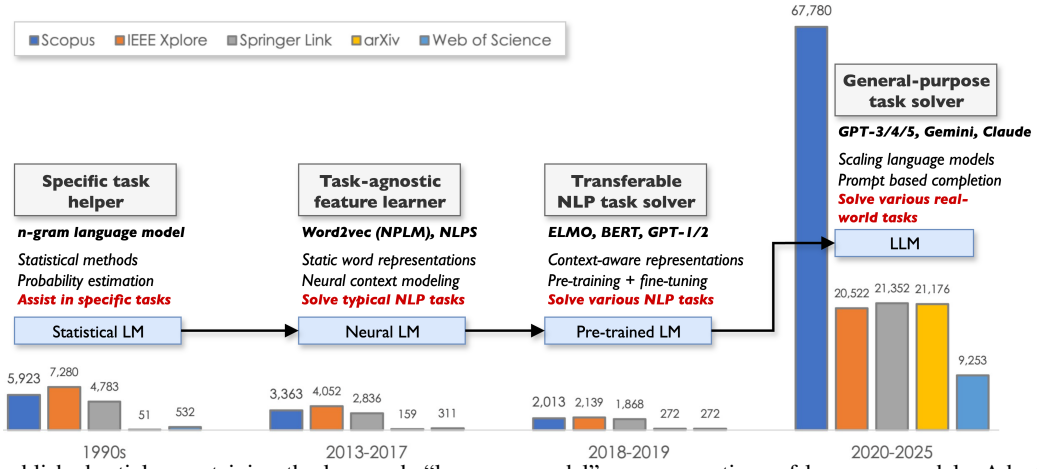

Fig. 1. Trend of published articles containing the keywords "language model" over generations of language models. Adapted from [24].

novel and context-aware outputs [18]. Unlike rule-based systems, generative approaches rely on probabilistic inference to adapt dynamically to diverse tasks [19]. Transformer-based models have demonstrated strong performance in generating structured text, enabling applications from educational tools to the design of accessible interfaces [20].

Large language models (LLMs) represent a subset of generative AI that is particularly suited to this purpose. Pre-trained on massive corpora, these models can perform a wide range of tasks under zero-shot and few-shot settings, including restructuring HTML code or correcting accessibility barriers [19]. However, they also present challenges, as they may replicate training biases, produce inconsistent outputs, or overlook WCAG-specific requirements [8]. Prompt engineering has emerged as a key practice to address these issues. By designing precise instructions or providing structural scaffolding, prompts can reduce ambiguity and improve the quality of generated outputs [19]. This approach has proven effective in accessibility-related tasks, such as repairing missing alternative text or enforcing semantic markup [21].

Although WCAG provides a solid normative foundation, its implementation remains challenging. Automated tools frequently miss contextual issues [13], while manual evaluations are time-consuming and require expert knowledge [22]. These limitations have driven growing interest in generative AI, particularly LLMs, to support accessibility remediation through automation [23]. Therefore, there is a need to retrofit [11] inaccessible web pages using a modular and explainable workflow that coordinates input extraction, prompt-based generation, and accessibility validation, making the process scalable, transparent, and maintainable.

III. Related Work

As illustrated in Fig. 1, language models have evolved considerably over four generations [24], yet their use in web accessibility remediation is relatively recent [8], [21], [25]. Results indicate that LLMs can correct violations detected by tools such as WAVE, producing results comparable to manual audits [9]. Reported success rates often exceed 70% in fixing accessibility issues in both self-generated and external code, although semantic aspects remain inconsistent [6]. Performance improves when structured prompts or visual cues are included, especially for tasks like contrast analysis and landmark detection [7]. Evaluation studies confirm that LLM-based scripts can outperform conventional tools in detecting WCAG success criteria that usually require manual inspection, identifying overlooked issues with high accuracy [10]. Comparative analyses also show that effectiveness depends strongly on prompt design and task specificity [26], underscoring the importance of prompt engineering for accessibility tasks.

Parallel work has examined web scraping as a source for training or guiding LLMs. Some studies propose leveraging structural patterns to enhance prompt design [27], while others highlight the risk of losing critical semantics when converting HTML into plain text [28]. Concerns have also been raised about quality degradation from training on unfiltered scraped data, stressing the need for careful dataset curation.

Unlike other domains where generative AI has been extensively studied, accessibility research remains scarce and exploratory [25]. Empirical validation is still limited, and challenges arise from the complexity of generating valid HTML compared to more constrained programming languages [29].

Key takeaway: To the best of our knowledge, this is the first study to systematically evaluate LLM-based accessibility remediation via structured prompt engineering, analyzing input formats, template usage, and token consumption, and triangulating results with automated, expert, and real-user validations.

IV. Methodology

To ensure methodological rigor and reproducibility, a five-phase methodology was established, as illustrated in Fig. 2. The design follows both practical and analytical criteria, aligned with the research objective of evaluating whether LLMs can generate accessible web content through prompt-based remediation, and examining the influence of input format, template guidance, and token efficiency on accessibility outcomes.

A. Research Questions

This experiment was designed to address the following research questions:

- **RQ1.** Can non-specialized LLMs generate accessible web outputs?
- **RQ2.** Does the input format (HTML or Markdown) affect the accessibility quality?

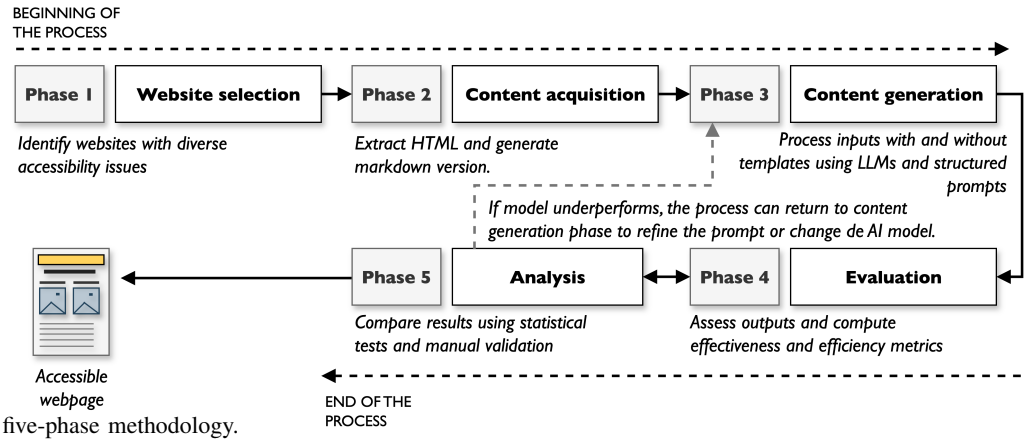

Fig. 2. Proposed five-phase methodology.

- **RQ3.** Does providing a structured template enhance accessibility outcomes?
- **RQ4.** Is there a relationship between generation cost (tokens) and achieved accessibility?

V. Experimental Setup

This section presents the experimental setup, evaluation metrics and statistical analysis methods.

A. Website Selection

Two websites were selected for evaluation, as summarized in Table I. Although the sample is small, it generates 20 versions to evaluate, consistent with the median of 13 reported in recent accessibility studies [25]. The goal is to test whether generative AI can improve web accessibility using only a prompt-based workflow without accessibility expertise. Both sites present common issues in navigation, multimedia, and forms, and include a manual version, corrected to meet WCAG requirements with automated tool verification.

The selected sites represent controlled cases, yet broader generalization would only require adapting templates to the type of content being remediated. For example, if the input is an HTML page built with a dynamic frontend framework, a corresponding dynamic template should be applied. In this study, because the target websites are static university pages, a Bootstrap-based template was prepared to capture their typical structure.

B. Content Acquisition

To retrieve web content while preserving structural integrity, five scraping strategies were evaluated:

- BeautifulSoup, a traditional parser for static HTML content subsequently converted into Markdown using the markdownify library.
- Playwright, capable of rendering JavaScript-driven pages and supporting content snapshotting.
- Jina Reader, optimized for transforming websites into clean, LLM-friendly Markdown.
- Crawl4AI, an open-source semantic crawler preserving metadata and accessibility structures.
- Firecrawl, a commercial alternative to Crawl4AI for generating LLM-ready datasets.

1) Comparative Testing: A comparative analysis was performed to evaluate content completeness, structural fidelity, and processing efficiency across scraping methods. As shown in Table II, Crawl4AI consistently achieved the best balance between output quality and processing time. Due to its support for dynamic content and Markdown extraction, it was selected for all subsequent experiments.

C. Content Generation

The extracted content was processed in two formats: as raw HTML and later converted into a more LLM-friendly Markdown representation [30].

1) LLMs Selection: The rationale for selecting GPT-4o and Gemini 2.0 Flash is grounded in both literature and practical considerations. A recent review on AI and accessibility [25] found that GPT-based models were the most frequently adopted (54% of studies), followed by Gemini (16%), which is consistent with the mapping study [31]. Since the goal of this study is not to determine which LLM is superior but rather to assess whether LLMs can remediate web accessibility, the two most widely used models were selected to provide a representative and comparable baseline. While internal differences in parameter count, training data, and token processing may exist, both GPT-4o and Gemini 2.0 Flash belong to the latest generation of proprietary frontier models, offering competitive multimodal and large-context capabilities.

2) Prompt Engineering Strategies: Effective prompt engineering is essential to guide LLMs in generating accessible HTML. Prior work shows that structured prompts improve performance in correcting accessibility violations [32] and organizing scraped content [33]. To reduce ambiguity and hallucinations, prompts included explicit WCAG-oriented framing and contextual cues, consistent with findings in [34].

Two strategies were tested: system-level instructions with WCAG guidance, and optional injection of accessibility-aware templates. These were applied via the OpenAI API using zero-shot and one-shot (using a template) prompting to assess the effect of contextual examples and structural scaffolding on accessibility outcomes [35].

a) System Instruction: The system message specified strict accessibility requirements and output format expectations, focusing on evaluating, correcting, and regenerating HTML

TABLE I
DESCRIPTION OF EVALUATED WEBSITES

Id	Description	URL
Site 1	University homepage created by the University of Washington to illustrate common accessibility issues.	Visit
Site 2	Updated version of Site 1 adapted with Bootstrap components to improve accessibility, maintained by Duke University.	Visit

TABLE II
COMPARISON OF WEB SCRAPING TECHNIQUES

Site	Method	Chars	Time	Markdown support	Dynamic content
Site 1	BeautifulSoup	10,367	1.95	N	N
	Playwright	12,237	7.35	N	Y
	Jina Reader	12,252	3.68	Y	Y
	Firecrawl	11,083	8.61	Y	Y
	Crawl4AI	**12,393**	**4.58**	**Y**	**Y**
Site 2	BeautifulSoup	12,281	3.44	N	N
	Playwright	16,292	3.20	N	Y
	Jina Reader	16,307	2.10	Y	Y
	Firecrawl	11,180	7.72	Y	Y
	Crawl4AI	**16,454**	**2.49**	**Y**	**Y**

according to WCAG 2.2 standards and following the techniques for instruction-based prompting as outlined in [32].

System Message: Accessibility correction

Identity
* You are a helpful assistant expert on web accessibility WCAG that evaluates and corrects HTML code.
* You will be given code and you will analyze it.

Instructions
1. You will create a new webpage from input code but WCAG accessible.
2. Revise color contrast, alt text, ARIA labels, heading levels, and use semantic HTML elements.
3. If images, scripts, or CSS files use relative URLs, convert them to absolute URLs preserving the relative path structure.
4. Include all required scripts in the `<head>` section.
5. Inline all critical CSS rules within the `<head>` section.
6. Detect the language and set the appropriate `lang` attribute.
7. Ensure the generated HTML is complete and WCAG accessible.
8. You will double check every step you do.
9. You will provide a numbered list of the improvements you made.
10. The HTML code must be a complete HTML code, not a fragment.
11. Return the result as a JSON object: {Procedure: str, HTML: str}.
12. If you find any error in the JSON object, you will correct it.

b) Template-based Prompt Injection: A Bootstrap-based accessibility template was added to the conversation context, providing a semantic and responsive base structure for the LLM to adapt. This template-guided prompting approach has been shown to enhance structural fidelity and semantic consistency in related domains [33].

User Message: Template injection

The following is a template to use as a base for the HTML code you will generate. Use Bootstrap classes to make the page responsive and accessible:

```
<html lang="en">
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet" href="://bootstrap.css">
<title>{{title}}</title>
</head>
<body>
<main class="container">
{{content}}
</main>
</body>
</html>
```

Generated Variants: For each model and site, four variants were produced: (1) raw HTML, (2) HTML with template, (3) Markdown, and (4) Markdown with template. This yielded eight versions per site, enabling systematic comparative analysis.

D. Evaluation

To assess the accessibility and usability of the generated web content, we applied a multi-layered evaluation strategy combining automated tools, expert heuristic reviews, and real-user validation. This triangulated approach was designed to capture both conformance with technical standards and practical challenges faced by blind users.

1) Automated Evaluation: Based on the recent literature review of AI in web accessibility [25], WAVE emerged as the most frequently used tool, reported in 29% of the studies. To complement it with a browser-integrated solution that also provides a single aggregated accessibility score, we included Google Lighthouse, which appeared in 4% of the studies. Together, these tools enabled both detailed error-level analysis and an overall accessibility quality assessment.

2) Metrics: Accessibility was measured through WAVE errors (WCAG 2.2 violations such as missing labels or headings), contrast errors (insufficient color contrast), WAVE alerts (potential issues requiring manual review), WAVE features (correctly implemented accessible elements), and the Lighthouse accessibility score as an overall indicator. Efficiency was evaluated by prompt tokens (input size), completion tokens (generated output), total tokens (combined prompt and output), and the estimated API cost in USD.

3) Manual Experts Evaluation: A manual evaluation was conducted using a structured heuristic method [36] to assess accessibility barriers for blind users [37].

4) Blind User Evaluation: Manual evaluation was performed with blind users through the Cognitive Walkthrough (CW) method [38]. The inclusion of this real-user validation aimed to complement automated and heuristic evaluations, ensuring that practical accessibility issues could also be identified.

E. Analysis

To compare accessibility and efficiency outcomes across models, input types, and prompting strategies, we applied a structured statistical analysis. Statistical significance was determined using a standard threshold of $p = 0.05$.

1) Descriptive Statistics: For each generation, we calculated mean, median, standard deviation, min, and max values for WAVE metrics, Lighthouse scores, token usage, and API cost.

2) Comparative Analysis: Pairwise comparisons were conducted across:

- Model: GPT-4o vs. Gemini 2.0 Flash
- Input format: HTML vs. Markdown
- Template use: with vs. without template

The Shapiro-Wilk test was applied to evaluate the normality of the data distributions. As most variables deviated significantly from normality ($p < 0.05$), Mann–Whitney U tests were used to assess differences. This non-parametric approach is appropriate for small sample sizes and skewed data.

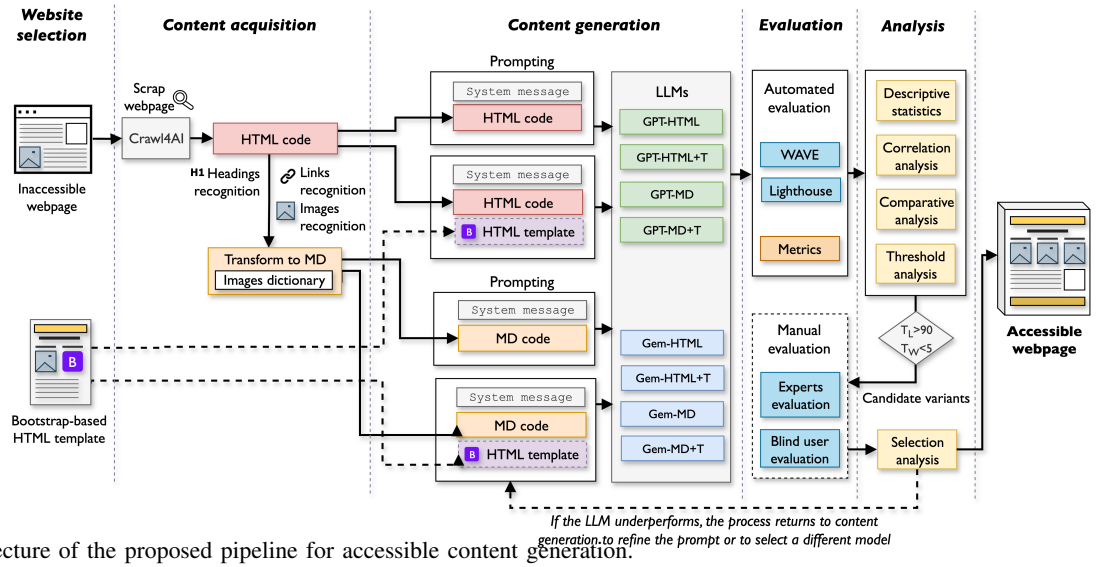

Fig. 3. Architecture of the proposed pipeline for accessible content generation.

3) Correlation Analysis: Spearman's rank correlation coefficient (ρ) was used to explore relationships between continuous variables, including total token usage, WAVE errors, and Lighthouse scores. This method was selected due to the non-parametric and ordinal nature of the data.

4) Threshold Analysis: To support a deeper error analysis and ensure that only promising variants advance to expert and user validation, a threshold model was defined. This mechanism allowed us to filter out underperforming generations and focus manual evaluation efforts on the most accessible outputs. Threshold analysis can be represented mathematically using (Equation 1).

$$y = \begin{cases} 0 & \text{if } x < T \\ 1 & \text{if } x \geq T \end{cases} \tag{1}$$

where y is the decision outcome, x is the observed metric, and T is the predefined threshold. Two complementary thresholds were established: Lighthouse accessibility score T_L and WAVE error count T_W. We set $T_L = 90$, as scores equal to or above this value are considered *good* accessibility by Google Lighthouse. For WAVE errors, we defined $T_W = 5$, since values below this level indicate a low incidence of critical WCAG violations.

F. Pipeline Architecture for Accessibility Remediation

Finally, the full pipeline of the experiment is illustrated in Fig. 3, summarizing the five core phases of the methodology. The architecture supports multiple configurations depending on the input format, the use of templates, and the chosen LLM. Therefore, this setup allows for the evaluation of multiple generative combinations per site. In case of model underperformance, the process can return to prompt refinement or model selection. This loop supports an iterative and resilient remediation workflow grounded in measurable outcomes.

VI. Results

This section summarizes the experimental outcomes based on the methodology in Section V.

A. Dataset Summary

Table III presents the resulting dataset [39], including columns for Lighthouse scores, WAVE metrics, and processing costs for all variants across two websites, as well as rows for the original and manually corrected versions. The experiments combined two LLMs (GPT-4o and Gemini 2.0 Flash) with different prompting strategies (HTML vs. Markdown, with and without templates).

B. Lighthouse Accessibility Improvements

Generated pages achieved an average Lighthouse score of 93.25. Site 1 scored higher on average (98.13 vs. 88.37). As shown in Fig. 4, both models produced substantial gains, but Mann–Whitney U tests revealed no significant differences between models scores across both sites ($p = 0.7089$). Similarly, template-based variants scored higher on average, but the difference only was statistically significant on Site 2 ($p_{site1} = 0.1138$, $p_{site2} = 0.0294$).

C. WAVE Metrics

Fig. 5 shows the improvements across all WAVE evaluation metrics after LLM-based remediation. Relative gains were computed using the original page as baseline, applying (Equation 2) for error-related metrics (errors, contrast issues, alerts).

$$\text{Improvement} = \frac{\text{Original} - \text{Generated Mean}}{\text{Original}} \times 100 \tag{2}$$

For Site 1, WAVE errors decreased by 96.1%, contrast errors by 59.4%, and alerts by 46.6%. For Site 2, reductions reached 89.6%, 51.2%, and 60.2%, respectively, yielding an average error decrease of 92.85%. Across both sites, Mann–Whitney U tests revealed that Gemini generated significantly fewer errors ($p = 0.0435$) and fewer alerts ($p = 0.0151$), with no significant difference in contrast errors ($p = 0.4870$).

TABLE III
SUMMARY OF THE DATASET OBTAINED FROM EXPERIMENTS

Site	ID	Source	Input type	Template	Lighthouse Score	WAVE Metrics Errors	Contrast errors	Alerts	Features	Prompt tokens	Completion tokens	Total tokens	Cost
Site 1	V01	Original	-	-	69	16	4	11	1	-	-	-	
	V01	Manual	-	-	100	0	1	3	59	-	-	-	-
	V03	GPT-4o	HTML	No	96	1	0	9	19	3,861	4,306	8,167	0.020418
	V04	GPT-4o	HTML	Yes	100	1	3	6	17	6,790	3,345	8167	0.025337
	V05	GPT-4o	Markdown	No	100	1	0	6	18	3,914	3,642	8167	0.018890
	V06	GPT-4o	Markdown	Yes	100	1	2	6	17	6,785	3,311	8167	0.025240
	V07	Gemini	HTML	No	96	0	0	10	17	3,890	8,414	8167	0.030760
	V08	Gemini	HTML	Yes	100	1	3	5	26	6,769	7,831	8167	0.036500
	V09	Gemini	Markdown	No	96	0	0	1	25	3,943	10,118	8167	0.035153
	V10	Gemini	Markdown	Yes	97	0	5	4	22	6,764	7,269	8167	0.035083
Site 2	V11	Original	-	-	58	24	21	11	3	-	-	-	
	V12	Manual	-	-	100	0	0	0	24	-	-	-	
	V13	GPT-4o	HTML	No	87	4	6	7	23	4,475	5,344	8167	0.024547
	V14	GPT-4o	HTML	Yes	93	4	4	5	15	7,404	3,493	8167	0.027243
	V15	GPT-4o	Markdown	No	75	5	3	8	19	4,528	5,167	8167	0.024238
	V16	GPT-4o	Markdown	Yes	96	2	5	7	15	7,399	3,081	8167	0.026200
	V17	Gemini	HTML	No	80	2	33	4	28	4,583	10,746	8167	0.038323
	V18	Gemini	HTML	Yes	93	1	5	0	24	7,462	7,996	8167	0.038645
	V19	Gemini	Markdown	No	88	2	23	1	26	4,636	10,443	8167	0.037698
	V20	Gemini	Markdown	Yes	95	0	3	3	24	7,457	6,496	8167	0.034882

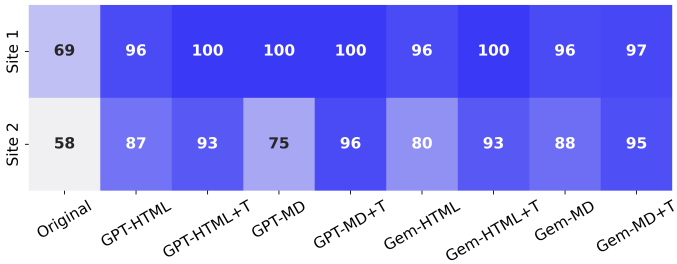

Fig. 4. Comparison of Lighthouse accessibility scores per site.

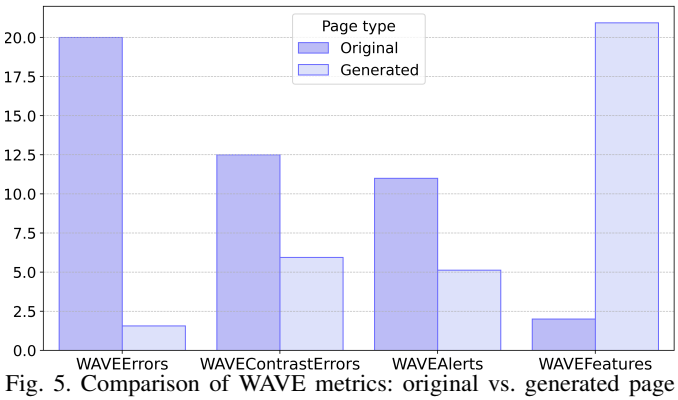

Fig. 5. Comparison of WAVE metrics: original vs. generated page averages.

D. Token Usage

As shown in Fig. 6, the average consumption was 11,979 tokens (GPT-4o: 9,606; Gemini: 14,352). Mann–Whitney U tests confirmed significant differences in total tokens ($p = 0.0286$) across both sites, highlighting GPT-4o's lower usage. However, Spearman tests showed no significant correlations between tokens ($\rho = -0.2196$, $p = 0.4138$) and either Lighthouse scores or WAVE errors ($\rho = -0.2276$, $p = 0.3965$).

E. Correlation Analysis between Tools

Spearman tests revealed moderate negative correlations between WAVE errors ($\rho = -0.5843$, $p = 0.0175$) and contrast errors ($\rho = -0.5643$, $p = 0.0228$) with Lighthouse scores. This indicates that lower critical WAVE violations are moderately associated with higher Lighthouse performance.

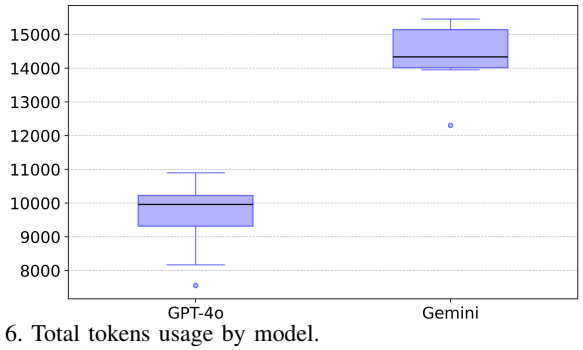

Fig. 6. Total tokens usage by model.

F. Threshold analysis

To focus expert and user validation on promising outputs, we applied a dual threshold: Lighthouse accessibility score $T_L \geq 90$ and WAVE errors $T_W \leq 5$. As shown in Table IV, variants meeting both criteria were advanced, while the rest were discarded. Overall, 12/16 variants (75%) passed. Selection was balanced across models (GPT-4o: 6/8; Gemini 2.0: 6/8) and strongly favored template use (with template: 8/8; without template: 4/8). All Site 1 variants passed (8/8), whereas only 4/8 Site 2 variants met the threshold. To benchmark these thresholds, original and manual pages were included.

G. Manual Experts Evaluation

1) Quantitative Results: Following the evaluation protocol in [37], a manual review was conducted using the Barrier Walkthrough (BW) method [36], which classifies barriers by impact and persistence, assigning severity levels from None to Critical. The evaluation covered 24 barriers (see Table V) and results for each variant are summarized in Table VI. Expert scores confirmed consistent improvement

over the original. Gemini variants achieved better scores, with the best case being Gem-MD+T. By contrast, GPT variants—though meeting thresholds—still accumulated up to five critical barriers, highlighting that expert inspection remains essential to uncover residual problems not penalized by automated tools.

2) Qualitative Observations: Despite high automated scores, functional issues were found:

- Missing submenu navigation mostly without templates.
- Visual inconsistencies showing misaligned layouts.
- JS issues with broken carousels or form missing behavior.
- Image rendering due to broken paths.

H. Blind User Evaluation

To provide a human-centered evaluation with real screen-reader users, we applied the Cognitive Walkthrough (CW) method [38]. This task-based usability inspection evaluates learnability and semantic accessibility involving blind participants directly performing predefined tasks. Unlike automated tools, this method captures experiential barriers encountered during screen-reader interaction.

1) User Demographics: The evaluation included four participants (see Table VII): two evaluators with visual impairments, one facilitator who guided the session, and one timekeeper who recorded task completion times and notes (see Fig. 7).

2) Task Definition: As shown in Table VIII, representative tasks were defined to evaluate usability. Evaluators could skip a task if they were unable to complete it within the 120-second limit, which was derived from expert evaluation.

3) Results: Evaluators performed tasks with little external assistance, especially the P2 evaluator who relied mainly on tabs and arrow keys and less on NVDA commands, while P1 used a combination of both. Tasks were completed with an average of 12.92 seconds per task. P1 required 20.22 minutes to complete 90/96 (94%) tasks, while P2 required 29.46 minutes to complete 80/96 (83%), resulting in an overall success rate of 88.54%, which increases to 95.83% for only LLM-generated variants. The manual variant—marked as WCAG conformant by automated tools—was not the most usable (see Table IX). This highlights the well-documented gap between conformance and usability [13], [15], since pages that are technically WCAG-conformant may overload screen readers, while simplified generated structures may inadvertently improve navigation efficiency. This demonstrates that WCAG conformance does not mean perfect usability.

TABLE IV
VARIANTS MEETING THE THRESHOLD

ID	Variant	Lighthouse	WAVE Errors
V01	Original (Site 1)	69	16
V02	Manual (Site 1)	100	1
V03	GPT-HTML (Site 1)	96	1
V04	GPT-HTML+T (Site 1)	100	1
V05	GPT-MD (Site 1)	100	1
V06	GPT-MD+T (Site 1)	100	1
V07	Gem-HTML (Site 1)	96	0
V08	Gem-HTML+T (Site 1)	100	1
V09	Gem-MD (Site 1)	96	0
V10	**Gem-MD+T (Site 1)**	**97**	**0**
V11	Original (Site 2)	58	24
V12	Manual (Site 2)	100	0
V14	GPT-HTML+T (Site 2)	93	4
V16	GPT-MD+T (Site 2)	96	2
V18	Gem-HTML+T (Site 2)	93	1
V20	**Gem-MD+T (Site 2)**	**95**	**0**

TABLE V
LIST OF EVALUATED ACCESSIBILITY BARRIERS

ID	Barrier	Principle	Criteria
B01	Rich images lacking text	Perceivable	1.1.1
B02	Color is necessary	Perceivable	1.4.1, 1.4.3
B03	Opaque or missing objects	Perceivable	1.1.1
B04	Functional images lacking text	Perceivable	1.1.1, 1.4.5
B05	Tables with no structural	Perceivable	1.3.1, 1.3.2
B06	Forms that are badly linearized	Perceivable	1.3.2
B07	Broken functionality	Operable	2.1.1, 2.1.2
B08	Generic links	Operable	2.4.4, 2.4.9
B09	Ambiguous links	Operable	2.4.4, 2.4.9
B10	Dynamic menus in JS or CSS	Operable	2.1.1, 2.1.3, 4.1.2
B11	Mouse events	Operable	2.1.1, 2.1.3
B12	Keyboard traps	Operable	2.1.2
B13	Too many links	Operable	2.4.4, 2.4.9
B14	Forms with no LABEL tags	Operable	3.3.2, 1.3.1
B15	Non separated links	Operable	2.4.4, 2.4.9
B16	Skip links not implemented	Operable	2.4.1
B17	No keyboard shortcuts	Operable	2.1.1, 2.1.4
B18	No page headings	Operable	1.3.1, 2.4.6, 2.4.10
B19	Page without titles	Operable	2.4.2
B20	Difficulty locating buttons	Operable	2.4.3, 1.3.2
B21	Form with redirect	Understandable	3.2.2, 3.2.5
B22	New windows	Understandable	3.2.1, 3.2.5
B23	Language markup	Understandable	3.1.1, 3.1.2
B24	Dynamic changes	Robust	4.1.2, 4.1.3

TABLE VI
EXPERT HEURISTIC EVALUATION SCORES

ID	Variant	None	Minor	Significant	Critical
V01	Original (Site 1)	7	4	7	6
V02	Manual (Site 1)	21	0	2	1
V03	GPT-HTML (Site 1)	15	0	4	5
V04	GPT-HTML+T (Site 1)	16	2	2	4
V05	GPT-MD (Site 1)	18	0	3	3
V06	GPT-MD+T (Site 1)	17	1	3	3
V07	Gem-HTML (Site 1)	18	0	3	3
V08	Gem-HTML+T (Site 1)	17	2	3	2
V09	Gem-MD (Site 1)	20	0	2	2
V10	**Gem-MD+T (Site 1)**	**20**	**1**	**1**	**2**
V11	Original (Site 2)	7	3	6	8
V12	Manual (Site 2)	21	0	2	1
V14	GPT-HTML+T (Site 2)	17	2	2	3
V16	GPT-MD+T (Site 2)	15	4	2	3
V18	Gem-HTML+T (Site 2)	18	1	2	3
V20	**Gem-MD+T (Site 2)**	**21**	**1**	**1**	**1**

VII. DISCUSSION

A. RQ1. Model Outputs in Practice

Results revealed that without templates, GPT-4o tended to preserve source fidelity but retained several operability issues

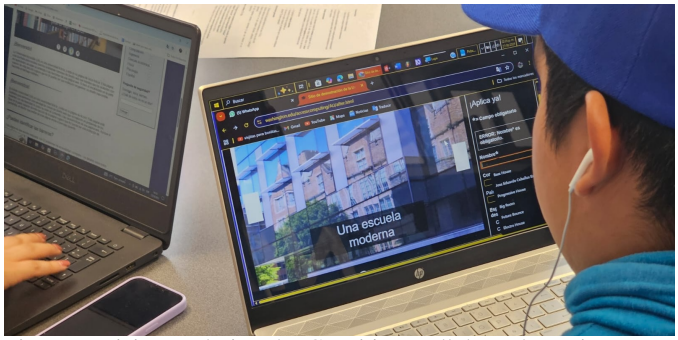

Fig. 7. Participants during the Cognitive Walkthrough session.

TABLE VII
PARTICIPANT DEMOGRAPHICS FOR BLIND USER EVALUATION

ID	Role	Gender	Sight Condition	Screen reader experience
P1	Evaluator	F	Blind	NVDA (4 years)
P2	Evaluator	M	Legally Blind	NVDA (1 year)
P3	Timekeeper	M	Sighted	-
P4	Facilitator	M	Sighted	-

TABLE VIII
DEFINED TASKS FOR COGNITIVE WALKTHROUGH METHOD

ID	Task description
T1	Find the *About* link under the main dropdown menu
T2	Localize the *Welcome* header
T3	Read the an item of the news carousel
T4	Localize the amount of CS students last year in *enrollments* table
T5	Verify proper input validation after a form submission
T6	Localize the *info/known issues* link

and sometimes removed interactive elements (e.g., dropdown menus), yielding pages that were only partially usable. Gemini, in contrast, applied WCAG-like patterns more proactively and produced visually refined layouts, but template-free runs occasionally introduced new contrast violations. With template guidance, both models converged toward desired behavior, with residual issues (mainly minor JS integration) persisting in a few cases.

1) Evaluation with Blind Users: CW showed that 8/12 generated variants achieved full task completion for both evaluators, whereas original and manual variants tended to be slower and in all cases, incomplete. The most difficult tasks were those requiring interacting with news carousel (T3) and table navigation (T4), taking longer times to complete. Results with blind users highlight that, beyond automated evaluation, real usage exposes navigation challenges that remain critical for everyday accessibility, reinforcing the need to include users with disabilities in the evaluation loop.

2) Accuracy and F1 Analysis: To benchmark evaluation layers, we defined a theoretical upper bound with zero BW barriers and per-task *ideal* times set to the 10th percentile of observed completion times among completed runs. With that reference, BW accuracy for a variant v was defined as the ratio of "None" barriers identified relative to the total number of barriers (24), following (Equation 3).

TABLE IX
COGNITIVE WALKTHROUGH SUMMARY RESULTS

ID	Variant	Total Avg (s)	Completion	% Success
V01	Original (Site 1)	Incomplete	4/12	33.33%
V02	Manual (Site 1)	Incomplete	9/12	75%
V03	GPT-HTML (Site 1)	Incomplete	11/12	91.67%
V04	GPT-HTML+T (Site 1)	Incomplete	11/12	91.67%
V05	GPT-MD (Site 1)	18.70	12/12	100%
V06	GPT-MD+T (Site 1)	16.44	12/12	100%
V07	Gem-HTML (Site 1)	16.39	12/12	100%
V08	Gem-HTML+T (Site 1)	Incomplete	10/12	83.33%
V09	Gem-MD (Site 1)	12.13	12/12	100%
V10	**Gem-MD+T (Site 1)**	**11.41**	**12/12**	**100%**
V11	Original (Site 2)	Incomplete	8/12	66.67%
V12	Manual (Site 2)	Incomplete	11/12	91.67%
V14	GPT-HTML+T (Site 2)	11.60	12/12	100%
V16	GPT-MD+T (Site 2)	Incomplete	10/12	83.33%
V18	Gem-HTML+T (Site 2)	9.22	12/12	100%
V20	**Gem-MD+T (Site 2)**	**7.46**	**12/12**	**100%**

TABLE X
ACCURACY AND F_1 OF COMPLETED-TASK VARIANTS

ID	Variant	BWAcc	BWF1	CWAcc	CWF1
V05	GPT-MD (Site 1)	0.75	0.86	0.40	0.57
V06	GPT-MD+T (Site 1)	0.71	0.83	0.45	0.62
V07	Gem-HTML (Site 1)	0.75	0.86	0.46	0.63
V09	Gem-MD (Site 1)	0.83	0.91	0.61	0.76
V10	**Gem-MD+T (Site 1)**	**0.83**	**0.91**	**0.65**	**0.79**
V14	GPT-HTML+T (Site 2)	0.71	0.83	0.56	0.71
V18	Gem-HTML+T (Site 2)	0.75	0.86	0.70	0.82
V20	**Gem-MD+T (Site 2)**	**0.88**	**0.93**	**0.86**	**0.93**

$$\mathrm{BW}_{Acc}(v) \;=\; \frac{\mathrm{None}(v)}{24} \tag{3}$$

Since the ideal has no barriers, Precision is always equal to 1, and the F_1 score reduces to a function of Recall with (Equation 4).

$$\mathrm{BW}_{F_1}(v) \;=\; \frac{2 \cdot \mathrm{BW}_{Acc}(v)}{1 + \mathrm{BW}_{Acc}(v)} \tag{4}$$

For CW, time-based accuracy is bounded in $[0, 1]$ as (Equation 5).

$$\mathrm{Acc}_i(v) \;=\; \begin{cases} \min\!\big(1,\; t_i^*/t_i(v)\big), & \text{if task } i \text{ completed} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Where $t_i(v)$ is the variant's completion time for task i and t_i^* is its ideal time (10th percentile). Variant-level CW accuracy is then the mean across N tasks, following (Equation 6).

$$\mathrm{CW}_{Acc}(v) \;=\; \frac{1}{N} \sum_{i=1}^{N} \mathrm{Acc}_i(v) \tag{6}$$

With the corresponding F_1 score defined with (Equation 7).

$$\mathrm{CW}_{F_1}(v) \;=\; \frac{2 \cdot \mathrm{CW}_{Acc}(v)}{1 + \mathrm{CW}_{Acc}(v)} \tag{7}$$

Table X shows that the most usable cases achieved CW_{Acc} of 0.76 on average (Site 1: 0.65; Site 2: 0.86). Gemini consistently outperformed GPT-4o on both sites, especially in Markdown with templates. Expert BW scores followed a similar trend, confirming convergence between expert and blind-user evaluations.

Key takeaway: As shown in Fig. 8a, GPT-4o reached slightly higher Lighthouse scores, while Gemini yielded more consistent results and stronger BW and CW metrics. All generations used identical parameters ($T = 0.1$), yet results suggest Gemini may have been trained with broader WCAG exposure and generates more visually refined layouts, whereas GPT-4o tends to prioritize fidelity but often omitting enhancements unless explicitly prompted.

B. RQ2. Effect of Input Format

Automated evaluations showed slightly higher values in favor of Markdown, but the difference from HTML was not significant ($p = 0.7089$). However, expert and user reviews highlighted practical contrasts: HTML preserved layouts but also inherited source barriers, while Markdown promoted cleaner structures yet was more volatile without templates.

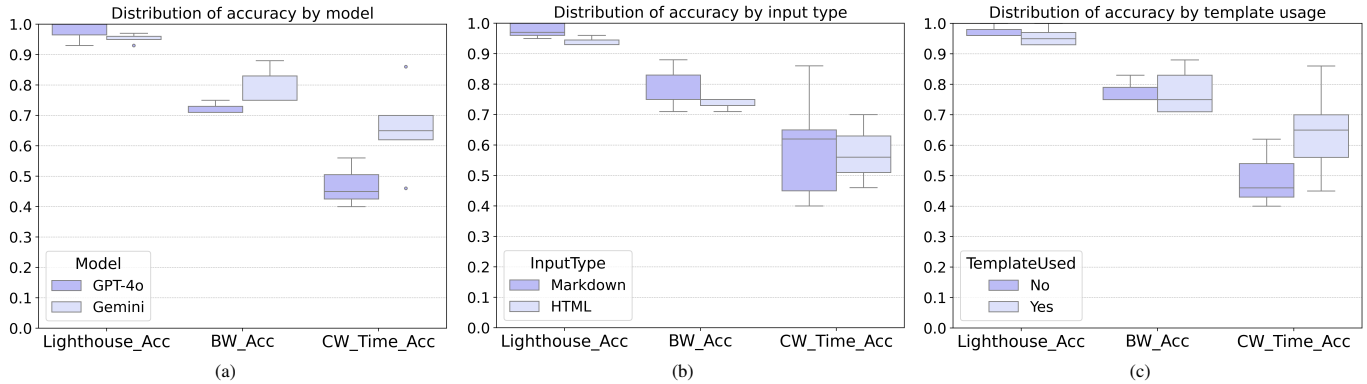

Fig. 8. Distribution of accuracy across evaluation methods (Lighthouse, BW, CW) grouped by (a) model, (b) input type, and (c) template usage.

With templates, Markdown variants achieved the best outcomes, with Gem-MD+T emerging as the top performer.

Key takeaway: While automated tools showed slightly input-related differences, practical evaluations in Fig. 8b, confirmed that Markdown with template guidance produced the most usable and reliable outputs, whereas raw HTML, although preserving spatial layout, tended to replicate original errors.

C. RQ3. Effect of Templates

Template guidance consistently improved results across all layers. While automated tools showed modest gains, template variants reached higher Lighthouse scores and fewer WAVE issues. Threshold analysis confirmed 8/8 template outputs passed, versus only half without templates. Expert BW further showed fewer Critical operability barriers (e.g., dynamic menus, broken functionality). Blind-user walkthroughs validated these trends, with Gem-MD+T achieving 100% task success and the best times.

Key takeaway: Templates proved essential for reliable remediation, indicating that structural guidance benefits usability even when Lighthouse scores remain high in both cases (see Fig. 8c).

D. RQ4. Generation Cost vs. Accessibility

Token usage alone was not a predictor of quality. It is true that Gemini consumed more tokens, and that template-guided variants achieved superior BW and CW, but Markdown further reduced token counts relative to HTML while maintaining better accessibility. These findings show that template structure and input optimization—not raw token volume—drive the balance between cost and conformance.

Key takeaway: Gemini consumed more tokens (see Fig. 6), yet achieved better results when guided by templates, while Markdown inputs reduced token cost without compromising results—highlighting the importance of prompt structure over raw token volume.

E. Limitations

Automated tools were used as proxy benchmarks for WCAG conformance, which could not fully capture real-world accessibility. This limitation was mitigated by incorporating a complementary tool and triangulating results with user-based validations. Only two LLMs were analyzed, which may limit generalizability; to reduce this risk, we selected the most advanced and widely used models in accessibility research [25], [31]. Two websites were selected for evaluation and, although the sample is small, it produced 20 variants for analysis, consistent with the median of 13 reported in recent accessibility evaluation studies [25]. The sites also provided manual version conformant to WCAG, which are rarely available at scale. Finally, static sites may not capture the complexity of dynamic real-world environments; however, the methodology was designed to remain extensible through template injection tailored to different website types, enabling broader adaptation.

VIII. Conclusions and Future Work

This study demonstrates the potential of LLMs to remediate web accessibility when guided by structured prompting strategies. By combining web scraping, prompt engineering, and evaluation layers—including blind users—, the approach achieved substantial improvements in WCAG conformance and usability with template-based prompts and Markdown input, consistently yielding the best results. The methodology provides a replicable foundation for scaling automated accessibility remediation. Future work could expand to dynamic and mobile-first sites, integrate other AI techniques for context enrichment (e.g., RAG), and study trade-offs between smaller models. Visual prompting (e.g., screenshots) may improve layout-sensitive tasks by preserving spatial relations without replicating HTML issues. Finally, future studies could investigate whether the threshold analysis can be enhanced to refine the selection process.

References

[1] S. L. Henry, “WCAG 2 Overview,” Mar. 2024. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/
[2] WebAIM, “The WebAIM Million,” Institute for Disability Research, Policy, and Practice. Utah State University, Tech. Rep., Mar. 2025. [Online]. Available: https://webaim.org/projects/million/
[3] ——, “Screen Reader User Survey #10 Results ,” Feb. 2024. [Online]. Available: https://webaim.org/projects/screenreadersurvey10/
[4] WHO, “World report on vision,” World Health Organization, Tech. Rep. 14, 2019. [Online]. Available: https://iris.who.int/bitstream/handle/10665/328717/9789241516570-eng.pdf?sequence=18

[5] A. Hemmat *et al.*, "Research directions for using LLM in software requirement engineering: a systematic review," *Frontiers in Computer Science*, vol. 7, p. 1519437, Mar. 2025, doi: 10.3389/fcomp.2025.1519437.

[6] W. Aljedaani *et al.*, "Does ChatGPT Generate Accessible Code? Investigating Accessibility Challenges in LLM-Generated Source Code," *Proceedings of the 21st International Web for All Conference*, pp. 165–176, May. 2024, doi: 10.1145/3677846.3677854.

[7] A. Ahmed *et al.*, "From Code to Compliance: Assessing ChatGPT's Utility in Designing an Accessible Webpage – A Case Study," Jan. 2025, arXiv preprint arXiv.2501.03572.

[8] K. S. Fuglerud *et al.*, "Exploring the Use of AI for Enhanced Accessibility Testing of Web Solutions," *Studies in health technology and informatics*, vol. 320, pp. 453–460, Nov. 2024, doi: 10.3233/SHTI241041.

[9] A. Othman, A. Dhouib, and A. Nasser Al Jabor, "Fostering websites accessibility: A case study on the use of the Large Language Models ChatGPT for automatic remediation," *ACM International Conference Proceeding Series*, pp. 707–713, Jul. 2023, doi: 10.1145/3594806.3596542.

[10] J.-M. López-Gil and J. Pereira, "Turning manual web accessibility success criteria into automatic: an LLM-based approach," *Universal Access in the Information Society*, vol. 24, pp. 837–852, Mar. 2025, doi: 10.1007/s10209-024-01108-z.

[11] J. Lazar, "A Framework for Born-Accessible Development of Software and Digital Content," *Lecture Notes in Computer Science*, pp. 333–338, 2023, doi: 10.1007/978-3-031-42293-5_32.

[12] S. L. Henry, "Introduction to Web Accessibility," Mar. 2024. [Online]. Available: https://www.w3.org/WAI/fundamentals/accessibility-intro/

[13] C. Power *et al.*, "Guidelines are only half of the story: Accessibility problems encountered by blind users on the Web," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 433–442, 2012, doi: 10.1145/2207676.2207736.

[14] M. Bajammal and A. Mesbah, "Semantic web accessibility testing via hierarchical visual analysis," *Proceedings - International Conference on Software Engineering*, pp. 1610–1621, May. 2021, doi: 10.1109/ICSE43902.2021.00143.

[15] G. Brajnik, "Beyond Conformance: The Role of Accessibility Evaluation Methods," in *Web Information Systems Engineering – WISE 2008 Workshops*. Springer Berlin Heidelberg, 2008, pp. 63–80, doi: 10.1007/978-3-540-85200-1_9.

[16] J. Wood and D. Joshi, "Conflict-RAG: Understanding Evolving Conflicts Using Large Language Models," in *2024 IEEE International Conference on Big Data (BigData)*. IEEE, Dec. 2024, pp. 5459–5467, doi: 10.1109/BigData62323.2024.10825676.

[17] I. Naing *et al.*, "A Reference Paper Collection System Using Web Scraping," *Electronics*, vol. 13, p. 2700, Jul. 2024, doi: 10.3390/electronics13142700.

[18] P. Acosta-Vargas *et al.*, "Generative Artificial Intelligence and Web Accessibility: Towards an Inclusive and Sustainable Future," *Emerging Science Journal*, vol. 8, pp. 1602–1621, Aug. 2024, doi: 10.28991/ESJ-2024-08-04-021.

[19] W. Alsakran and R. Alabduljabbar, "Exploring the Potential of LLMs and Attributed Prompt Engineering for Efficient Text Generation and Labeling," in *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*. IEEE, Nov. 2024, pp. 244–252, doi: 10.1109/FLLM63129.2024.10852475.

[20] H. Du *et al.*, "Interactive Rubric Generator for Instructor's Assessment Using Prompt Engineering and Large Language Models," in *2024 IEEE Frontiers in Education Conference (FIE)*. IEEE, Oct. 2024, pp. 1–9, doi: 10.1109/FIE61694.2024.10893448.

[21] G. Delnevo, M. Andruccioli, and S. Mirri, "On the Interaction with Large Language Models for Web Accessibility: Implications and Challenges," *Proceedings - IEEE Consumer Communications and Networking Conference, CCNC*, 2024, doi: 10.1109/CCNC51664.2024.10454680.

[22] P. Acosta-Vargas, L. Antonio Salvador-Ullauri, and S. Lujan-Mora, "A Heuristic Method to Evaluate Web Accessibility for Users with Low Vision," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2939068.

[23] Y. Chang *et al.*, "A Survey on Evaluation of Large Language Models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, Mar. 2024, doi: 10.1145/3641289.

[24] W. X. Zhao *et al.*, "A Survey of Large Language Models," Mar. 2025, arXiv preprint arXiv.2303.18223.

[25] G. Vera-Amaro and J. R. Rojano-Cáceres, "Towards accessible website design through artificial intelligence: A systematic literature review," *Information and Software Technology*, vol. 186, no. C, Oct. 2025, doi: 10.1016/j.infsof.2025.107821.

[26] C. Duarte *et al.*, "Expanding Automated Accessibility Evaluations: Leveraging Large Language Models for Heading-Related Barriers," in *Companion Proceedings of the 30th International Conference on Intelligent User Interfaces*. New York, NY, USA: ACM, Mar. 2025, pp. 39–42, 10.1145/3708557.3716329.

[27] A. Namoun *et al.*, "Web Design Scraping: Enabling Factors, Opportunities and Research Directions," *ICITEE 2020 - Proceedings of the 12th International Conference on Information Technology and Electrical Engineering*, pp. 104–109, Oct. 2020, doi: 10.1109/ICITEE49829.2020.9271770.

[28] E. Benavides-Astudillo *et al.*, "Comparative Study of Deep Learning Algorithms in the Detection of Phishing Attacks Based on HTML and Text Obtained from Web Pages," in *Communications in Computer and Information Science*, 2023, pp. 386–398, doi: 10.1007/978-3-031-24985-3_28.

[29] S. K. Oswal and H. K. Oswal, "Examining the Accessibility of Generative AI Website Builder Tools for Blind and Low Vision Users: 21 Best Practices for Designers and Developers," in *2024 IEEE International Professional Communication Conference (ProComm)*. IEEE, 2024, pp. 121–128.

[30] J. He *et al.*, "Does Prompt Formatting Have Any Impact on LLM Performance?" 2024, arXiv preprint arXiv:2411.10541v1.

[31] M. Campoverde-Molina and S. Luján-Mora, "Artificial intelligence in web accessibility: A systematic mapping study," *Computer Standards & Interfaces*, vol. 96, p. 104055, Mar. 2026, doi: 10.1016/j.csi.2025.104055.

[32] C. Huang *et al.*, "ACCESS: Prompt Engineering for Automated Web Accessibility Violation Corrections," Jan. 2024, arXiv preprint arXiv.2401.16450.

[33] A. Ahluwalia and S. Wani, "Leveraging Large Language Models for Web Scraping," Jun. 2024, arXiv preprint arXiv.2406.08246.

[34] E. Ha *et al.*, "AI-based nanotoxicity data extraction and prediction of nanotoxicity," *Computational and Structural Biotechnology Journal*, vol. 29, pp. 138–148, Jan. 2025, doi: 10.1016/j.csbj.2025.03.052.

[35] Y. Song *et al.*, "A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and Opportunities," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–40, Dec. 2023, doi: 10.1145/3582688.

[36] G. Brajnik, "A Comparative Test of Web Accessibility Evaluation Methods," in *10th international ACM SIGACCESS conference on Computers and accessibility*, 2008, doi: 10.1145/1414471.1414494.

[37] G. Vera-Amaro and J. R. Rojano-Cáceres, "Understanding Accessibility Needs of Blind Authors on CMS-Based Websites," 2025, arXiv preprint arXiv.2508.15045.

[38] T. Mahatody, M. Sagar, and C. Kolski, "State of the art on the cognitive walkthrough method, its variants and evolutions," *International Journal of Human-Computer Interaction*, vol. 26, no. 8, pp. 741–785, Jul. 2010, doi: 10.1080/10447311003781409.

[39] G. Vera-Amaro and J. R. Rojano-Cáceres, "Dataset for Accessible Web Content Generation Using LLMs: An Empirical Study on Prompting Strategies and Template-Guided Remediation," 2025, Mendeley Data, doi: 10.17632/zybws98spf.2.

Guillermo Vera-Amaro received the degree of Bachelor in Informatics in 2002 and the degree of Master in Computer Science in 2008. He is currently pursuing the Ph.D. degree in Computer Science at Universidad Veracruzana. His research interests include web accessibility, HCI, software engineering, AI and web development.

José Rafael Rojano-Cáceres received a Ph.D. in Computer Science from Tecnológico de Monterrey, currently he is a full-time professor at Universidad Veracruzana. His research interests include accessibility, Human-Computer Interaction, and inclusive technologies for people with sensory disabilities, such as blindness and deafness. He is a member of the Mexican National System of Researchers (SNII) and holds a PRODEP academic profile.