AccessAI: Exploring AI-Driven Support for Smart Digital Accessibility Compliance

Heidi Kandil
Quality Tower
PwC ETIC
Cairo, Egypt
e-mail: heidi.t.kandil@gmail.com

Abstract—**Digital accessibility compliance remains a significant challenge for organizations due to the cost, time, and specialized expertise required for traditional accessibility audits. While automated testing tools such as axe-core and WAVE can identify accessibility violations against the Web Content Accessibility Guidelines (WCAG), they often lack contextual guidance, prioritization, and iterative remediation support. This work-in-progress paper introduces AccessAI, a conceptual AI-driven accessibility auditing framework aligned with Smart Accessibility principles. AccessAI integrates automated accessibility evaluation using established engines, interactive dashboarding for compliance visualization, and conversational chatbot assistants powered by Large Language Models to support continuous compliance. The framework targets perceivable and operable accessibility dimensions under WCAG 2.2 Levels A, AA and AAA, addressing both physical and cognitive barriers to digital inclusion. This paper reviews the state of the art, identifies knowledge gaps, outlines the proposed system architecture with concrete technical details, and describes a three-phase mixed-method evaluation methodology.**

Keywords—digital accessibility; WCAG compliance; automated evaluation; conversational AI; smart accessibility.

I. INTRODUCTION

Smart Accessibility emphasizes the use of intelligent, adaptive systems to proactively support inclusive digital experiences [1]. As digital services become increasingly embedded in everyday life, accessibility compliance has become a critical requirement, reinforced by international standards such as the Web Content Accessibility Guidelines (WCAG 2.2), developed by the World Wide Web Consortium (W3C) [2]. Organizations are expected not only to achieve compliance but also to sustain it across continuous updates and iterative development cycles.

Despite growing awareness, accessibility remains difficult to operationalize. Manual audits conducted by accessibility specialists require significant time and expertise, making them costly and difficult to scale across large or frequently changing systems [3]. Existing automated tools such as axe-core [4], WAVE [5], and Lighthouse [6] help identify accessibility violations, but they typically cover only 30–50% of WCAG success criteria automatically [7] and produce static reports that lack sufficient explanation, prioritization, and actionable remediation guidance.

From a theoretical perspective, the limitations of current tools align with challenges identified in Human–Computer Interaction (HCI) research. Studies by Bi et al. [8] and Alshayban et al. [9] have shown that developers frequently struggle with interpreting accessibility audit results and translating them into effective code changes. The absence of contextual, conversational support in existing tools contributes to this difficulty.

AccessAI is proposed as a Smart Accessibility-aligned concept designed to move beyond static accessibility validation. By integrating automated evaluation, interactive dashboards, and conversational AI assistance powered by Large Language Models (LLMs), AccessAI aims to embed accessibility more naturally into modern development processes. The key contributions of this paper are: (1) a review of the state of the art identifying knowledge gaps; (2) a conceptual architecture for an integrated accessibility support framework; (3) a specification of the accessibility dimensions targeted; and (4) a detailed three-phase evaluation methodology.

The remainder of this paper is structured as follows. Section II reviews related work. Section III presents the problem context. Section IV describes the AccessAI conceptual design. Section V outlines the evaluation methodology. Section VI discusses implications and limitations, and Section VII presents the conclusion and future work.

II. RELATED WORK

This section reviews the state of the art across three areas relevant to AccessAI and identifies the knowledge gap that the proposed research aims to fill.

A. Automated Accessibility Evaluation Tools

Automated accessibility evaluation tools have been widely adopted to support compliance checking at scale. Tools such as axe-core [4], WAVE [5], Google Lighthouse [6], and pa11y [10] perform rule-based analysis of web content against WCAG success criteria. Axe-core, developed by Deque Systems, is an open-source engine that evaluates Document Object Model (DOM) elements against a configurable rule set and integrates with Continuous Integration/Continuous Deployment (CI/CD) pipelines. WAVE provides a browser-based overlay that visually highlights accessibility issues. Pa11y provides a command-line interface for automated testing.

Despite their widespread use, these tools have well-documented limitations. Frazao and Duarte [7] found that automated tools can typically detect only 30–50% of WCAG 2.2 success criteria automatically. Vigo et al. [11] conducted a comparative study of six automated evaluation tools and observed significant variance in the issues reported. Abou-Zahra [3] emphasized that automated tools are best used alongside manual evaluation. Paterno and Schiavone [12] noted that the lack of prioritization in tool outputs makes it difficult for developers to allocate remediation effort effectively.

B. AI and Machine Learning in Accessibility

Recent research has explored the application of AI to accessibility tasks. Gleason et al. [13] developed automated methods for generating image alt-text using computer vision models. Zhang et al. [14] applied deep learning to detect accessibility barriers in mobile application interfaces. Prakash et al. [15] investigated the use of natural language processing to assess the readability and cognitive accessibility of web content. Chiou et al. [16] explored automated captioning for video content.

The emergence of Large Language Models (LLMs) has opened new possibilities. Fan et al. [17] conducted a systematic review of LLMs for software engineering and found promising but uneven results for accessibility-related code generation. Peng et al. [18] demonstrated that AI-assisted coding tools can significantly improve developer productivity. However, the integration of LLMs into interactive, developer-facing accessibility workflows remains largely unexplored.

C. Developer-Facing Accessibility Support

Research in HCI has identified persistent barriers to developer engagement with accessibility. Bi et al. [8] found that developers often lack sufficient knowledge of WCAG criteria and find automated tool outputs difficult to act upon. Alshayban et al. [9] analyzed accessibility issues in over 1,000 Android applications and concluded that many violations persist due to inadequate developer support rather than technical infeasibility. Moreno et al. [19] proposed method-level accessibility guidelines but found developers needed additional interactive support.

Conversational interfaces have been proposed as a mechanism for improving developer guidance. Research on chatbot-assisted development [18] suggests that natural language interaction can reduce cognitive overhead. Retrieval-Augmented Generation (RAG) [20] has emerged as a technique for grounding LLM responses in domain-specific knowledge bases. Applying conversational AI with RAG to accessibility support represents a gap that AccessAI seeks to address.

D. Identified Knowledge Gap

No existing solution integrates automated evaluation, visualization, and conversational AI into a unified accessibility support framework. Automated engines operate in isolation without interactive remediation support [7][11]; visualization of compliance status is rare and typically limited to simple issue lists [12]; conversational AI has not yet been applied to provide contextual, iterative accessibility guidance within development workflows [17][18].

AccessAI is designed to fill this gap. Table I summarizes these differences.

TABLE I. COMPARISON OF ACCESSIBILITY SUPPORT APPROACHES.

Feature	Manual Audit	Auto Tools	AI-Only	AccessAI
Scalability	Low	High	Medium	High
Contextual Guidance	High	Low	Medium	High
Iterative Support	Low	Low	Low	High
Visualization	N/A	Limited	N/A	Integrated
Conversational AI	N/A	N/A	Partial	Integrated
Continuous Validation	Low	Medium	Low	High

III. PROBLEM CONTEXT AND RESEARCH GAP

Current accessibility practices generally fall into two categories: expert-led manual audits and rule-based automated testing tools [3]. Manual audits provide contextual depth and accuracy but are resource-intensive. Automated tools offer scalability but lack interpretability [7][11].

From a Smart Accessibility perspective, accessibility support systems should be adaptive, explanatory, and capable of supporting continuous improvement [1]. Developers frequently require clarification, follow-up guidance, and confirmation that applied fixes have resolved the original issue. Traditional tools rarely support this iterative and conversational workflow.

The two primary challenges addressed by this research were identified through: (1) the literature reviewed in Section II, which documents the limitations of automated tools [7][11] and developer difficulties [8][9]; (2) analysis of existing tool outputs, which revealed a consistent pattern of static, non-interactive reporting; and (3) preliminary discussions with five web developers who reported that understanding why an issue matters and how to fix it are their primary pain points. The resulting challenges are: (1) limited guidance and contextual support in existing tools, and (2) insufficient support for iterative remediation and continuous validation.

IV. ACCESSAI CONCEPTUAL DESIGN

Accessibility refers to a wide range of potential exclusion causes, including visual, auditory, motor, and cognitive impairments [2]. AccessAI focuses primarily on the Perceivable and Operable principles of WCAG 2.2 at conformance Levels A, AA and AAA. Specifically, AccessAI targets: alternative text for non-text content (SC 1.1.1), color contrast (SC 1.4.3, 1.4.6), keyboard accessibility (SC 2.1.1, 2.1.2), focus management (SC 2.4.3, 2.4.7), form labeling (SC 1.3.1), heading structure (SC 1.3.1), and link purpose (SC 2.4.4). The Understandable and Robust principles are addressed to a lesser extent.

AccessAI is intended to be integrated at two stages: (1) during active development, operating alongside the code editor and browser developer tools for real-time feedback; and (2) during continuous integration, running as an automated check within CI/CD pipelines on each code commit [21].

The architecture consists of four integrated components: (1) automated accessibility evaluation, (2) dashboarding and visualization, (3) conversational chatbot assistance, and (4) iterative retesting. Figure 1 illustrates the system architecture.

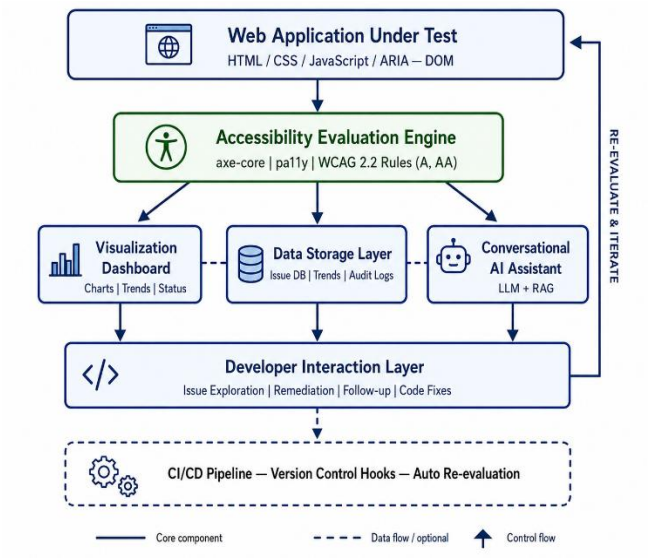

Figure 1. AccessAI system architecture.

A. Automated Accessibility Evaluation Engine

The system builds on established automated accessibility evaluation tools as its core analysis layer. These tools programmatically assess web content against defined accessibility standards and generate structured results describing identified issues, their severity, affected elements, and references to relevant guidelines. An example of such a tool is axe-core, which provides a widely used programmatic interface for running accessibility checks.

TABLE II. EXAMPLE AXE-CORE EVALUATION OUTPUT.

Rule ID	Impact	WCAG SC	Element	Description
image-alt	Critical	1.1.1	img.hero	Image missing alt text
color-contrast	Serious	1.4.3	p.intro	Ratio 2.8:1 < 4.5:1
label	Serious	1.3.1	input#email	Form input lacks label
heading-order	Moderate	1.3.1	h4.sect	Heading level skipped

The engine assesses structural semantics, keyboard accessibility, color contrast ratios (WCAG thresholds of 4.5:1 for normal text and 3:1 for large text), alternative text for images and media, and form labeling and Accessible Rich Internet Applications (ARIA) attribute correctness. This structured output feeds into both the dashboard and conversational assistant.

B. Dashboarding and Visualization Interfaces

AccessAI proposes interactive dashboards to visualize accessibility findings, drawing on information visualization principles [22]. The dashboard provides three views: (1) a compliance summary with proportional breakdown by impact level; (2) a trend chart displaying issue counts over successive evaluation runs; and (3) a detail view listing individual violations with remediation status and direct links to the conversational AI assistant. Figure 2 illustrates the proposed dashboard layout.

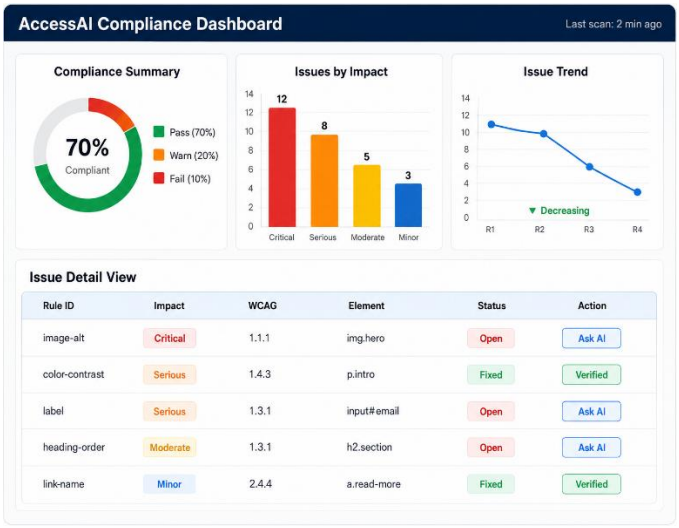

Figure 2. Proposed AccessAI dashboard layout.

C. Conversational Chatbot Assistance

A distinguishing aspect of AccessAI is the integration of a conversational AI assistant powered by an LLM with Retrieval-Augmented Generation (RAG) [20]. The chatbot supports three interaction patterns.

First, issue explanation: when a developer selects a violation from the dashboard, the chatbot provides a natural language explanation referencing the relevant WCAG success criterion. For example, for a color-contrast violation (SC 1.4.3) with a measured ratio of 2.8:1, the chatbot explains the 4.5:1 threshold and describes the impact on users with low vision.

Second, remediation guidance: the chatbot provides context-specific remediation advice with code snippets. For the contrast example, it suggests specific Cascading Style Sheets (CSS) changes to meet the 4.5:1 threshold.

Third, follow-up interaction: developers can ask clarifying questions, request alternative approaches, or inquire about related criteria. Figure 3 illustrates the workflow with a concrete example.

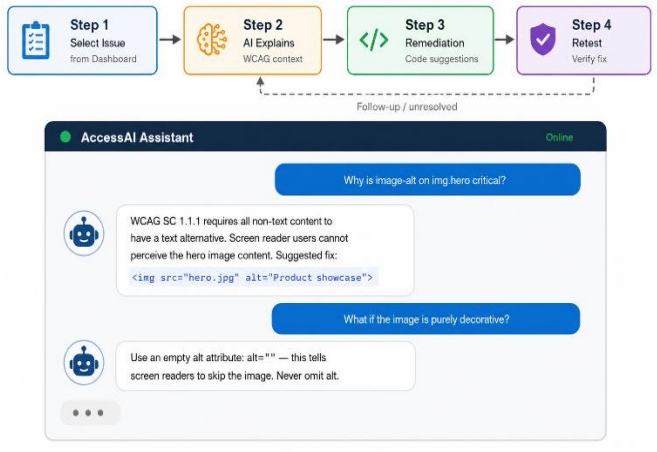

Figure 3. Conversational AI interaction workflow with example.

The RAG architecture grounds the LLM's responses in a knowledge base containing WCAG 2.2 documentation, W3C Web Accessibility Initiative (WAI) techniques, and the specific evaluation results from the current audit. This mitigates hallucination risk compared to unconstrained generation [20].

D. Iterative Retesting and Continuous Validation

After a developer applies a fix, the evaluation engine is re-invoked to verify resolution. The dashboard updates to reflect the new compliance status, and the chatbot confirms the fix or identifies remaining issues. This feedback loop aligns with agile and CI/CD workflows [21], supporting accessibility as an ongoing process. The retesting mechanism integrates with version control hooks for automated checks on each commit.

V. Evaluation Methodology

The evaluation follows a three-phase mixed-method approach. Figure 4 illustrates the methodology and timeline.

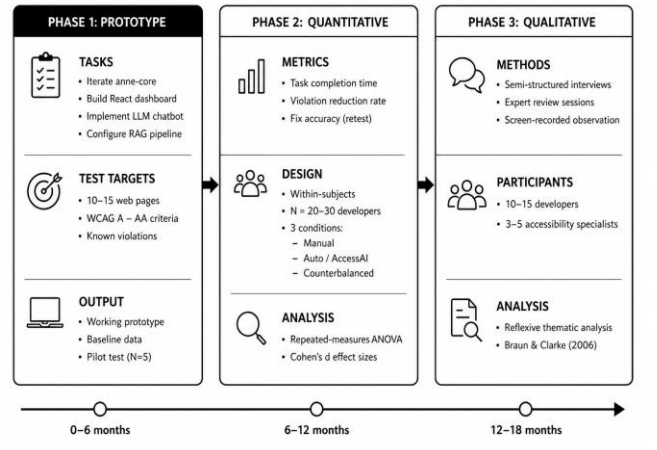

Figure 4. Three-phase evaluation methodology.

A. Phase 1: Prototype Development

A functional prototype will be developed that integrates three main components: (1) an automated evaluation engine accessed via a programmatic API; (2) an interactive web-based dashboard for visualizing results; and (3) an AI-powered assistant supported by a retrieval pipeline over a structured knowledge base of relevant standards and best practices. The system will be delivered in multiple forms, including a browser-based tool and a standalone web application.

The prototype will be evaluated on a set of web pages with pre-identified accessibility issues spanning different levels of conformance. These will include both controlled test pages with known violations and real-world examples drawn from publicly available accessibility datasets.

B. Phase 2: Quantitative Evaluation

A within-subjects experimental design with 20–30 participants (developers with varying accessibility experience) will compare three conditions using counterbalanced task assignment: (1) manual audit using only WCAG documentation and browser developer tools; (2) automated tools using axe-core with its default report output; and (3) AccessAI using the full prototype with dashboard and chatbot.

Three metrics will be measured: (1) task completion time, defined as the time taken to identify and remediate accessibility violations; (2) violation reduction rate, defined as the percentage reduction in violations after one remediation cycle; and (3) fix accuracy, defined as the percentage of remediation attempts that pass automated retesting. Statistical analysis will use repeated-measures ANOVA with post-hoc pairwise comparisons and Cohen's d effect sizes.

C. Phase 3: Qualitative Evaluation

Qualitative data will be collected through: (1) semi-structured interviews with 10–15 developers; (2) expert review sessions with 3–5 accessibility specialists; and (3) screen-recorded observation of participant interactions. Analysis will follow reflexive thematic analysis [23]. Table III summarizes the evaluation design.

TABLE III. Evaluation Metrics and Data Collection.

Phase	Metric	Method	Participants
Quant.	Task completion time	Timed tasks, 3 conditions	20–30 developers
Quant.	Violation reduction	Successive eval cycles	20–30 developers
Quant.	Fix accuracy	Retest pass rate	20–30 developers
Qual.	Perceived usefulness	Semi-structured interviews	10–15 developers
Qual.	Guidance quality	Expert review sessions	3–5 specialists
Qual.	Interaction patterns	Observation + thematic analysis	10–15 developers

TABLE IV. HYPOTHESIZED OUTCOMES BY CONDITION.

#	Metric	Manual	Auto Only	AccessAI
H1	Completion time	Longest	Moderate	Shortest
H2	Violation reduction	Moderate	Low	Highest
H3	Fix accuracy	High	Low–Med	High
H4	Developer confidence	Low	Low–Med	High

VI. DISCUSSION

As an early-stage conceptual framework, AccessAI presents design direction and research intent rather than finalized empirical findings. The central premise that combining automation, visualization, and conversational AI can reduce both technical and cognitive barriers to accessibility compliance is supported by prior work on developer challenges [8][9] and conversational code assistance [18].

Compared with existing solutions, AccessAI's contribution lies in integrating capabilities that currently exist only in isolation. Automated tools excel at scalable detection but do not support remediation [4][7]. AI-based approaches show promise for remediation but lack systematic detection [17]. AccessAI combines these strengths while adding visualization and iterative retesting.

A. Limitations and Constraints

Several limitations must be acknowledged. First, LLM-based guidance may produce inaccurate remediation advice (hallucination), particularly for complex ARIA patterns. The RAG approach [20] mitigates but does not eliminate this risk. Second, automated evaluation covers only a subset of WCAG success criteria [7]; AccessAI does not replace manual auditing for criteria requiring human judgment. Third, effectiveness may vary with developer experience level. Fourth, the current scope excludes mobile-native accessibility testing. Fifth, reliance on specific LLM providers introduces dependency and cost considerations.

B. Ethical Considerations

AI-driven accessibility guidance raises ethical concerns regarding trust and responsible use. Developers may over-rely on AI-generated remediation without critical evaluation. The system must communicate confidence levels and encourage verification. Accessibility compliance has legal implications [24], and AI-generated guidance must not substitute for expert review.

C. Challenges and Risks

Several challenges are anticipated: (1) RAG knowledge base currency. WCAG guidelines evolve, requiring ongoing corpus maintenance; (2) evaluation engine accuracy, false positives and negatives from axe-core may propagate; (3) LLM response latency, interactive chatbot performance depends on API response times; and (4) generalizability, the evaluation will use a limited set of test pages.

D. Immediate and Mid-Term Future Plans

Immediate plans (0–6 months): Complete Phase 1 prototype development, including axe-core integration, React dashboard implementation, and LLM chatbot with RAG pipeline. Conduct preliminary pilot testing with 5 developers to refine the prototype.

Mid-term plans (6–18 months): Execute Phases 2 and 3 of the evaluation. Expand automated evaluation capabilities. Investigate integration with popular development environments (VS Code, IntelliJ). Explore multi-modal accessibility support, including voice-guided remediation. Conduct longitudinal studies in real-world development environments.

VII. CONCLUSION AND FUTURE WORK

This paper introduced AccessAI, a conceptual Smart Accessibility-aligned framework for improving digital accessibility compliance. The framework integrates automated evaluation engines (axe-core, pa11y), visualization dashboards, and conversational chatbot support powered by Large Language Models with Retrieval-Augmented Generation. AccessAI targets the perceivable and operable WCAG 2.2 dimensions at Levels A, AA and AAA, addressing both detection and remediation phases.

The review of related work identified a knowledge gap: no existing solution integrates automated evaluation, visualization, and conversational AI into a unified, iterative accessibility support framework. AccessAI addresses this gap through a four-component architecture designed for integration at both development-time and CI/CD stages. A three-phase evaluation methodology has been defined with quantitative metrics and qualitative analysis.

Future work will focus on expanding coverage to additional WCAG criteria, enhancing chatbot explainability, integrating with mainstream development environments, and conducting longitudinal evaluations. The proposed direction highlights the potential of AI-driven accessibility support systems to make accessibility more scalable, actionable, and embedded within modern development practices.

REFERENCES

[1] G. Brajnik, "Conceptual frameworks for web accessibility analysis," in *Proc. Int. Cross-Disciplinary Workshop on Web Accessibility (W4A)*, ACM, 2008, pp. 30–38.

[2] World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 2023.

[3] S. Abou-Zahra, "Web accessibility evaluation," in *Web Accessibility: A Foundation for Research*, S. Harper and Y. Yesilada, Eds. London: Springer, 2008, pp. 79–106.

[4] Deque Systems, "axe-core: Accessibility testing engine for websites and other HTML-based user interfaces," Deque Systems, 2024. [Online]. Available: https://github.com/dequelabs/axe-core

[5] WebAIM, "WAVE: Web accessibility evaluation tool," WebAIM, Utah State University, 2024. [Online]. Available: https://wave.webaim.org/

[6] Google Developers, "Lighthouse: Automated auditing, performance metrics, and best practices for the web," Google LLC, 2024. [Online]. Available: https://developer.chrome.com/docs/lighthouse/

[7] T. Frazao and C. Duarte, “Comparing accessibility evaluation plug-ins,” in *Proc. 17th Int. Web for All Conf. (W4A)*, ACM, 2020, pp. 1–10.

[8] T. Bi, X. Xia, D. Lo, J. Grundy, and T. Zimmermann, “Accessibility in software practice: A practitioner’s perspective,” *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 4, pp. 1–26, 2022.

[9] A. Alshayban, I. Ahmed, and S. Malek, “Accessibility issues in Android apps: State of affairs, sentiments, and ways forward,” in *Proc. 42nd Int. Conf. on Software Engineering (ICSE)*, ACM, 2020, pp. 1323–1334.

[10] pa11y, “pa11y: Your automated accessibility testing pal,” Team pa11y, 2024. [Online]. Available: https://pa11y.org/

[11] M. Vigo, J. Brown, and V. Conway, “Benchmarking web accessibility evaluation tools: Measuring the harm of sole reliance on automated tests,” in *Proc. 10th Int. Cross-Disciplinary Conf. on Web Accessibility (W4A)*, ACM, 2013, pp. 1–10.

[12] F. Paterno and A. G. Schiavone, “The role of tool support in public policies and accessibility,” in *Proc. ACM SIGACCESS Conf. on Computers and Accessibility (ASSETS)*, ACM, 2015, pp. 1–8.

[13] C. Gleason, A. Pavel, X. Liu, P. Carrington, L. Chilton, and J. Bigham, “Making memes accessible,” in *Proc. 21st Int. ACM SIGACCESS Conf. on Computers and Accessibility (ASSETS)*, ACM, 2019, pp. 367–376.

[14] X. Zhang, A. S. Ross, A. Caspi, J. Fogarty, and J. Wobbrock, “Interaction proxies for runtime repair and enhancement of mobile application accessibility,” in *Proc. CHI Conf. on Human Factors in Computing Systems*, ACM, 2017, pp. 6024–6037.

[15] J. Prakash, S. Kane, and R. Azenkot, “Examining the readability of web content for people with cognitive disabilities,” in *Proc. 22nd Int. ACM SIGACCESS Conf. on Computers and Accessibility (ASSETS)*, ACM, 2020, pp. 1–4.

[16] E. Chiou, K. W. Higgins, and R. E. Ladner, “Automated captioning for accessibility,” *J. Access. Des. All*, vol. 10, no. 1, pp. 1–25, 2020.

[17] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, “Large language models for software engineering: A systematic literature review,” *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, pp. 1–60, 2024.

[18] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, “The impact of AI on developer productivity: Evidence from GitHub Copilot,” *Commun. ACM*, vol. 67, no. 3, pp. 54–63, 2024.

[19] L. Moreno, P. Martínez, and B. Ruiz-Mezcua, “Disability standards for multimedia on the web,” *IEEE Multimedia*, vol. 15, no. 4, pp. 52–54, 2008.

[20] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 9459–9474.

[21] M. Fowler and M. Foemmel, “Continuous integration,” ThoughtWorks, 2006.

[22] B. Shneiderman, “The eyes have it: A task by data type taxonomy for information visualizations,” in *Proc. IEEE Symposium on Visual Languages*, IEEE, 1996, pp. 336–343.

[23] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[24] European Commission, “European Accessibility Act,” Directive (EU) 2019/882, Jun. 2019.