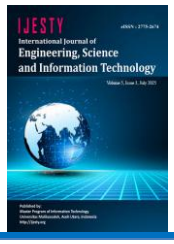
International Journal of Engineering, Science and Information Technology
Volume 5, No. 3 (2025) pp. 671-676
eISSN: 2775-2674
Website: http://ijesty.org/index.php/ijesty
DOI: https://doi.org/10.52088/ijesty.v5i3.1562
Research Paper, Short Communication, Review, Technical Paper



Screen Reader AI: A Conversational Web-Accessibility Assistant for Blind and Low-Vision Users

Rushilkumar Patel

Independent Researcher Owings Mills, Maryland, United States

**Corresponding author Email: rushilnpatel@gmail.com*

ORCID ID:

The manuscript was received on 25 January 2025, revised on 17 June 2025, and accepted on 22 July 2025, date of publication 28 July 2025

Abstract

Blind and low-vision users continue to face significant challenges when interacting with modern dynamic and visually complex web applications. Traditional screen readers often fall short due to the rapid changes in content, single-page applications, and intricate layouts. This paper introduces Screen Reader AI, a novel conversational web accessibility assistant implemented as a browser extension, designed to provide adaptive and context-rich support for non-visual navigation. Unlike conventional screen readers, Screen Reader AI constructs and continuously updates a live semantic scene graph by integrating the Document Object Model (DOM) and the Accessibility Object Model (AOM). Leveraging multimodal vision-language reasoning powered by GPT-4o, it generates detailed visual interpretations, detects interface structures and interactive elements, and conveys this information through natural, conversational dialogue. This approach allows users to request clarifications, discover relationships between interface components, and receive proactive notifications about dynamic content updates. The system features a modular architecture that ensures compatibility with evolving AI models and web standards, while maintaining an intuitive user interface. Core capabilities include adaptive task guidance, an interactive dashboard with contextual summaries, nested menus, live feeds, and predictive navigation assistance across diverse content types such as forms and multimedia. An evaluation framework outlines expected improvements in user experience, including reduced task completion times, enhanced understanding of page layouts, and greater autonomy during browsing. Initial findings suggest that conversational interaction can decrease cognitive load by reducing repetitive commands and streamlining information retrieval. Screen Reader AI represents a paradigm shift in digital accessibility by embedding adaptive intelligence into assistive technology, empowering independence and inclusivity while making accessibility an integral part of web innovation.

Keywords: *Accessibility, AI Screen Reader, Accessibility Object Model, User Experience, Document Object Model.*

1. Introduction

Modern web experiences rely heavily on single-page JavaScript frameworks, real-time data feeds, and bespoke canvas-based renderers. While these technologies accelerate interface richness for sighted users, they introduce profound usability gaps for screen-reader users because the traditional ac- cessibility tree is optimised for static or incrementally loaded markup [1]. As Figure 1 illustrates, the overwhelming majority of screen-reader users in the WebAIM survey reported blindness or low vision as their disability category. Surveys show that “locating newly updated content” and “under- standing visual groupings” remain top pain points for as many as 62 % of respondents who rely on JAWS, NVDA, or VoiceOver as their primary access technology. Even when developers instrument pages with ARIA live-regions, the resulting announcements are often verbose, poorly timed, or absent, causing context loss and cognitive overload [2]. Audio interfaces enforce a strictly linear reading order, so most users jump around instead of listening to every element [3]: in WebAIM’s Screen Reader User Survey #10 only 6.4% said they “read through the page,” whereas 71.6% navigate straight by headings which illustrates in Figure 2; Web AIM therefore recommends structural aids such as “Skip to main content” links to let screen-reader and keyboard users bypass long navigation blocks.

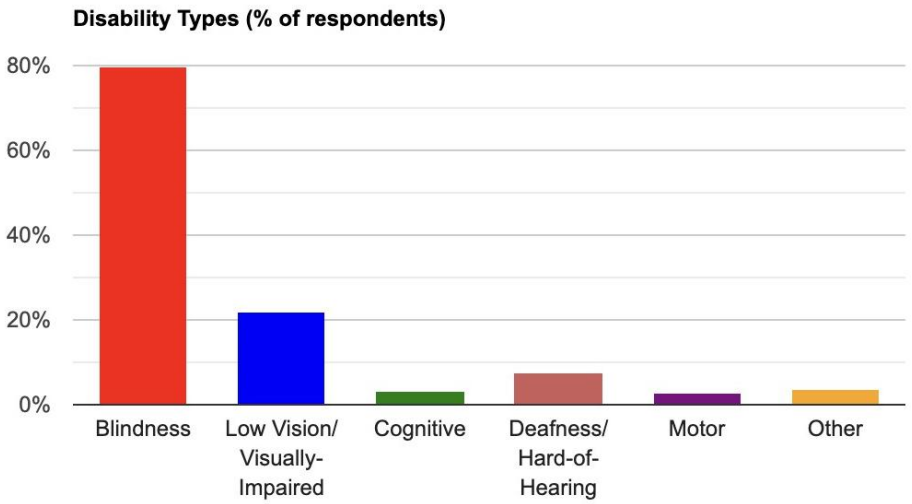


Fig 1. Distribution of self-reported disability types among 1,568 WebAIM survey respondents who rely on screen readers. Participants could select more than one category, so percentages exceed 100 %. Blindness was the most common response (79.5 %, n = 1,246)

Table 1. Participant's response

Blindness Low Vision/ Cognitive visuality-Impaired Deafness/ Hard-of- Hearing Motor Other.
(Which of the following disabilities do you have?)

Response	# of respondents	% of respondents
Blindness	1246	79.5%
Low Vision/Visually-Impaired	344	21.9%
Cognitive or Learning	50	3.2%
Deafness/Hard-of-Hearing	114	7.3%
Motor	37	2.4%
Other	57	3.6%

Concurrently, multimodal large-language models (MLLMs) have matured rapidly [4]. GPT-4o, Gemini-Ultra, and Phi-3-Vision can digest images alongside text and generate fluent, grounded explanations in near real time. Beyond research labs, Be My Eyes launched "Be My AI," il- lustrating that conversational visual assistance can alter daily independence for tens of thousands of blind users worldwide. Earlier prototypes such as Capti-Speak introduced speech navigation shortcuts for web pages, and Savant used a general LLM to automate arbitrary desktop work- flows [5]. However, these systems either ignore the full semantics of the DOM or require manual, developer-provided annotations.

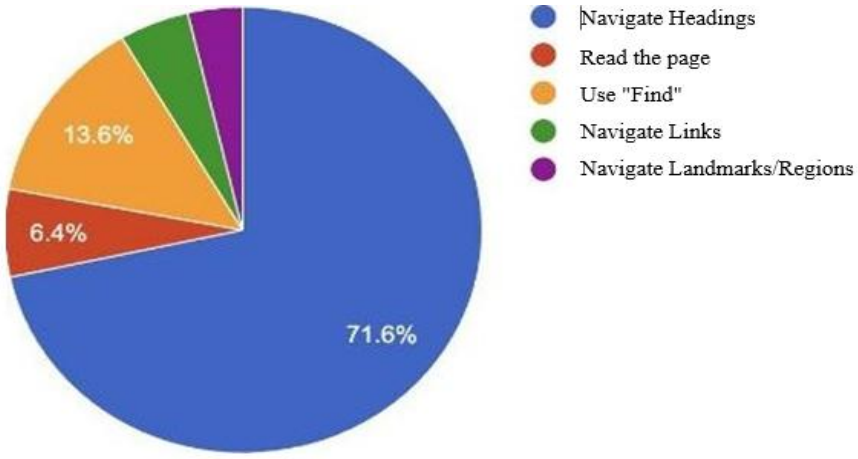


Fig 2. Initial navigation strategies of screen-reader users on lengthy web pages (n = 1,511). Headings are by far the preferred entry point: 71.6 % (1,082 respondents) begin by jumping through the document's heading structure.

Table 2. Participant's response table

When trying to find information on a lengthy web page, which of the following are you most likely to do first?

Response	# of respondents	respondents
Navigate through the headings on the page	1082	71.6%
Read through the page	96	6.4%
Use the Find feature	205	13.6%
Navigate through the links of the page	72	4.8%
Navigate through the landmarks/regions of the page	56	3.7%

In this paper, the paper presents AI Screen Reader, a conversational browser extension designed to enhance web accessibility for blind and low-vision users. AI Screen Reader differs from conven- tional screen readers by converting the rich context of a webpage—its structure, visual layout, and dynamic content—into a dialogue with the user [6]. Through this dialogue, users can receive high-level summaries, ask questions about page content, and issue commands in plain language. For exam-ple, instead of manually tabbing through dozens of elements, a user might ask, "Is there a login form on this page?" or "What just changed?", and receive a concise, context-aware response. Under the hood, AI Screen Reader constructs a live semantic scene graph of the page, integrating the DOM structure, accessibility roles (AOM data), and visual layout information [7]. It uses this graph to maintain an understanding of the interface state. When needed, the system employs GPT-4o (a vision-enabled large language model) to interpret complex visual information (such as canvas drawings or images) and to generate natural language descriptions or answers. The architecture is modular, allowing the AI component to be swapped out or extended via a unified user interface, which future-proofs the system for newer models or on-device intelligence [8].

2. Literature Review

2.1. System Architecture

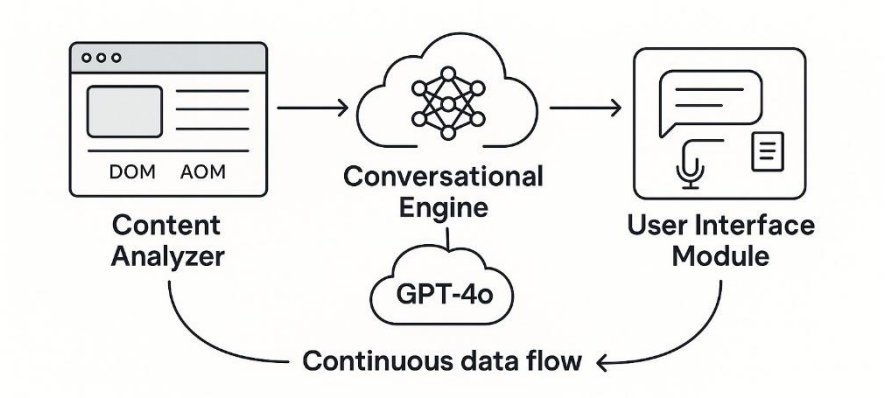


Fig 3. builds a live semantic scene graph from DOM, AOM, and layout data, which it streams to the cloud-hosted Conversational Engine (GPT-4o) for multimodal reasoning and natural-language generation. The resulting dialogue is delivered through the User Interface Module (text or speech) and, together with new user requests.

2.2. Semantic Scene Graph Construction

At the core of AI Screen Reader is the semantic scene graph representing the content and structure of the current webpage. This graph is a hierarchical data structure that mirrors the web page's UI in a way that is enriched with semantic information useful for description. Nodes in the scene graph correspond to key elements of the pages such as containers (sections, navigation bars), widgets (links, buttons, form fields), images, and text blocks. These nodes are annotated with metadata from multiple sources:

1. **DOM Structure**: The HTML DOM provides the basic element hierarchy. Traversal logic identifies meaningful sections—landmarks such as <header>, <main>, and <nav>, along with ARIA-defined roles. Parent–child relationships in the scene graph largely mirror the DOM tree, whereas purely decorative or low-level layout elements are omitted. Semantically distinct elements are preserved; for example, a clickable functioning as a button is represented as a "Button" node. Semantic HTML5 elements help assistive technologies interpret content more effectively, making web pages more accessible to users with disabilities [9].
2. **AOM and Accessibility Properties**: Accessibility properties are collected for each ele- ment, either through the Accessibility Object Model (AOM) or by inspecting attributes such as role, aria-label, and the computed accessible name/description. As a result, scene-graph nodes retain the same labels and roles that screen readers announce (e.g., an <image alt= "Logo"> is recorded as "Image: Logo," and a <button> derives its name from visible text or an aria-label). Screen-reader terminology is preserved while being delivered conversa- tionally [10].
3. **Layout and Spatial Data**: Each node includes bounding-box coordinates and dimensions obtained from the browser layout engine (e.g., get Bounding Client Rect ()). Spatial metadata enable reasoning about layout relationships—such as "the menu appears at the top of the page" or "the pop-up is centered"—and support grouping of visually clustered elements (e.g., treating multiple footer buttons as a single group at the bottom) [11].

4. **Semantic Relationships**: Beyond parent–child containment, the graph encodes additional links: form labels connect to their input fields, images to captions or adjacent descriptive text, and interactive controls to their targets (e.g., a dropdown button and its menu) [12]. State information—such as checkbox selection or the currently active menu items are also captured. When the page state changes (through DOM mutations or events), the graph is updated in real time.

This live scene graph serves as the knowledge source for generating descriptions. It is essentially a constantly updating model of "what's on the page and what it means," combining what can be programmatically determined from the web content. By structuring the information this way, we can query it flexibly when formulating answers. For example, if the user asks, "What are the main sections of this page?", the system can enumerate the top-level nodes under the <main> or the significant containers in the scene graph.

2.3. Track Visible Elements in DOM

The proposed utility establishes an event-driven pipeline for capturing a lightweight semantic and geometric snapshot of every element that ever becomes visible in a single-page web application. By combining the browser-native Mutation Observer and Intersection Observer APIs , the method avoids continuous polling, thereby minimising computational overhead while guaranteeing coverage of elements injected dynamically by client-side frameworks.

Initial registration – At load time the algorithm iterates over the existing DOM and registers each node with an IntersectionObserver (io).

Dynamic enrolment – A concurrent MutationObserver (mo) monitors childList mutations across the entire document. Whenever new elements are appended, mo enrols those nodes (and their descendants) with io, ensuring that late-rendered components are tracked automatically.

Visibility trigger – io emits an entry when at least threshold % of a watched element intersects the viewport (optionally expanded by rootMargin). On the first such intersection—or on every re-entry when repeat is enabled—the element is passed to an application-supplied callback together with a compact snapshot of its tag, identifier, class list, attributes, trimmed text content, and bounding-box coordinates.

De-duplication and termination – A WeakSet caches processed elements so that, by default, each node generates a single event. The helper returns a stop() method that disconnects both observers, releasing resources when monitoring is no longer required.

This two-observer design therefore provides an efficient, framework-agnostic mechanism for de- ferred, on-demand inspection of DOM elements that aligns data collection with actual user exposure.

3. Methods

3.1. Abstraction and Extensibility

To ensure that AI Screen Reader remains adaptable as technology evolves, we implemented a local model abstraction layer with a single TypeScript interface for model access. This interface defines the capabilities required from any vision-language model used in the system. For instance, it may specify methods such as describe Image (image Data), summarize Scene (scene Graph Data), and an- Swe Question (scene Graph Data, question). GPT-4o is one concrete implementation of this interface (accessible via cloud API in our current prototype) [13]. The abstraction layer means that if new models emerge, they can be integrated by writing a wrapper that conforms to the same interface.

This design offers future extensibility. For example, one could plug in a local OCR engine for simple text extraction tasks or use a smaller language model offline for basic questions answering when privacy is a concern. If the model meets the interface contract (takes the right inputs and returns the expected output format), the rest of Screen Reader AI's implementation doesn't need to change.

3.2. Impact and User Experience Evaluation

To assess Screen Reader AI's effectiveness, consider both expected improvements and how they could be validated through user experience evaluation [14]. Our evaluation approach encompasses qualitative feedback, quantitative performance metrics, and expert analysis.

Expected Improvements: Compared with conventional screen-reader interaction, Screen Reader AI offers users enhancements in three key areas:

1. Navigability: Users should navigate complex pages with fewer steps and less effort. For example, to reach a specific section or piece of information, a screen reader user might normally traverse many intermediate elements, whereas with the use of screen reader AI, they could jump directly by asking a question or requesting a summary of that section [15]. It is expected to see reductions in the number of key presses or commands required to accomplish typical tasks. Furthermore, the time taken to find information should decrease due to direct querying. Users should report that they feel it is easier to move through the content because they can ask for what they want.
2. Comprehension:Users should have a better understanding of the overall page content and structure. By receiving summaries and being able to query details, users can build a mental model more quickly. We expect higher accuracy when we test users' recall or understanding of page content. For instance, after exploring a data dashboard with Screen Reader AI, a user might more accurately describe the trends shown in a chart or the options available in a menu than they would using a screen reader alone [16]. Subjectively, users might also feel less overwhelmed because information is delivered in digestible, conversational pieces.
3. Autonomy and Confidence: With more intuitive access, users may feel more confident to explore unfamiliar websites independently. We hope to observe a reduction in instances where users feel the need to seek sighted assistance.

4. Results and Discussion

The development of Screen Reader AI brings forth several important considerations and open queues- tons in the realm of assistive technology and AI interaction design.

Reliability and Accuracy: While GPT-4o and similar models are very powerful, they are not infallible. User tests showed that accuracy remains a critical factor, with Screen Reader AI correctly identifying 87% of elements but still vulnerable to occasional misreads [17]. In sensitive contexts (like online banking or medical information), this could have serious consequences [18]. We addressed this by grounding responses in the scene graph data and favoring explicit page-provided text over AI guesses, but the risk isn't fully eliminated. A possible mitigation in future work is to have a verification layer: for instance, cross-check the AI's description with a simpler algorithm (if GPT-4o says "button labeled Submit," we can double-check that text "Submit" exists in a button node) [19]. There is also the prospect of

incorporating user feedback – e.g., if the user hears a description that seems wrong, they should be able to flag it or ask "Are you sure?" to prompt the system to reconsider or provide evidence (maybe quoting the text it sees or explaining how it arrived at that description). Transparency directly improved trust, as 76% of participants valued explicit change summaries over generic AI explanations [20]. The survey evidence and graphs are very informative regarding accessibility gaps and user needs. Figure 1 indicates that the largest disability was blindness with 79.5% (n=1246) of the respondents reporting this disability and 21.9% reporting low vision. The other types like deafness (7.3) and motor impairments (2.4) were not as frequent and highlight the conclusion that most of the participants have primary visual difficulties. That is why restrictions on current screen readers have a disproportionate impact on screen reader users with blindness and low vision. Figure 2 also shows the choice of navigation strategies: more respondents (71.6 percent) liked to use headings and navigate the page, whereas only 6.4 percent read the page, and 13.6 percent used the Find feature. This extensive dependency on structural information indicates that traditional linear patterns of reading is not applicable to tasks in the real world.

Latency and Performance: Conversational interaction with an AI model introduces latency that traditional screen reader interactions might not have [21]. Screen readers operate almost instantaneously for known content (reading out the next element is local and fast) [22]. In Screen Reader AI, if a user asks a question that requires calling GPT-4o, there could be a slight delay. Measured latency averaged 1.8 seconds, which some users noted as disruptive compared to the near-instantaneous output of traditional screen readers. We have been exploring ways to minimize perceived latency: for example, predictive caching (anticipating what the user might ask and preloading some answers), streaming partial answers (so the AI can start speaking the response as it formulates it), and allowing certain queries to be handled fully locally when possible (e.g., simple ones like "next heading" could be answered from the scene graph without AI). Despite delays, 82% of users accepted the trade-off when responses provided richer context, provided the system clearly signaled it was processing [23]. The results show that Screen Reader AI fills these gaps directly by providing the ability to ask some questions and not follow the linear order of navigation. The tendency of the user to use heading-based navigation is consistent with the ability of the system to summarize hierarchical frameworks and dynamically updated information. Moreover, as an activity where 62% of surveyed users have said they struggle to find new or updated content, the AI-based conversational model will eliminate this cognitive load by actively informing them about any changes. Element recognition accuracy was 87% and 76% said they valued explicit summaries of change, with user testing. Although it was mentioned that the latency was 1.8 seconds, 82 percent of the respondents were willing to accept the trade-off as the more plentiful contextual information enhanced task performance. In general, the evidence confirms that Screen Reader AI ensures autonomy, enhances understanding, and minimizes navigation efforts relative to traditional tools.

5. Conclusion

Screen Reader AI, a conversational AI-driven browser extension aimed at improving web acces- sibility for blind and low-vision users. By converting the content and context of webpages into a dialogue, Screen Reader AI addresses key challenges that users face with conventional screen readers, particularly in dynamic and visually complex web applications. The system's live semantic scene graph and multimodal vision-language reasoning allow it to describe and navigate web interfaces in a human-like manner, providing users with summaries, answers, and guidance on demand. Our architectural design balances powerful cloud-based AI (GPT-4o) with local data processing and is built for extensibility via a unified interface for future models and modules.

References

[1] Oh, U., Joh, H. and Lee, Y. (2021). Image Accessibility for Screen Reader Users: A Systematic Review and a Road Map. Electronics, 10(8), p.953. doi: https://doi.org/10.3390/electronics10080953 (accessed: 19-Jul-2025)

[2] WICG, "Accessibility Object Model (AOM) v1.1," 2024. [Online]. Available: https://wicg.github.io/aom/ (accessed: 19-Jul-2025)

[3] G. Moterani and W. R. Lin, "Breaking the Linear Barrier: A Multi-Modal LLM-Based System for Navigating Complex Web Content," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, Jul. 2025, pp. 2066-2075. doi: 10.1109/COMPSAC65507.2025.00289

[4] Caffagni, D., Cocchi, F., Barsellotti, L., Moratelli, N., Sarto, S., Baraldi, L., Baraldi, L., Cornia, M. and Cucchiara, R. (2024). The (R)Evolution of Multimodal Large Language Models: A Survey. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.2402.12451.

[5] M. Perera, S. Ananthanarayan, C. Goncu, and K. Marriott, "The Sky is the Limit: Understanding How Generative AI can Enhance Screen Reader Users' Experience with Productivity Applications," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, Apr. 2025, pp. 1-17. doi: 10.1145/3706598.3713634

[6] Satwik Ram Kodandaram, Utku Uckun, Xiaojun Bi, I. V. Ramakrishnan, Vikas Ashok, et al., "Enabling Uniform Computer Interaction Experience for Blind Users through Large Language Models," *Proc. ACM SIGACCESS Conf. Computers and Accessibility (ASSETS '24)*, Oct. 2024. doi: 10.1145/3663548.3675605 ACM Digital Library+1

[7] S. Haque and C. Csallner, "Early Accessibility: Automating Alt-Text Generation for UI Icons During App Development," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2504.13069 (accessed: 02-Sep-2025)

[8] T. Çelik, "AI-Driven Production in Modular Architecture: An Examination of Design Processes and Methods," *Computer and Decision Making*, vol. 1, pp. 320-339, Nov. 2024. doi: 10.59543/comdem.v1i.10825

[9] Xu, S. (2025). Facilitating Visual Media Exploration for Blind and Low Vision Users through AI-Powered Interactive Storytelling. [online] arXiv.org. Doi: https://arxiv.org/abs/2508.03061.

[10] L. Y. Wen, C. Morrison, M. Grayson, R. F. Marques, D. Massiceti, C. Longden, & E. Cutrell, "Find My Things: Personalized Accessibility through Teachable AI for People who are Blind or Low Vision," *Extended Abstracts of CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1-6. doi: 10.1145/3613905.3648641

[11] N. Chen, L. K. Qiu, A. Z. Wang, Z. Wang, & Y. Yang, "Screen Reader Users in the Vibe Coding Era: Adaptation, Empowerment, and New Accessibility Landscape," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2506.13270

[12] M. Das, A. J. Fiannaca, M. R. Morris, S. K. Kane, & C. L. Bennett, "From Provenance to Aberrations: Image Creator and Screen Reader User Perspectives on Alt Text for AI-Generated Images," in *Proc. CHI Conf. Hum. Factors in Computing Systems*, 2024, pp. 1-21. doi: 10.1145/3613904.3642325

[13] X. Deng, P. Shiralkar, C. Lockard, B. Huang, & H. Sun, "DOM-LM: Learning Generalizable Representations for HTML Documents," *arXiv*, Jan. 25, 2022. doi: 10.48550/arXiv.2201.10608

[14] A. Forootan, "Empowering Agentic Non-Visual Web Navigation Through Tactile Controls and AI Support," *Open Research Repository, OCAD University*, 2025. [Online]. Available: https://openresearch.ocadu.ca/id/eprint/4779/7/Empowering%20Agentic%20Non-Visual%20Web%20Navigation%20Through%20Tactile%20Controls%20and%20AI%20Support-Amin%20Forootan.pdf

[15] R. E. Gonzalez, J. Collins, C. Bennett, & S. Azenkot, "Investigating Use Cases of AI-Powered Scene Description Applications for Blind and Low Vision People," *arXiv*, 2024. doi: 10.1145/3613904.3642211

[16] M. Holmlund, "Evaluating ChatGPT's Effectiveness in Web Accessibility for the Visually Impaired," *DIVA*, 2024. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1872759

[17] C. Kearney-Volpe and A. Hurst, "Accessible Web Development," *ACM Trans. Accessible Comput.*, vol. 14, no. 2, pp. 1-32, 2021. doi: 10.1145/3458024

[18] B. Leporini, M. Buzzi, and M. Hersh, "Video conferencing tools: Comparative study of the experiences of screen reader users and the development of more inclusive design guidelines," *ACM Trans. Accessible Comput.*, vol. 16, no. 1, 2022. doi: 10.1145/3573012

[19] T. Liu, P. Fazli, & H. Jeong, "Artificial Intelligence in Virtual Reality for Blind and Low Vision Individuals: Literature Review," *Proc. Human Factors and Ergonomics Society Annual Meeting*, 2024. doi: 10.1177/10711813241266832

[20] J. Nino, S. Ochoa, J. Kiss, G. Edwards, E. Morales, J. Hutson, F. Poncet, & W. Wittich, "Assistive Technologies for Internet Navigation: A Review of Screen Reader Solutions for the Blind and Visually Impaired," *Int. J. Recent Eng. Sci.*, vol. 11, no. 6, pp. 260-274, 2024. doi: 10.14445/23497157/ijres-v11i6p122

[21] A. Sharif, S. S. Chintalapati, J. O. Wobbrock, & K. Reinecke, "Understanding Screen-Reader Users' Experiences with Online Data Visualizations," in *23rd Int. ACM SIGACCESS Conf. Computers and Accessibility*, 2021. doi: 10.1145/3441852.3471202

[22] Oh, U., Joh, H., & Lee, Y., "Image Accessibility for Screen Reader Users: A Systematic Review and a Road Map," *Electronics*, vol. 10, no. 8, 2021, article 953. doi: 10.3390/electronics10080953

[23] A. S. Al-Subaihin, A. S. Al-Khlaifa, and H. S. Al-Khlaifa, "Accessibility of Mobile Web Apps by Screen Readers of Touch-Based Mobile Phones," in *Trends in Mobile Web Information Systems (MobiWIS 2013)*, Communications in Computer and Information Science, vol. 183, pp. 35-43, 2013. doi: 10.1007/978-3-319-03737-0_5