UC Irvine

Research Statement

Syed Fatiul Huq

When developers are tasked with creating a software application, accessibility is rarely considered as an essential requirement, despite how an inaccessible application can be difficult or downright impossible to use for the 16% of the global population who experience some form of disability [11]. Attributed to a lack of organizational prioritization and pedagogical awareness, software inaccessibility has been addressed with engineering techniques that automatically detect inaccessible components in web, mobile and desktop applications. Built on rules and heuristics, such techniques are easily scalable, and effective in passing compliance checks. *But how well do they represent what disabled people actually experience?*

My research addresses this question and builds LLM-based approaches that put disabled people's experience at the center of accessibility testing, using AI to streamline their insight rather than replace it. Sitting at the intersection of software engineering, human-computer interaction and applied AI, my work combines empirical studies of practice, user studies with disabled testers, and experimentation with LLM coding agents that increasingly write the software people use. It has produced three first-author papers at premier human-computer interaction and software engineering venues (CHI and ICSE respectively), including an ICSE 2025 Distinguished Paper, and further collaborative papers at ICSE, FSE, ASE, and ASSETS.

Thread 1: How are accessible software developed?

To understand why the software development ecosystem produces inaccessible products, interview and survey studies have been conducted with software professionals [12], [14], [15]. Common obstacles included lack of awareness about accessibility concepts and best practices, along with a lack of incentive to put in the time and effort to learn and execute them. In "#A11yDev" (CHI '23) [10], I aimed to further our understanding of the space by looking into social media, Twitter specifically, where conversations on accessibility converged user and developer silos. Across three viewpoints of process, product and people, we found obstacles ranging from the individual to the organization, in development practices and institutional cultures. Conversely, an increasing advocacy for and training on accessibility in software companies, paired with the development of accessibility-focused design systems, frameworks and testing tools, shone a light on better roads ahead.

The most prevalent industry practice for incorporating accessibility, as observed in "#A11yDev", was the use of automated scanners for scalable accessibility testing. However, such tools underperformed in coverage and accuracy. As part of the SEAL lab at UCI, I collaborated in developing highly effective and specialized accessibility testing techniques. This included "over-accessibility" issues where Android app functionality became insecurely exposed through accessibility services (ASE '22) [13], hindrances caused by third-party advertisements embedded on Android app UI (ICSE '24) [9], locatability and actionability problems when navigating websites using screen readers (ASE '25) [8], and reflow-related issues in responsive web apps (FSE '26) [3]. To elevate the quality of our techniques, I led user studies with people with disabilities, who helped us recognize and characterize accessibility issues and evaluate the effectiveness of the developed tools.

These studies also inspired me to take a more systematic look at similar accessibility detection techniques in research, and bridge their claims with real users' feedback. I conducted a mixed-methods study combining a systematic literature review and user studies with blind individuals (CHI '26) [4].

Upon synthesis, a shared language was developed to characterize and categorize screen reader issues on Android. A total of 28 issue types emerged across five categories: labeling, navigation, activation, dynamic change and feedback. It also revealed key trends and limitations of automated interventions: heuristics are scalable, but fall short of semantics-reliant issues, and an upwards trend towards fidelity through dynamic GUI testing.

From our categorization, we developed MCAG (Mobile Content Accessibility Guidelines), as a mobile-focused companion for WCAG, which serves as a foundational guideline for web accessibility. It provides a systematic reference to understand accessibility barriers and find software solutions that address them.

In a recent data-mining study of Moodle's issue repository (ASSETS '26), my colleague and I examined how accessibility issues are reported and resolved in an open source project. We report similar observations, such as a dependence on automated auditing, along with new insights on what resources are used to describe, recreate, test and fix issues. Most importantly, we observe how dedicated organizational efforts to improve accessibility have not only benefited the software's accessibility, but reduced time and effort in resolving issues as well.

Thread 2: Can LLMs improve software accessibility?

Addressing the biggest gap of existing checkers — lack of subjective and contextual understanding — we built GenA11y (FSE '25) [6], an LLM-powered accessibility issue detection technique. GenA11y utilized the natural language, source code and image processing capabilities of LLMs. While prior techniques checked, for instance, whether an alt text existed for an image, GenA11y could diagnose whether the appropriateness and accuracy of existing alt text. GenA11y covers 37 WCAG success criteria, with future efforts to incorporate barriers caused by dynamically changing components.

As LLM technologies advanced to become autonomous coding agents, quickly democratizing the process of developing web applications, I asked whether this will lead to a more accessible internet or not. In a work currently under review [1], I conducted an empirical study with a set of three coding agents (Codex, Claude Code and OpenCode) and four models (GPT, Sonnet, DeepSeek and Qwen), engineering the harnesses for at-scale development and evaluating the accessibility of their generated websites. At the same time, I retrieved agent trajectories (sequence of reasoning and actions the agent executed) and processed patterns therein to glean a more in-depth understanding of agent behavior.

The results were less than ideal, as all of the websites included some number of accessibility violations. Color contrast issues persisted across all 90 AI generated websites, with feature-specific issue prevalence, such as labeling mismatch in forms, heading and table issues in textual content, and more.

The experiments then expanded to incorporate agentic skills, to see whether context and tooling dedicated to auditing and fixing accessibility can steer models towards accessible code generation. While still not perfect, a consistent improvement was observed, especially when employing specialized interventions, such as scripts that calculate on-screen color contrast. Analyzing agent trajectories also revealed a heightened reasoning about accessibility compared to no-skill runs, which influenced agent behavior. The results suggest a pathway towards accessible websites in the agentic era, by employing careful and purposive prompt, context and harness engineering.

Thread 3: Streamlining user feedback as accessibility reports

Despite the development of progressively improving automated testing techniques, none were able to emulate the full spectrum of accessibility issues experienced by actual users. "#A11yDev" reveal industry advocacy for increasing user testing. But user testing is considered too expensive and time-consuming by developers. Inspired by the need to make user testing more popular and attainable, and with access to LLMs as a technology capable of processing natural language, I built Reca11 (ICSE '25) [7].

In collaboration with Fable, an organization connecting software companies with disabled testers, I analyzed their user testing process to pinpoint aspects of the process that can be automated without removing the tester's feedback. Reca11 takes as input the transcript from a test session and, using LLMs with engineered prompts and contexts, generates a test report for developers. The results demonstrated that Reca11 can alleviate the effort of analyzing multiple hour long sessions, along with the expertise required to parse accessibility feedback and connect to the software under test. Furthermore, the generated reports not only covered expert-annotated entries (98% recall, 85% precision), but unearthed new ones missed by the former. This work received a Distinguished Paper Award and the findings inspired Fable to deploy a test report summarization tool on their platform [5]. The work also demonstrated the potential for shifting the paradigm of accessibility testing, and usability testing at large, to bring in human feedback into software development with the help of LLMs.

Continuing collaboration with Fable, I also led a study that compares the usability testing performance of testers with cognitive disabilities, against non-disabled testers. Currently under review [2], this work found that cognitive testers do not fall behind in conventional usability metrics, while providing more nuanced usability feedback. This study has also expanded my understanding of accessibility in the cognitive disabilities space.

Research agenda

1. Foreground disabled perspectives through software

My work on automating accessibility feedback from disabled testers has shown me the promise and importance of popularizing user testing as the next paradigm for software accessibility. The ultimate goal is to build a robust, scalable and accurate system, across boundaries of platform (web, mobile, desktop), disability (vision, motor, cognitive, etc.), and software artifact (product, code, design artifact).

The research core of this system will also evolve, from the current software engineering-focused design and implementation, to incorporating HCI methods and theories. For instance, participatory design, with disabled testers and developers, is necessary to build both sides of the system's interfaces: the tester-side recording tool, with attention to privacy, consent, and usability, and the developer-side interface, with a focus on creating actionable reports. The goal will shift from accuracy of the system towards its adaptability by the stakeholders.

2. Accessibility in the LLM era

LLMs now hold many roles in software engineering, each of which needs to be studied under the scope of accessibility. As *developers*, coding agents produce inaccessible websites. My empirical study on agents indicates that engineering harnesses with agentic interventions and analyzing their trajectories to find behavioral patterns are effective strategies to steer LLMs toward accessibility-aware development.

As *coding tools*, agents provide an ever evolving set of interfaces for programming: chatbots, IDE extensions, consoles and more. But in our accelerated churn, are we leaving behind disabled programmers, stretching again the disability gap? I will evaluate the accessibility implications of these interfaces, bringing in facets of trust, verifiability and creativity.

As *testers*, LLM-based detectors can judge systems based on semantic, contextual and multimodal input. Plugged to browser workers, LLMs can also simulate web navigation behavior and dynamically check for violations. But are they similarly vulnerable to the common pitfalls we found in our CHI '26 study [4]? I will investigate LLM-powered accessibility evaluators against human experiences and create ones that can complement tester feedback with scalable automation.

The common thread is that LLM's potential for enhancing accessibility in software use must be carefully validated and subsequently harnessed, while keeping disabled people's feedback as ground truth.

3. Human implications of software

Accessibility is my entry point into a broader question: whose experiences does software assume? My background in Bangladesh has made me acutely aware of how systems standing on the foundations of a different culture and language can further discriminate the already marginalized. With the rapid growth and adoption of expensive LLMs in our essential technologies, the promise of democratizing technological advancement in the era of AI may be misguided and ultimately counterproductive. The fundamentals of accessibility research has motivated and taught me the techniques of centering the human in developing and investigating software. I will inquire the quality of existing software systems and new ones built for and by LLMs under the dimensions of varying languages, cultural contexts, and economic and literacy levels.

In essence, I will pursue one question: *how can software be built and evaluated in ways grounded in the experiences of the people it serves?*

References

[1] **S. F. Huq**, M. A. Haider, A. J. Saifullah, T. Zimmermann, and S. Malek, "Vibing accessibly: Investigating the accessibility of websites generated by llm coding agents," Under review.

[2] **S. F. Huq**, K. Kalcevich, P. Pidathala, M. Asad, and S. Malek, "Web usability testing with people with cognitive disabilities: A mixed-methods empirical case for inclusion and accessible design," Under review.

[3] Y. He, Z. He, **S. F. Huq**, and S. Malek, "Reflair: Detecting responsive layout reflow issues using multimodal generative ai," in *Proceedings of the ACM on Software Engineering*, vol. 3, ACM New York, NY, USA, 2026, pp. 2905–2927.

[4] **S. F. Huq**, Z. He, Y. He, and S. Malek, "Bridging the gap between automated intervention and actual user experience: A mixed-methods study on mobile accessibility issues for screen reader users," in *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2026, pp. 1–27.

[5] K. Pereira. "AI Insights," Fable, Accessed: Oct. 13, 2026. [Online]. Available: https://makeitfable.com/tutorial/ai-insights/.

[6] Z. He, **S. F. Huq**, and S. Malek, "Enhancing web accessibility: Automated detection of issues with generative ai," in *Proceedings of the ACM on Software Engineering*, vol. 2, ACM New York, NY, USA, 2025, pp. 2264–2287.

[7] **S. F. Huq**, M. Tafreshipour, K. Kalcevich, and S. Malek, "Automated generation of accessibility test reports from recorded user transcripts," in *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, IEEE, 2025.

[8] S. Jain, **S. F. Huq**, Z. He, and S. Malek, "Automated detection of web application navigation barriers for screen reader users," in *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2025, pp. 1906–1918.

[9] Z. He, **S. F. Huq**, and S. Malek, "'i tend to view ads almost like a pestilence': On the accessibility implications of mobile ads for blind users," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[10] **S. F. Huq**, A. Alshayban, Z. He, and S. Malek, "#A11ydev: Understanding contemporary software accessibility practices from twitter conversations," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–18.

[11] W. H. Organization, *Disability fact sheet - who*, 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/disability-and-health.

[12] T. Bi, X. Xia, D. Lo, J. Grundy, T. Zimmermann, and D. Ford, "Accessibility in software practice: A practitioner's perspective," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–26, 2022.

[13] F. Mehralian, N. Salehnamadi, **S. F. Huq**, and S. Malek, "Too much accessibility is harmful! automated detection and analysis of overly accessible elements in mobile apps," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–13.

[14] D. Miranda and J. Araujo, "Studying industry practices of accessibility requirements in agile development," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 1309–1317.

[15] S. Azenkot, M. J. Hanley, and C. M. Baker, "How accessibility practitioners promote the creation of accessible products in large companies," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–27, 2021.