International Journal of Engineering & Scientific Research
Vol. 13 Issue 4, May 2025
ISSN: 2347-6532 Impact Factor: 6.660
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Implementing Web Accessibility in Enterprise Applications

Seema B Shrikant

Abstract

As web applications continue to grow in complexity and reach, accessibility is no longer a design consideration - it is a core engineering responsibility. This paper presents a practical approach to implementing the Web Content Accessibility Guidelines (WCAG) across large-scale enterprise platforms, with a focus on version 2.1 and considerations for evolving standards introduced in WCAG 2.2. Drawing from real-world experience in financial technology systems, we explore strategies for improving keyboard navigation, screen reader compatibility, and visual contrast, while addressing the challenges of retrofitting accessibility into legacy codebases. We also discuss the integration of automated validation tools and the role of semantic markup in building inclusive systems. Our proposed framework demonstrates how accessibility, when embedded into development workflows and architecture, can lead to more resilient, user-centered digital products that adapt to future standards.

Copyright © 2025 International Journals of Multidisciplinary Research Academy. *All rights reserved.*

Keywords:

Web Accessibility
WCAG 2.1
Inclusive Design
Semantic HTML
Keyboard Navigation
Accessibility Engineering

Author correspondence:

Seema B Shrikant,

Product Manager, Visa Inc., Austin-TX, USA

Email: seemabshrikant@gmail.com

1. Introduction

Accessibility in digital systems is often treated as a downstream design concern, addressed late in the development process or during compliance audits. However, as web applications scale in complexity and global reach, accessibility must be recognized as a foundational engineering responsibility. According to the World Health Organization, over one billion people - approximately 16% of the global population - live with some form of disability, including visual, auditory, motor, and cognitive impairments [1]. As governments, banks, healthcare providers, and educational institutions increasingly digitize their services, the need for inclusive systems that can serve diverse populations around the world has become more critical than ever.

Beyond ethical considerations, accessibility is also a legal and economic imperative. Countries across the world are enacting or strengthening digital accessibility legislation, including the Americans with Disabilities Act (United States), the Equality Act (United Kingdom), the European Accessibility Act (European Union), and similar laws across Asia

and Australia. These policies reflect a global recognition that digital exclusion has serious implications - not only for individuals, but for society at large.

The Web Content Accessibility Guidelines (WCAG), developed by the World Wide Web Consortium, offer a comprehensive framework for making web content accessible to people with disabilities across varied cultural and technological contexts. These guidelines are organized into four core principles - perceivable, operable, understandable, and robust - which collectively serve as the foundation for inclusive design and development practices. This paper draws primarily on version 2.1 of the WCAG standard, which was implemented in a range of enterprise-level web applications serving users in North America, Europe, and Asia.

2. Accessibility Standards and Industry Practices

The pursuit of digital accessibility has become a global imperative as web-based platforms increasingly serve as primary channels for public services, commerce, education, and financial systems. According to the World Health Organization, more than one billion individuals - approximately sixteen percent of the global population - experience some form of disability [1]. This demographic includes users with permanent, temporary, and situational impairments, each of whom may rely on assistive technologies to engage with digital content. Consequently, accessibility is not solely a matter of social inclusion; it is a critical engineering and legal requirement in modern software development.

The most widely accepted framework for addressing accessibility in web systems is the Web Content Accessibility Guidelines (WCAG), developed by the World Wide Web Consortium (W3C). These guidelines are internationally recognized and serve as the foundation for most digital accessibility legislation. WCAG is organized around four fundamental principles: perceivable, operable, understandable, and robust. These principles ensure that digital content can be accessed and interpreted by users with diverse needs and assistive technologies.

This study is grounded in the implementation of WCAG 2.1, which expanded upon WCAG 2.0 by introducing new criteria aimed at improving access for users with low vision, cognitive limitations, and those using mobile devices. Key additions included success criteria for non-text contrast, gesture support, and input mechanisms. These enhancements

required significant modifications to the frontend architecture of large-scale web applications, particularly those with shared component libraries and legacy design patterns.

In October 2023, the W3C published WCAG 2.2, introducing nine new success criteria with a focus on mobile accessibility and cognitive support [2]. These updates include improvements to focus visibility, input target sizing, and mechanisms for reducing redundant user input. While this paper centers on engineering solutions aligned with WCAG 2.1, the methodologies described are designed to be forward-compatible with the evolving requirements of WCAG 2.2.

In addition to international standards, numerous jurisdictions have enacted legislation mandating accessibility compliance, often referencing WCAG as the benchmark. The Americans with Disabilities Act (ADA) in the United States, the European Accessibility Act (EAA) in the European Union, and similar frameworks in Canada, Australia, and India illustrate the widespread regulatory alignment around digital inclusion. These laws have made accessibility compliance a strategic concern for organizations operating globally.

To support the implementation of these standards, engineering teams frequently employ automated testing tools such as Lighthouse and Axe, which evaluate web content against WCAG guidelines and generate diagnostic reports [3][4]. While these tools are valuable for initial compliance assessment, they are limited in their ability to detect more complex issues - such as incorrect tab order, improper use of Accessible Rich Internet Applications (ARIA) attributes, or poor screen reader behavior. Comprehensive validation therefore requires a combination of automated and manual testing methodologies.

Despite the availability of tools and standards, organizations continue to face challenges in scaling accessibility. Legacy systems often exhibit poor semantic structure, non-compliant visual elements, and interaction models that are incompatible with assistive technologies. Retrofitting these systems necessitates architectural refactoring, component redesign, and the incorporation of accessibility as a core engineering principle.

This paper presents an applied engineering framework for achieving scalable accessibility within enterprise systems. The approach integrates WCAG 2.1 compliance into the software development lifecycle, supported by semantic markup, reusable component patterns, automation, and continuous validation. The proposed methodology is informed by

practical implementation experience and is adaptable to future standards, including WCAG 2.2.

3. Methodology

This section outlines the engineering methodology employed to implement accessibility improvements across a large-scale web platform. The objective was to align the system with the Web Content Accessibility Guidelines (WCAG) 2.1 while ensuring scalability, maintainability, and forward compatibility with WCAG 2.2. The methodology was structured around four key domains: semantic markup, keyboard accessibility, screen reader support, and contrast compliance. These initiatives were executed in conjunction with the development of reusable interface components and the integration of accessibility checks into continuous integration pipelines.

3.1 Semantic Structure and HTML Refactoring

The foundation of the accessibility enhancement process involved enforcing semantic HTML across all user interface components. Many legacy elements were constructed using div or span tags with nested event handlers, which are not inherently accessible to assistive technologies. These elements were replaced with appropriate HTML5 elements, such as button, nav, header, main, and section, which are natively recognized by screen readers and provide more predictable behavior for keyboard navigation.

In parallel, redundant or misused Accessible Rich Internet Applications (ARIA) attributes were removed or corrected to prevent conflicting announcements or inaccurate element roles. For custom components such as modal dialogs, carousels, and dropdown menus, proper ARIA roles (e.g., dialog, menu, listbox) and state attributes (e.g., aria-expanded, aria-hidden) were introduced to ensure dynamic content could be interpreted correctly by assistive technologies.

3.2 Keyboard Navigation and Focus Management

Keyboard accessibility was addressed by designing all interactive components to be operable using only the keyboard. Tab order was explicitly defined to ensure logical progression through input fields, buttons, and links. Focus indicators were made visible and consistent across all components, and custom elements - such as sliders and accordions - were augmented with key event listeners to support navigation via the arrow keys, escape key, and spacebar.

Special attention was given to managing focus within modal interfaces and dynamic overlays. When a modal is activated, focus is programmatically moved to the modal container and trapped until the user closes the interface. Upon dismissal, focus returns to the previously active element, preserving navigation context. This behavior was achieved using JavaScript focus management and ARIA attributes such as aria-modal and aria-labelledby.

3.3 Screen Reader Compatibility

Screen reader accessibility was tested and enhanced using semantic elements and appropriate labeling techniques. For form fields, the label element was associated directly with input elements via the for and id attributes. Where visual design constraints prevented visible labels, aria-label and aria-labelledby attributes were used to provide descriptive metadata for screen readers.

In dynamic interfaces where content updates without a full page reload, live region attributes such as aria-live="polite" and aria-atomic="true" were employed to announce changes without disrupting the user experience. For example, when a user applied a filter or added an item to a cart, screen reader users received immediate feedback without losing focus context.

3.4 Visual Contrast and Design Tokens

Color contrast was evaluated using the WCAG-defined contrast ratio formula, which calculates the luminance difference between foreground text and background elements. All interactive and informational elements were tested against the minimum thresholds of 4.5:1 for normal text and 3:1 for large-scale text. Visual styles were enforced through a design token system, wherein predefined variables governed color usage across themes and components. This allowed for consistent enforcement of contrast ratios and simplified future updates.

Interactive states - such as hover, focus, and active - were also updated to meet contrast requirements, ensuring users with visual impairments could identify clickable or focusable elements through color differentiation, border outlines, or iconography. To ensure compliance with WCAG 2.1 contrast requirements, all interactive elements were evaluated using minimum contrast thresholds of 4.5:1 for normal text and 3:1 for large-scale text. A standardized color system was used to enforce visual consistency and accessibility across the platform. Figure 1 illustrates a common improvement: replacing low-contrast buttons with accessible alternatives that meet WCAG guidelines.

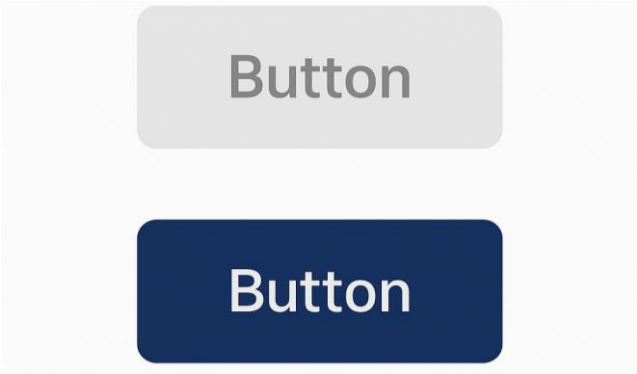

Figure 1: *Visual comparison of button contrast before and after accessibility improvements. The upper button fails WCAG contrast requirements due to low text-to-background contrast, while the lower button demonstrates sufficient contrast for readability and compliance*.

3.5 Continuous Integration and Automation

To ensure consistency and long-term maintainability, automated accessibility checks were incorporated into the software development lifecycle. The Axe accessibility testing engine was integrated into the continuous integration (CI) pipeline to scan modified code for violations of WCAG 2.1. Pull requests that introduced regressions or non-compliant patterns were flagged automatically, enabling developers to address issues before deployment.

In addition, Lighthouse reports were used as part of post-deployment validation to assess real-world accessibility scores across critical user flows. Manual exploratory testing was scheduled during each release cycle to supplement automated scans, using screen readers such as NVDA (NonVisual Desktop Access) and VoiceOver.

Accessibility Area	**Engineering Focus**	**WCAG Reference**
Semantic HTML	Replaced non-semantic tags with HTML5 elements	1.3.1, 4.1.2
Keyboard Navigation	Tab order, focus indicators, event handling	2.1.1, 2.4.3
Screen Reader Support	Labels, ARIA attributes, live regions	1.3.1, 4.1.2
Contrast Compliance	Luminance ratio calculation, design tokens	1.4.3, 1.4.11

Table 1: *Accessibility Enhancements by Domain*

4. Results

The accessibility enhancements led to measurable improvements in platform compliance, user experience, and development quality. This section presents the outcomes based on audit scores, component refactoring, usability observations, and validation through assistive technology testing.

4.1 Accessibility Audit Scores

Post-implementation audits conducted using Lighthouse indicated substantial improvements in accessibility scores across all major user flows. Pages that previously exhibited common issues - such as insufficient contrast, missing form labels, and inconsistent focus behavior - achieved scores consistently above 90 out of 100. These improvements were validated across multiple browsers and device types, including desktop and mobile platforms.

4.2 Component and Module Refactoring

As part of the initiative, accessibility updates were applied across more than 50 distinct interface modules. These included high-traffic workflows, interactive components, and shared UI elements. Updates involved applying semantic HTML, ensuring keyboard operability, correcting ARIA usage, and updating visual design tokens for contrast compliance. The revised components now serve as standardized templates for new development, supported by internal accessibility guidelines.

4.3 Improved Usability and Developer Adoption

Following the rollout, internal usability testing with keyboard-only navigation and screen readers demonstrated notable improvements in navigability, content comprehension, and user flow completion. Key tasks - including form submissions, modal interactions, and content filtering - were completed without assistive barriers. In parallel, engineering teams adopted the accessibility framework and checklist as part of routine development and code review processes, improving the consistency of accessible coding practices across teams.

4.4 Assistive Technology Validation

Manual testing was conducted using popular screen readers such as NonVisual Desktop Access (NVDA) and VoiceOver. Pages were successfully navigated using only the keyboard, with appropriate announcements for headings, buttons, input fields, and dynamic content updates. Modal dialogs correctly implemented focus trapping and restoration, and

alert messages were reliably announced through live regions. These results confirmed that accessibility improvements were effective not only in automated audits but also in real-world usage scenarios involving assistive technologies.

Tool	Type	Platform	Scope
Lighthouse	Automated	Web, CI	WCAG audits, scores, performance overview
NVDA	Manual	Windows	Screen reader testing
VoiceOver	Manual	macOS, iOS	Screen reader and mobile validation

Table 2: *Testing Tools Used and Their Scope*

5. Discussion

The implementation of accessibility standards within a large-scale enterprise system requires a balance between technical feasibility, development velocity, and long-term maintainability. While the results of this initiative indicate clear improvements in compliance, usability, and developer engagement, several insights emerged that are relevant to broader engineering contexts.

First, the integration of accessibility into the software development lifecycle proved more effective than isolated remediation efforts. Embedding semantic standards, keyboard operability, and design constraints into shared component libraries not only reduced the likelihood of recurring issues but also normalized accessible practices among engineering teams. However, establishing these conventions required deliberate coordination across product, design, and quality assurance roles.

Second, although automated tools such as Lighthouse and Axe accelerated compliance verification, they were insufficient for detecting context-sensitive issues such as improper use of ARIA attributes, screen reader misbehavior, or illogical focus patterns. Manual validation - particularly with assistive technologies - was essential to uncovering these edge cases and ensuring a robust user experience.

Finally, while the methodology described in this paper was developed in alignment with WCAG 2.1, it was intentionally designed to be adaptable. The release of WCAG 2.2 underscores the need for systems that can evolve with changing accessibility standards. The modular and testable nature of the accessibility framework implemented during this initiative positions the platform to accommodate future requirements with minimal rework.

These observations suggest that accessibility, when treated as an engineering discipline rather than an external constraint, contributes meaningfully to product quality, resilience, and inclusivity.

6. Conclusion

This paper presented a structured engineering methodology for implementing accessibility within a large-scale web platform, guided by the principles of the Web Content Accessibility Guidelines (WCAG) 2.1. The initiative involved systematic updates across interface modules, including semantic HTML refactoring, keyboard operability, screen reader compatibility, and contrast ratio enforcement. These improvements were integrated into shared component libraries and validated through both automated tools and manual testing with assistive technologies.

The results demonstrate that accessibility, when embedded into foundational architecture and development workflows, can be scaled effectively across complex digital systems. Audit scores improved significantly, usability for assistive technology users increased, and development teams adopted sustainable accessibility practices. Furthermore, the engineering approach was structured to support forward compatibility with WCAG 2.2 and future iterations of accessibility standards.

Future work will focus on expanding accessibility support to mobile-specific workflows, integrating localization and multilingual accessibility considerations, and increasing user feedback loops with individuals who rely on assistive technologies. As accessibility guidelines evolve, ongoing alignment with emerging best practices will be necessary to ensure inclusive access for all users, regardless of device, ability, or context.

References

[1] World Health Organization, *Disability and Health*, 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/disability-and-health

[2] World Wide Web Consortium (W3C), *Web Content Accessibility Guidelines (WCAG) 2.2*, Oct. 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/

[3] Deque Systems, *Axe: Accessibility Testing Engine*. [Online]. Available: https://www.deque.com/axe/

[4] Google Developers, *Lighthouse: Automated Tool for Improving the Quality of Web Pages*. [Online]. Available: https://developer.chrome.com/docs/lighthouse/

[5] United States Department of Justice, *Americans with Disabilities Act (ADA)*. [Online]. Available: https://www.ada.gov/

[6] European Commission, *European Accessibility Act*, 2019. [Online]. Available: https://ec.europa.eu/social/main.jsp?catId=1202

[7] WebAIM, *Contrast Checker*. [Online]. Available: https://webaim.org/resources/contrastchecker/

[8] Mozilla Developer Network, *ARIA: Accessible Rich Internet Applications*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA

[9] WebAIM, *WAVE Web Accessibility Evaluation Tool*. [Online]. Available: https://wave.webaim.org/

[10] NV Access, *NonVisual Desktop Access (NVDA)*. [Online]. Available: https://www.nvaccess.org/

[11] Apple Inc., *VoiceOver for macOS and iOS*. [Online]. Available: https://www.apple.com/accessibility/mac/vision/