Emperor International Journal of Applied Scientific Research
ISSN: 2581-964X Mayas Publication® www.mayas.info
Volume-VIII Issue-VII July 2026

A Semantic-First Methodology for Inclusive Web Design: A Four-Stage Framework for Reducing Navigational Barriers in Visually Impaired User Interfaces

Manav Chawla
Department of Computer Science & Engineering,
Vaish College of Engineering Maharshi Dayanand University,
Rohtak, Haryana, India

Dr. Bijender Bansal
Department of Computer Science & Engineering,
Vaish College of Engineering Maharshi Dayanand University,
Rohtak, Haryana, India

Dr. Vineet Kumar Lohan
Department of Computer Science & Engineering,
Vaish College of Engineering Maharshi Dayanand University,
Rohtak, Haryana, India

Dr. Deepak Goyal
Department of Computer Science & Engineering,
Vaish College of Engineering Maharshi Dayanand University,
Rohtak, Haryana, India

Abstract

Modern web accessibility research has extensively documented the barriers that visually impaired users face when navigating Single Page Application (SPA) interfaces. However, existing approaches to barrier remediation remain predominantly reactive, addressing individual WCAG compliance violations rather than the underlying structural causes of navigational inefficiency. This paper proposes a **Semantic-First Methodology** comprising four iterative stages Heuristic Semantic Audit, Structural Deconstruction, ARIA-Native Hybridization, and Navigational Parity Validation designed to systematically address the root causes of screen reader navigational barriers in complex web interfaces. The methodology was applied to a test dataset of five high-complexity UI components, and its effectiveness was evaluated through comparative pre- and post-implementation measurement of

task completion time, task success rate, and the Navigational Efficiency Gap (NEG) metric. Preliminary results indicate an average reduction of approximately 74% in the NEG across the five tested components, a reduction in average task completion time for screen reader users from 94.6 seconds to 20.4 seconds, and achievement of WCAG 2.2 Level AA compliance on all tested components following methodology application. A structured comparison with three existing accessibility approaches demonstrates that the proposed methodology addresses barrier categories that compliance-only and component-level approaches do not cover. Limitations of the study, including testing environment scope and the preliminary nature of the evaluation, are explicitly discussed.

Keywords: Web Accessibility; Inclusive Design; Screen Reader; WCAG 2.2; Semantic HTML; ARIA; Single Page Applications; Navigational Efficiency Gap; Focus Management; Barrier Remediation; Visually Impaired User.

I.INTRODUCTION

The challenge of web accessibility for visually impaired users has been the subject of sustained research attention for over two decades. Despite the existence of established guidelines, notably the Web Content Accessibility Guidelines (WCAG 2.2) [1], and a growing body of literature documenting the nature and severity of accessibility barriers, the practical navigational experience of screen reader users on modern web environments remains significantly degraded relative to that of sighted users [2, 3].

A fundamental limitation of the current dominant approach to web accessibility is its reactive and compliance-centric character. Accessibility is typically addressed through post-development auditing, in which automated tools scan deployed web pages for WCAG success criteria violations and developers apply targeted fixes to the identified violations. While this approach can achieve technical compliance, it does not address the structural root causes of navigational inefficiency for screen reader users, which are frequently embedded in the architectural decisions made during the initial design and development of a web interface [4].

The transition of modern web development toward JavaScript-driven Single Page Application (SPA) frameworks such as React, Angular, and Vue has intensified this problem. These frameworks encourage component-based development patterns in which UI elements are constructed from semantically empty HTML div and span containers, systematically removing the native HTML5 semantic properties that browsers communicate to the Accessibility Tree. The resulting structural condition commonly described as a Semantic Void cannot be fully addressed through post-development patching of individual WCAG violations, as it represents a pervasive architectural characteristic rather than a collection of isolated defects [5].

This paper addresses the gap between compliance-focused reactive approaches and the structural remediation required to meaningfully improve screen reader navigational efficiency in SPA environments. A Semantic-First Methodology is proposed, comprising four iterative stages that target the root causes of navigational barriers at the architectural level. The methodology is evaluated through controlled pre- and post-implementation comparison on a dataset of five representative high-complexity UI components, and compared against three existing accessibility approaches to establish its relative contribution.

The remainder of this paper is organized as follows. Section II reviews related work on existing accessibility remediation approaches. Section III presents the proposed four-stage methodology. Section IV describes the experimental evaluation design. Section V presents results and comparative analysis. Section VI discusses limitations. Section VII provides recommendations. Section VIII concludes the paper.

RELATED WORK

Existing approaches to web accessibility remediation can be broadly categorized into three types: compliance-based auditing, component-level design systems, and overlay-based tools. Each approach addresses a subset of the accessibility barrier landscape, and each has documented limitations that the methodology proposed in this paper seeks to address.

A. Compliance-Based Auditing Approaches

The dominant approach to web accessibility remediation is the application of automated WCAG compliance auditing tools, including Axe-Core, WAVE, and Google Lighthouse. Vigo et al. [6] established through empirical benchmarking that automated tools reliably detect approximately 30 to 40 percent of actual accessibility barriers, with the undnot reliably predict a reduction in the problems encountered by blind users in real task scenarios, providing empirical evidence that compliance auditing and praetected proportion consisting predominantly of behavioral and contextual barriers that require dynamic interaction to manifest. Power et al. [7] demonstrated that conformance to WCAG Level A does ctical navigational usability are not equivalent.

The fundamental limitation of compliance-based approaches is their focus on individual element-level violations rather than the global structural and behavioral architecture of the interface. An interface can satisfy every applicable WCAG success criterion while retaining focus management failures, missing live region announcements, and semantic structure deficiencies that collectively produce a severely degraded screen reader navigation experience [3].

B. Component-Level Design System Approaches

A second category of approach addresses accessibility at the component design level. IBM's Carbon Design System [8] and Microsoft's Inclusive Design Toolkit [9] provide pre-built, accessibility-verified UI components and documented keyboard interaction patterns that development teams can adopt as the building blocks of new interfaces. These resources represent a significant advancement over purely compliance-based approaches, as they embed accessibility properties into the component architecture rather than applying them as post-development corrections.

The limitation of component-level design system approaches is their scope: they address the design of individual components in isolation and are applicable primarily to new development using the specific design system's component library. They do not provide a methodology for evaluating or remediating the global navigational architecture of existing arbitrary SPA interfaces, and they do not address the interaction between components as a user navigates across an entire interface in the course of a realistic task.

C. Overlay-Based Approaches

Accessibility overlay tools represent a third category, providing automated JavaScript-based modifications to deployed web pages intended to improve screen reader compatibility without requiring source code changes. Overlay tools have been critically evaluated in accessibility research and practitioner literature [10]. Their primary limitation is that JavaScript-based modifications cannot reliably reconstruct the native HTML semantic structure and browser-level Accessibility Tree properties that are absent from Div-heavy SPA component architectures, and may in some cases introduce additional auditory clutter or focus management conflicts that further degrade the screen reader experience.

D. Gap Addressed by This Paper

The review of existing approaches identifies a consistent gap: no established methodology targets the structural root causes of SPA-specific screen reader navigational inefficiency at the architectural level, through a systematic process applicable to existing arbitrary interfaces. The Semantic-First Methodology proposed in this paper is designed to address this gap.

Proposed Methodology

The Semantic-First Methodology comprises four iterative stages applied sequentially to each target UI component or interface section. The methodology is guided by the principle of Programmatic Determinism: for every dynamic visual change in the interface, there must be a corresponding, deterministic, and correctly sequenced change in the information communicated to the Accessibility Tree. The four stages are described below.

A. ***Stage 1: Heuristic Semantic Audit***

The first stage establishes a quantitative baseline by mapping the visual hierarchy of the interface against its programmatic hierarchy as exposed in the Accessibility Tree. The audit employs a dual-layer observation approach: Chrome DevTools Accessibility Tree Inspection is used to verify computed accessibility properties for each interactive element in real time, while Axe-Core v4.8 is integrated as a runtime monitor during active navigation sessions to capture dynamic barrier instances that static code analysis cannot detect.

The audit produces two outputs for each evaluated component: a baseline Navigational Efficiency Gap (NEG) value, calculated as the ratio of mean screen reader task completion time to mean sighted user task completion time on an identical task, and a barrier inventory classifying identified deficiencies into four categories: Semantic Mismatch, Focus Fragmentation, Auditory Clutter, and State Synchronization Failure. The baseline NEG value and barrier inventory together constitute the functional specification that guides the subsequent remediation stages.

B. ***Stage 2: Structural Deconstruction***

The second stage targets Semantic Mismatch barriers through systematic refactoring of the component's HTML structure. The core operation of this stage is the replacement of semantically empty div and span container elements with semantically appropriate native HTML5 elements. Generic div wrappers serving as page region containers are replaced with landmark elements including main, nav, header, footer, and aside. Generic div elements serving as interactive controls are replaced with native button, select, and input elements as appropriate to their function.

This stage is guided by the hierarchy of native semantics principle: native HTML5 elements carry built-in accessibility properties including automatic keyboard support, built-in ARIA roles, and native state management that are communicated reliably across browser-screen reader combinations without requiring additional ARIA attributes. The adoption of native elements as the foundational layer reduces both the quantity of ARIA attributes required and the risk of attribute conflicts that produce Auditory Clutter. The output of this stage is a refactored component structure in which the programmatic hierarchy of the interface, as represented in the Accessibility Tree, correctly reflects the visual and functional hierarchy of the interface.

C. ***Stage 3: ARIA-Native Hybridization***

The third stage addresses barrier categories that cannot be resolved through native HTML5 element substitution alone, specifically State Synchronization Failure and Focus Fragmentation in dynamic, JavaScript-driven contexts. At the points

where native HTML5 elements reach their functional limits in representing complex dynamic UI patterns, WAI-ARIA attributes are applied as a targeted metadata layer.

Three specific technical interventions are applied in this stage. First, Focus Management utilities are implemented for all dynamic state transitions: when a modal dialog opens, focus is programmatically moved to the dialog container or its first interactive element; when the dialog closes, focus is returned to the triggering element. When a SPA route transition occurs, focus is moved to the new page's primary heading or main landmark. Second, ARIA Live Regions are applied to interface regions that receive dynamic content updates, using aria-live="polite" for non-critical updates such as search result count changes and aria-live="assertive" for time-sensitive notifications such as form validation errors. Third, ARIA relationship attributes including aria-describedby, aria-labelledby, and aria-controls are applied to establish programmatic associations between related elements that are visually proximate but structurally separated in the DOM.

D. ***Stage 4: Navigational Parity Validation***

The fourth stage re-evaluates the refactored component against the same metrics recorded in Stage 1 to quantify the improvement achieved. Automated re-auditing using Axe-Core and Lighthouse verifies WCAG 2.2 Level AA compliance. Manual screen reader walkthrough using NVDA verifies the qualitative navigational experience, specifically confirming correct focus management behavior, appropriate live region announcements, and absence of auditory clutter. The post-implementation NEG value is calculated using the same task scenario and measurement protocol as the Stage 1 baseline.

The methodology is considered to have achieved its objective for a given component when two conditions are satisfied: the post-implementation NEG value represents a reduction of at least 60% relative to the baseline value, and the component achieves a zero-violation status in both automated Axe-Core scanning and manual NVDA walkthrough. Components that do not satisfy both conditions are returned to Stage 2 for further structural refinement before re-evaluation.

Experimental Design

A. ***Test Dataset***

The methodology was applied and evaluated on the same five UI component types used for baseline measurement: a Mega Menu navigation structure (DS-01), a Dynamic AJAX Search Bar (DS-02), a Multi-Step Progress Form (DS-03), a Sortable Data Table (DS-04), and a Modal Dialog Overlay (DS-05). Each component was sourced from publicly available web applications that had received passing scores in Axe-Core automated audits prior to evaluation, ensuring that the baseline represents real-world compliance-passing interfaces.

B. ***Evaluation Metrics***

Three metrics were recorded for each component before and after methodology application. The Navigational Efficiency Gap (NEG) is defined as the ratio of mean screen reader task completion time to mean sighted user task completion time on an identical task scenario. The Task Success Rate (TSR) is the proportion of screen reader navigation sessions in which the user completed the assigned task without abandoning or requiring a page refresh. WCAG 2.2 Level AA Compliance Status is recorded as a binary pass/fail based on zero-violation Axe-Core audit results combined with zero-defect manual NVDA walkthrough.

C. ***Testing Environment***

All pre- and post-implementation evaluations were conducted in a standardized technical environment: NVDA v2024 screen reader with Google Chrome v120 on Windows 11 for screen reader sessions, and standard keyboard-and-mouse navigation with Google Chrome v120 for sighted user sessions. It is acknowledged that this represents a single browser-screen reader combination; findings may differ across JAWS with Edge or VoiceOver with Safari, which is discussed as a limitation in Section VI.

Results And Comparative Analysis

A. Pre- and Post-Implementation Results

Table I presents the quantitative results before and after application of the Semantic-First Methodology across all five test components. Given the preliminary and limited-scope nature of this evaluation, results are presented as indicative observations. Standard deviation values and formal statistical significance testing are not reported, reflecting the exploratory scope of this study, which is further discussed in Section VI.

I.TABLE

PRE- AND POST-IMPLEMENTATION COMPARISON OF NAVIGATIONAL EFFICIENCY METRICS

Component	NEG Before	NEG After	TSR Before	TSR After	WCAGAA
Main Menu (DS-01)	19.2×	4.1×	92%	98%	✓ Pass
Live Search (DS-02)	28.7×	6.3×	40%	92%	✓ Pass
Step Form	18.3×	4.2×	75%	92%	✓

(DS-03)					Pass
Data Table (DS-04)	17.5×	4.9×	55%	88%	✓ Pass
Modal Popup (DS-05)	23.3×	4.4×	85%	97%	✓ Pass
Average	21.4×	4.8×	69%	93%	100%

The observed post-implementation NEG values range from 4.1× to 6.3× across the five components, representing reductions from baseline values ranging from 17.5× to
28.7×. The average NEG reduction across all components is approximately 74%, exceeding the 60% threshold specified as the methodology's success criterion. Average task success rate increased from 69% to 93% across the dataset.

All five components achieved zero-violation status in both automated Axe-Core auditing and manual NVDA walkthrough following methodology application, indicating full WCAG 2.2 Level AA compliance.

The Live Search component (DS-02) recorded the largest absolute NEG reduction, from 28.7× to 6.3×, and the largest improvement in task success rate, from 40% to 92%. This component benefited most from the ARIA Live Region implementation in Stage 3, which addressed the primary barrier of silent AJAX result updates that had previously left screen reader users without feedback during dynamic filtering operations.

B. Task Completion Time Comparison

Table II presents the comparison of mean task completion times for screen reader users before and after methodology application, alongside sighted user times for reference.

II.TABLE

TASK COMPLETION TIME: SCREEN READER USERS BEFORE AND AFTER METHODOLOGY APPLICATION

Component	Sighted (s)	SR Before (s)	SR After (s)	Reduction
Main Menu (DS-01)	2.5	48.0	10.3	78.5 %
Live Search (DS-02)	4.0	115.0	25.2	78.1 %
Step Form (DS-03)	3.0	55.0	12.6	77.1 %
Data Table (DS-04)	8.0	140.0	39.2	72.0 %
Modal Popup (DS-05)	1.5	35.0	6.6	81.1 %
Average	3.8	78.6	18.8	78.5 %

The observed average task completion time for screen reader users reduced from 78.6 seconds before methodology application to 18.8 seconds after, representing an average reduction of approximately 78.5% across the five components. While post-implementation screen reader completion times remain above sighted user times in all cases, the residual NEG values of 4.1× to 6.3× suggest that some navigational overhead is an inherent characteristic of the sequential, auditory-feedback-dependent navigation model rather than a correctable interface deficiency.

C. Comparison with Existing Approaches

Table III presents a structured comparison of the Semantic-First Methodology against three existing accessibility approaches across six evaluation dimensions. The comparison is based on documented characteristics of each approach as reported in

the literature reviewed in Section II, and is intended to illustrate the relative positioning of the proposed methodology rather than to make quantitative superiority claims.

III.TABLE

COMPARISON OF EXISTING ACCESSIBILITY APPROACHES VS. PROPOSED METHODOLOGY

Feature / Criterion	Compliance Auditing	Design Systems	Overlay Tools	Proposed Methodology
Primary Focus	WCAG pass/fail	New components	Post-de ploy fix	Structural root cause
Timing in Lifecycle	Post-development	Design phase	Post-development	Shift-left integration
Detects Dynamic Barriers	Partial (∼35%)	Yes (new builds)	Partial	Yes (runtime monitoring)
Addresses Focus Management	No	Compo nent-lev el	Limited	Yes (all transitions)
Applicable to Existing SPAs	Yes	No	Yes	Yes
Measures User Efficiency	No	No	No	Yes (NEG metric)
WCAG 2.2 AA Achieved	Goal only	Yes (new builds)	Partial	Yes (all components)

The comparison indicates that the Semantic-First Methodology addresses several dimensions not covered by existing approaches, including the measurement of user-side navigational efficiency, the systematic targeting of dynamic barrier

categories in existing SPA interfaces, and the integration of both automated and manual evaluation within a structured iterative process. It is acknowledged that the comparison is based on documented approach characteristics rather than direct experimental comparison under identical conditions, which would be required to make stronger claims of superiority.

LIMITATIONS

This study has several important limitations that should be considered when interpreting the reported findings.

1) ***Testing Environment Scope***

All evaluations were conducted using a single browser-screen reader combination: NVDA v2024 with Google Chrome v120 on Windows 11. Prior research documents significant behavioral differences between NVDA, JAWS, and VoiceOver in how they interpret ARIA attributes and announce dynamic content [5, 11]. The reported NEG values and methodology effectiveness observations may not directly generalize to other screen reader and browser combinations. Future work should replicate the evaluation across JAWS with Microsoft Edge and VoiceOver with Safari to assess cross-environment consistency.

2) ***Sample Size and Statistical Rigor***

This study represents a preliminary evaluation with a limited number of component instances. Standard deviation values and formal statistical significance testing are not reported, as the exploratory scope of the study does not support statistically validated conclusions. The reported NEG values and task success rates should be interpreted as indicative preliminary findings. Future work should conduct a larger-scale study with a sufficient participant sample ($n \geq 20$ per group per component) and appropriate statistical analysis, including measures of variance and significance testing, to validate and extend these preliminary observations.

3) ***Component Selection Scope***

The five component types evaluated, while representative of common SPA UI patterns, do not constitute exhaustive coverage of all high-complexity component categories present in modern web applications. Date pickers, autocomplete inputs, virtualized lists, and drag-and-drop interfaces represent additional barrier categories not addressed in this study. Generalization of methodology effectiveness beyond the specific components evaluated should be made with caution.

4) ***Methodology Application Subjectivity***

The application of Stages 2 and 3 of the methodology involves developer judgment in selecting appropriate native HTML5 elements and ARIA attributes for specific components. Different developers applying the methodology to the same component may make different implementation decisions, potentially producing

different outcome metrics. Future work should develop more prescriptive implementation specifications to reduce inter-implementer variability.

RECOMMENDATIONS

Based on the preliminary findings of this study, the following recommendations are proposed for web development teams and accessibility practitioners. These are presented as evidence-informed guidance rather than prescriptive conclusions.

1) Native HTML5 as the Foundational Layer

The Structural Deconstruction stage of the methodology produced consistent barrier reductions across all tested components, suggesting that prioritizing native HTML5 semantic elements over custom div-based implementations as the default architectural choice may reduce the frequency and severity of screen reader accessibility failures. Development teams should consider establishing native element usage as a coding standard rather than an accessibility-specific requirement.

2) Programmatic Focus Management as a Development Requirement

Focus Fragmentation was observed as a barrier category across all five tested components before methodology application. The consistent post-implementation improvement following programmatic focus management implementation suggests that explicit focus destination specification for all dynamic state transitions should be treated as a core development requirement, comparable in priority to visual design correctness.

3) Hybrid Evaluation as Standard Practice

The observation that all pre-implementation components passed automated WCAG audits while recording substantial NEG values suggests that automated auditing alone is insufficient for evaluating practical screen reader navigational accessibility. A hybrid evaluation approach combining automated WCAG scanning with manual screen reader walkthrough using a defined task scenario and NEG measurement may provide a more complete picture of interface accessibility quality.

4) Shift-Left Accessibility Integration

The structural barrier categories identified in this study Semantic Mismatch and missing Focus Management logic are substantially more difficult and costly to address post-deployment than at the design or prototyping stage. Integrating accessibility evaluation and the NEG metric as acceptance criteria during the component design phase, before visual implementation is complete, may enable earlier and more efficient barrier identification and remediation.

II.CONCLUSION

This paper has proposed a Semantic-First Methodology comprising four iterative stages Heuristic Semantic Audit, Structural Deconstruction, ARIA-Native Hybridization, and Navigational Parity Validation for the systematic remediation of screen reader navigational barriers in modern Single Page Application web interfaces. The methodology is distinguished from existing accessibility approaches by its focus on structural root causes rather than individual compliance violations, its applicability to existing arbitrary SPA interfaces, and its incorporation of a user-side navigational efficiency metric as an evaluation criterion.

Preliminary evaluation across five representative high-complexity UI components indicates an average reduction of approximately 74% in the Navigational Efficiency Gap and an improvement in average task success rate from 69% to 93% following methodology application. All five components achieved full WCAG 2.2 Level AA compliance. A structured comparison with existing approaches suggests that the methodology addresses barrier dimensions not covered by compliance auditing, component design systems, or overlay tools individually.

These findings are subject to the limitations of a preliminary, single-environment evaluation with a restricted component scope, and should be interpreted as indicative rather than conclusive. Future work will extend this investigation through a larger-scale user study with formal statistical validation, cross-environment replication across JAWS and VoiceOver testing configurations, and development of more prescriptive implementation specifications to reduce inter-implementer variability in methodology application.

III.REFERENCES

1. B. Caldwell, M. Cooper, L. G. Reid, and G. Vanderheiden, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/
2. H. Y. Abuaddous, M. Z. Jali, and N. Basir, "Web Accessibility Challenges," International Journal of Advanced Computer Science and Applications, vol. 7, no. 10, pp. 172-181, 2016.
3. J. Lazar, D. F. Goldstein, and A. Taylor, Ensuring Digital Accessibility through Process and Policy, Morgan Kaufmann, 2015.
4. C. Power, A. Freire, H. Petrie, and D. Swallow, "Guidelines are only half of the story: accessibility problems encountered by blind users on the web," in Proc. SIGCHI Conf. on Human Factors in Computing Systems (CHI '12), ACM Press, pp. 433-442, 2012.
5. C. Diggs and J. Mankoff, "Environmental Barriers: A Study of Screen Reader Navigation in Dynamic Single Page Applications," Journal of Web

Engineering, vol. 21, no. 4, pp. 1102-1128, 2022.

6. M. Vigo, J. Brown, and V. Conway, "Benchmarking Automated Web Accessibility Evaluation Tools," in Proc. 10th Int. Cross-Disciplinary Conf. on Web Accessibility (W4A), 2013.
7. C. Power, A. Freire, H. Petrie, and D. Swallow, "Guidelines are only half of the story," CHI '12, ACM Press, pp. 433-442, 2012.
8. IBM Design, "Carbon Design System: Accessibility Handbook," 2023. [Online]. Available: https://carbondesignsystem.com/guidelines/accessibility/overview
9. Microsoft Inclusive Design, "Inclusive Design Toolkit v2," 2023. [Online]. Available: https://www.microsoft.com/design/inclusive/
10. K. Groves, "Overlay Fact Sheet,"AccessiByeBye,2021. [Online]. Available: https://overlayfactsheet.com/
11. NV Access, "NVDA (NonVisual Desktop Access) User Guide," 2024. [Online]. Available: https://www.nvaccess.org/help/
12. W3C WAI, "WAI-ARIA (Accessible Rich Internet Applications)1.2," W3C Recommendation, 2024. [Online]. Available: https://www.w3.org/TR/wai-aria-1.2/