

AWAE: A Multi-Stakeholder Reporting Tool for Automated Web Accessibility Evaluation

Awoniyi Amos Tolulope¹, Chibuzor Emmanuel², Bill Ordu Ibuchukwu³,
Okesola Kikelomo Ibiwumi⁴, Ogunseitan Oluwaseun⁵

Department of Software Engineering^{1,2,3,4,5}

Department of Computer Science

School of computing, Babcock University Ilishan-Remo Ogun State, Nigeria

awoniyia@babcock.edu.ng, chibuzore2565@student.babcock.edu.ng,

ordui3091@student.babcock.edu.ng, okesolak@babcock.edu.ng,

ogunseitan0642@pg.babcock.edu.ng

Orcid: <https://orcid.org/0009-0009-4056-40661>, Orcid: <https://orcid.org/0009-0008-0692-25392>

Orcid: <https://orcid.org/0009-0000-4084-23743>, Orcid: <https://orcid.org/0000-0003-0944-14974>

Orcid: <https://orcid.org/0009-0001-5772-13655>

Abstract: Automated web accessibility evaluation tools play a central role in supporting Web Content Accessibility Guidelines (WCAG) compliance, yet established platforms largely deliver violation reports in a uniform format that underserves diverse stakeholders, including non-technical professionals such as designers, auditors, and end-users. The resulting comprehension gap inflates remediation effort, slows audit preparation, and limits stakeholders' ability to act on accessibility findings. This paper presents AWAE, a multi-stakeholder reporting tool for automated web accessibility evaluation that wraps the open-source axe-core engine in a data-transformation layer to produce four structurally distinct, audience-specific report views: Developer, Auditor, Designer, and End-User. AWAE evaluates pages against WCAG 2.2 success criteria. Empirical evaluation of the framework demonstrates strong quantitative consensus regarding accuracy, detail, and trust, while identifying critical operational insights, such as the need for in-viewport inspection features and prescriptive, code-level remediation paths. The findings indicate that tailored, multi-stakeholder reporting yields clear usability advantages and provides targeted benefits across operational roles.

Keywords: Web accessibility evaluation; WCAG 2.2; axe-core; automated testing; Multi-stakeholder reporting; User-centered design

I. INTRODUCTION

Web accessibility is a foundational requirement for inclusive digital participation. As government, education, financial, and consumer services migrate online, accessibility has shifted from a voluntary design consideration to a regulatory requirement. The Web Content Accessibility Guidelines (WCAG), maintained by the World Wide Web Consortium (W3C), provide the dominant technical standard for evaluating whether digital content is perceivable, operable, understandable, and robust (POUR) for users with disabilities [1]. Legislative frameworks such as the Americans with Disabilities Act (ADA) in the United States, the European Accessibility Act in the European Union, and Section 508 requirements for United States federal agencies have turned WCAG conformance from a best practice into a compliance obligation [2], [3], [4].

A range of automated tools has been developed to support WCAG conformance checking, including rule-based scanners such as axe-core and WAVE and, more recently, machine-learning-based approaches for detecting accessibility violations that are difficult to encode as static rules [5]. However, prior studies report that such tools



reliably detect only a subset of WCAG success criteria, typically those with objective, machine-verifiable conditions, while criteria requiring contextual or semantic judgment remain largely unautomated [6].

More fundamentally, existing automated evaluation tools are largely designed around a single, undifferentiated output format, a flat list of violations mapped to WCAG success criteria, regardless of who consumes that output. A developer needs actionable, code-level remediation guidance; an auditor needs conformance-level evidence mapped to legal requirements; an auditor needs a summarized risk or compliance score; and an end user or advocate needs a plain-language explanation of impact. Current tools are limited in their reporting along these lines, forcing every stakeholder to interpret the same generic output [7].

In this paper, we present AWAE, a multi-stakeholder framework and tool for automated web accessibility evaluation that tailors conformance reporting to the needs of distinct stakeholder groups. Our main contributions are as follows: (1) development of an architectural framework and design requirements necessary to deliver tailored reporting views for distinct stakeholder groups; (2) an automated data-transformation mechanism that maps raw accessibility engine outputs into stakeholder-specific reporting formats without diagnostic loss; and (3) an empirical stakeholder assessment evaluating the usability and functional efficacy of the multi-stakeholder reporting system compared to existing single-view tools.

Context and Motivation

Automated accessibility evaluation has converged on a small number of widely adopted engines, most notably the open-source axe-core library, which underpins commercial products such as axe DevTools as well as institutional validators including MAUVE [8]. These engines reliably surface WCAG violations together with severity classifications, affected DOM nodes, and references the violated success criterion. The scale of the underlying problem is well documented [9], [10]; the uniform structure of evaluator output, however, encodes an implicit assumption that the consumer is a software developer: results are presented as CSS selectors, HTML snippets, and impact tags language that is fluent for engineers and opaque to other audiences.

Accessibility work is, in practice, a multi-stakeholder activity. Designers need to know which colour-contrast, focus-order, or touch-target decisions failed and how that maps back to the design; auditors need compliance summaries structured for formal reporting; end-users need a plain-language indication of how usable a page is and what they can expect of it. Automated evaluation output is not usually comprehensible even to developers, with significant differences in rule implementation and output formatting across tool vendors making cross-tool interpretation difficult [3]. Forcing different stakeholders through the same developer-shaped report creates a comprehension gap that inflates remediation effort and limits non-technical stakeholders' ability to act on findings, a limitation consistent with recent evaluations of accessibility tooling more broadly [8]. Accessibility professionals experience a persistent gap between formal compliance output and user-centric reporting [11], underscoring the limitation of the existing tool landscape to non-developer audiences.

These findings collectively suggest that the deficiency lies less in evaluation accuracy than in output design: the same underlying violation data could, in principle, be reshaped to fit each audience's task, rather than forcing every stakeholder through a single developer-oriented report.

The opportunity, therefore, is not to replace existing evaluation engines but to wrap them in an audience-aware transformation layer that translates a single evaluation pipeline into structurally distinct, role-appropriate reports. AWAE pursues that opportunity for WCAG evaluation, building on axe-core while introducing four parallel report views (Developer, Designer, Auditor, and End-User) and an integrated educational layer of interactive WCAG card deck [12].

First, there is a limited number of deployed accessibility evaluators that differentiate their output structurally for the four stakeholder roles that consume accessibility findings: developers, designers, auditors, and end-users currently receive the same violation lists and technical descriptions, creating comprehension barriers for non-technical stakeholders and information overload for users seeking high-level summaries. Second, existing tools rarely embed an



educational scaffolding layer that explains the WCAG criterion in an interactive manner underlying each flagged violation, leaving users with error codes and rule references but no contextual learning opportunity. AWAE addresses these gaps directly through its multi-stakeholder reporting architecture and integrated WCAG card deck, and contributes empirical evidence of improvements in comprehension, task completion, or remediation efficiency across stakeholder roles. The main concept of this study is to explore three research questions (RQ):

RQ1: What architectural principles and design requirements are necessary to deliver effective, tailored reporting views for distinct stakeholder groups within web accessibility evaluation?

RQ2: How can data-transformation mechanisms effectively map raw accessibility engine outputs into stakeholder-specific reporting formats without loss of diagnostic fidelity?

RQ3: In what ways do multi-stakeholder assessments perceive the usability and functional efficacy of the proposed evaluation system compared to existing single-view accessibility tools?

II. LITERATURE REVIEW

A. Automated Accessibility Evaluation Engines

Automated accessibility evaluation has converged on a small set of widely adopted engines. axe-core [13] applies rule-based testing against WCAG 2.0, 2.1, and 2.2 Level A and AA success criteria by analysing the rendered DOM, ARIA attributes, CSS computed values, and element relationships. It classifies violations across four severity levels (critical, serious, moderate, minor) and emits structured JSON output containing rule identifiers, affected CSS selectors, WCAG criterion mappings, and remediation guidance. Its adoption as the detection backbone of commercial tools such as axe DevTools, and its wide use across the accessibility tooling landscape more broadly, makes it a de facto standard for WCAG-compliant automated evaluation [8].

WAVE [14] evaluates content against WCAG 2.0 and 2.1 through static DOM analysis and reports findings through an in-page visual annotation model overlaying icons across five categories: errors, alerts, features, structural elements, and contrast errors. While this provides immediate visual context, WAVE produces uniform output regardless of the user's role, offering identical presentation to developers, auditors, and non-technical stakeholders. axe DevTools, also built on axe-core, evaluates against WCAG 2.0, 2.1, and 2.2 and generates structured, code-level output, but its reporting remains oriented exclusively toward developers with technical expertise. [15] Supplements axe-core with proprietary checks and a weighted 0-to-100 scoring model; Pa11y [16] prioritises developer-pipeline integration, outputting JSON, CSV, or plain text for continuous integration workflows, but with minimal contextualisation for non-technical stakeholders. MAUVE [7] extends rule-based checking with a machine-learning component that predicts potential violations outside deterministic rule coverage, producing multi-dimensional scores across the four POUR principles, but its output remains structurally uniform across audiences. [8], In their analysis of transparency across automated accessibility tools, they found that this uniformity is a consistent design limitation rather than an incidental feature, confirming that the gap AWAE addresses is systemic rather than tool-specific.

B. Multi-Stakeholder Reporting in Accessibility Tools

A small body of work has experimented with adapting accessibility evaluation output to different audiences. [17], developing an automated web accessibility evaluation framework, produced outputs structured for both technical and non-technical users, but their differentiation remained at a binary level (technical versus lay audiences) without addressing the distinct needs of auditors, designers, or end-users as separate roles. [18], In their system for generating whole-app accessibility reports for mobile applications, they found that accessibility-focused engineers and testers were more satisfied with audit summaries produced by their system than with a baseline tool, and that the system helped them prioritize issues more quickly, providing empirical evidence that report design at the presentation layer affects usability independently of the underlying detection mechanism, though their study did not compare multiple role-specific report variants. As previously noted, [11] found that accessibility professionals experience a persistent gap between formal compliance output and user-centric reporting, reinforcing the case for systematic audience



differentiation. None of the reviewed work implements four structurally distinct report formats covering developer, auditor, designer, and end-user roles from a single evaluation pipeline.

Two gaps emerge from this review. First, no widely deployed accessibility evaluator differentiates its output structurally for the four stakeholder roles that consume accessibility findings; developers, designers, auditors, and end-users receive the same violation lists and technical descriptions, creating comprehension barriers for non-technical stakeholders and information overload for users seeking high-level summaries. Second, existing tools rarely embed an educational scaffolding layer that explains the WCAG criterion underlying each flagged violation, leaving stakeholders with additional cognitive load in fixing accessibility issues.

III. METHODOLOGY

A. System Overview

AWAE follows a three-tier architecture comprising a presentation layer for audience-specific report views, an application logic layer for evaluation orchestration, transformation, and report generation, and a data persistence layer for evaluation history, WCAG content, and user profiles. This separation follows directly from the incremental process model because the informational needs of a developer differ substantially from those of an auditor, and both differ again from those of a non-technical end-user; each reporting tier was developed, tested, and refined independently against feedback from representatives of that audience, without destabilizing the shared evaluation pipeline underneath. The conceptual system architecture is shown in Fig. 1. During the empirical assessment, participants navigated to the tool landing page and initiated the evaluation, after which they entered the designated target test URI into the evaluation input field. Participants then inspected and navigated the resulting evaluation report through the respective stakeholder-specific view corresponding to their assigned position (End User, Developer, Designer, or Auditor). Finally, participants assessed the extent to which the presented report format and diagnostic framing aligned with the informational requirements and operational context of their selected user role via the evaluation questionnaire presented in Table 2.

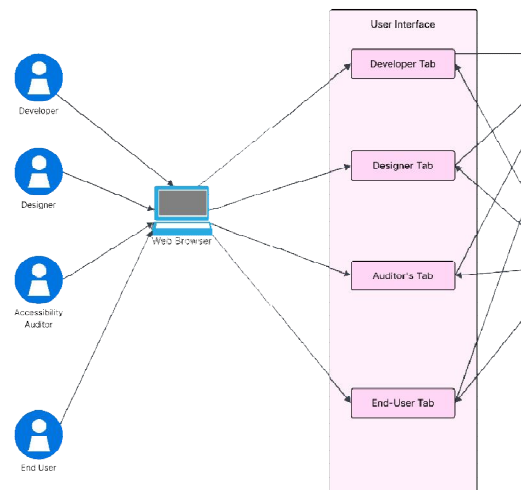


Fig. 1 System architecture of AWAE

Fig. 2 situates the same architecture described in Fig. 1 from the perspective of its four external stakeholder roles: developer, auditor, designer, and end-user, all of whom submit evaluation requests through the same entry point and are routed to a role-appropriate report by the audience selector, a routing component that maps each incoming request to its corresponding report template based on the user's declared role.



B. Core Architectural Components

The platform architecture is organized into four groups of components: user interface components, application logic components, data management components, and external integration components. Below we detail the components most relevant to the contribution of this paper.

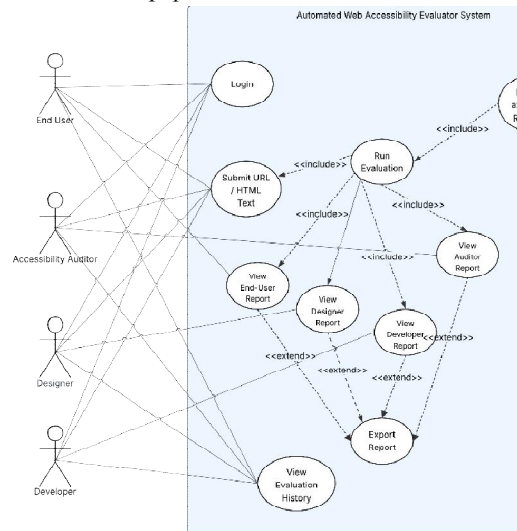


Fig. 2 System use case diagram showing actors

C. Front-End (Role-Specific Interfaces)

The front-end is implemented in React with TypeScript and Tailwind CSS, organized around four parallel report views and a shared evaluation input interface. The four views (Developer, Auditor, Designer, and End-User) are rendered from the same underlying evaluation result through a role-selection component that determines which transformation is applied. The shared Evaluation Input Interface accepts either a public URL or raw HTML, performs client-side URL format validation before submission, and provides progressive feedback during the evaluation pipeline. Fig. 3 shows the evaluation input interface used to capture URL or HTML submissions.

AWAE
Evaluate
WCAG Cards
History

Evaluate Website Accessibility

Enter a URL or upload an HTML file to run an automated WCAG 2.2 accessibility audit.

WCAG Version

Conformance Level

WCAG 2.2

Level AA

URL

HTML File

Enter website URL (e.g., https://example.com)

Evaluate

☐ Crawl full website (same origin)
Automatically excludes admin paths like /wp-admin, /admin, /administrator, and /wp-login.php.

Fig. 3 Evaluation input interface



D. Back-End Application Logic

The back-end is implemented as a set of Next.js API routes running on Node.js, organised around five core modules that together form the evaluation pipeline: The Evaluation Controller, the axe-core Integration Module, the Data Transformation Layer, the Report Generation Service, and the Export Controller, alongside a supporting Authentication Service.

The Evaluation Controller receives URL or raw HTML submissions from the front-end, invokes Playwright to render the target page in a headless browser context, and passes the rendered DOM to the axe-core Integration Module, which executes axe-core's rule set against the rendered page and returns the raw set of detected violations. The Data Transformation Layer takes this raw violation output and restructures it according to each of the four stakeholder-specific schemas. The Report Generation Service receives this transformed output and assembles the audience-specific report structure before returning it to the requesting client. The Export Controller handles post-generation export requests, converting report data into PDF (via react-pdf), JSON, or CSV format depending on the selected export type. The Authentication Service manages user registration, login, and session validation using bcrypt for password hashing and JSON Web Tokens (JWT) for stateless session management, ensuring that evaluation history retrieval and report export are restricted to authenticated users.

E. Accessibility Evaluation Engine (axe-core Integration)

The evaluation engine is built on the open-source axe-core library developed by Deque Systems. The platform invokes axe-core against the DOM rendered via Playwright and captures the resulting violation set, preserving the original Rule ID, impact level, description, affected nodes, and WCAG criterion mapping for downstream transformation.

F. Data Transformation Layer

The Data Transformation Layer serves as AWAE's reporting and projection engine, translating raw automated evaluation results into role-tailored representations. Upon completion of an audit run, the layer ingests the raw JSON payload produced by axe-core and executes four deterministic transformation pipelines, generating audience-specific data models without mutating the underlying evaluation source:

- **Developer View:** Extracts selector paths (target), contextual mark-up snippets (html), failure summaries, and native axe-core engine tags. These data points are augmented with normalized WCAG 2.2 Success Criterion mapping, automated severity classifications (critical, serious, moderate, minor), and code-level remediation patterns targeting DOM structure and ARIA implementation.
- **Auditor View:** Aggregates findings under the four foundational WCAG principles (Perceivable, Operable, Understandable, Robust) and target conformance tiers (A, AA). It compiles an automated compliance matrix containing pass/fail tallies restricted strictly to automatically testable success criteria, explicitly noting unverified criteria that necessitate manual inspection to avoid misleading compliance assumptions.
- **Designer View:** Filters for rule definitions directly attributable to visual styling, spatial tokens, and layout constraints. This includes color-contrast threshold failures (WCAG 1.4.3 and 1.4.11), touch-target dimensions (WCAG 2.5.8), and detectable focus-indicator defects or improper sequential ordering flags (such as positive tabindex anti-patterns). Deficiencies are contextualized with design-system remediation guidance, such as recommended contrast-ratio adjustments and minimum target-size tokens.
- **End-User View:** Abstract technical nomenclature (for example, CSS selectors, ARIA attributes) into plain-language functional impact summaries. Violations are mapped to user-facing assistive technology barriers, such as missing alternative text, unresponsive screen-reader landmarks, or trapping keyboard navigators, allowing non-technical stakeholders to assess direct real-world task disruption.

To balance query efficiency and storage overhead, the layer persists both the raw evaluation result and the four pre-computed views as structured JSONB documents within a single relational record in the PostgreSQL database. This



allows rapid retrieval across diverse client dashboards while eliminating redundant re-evaluations or on-the-fly serialization overhead.

G. Audience-Specific Reporting Features

The Developer Report categorizes violation data across four severity tiers (critical, serious, moderate, and minor). Each diagnostic entry is rendered as an expandable card detailing the target element's runtime CSS selector, a DOM-isolated HTML snippet of the non-compliant markup, the corresponding WCAG 2.2 Success Criterion, and context-specific remediation instructions. Interactive controls allow developers to isolate violations by WCAG principle (Perceivable, Operable, Understandable, and Robust) or target conformance level (A, AA, AAA). Fig. 4 illustrates this layout.

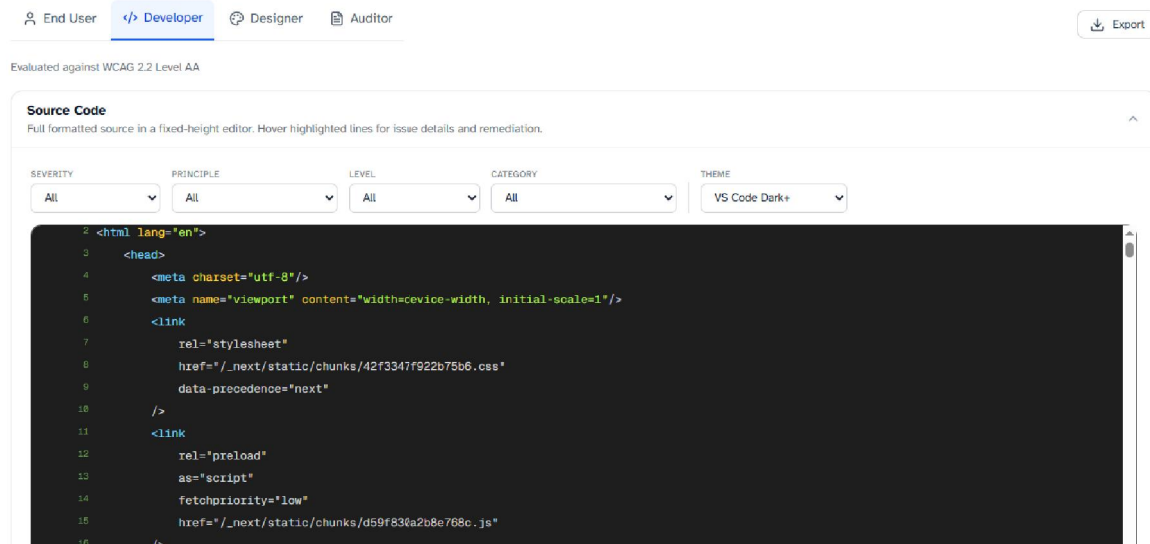


Fig. 4 Developer report view

The Auditor Report provides a structured compliance matrix mapped directly to the W3C Web Content Accessibility Guidelines (WCAG 2.2, Levels A-AAA). Each evaluated Success Criterion is catalogued by operational status (Pass, Fail, or Requires Manual Verification), violation frequency, and target conformance tier. To streamline regulatory compliance and reporting workflows, an executive summary panel aggregates the total issue count across the four POUR principles (Perceivable, Operable, Understandable, and Robust). As illustrated in Fig. 5.



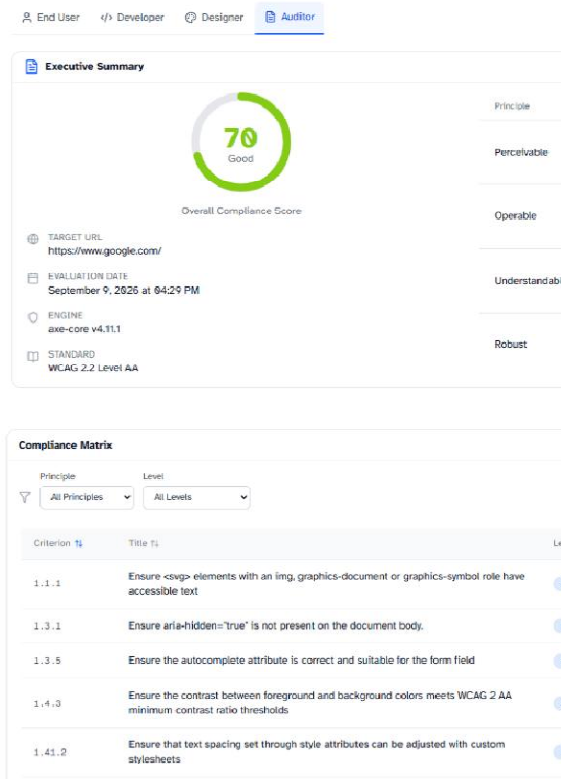


Fig. 5 Auditor report view

The Designer Report module identifies: (1) colour-contrast failures by calculating relative luminance ratios across foreground and background layer fills; (2) target-size violations by measuring interactive element bounds against the success criteria threshold, including spacing offset checks; and (3) potential focus-order conflicts using coordinate-based layout heuristics and flow annotations. These results are compiled into an actionable summary checklist, allowing designers to remediate issues as shown in Fig. 6.

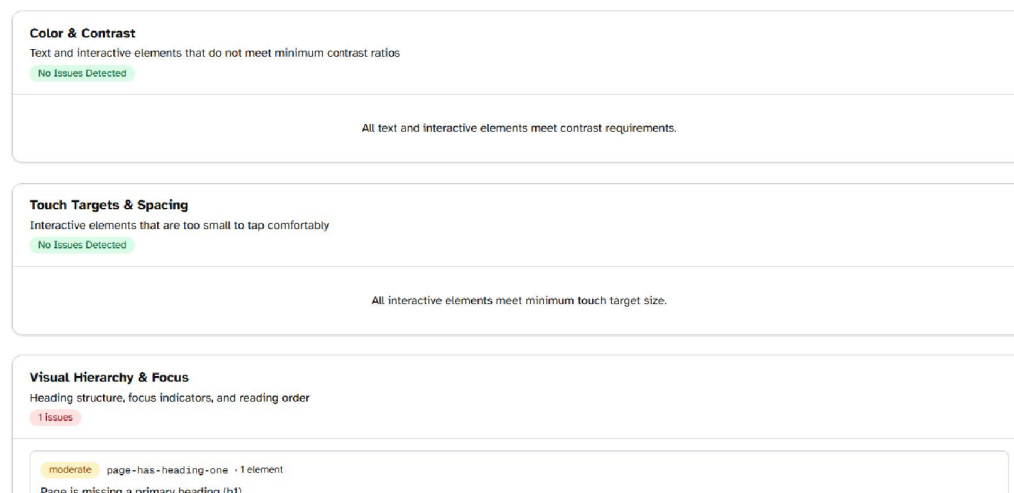


Fig. 6 Designer report view



The End-User Report displays a circular accessibility score range (0-100), a colour-coded compliance indicator, a plain-language summary of the most impactful issues, and a prioritised list of recommended improvements, with technical terminology abstracted away. An optional “Learn More” interaction opens the relevant WCAG card deck for any listed issue, presenting the underlying principle and its real-world significance without exposing CSS selectors, WCAG criterion codes, or severity tags. Fig. 7 shows an End-User Report view.

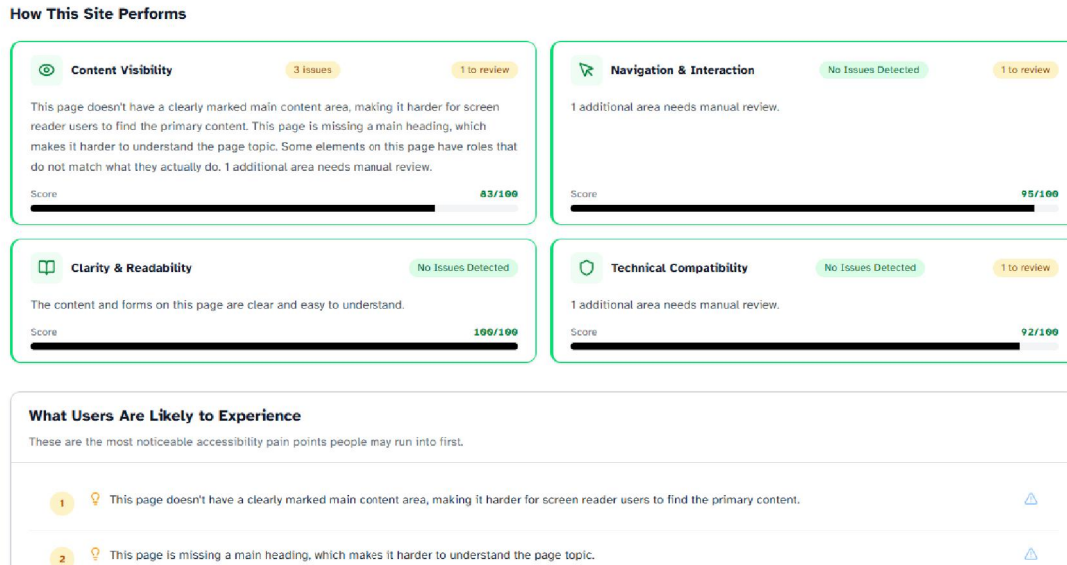


Fig. 7 End-user report view

H. Data Flow and Interaction Model

The cross-role data-flow model follows a fan-out pattern: a single evaluation pipeline, from URL submission through axe-core execution, is shared across all four audience roles, and the Data Transformation Engine branches at the processing stage to produce four structurally distinct report objects from the same underlying violation set. The four report objects and their shared parent evaluation record are written to PostgreSQL in a single transaction, ensuring that all four audience-specific views are immediately available for retrieval without re-evaluation. Subsequent requests for the same evaluation, whether for a different audience role or for historical review, are served entirely from the persistence layer. The WCAG Card Deck Mapper operates as a read-only enrichment service that queries the WCAG_CARDS table on demand rather than embedding card content in the evaluation record, keeping the evaluation schema stable as card content evolves. In functional testing, violation counts, severity classifications, and WCAG criterion mappings were found to be consistent across all four report types, verifying that a single evaluation can be transformed into four structurally distinct reports without loss of fidelity in the transformation process. The WCAG Card page is shown in Fig. 9.



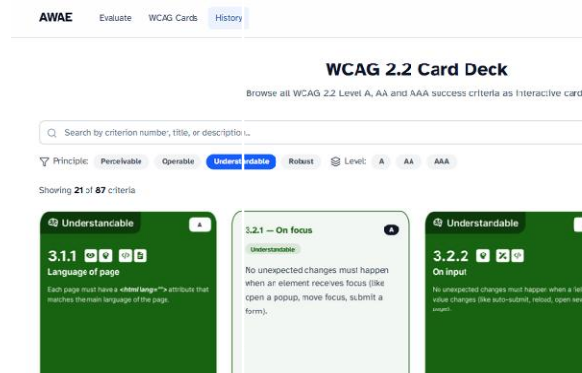


Fig. 8 End-user report view WCAG Interactive Card

I. Empirical User Study

The section provides information on the empirical assessment that was carried out. Participants were from a development team newly introduced to accessibility, consisting of five (5) team members and one (1) end user working on an ongoing web application. The time period of the data collection was from 15 September 2026 to 30 September 2026. The participants interacted with the proposed system and completed an assessment questionnaire. Before the assessment, verbal informed consent was sought. The study objectives, the tasks participants would complete, and the anticipated duration of their involvement were discussed. Participants were informed that their involvement was entirely voluntary and that they retained the right to withdraw at any point without penalty. All personal and demographic information was strictly anonymised. The demographic profile of evaluation participants is presented in Table 1.

Table I Demographic Profile of Evaluation Participants

Demographic Category	Characteristic	Frequency (n=6)
Gender	Male	4
	Female	1
Work Experience	0-4 Years	4
	5-9 Years	1
Stakeholder Role	Designer	2
	Auditor	1
	Developer	2
	End User	1

During the empirical assessment, participants navigated to the tool landing page and initiated the evaluation, after which they entered the designated target test URI into the evaluation input field. Participants then inspected and navigated the resulting evaluation report through the respective stakeholder-specific view corresponding to their assigned position (End User, Developer, Designer, or Auditor). Finally, participants assessed the extent to which the presented report format and diagnostic framing aligned with the informational requirements and operational context of their selected user role via the evaluation questionnaire presented in Table 2.

J. Questionnaire

Table II Questionnaire

S/N	Question	SD	D	N	A	SA
1	I understood what problems this report identified.					
2	The report presented information at the right level of detail for my role.					



3	I could tell which issues mattered most.					
4	I knew what action to take after reading the report.					
5	I trust the accuracy of this report.					
6	I could explain the findings to a colleague in a different role.					
7	Interview Question: What information was missing from the report? And what was unnecessary or confusing?					

IV. RESULTS AND DISCUSSION

To validate the developed AWAE, a comparative assessment was conducted with seven (7) current scientific models and nine (9) existing automated accessibility tools (Accessibility Checker, AccessMonitor, aCe, AChecker, MAUVE, Nibbler, Rocket Validator, TAW, and WAVE) that have been referenced in recent studies in the field. The aim of the evaluation is to determine the improvements addressed in the developed AWAE that distinguish it from existing tools. Table 3 presents the comparative assessment results.

Table III Comparison between AWAE and four established accessibility evaluation tools

(AD = Audience Differentiation; ES = Educational Scaffolding; RM = Reporting Model; AH = Authenticated History; DM = Deployment Model)

Tool	WCAG 2.2	AD	ES	RM	AH	DM
Scientific models						
[19]	No	No	No	Federated report	No	Service-based(web)
[20]	No	No	No	Ontology-based UI evaluation report (accessibility and usability combined)	No	Ontology-based framework (Research prototype)
[21]	No	No	Partial	Ontology/knowledge-modeling based evaluation report	No	research prototype
[22]	No	No	No	Declarative rule-based reporting.	No	Declarative model(research prototype)
[23]	No	No	No	Visual Complexity Score (VCS)	No	Automated tool (research prototype)
[24]	Yes	No (developer-oriented)	Limited, only 16 success criteria.	Error list with fix suggestions	No	CLI and browser plugin
[25]	Yes	No	Guideline simplification layer	Overall accessibility score and per-disability-type score	No	Online tool
Commercial open-source automated tools						
Accessibility Checker	Yes	No	Per-error explanation, who it affects,	Detailed error list with explanations	Yes	Web-based online tool



			and multiple fix options.			
AccessMonitor	No	No	Per- error explanation	Score-based (1-10)	Yes	Online tool and browser extension
aCe	Yes	No	Per- error explanation	Simple pass and fail compliance report	No	Web-based online tool
AChecker	No	No	Flags likely/potential problems for human intervention.	Line-by-line report	No	Online tool
MAUVE	No	Yes (End-User, Web Developer)	Per- error explanation	Multi-target reporting plus accessibility score	Yes	Online tool, Browser and CMS Plugins.
Nibbler	No	No	No	Overall score of 10 across categories (perceivable, robust, etc.)	No on free-tier, probably on paid	Web-based online tool
Rocket Validator	No	No	Per- error explanation, help guide	Site-wide summary plus detailed reports and charts.	Yes	Web-based SaaS as well as a WordPress plugin
TAW	No	No	No	Webpage annotations plus detailed report	No	Online tool, Desktop app, Firefox plugin
WAVE	Yes	Customizable view for different users	No	In-page issue annotation overlay	No	Online tool, Browser extension
AWAE	Yes	Four structurally distinct views: Developer, Auditor, Designer, End-User	Interactive WCAG 2.2 card deck	Role-specific report objects from one transformation layer	Yes (JWT-based)	Serverless web application

As outlined in the comparative data, existing scientific models [19], [20], [21], [22], [23] and several prominent commercial platforms, including AccessMonitor, AChecker, Nibbler, Rocket Validator, and TAW, do not support WCAG 2.2. While select tools such as Accessibility Checker, MAUVE, WAVE, and references [23] and [24] incorporate WCAG 2.2 criteria, AWAE distinguishes itself by integrating WCAG 2.2 compliance natively within a serverless architecture backed by JWT-based authentication.

Effective accessibility assessment requires tailoring diagnostic outputs to distinct stakeholder roles, a dimension where conventional solutions exhibit notable limitations. Most scientific models [19], [20], [21], [22], [23], along with traditional commercial tools (such as Accessibility Checker, aCe, and AChecker) deliver a single, undifferentiated compliance report. Although WAVE offers a customizable view and MAUVE provides limited differentiation for end-



users and web developers, their role-based granularity remains limited. In contrast, AWAE introduces four structurally distinct views tailored explicitly for developers, auditors, designers, and end-users, ensuring that each stakeholder receives targeted insights aligned with their specific workflow.

Educational support and remediation guidance vary widely across existing accessibility evaluators. The reviewed scientific models [19], [20], [21], [22], [23] offer either no scaffolding or purely declarative rule-based reporting without instructional depth. Among commercial alternatives, tools like AccessMonitor and Accessibility Checker provide per-error explanations or basic help guides, whereas platforms such as WAVE and TAW have limited educational scaffolding, focusing more on visual overlays and numerical scores. AWAE addresses this gap by incorporating an interactive WCAG 2.2 card deck, transforming standard compliance checking into an interactive educational resource that facilitates better understanding and faster remediation.

K. Stakeholders Assessment

The empirical evaluation of the accessibility evaluation framework yielded both positive quantitative consensus and critical qualitative feedback across diverse operational roles, encompassing designers, developers, auditors, and end users.

Quantitative assessments indicated high overall satisfaction with information detail, accuracy, and trustworthiness. Participants across all roles generally agreed or strongly agreed that the system accurately identified issues and presented information at an appropriate level of detail, as shown in Fig. 9. However, qualitative responses revealed distinct role-specific friction points regarding actionability and interface integration.

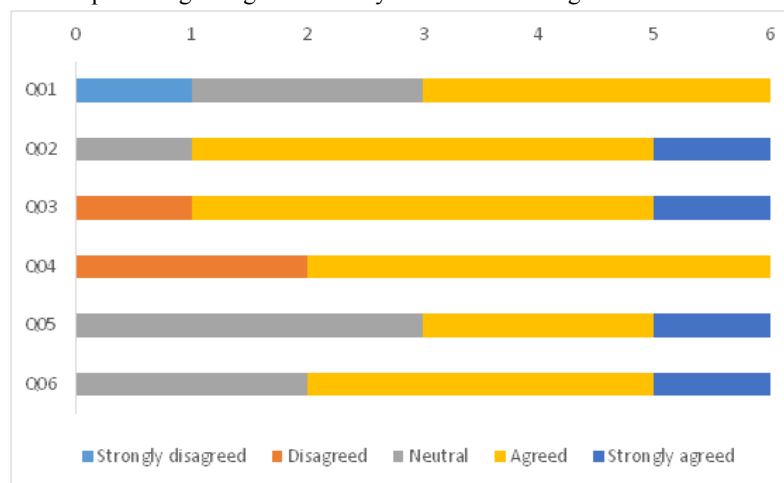


Fig. 9 Resulting analysis of stakeholder assessment

Participants confirmed that the reporting views successfully communicated compliance status and high-level findings, with auditors and end users reporting high confidence in task execution. While non-technical and oversight roles experienced high actionability clarity, developer feedback highlighted a distinct barrier. Specifically, participant 2, a developer, noted uncertainty regarding remediation, stating a lack of understanding of “how to solve the problem as a Developer.” Design professionals critiqued the structural separation between issue lists and viewports. Feedback emphasized the need for a more immersive, in-context experience, such as a floating toolbar or container with issue-type icons mapped directly to corresponding website sections to eliminate the cognitive overhead of scrolling.

Evaluating these findings directly against the core research questions provides key insights into the architectural, transformational, and usability dimensions of a multi-stakeholder reporting tool for automated web accessibility evaluation. Addressing RQ1, the data demonstrate that monolithic reporting schemas fail to serve diverse technical workflows adequately. While auditors and end users benefit from high-level compliance summaries, designers and developers require deep contextual integration. Consequently, architectural requirements for effective multi-stakeholder tools must prioritize spatial continuity and modular presentation layers. Transitioning from decoupled, stacked layouts



to in-viewport inspection architectures, such as interactive floating containers, prevents workflow fragmentation and aligns evaluation tools more closely with modern design and development cycles.

In response to RQ2, high quantitative agreement regarding information accuracy confirms that the system's data-transformation mechanism successfully maps raw accessibility engine outputs into audience-specific formats without compromising diagnostic integrity. However, developer feedback uncovers a critical boundary in transformation efficacy: surfacing raw violation metrics is insufficient for technical implementers. To bridge the gap between diagnostic detection and resolution, data-transformation pipelines must extend beyond textual translation to incorporate prescriptive, code-level remediation paths.

Regarding RQ3, compared to conventional single-view accessibility tools that impose a uniform interface across all users, the proposed framework demonstrates higher functional efficacy in addressing role-specific operational needs. The high ratings across problem identification, appropriate detail, and trust validate the core architectural premise of multi-stakeholder reporting. Furthermore, unearthing role-specific friction points, such as developers' demand for actionable code-level fixes and designer context-switching overhead, highlights the value of empirical stakeholder assessment in evolving diagnostic auditors into comprehensive, collaborative accessibility environments.

Stakeholder assessment was carried out using a small development team of six. However, the reliance on a single team and small sample size limits the broader applicability of the findings. Moreover, limited representation within individual stakeholder groups risks omitting diverse perspectives and nuances across the wider organizational ecosystem.

V. CONCLUSION

This study introduced AWAE, a multi-stakeholder reporting framework designed to bridge the accessibility comprehension gap by translating raw evaluation data from the axe-core engine into audience-specific views for developers, auditors, designers, and end users. By aligning WCAG compliance data with distinct operational contexts, the framework addresses the limitations inherent in uniform reporting formats.

The empirical stakeholder assessment demonstrates that tailored reporting yields strong quantitative consensus regarding accuracy, detail, and trust, while uncovering critical operational insights. Specifically, the findings confirm that multi-stakeholder views enhance comprehension for auditors and end users, while highlighting essential areas for system evolution, such as the integration of in-viewport inspection architectures for designers and prescriptive, code-level remediation paths for developers. Future work will expand empirical validation across larger organizational cohorts and incorporate a more immersive, in-context experience for designers and code-level remediation paths for developers, further enhancing collaborative web accessibility management.

ACKNOWLEDGMENT

The authors thank the School of Computing, Babcock University, for the institutional support provided during the preparation of this manuscript.

REFERENCES

1. World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.2." Accessed: Aug. 30, 2026. [Online]. Available: <https://www.w3.org/TR/WCAG22/>
2. N. Iannuzzi, M. Manca, F. Paterno, and C. Santoro, "Improving Accessibility of Public Administrations Web Sites through Large-Scale Automatic Validations," *ACM Journal on Responsible Computing*, vol. 2, no. 3, pp. 1–21, Sep. 2025, doi: 10.1145/3762815.
3. S. Kumar, J. Shree DV, and P. Biswas, "Comparing ten WCAG tools for accessibility evaluation of websites," *Technol. Disabil.*, vol. 33, no. 3, pp. 163–185, Aug. 2021, doi: 10.3233/TAD-210329.



4. T. A. Awoniyi and S. A. Idowu, "A SHIFT LEFT-BASED ACCESSIBILITY MODEL FOR SOFTWARE DEVELOPMENT PROCESS IMPROVEMENT Shift Left-based Accessibility Model for Software Development Process Improvement," CTICTR, vol. 2, no. 1, pp. 13–27, 2023.
5. P. Parvin, V. Palumbo, M. Manca, and F. Paternò, "The transparency of automatic accessibility evaluation tools," in Proceedings of the 18th International Web for All Conference, New York, NY, USA: ACM, Apr. 2021, pp. 1–5. doi: 10.1145/3430263.3452436.
6. I. Abu Doush, K. Sultan, M. A. Al-Betar, Z. Almeraj, Z. A. A. Alyasseri, and M. A. Awadallah, "Web accessibility automatic evaluation tools: to what extent can they be automated?," CCF Transactions on Pervasive Computing and Interaction, vol. 5, no. 3, pp. 288–320, Sep. 2023, doi: 10.1007/S42486-023-00127-8/METRICS.
7. N. Iannuzzi, M. Manca, F. Paternò, and C. Santoro, "Usability and transparency in the design of a tool for automatic support for web accessibility validation," Univers. Access Inf. Soc., vol. 23, no. 1, pp. 435–454, Mar. 2024, doi: 10.1007/s10209-022-00948-x.
8. [M. Manca, V. Palumbo, F. Paternò, and C. Santoro, "The Transparency of Automatic Web Accessibility Evaluation Tools: Design Criteria, State of the Art, and User Perception," ACM Trans. Access. Comput., vol. 16, no. 1, pp. 1–36, Mar. 2023, doi: 10.1145/3556979.
9. B. Martins and C. Duarte, "Large-scale study of web accessibility metrics," Univers. Access Inf. Soc., vol. 23, no. 1, pp. 411–434, Mar. 2024, doi: 10.1007/S10209-022-00956-X/TABLES/13.
10. [WebAIM, "The WebAIM Million." Accessed: Aug. 30, 2026. [Online]. Available: <https://webaim.org/projects/million/>
11. L. Seixas Pereira and C. Duarte, "Evaluating and monitoring digital accessibility: practitioners' perspectives on challenges and opportunities," Univers. Access Inf. Soc., vol. 24, no. 3, pp. 2553–2571, Aug. 2025, doi: 10.1007/S10209-025-01210-W/TABLES/3.
12. dequelabs, "GitHub - dequelabs/axe-core: Accessibility engine for automated Web UI testing • GitHub." Accessed: Aug. 30, 2026. [Online]. Available: <https://github.com/dequelabs/axe-core>
13. WAVE, "WAVE Web Accessibility Evaluation Tools." Accessed: Aug. 30, 2026. [Online]. Available: <https://wave.webaim.org/>
14. Google, "Lighthouse Chrome for Developers." Accessed: Aug. 30, 2026. [Online]. Available: <https://developer.chrome.com/docs/lighthouse>
15. Pally, "Pally is your automated accessibility testing pal • GitHub." Accessed: Aug. 30, 2026. [Online]. Available: <https://github.com/pally/pally>
16. J. Ara, C. Sik-Lanyi, and A. Kelemen, "Accessibility engineering in web evaluation process: a systematic literature review," Univers. Access Inf. Soc., vol. 23, no. 2, pp. 653–686, Jun. 2024, doi: 10.1007/S10209-023-00967-2/TABLES/11.
17. A. Swearngin et al., "Towards Automated Accessibility Report Generation for Mobile Apps," ACM Transactions on Computer-Human Interaction, vol. 31, no. 4, Sep. 2024, doi: 10.1145/3674967/ASSET/04282A88-91B5-4AAF-BD8B-DE35E76CC2DF/ASSETS/IMAGES/LARGE/TOCHI-2024-0044-F16.JPG.
18. J. R. Hilera, S. Otón, C. I. Martín-Amor, and C. F. Timbi-Sisalima, "Towards a Service-based Architecture for Web Accessibility Federated Evaluation," 2016, Accessed: Sep. 12, 2026. [Online]. Available: <http://observatorioWeb.ups.edu.ec/oaw/apirest.jsf>
19. T. Robal, J. Marenkov, and A. Kalja, "Ontology design for automatic evaluation of web user interface usability," PICMET 2017 - Portland International Conference on Management of Engineering and Technology: Technology Management for the Interconnected World, Proceedings, vol. 2017-January, pp. 1–8, Nov. 2017, doi: 10.23919/PICMET.2017.8125425.



20. P. Ingavelez-Guerra et al., "A proposal based on knowledge modeling and ontologies to support the accessibility evaluation process of learning objects," Congreso Argentino de Ciencias de la Informatica y Desarrollos de Investigacion, CACIDI 2018, Dec. 2018, doi: 10.1109/CACIDI.2018.8584355.
21. J. Pelzetter, "Using ontologies as a foundation for web accessibility tools," Proceedings of the 15th Web for All Conference : Internet of Accessible Things, W4A 2018, Apr. 2018, doi: 10.1145/3192714.3196316.
22. E. Michailidou, S. Eraslan, Y. Yesilada, and S. Harper, "Automated prediction of visual complexity of web pages: Tools and evaluations," Int. J. Hum. Comput. Stud., vol. 145, p. 102523, Jan. 2021, doi: 10.1016/J.IJHCS.2020.102523.
23. K. Boyalakuntla, A. Sri, M. Venigalla, and S. Chimalakonda, "WAccess -- A Web Accessibility Tool based on WCAG 2.2, 2.1 and 2.0 Guidelines," Jul. 2021, Accessed: Sep. 12, 2026. [Online]. Available: <https://arxiv.org/abs/2107.06799v2>
24. [J. Ara and C. Sik-Lanyi, "Automated evaluation of accessibility issues of webpage content: tool and evaluation," Scientific Reports 2025 15:1, vol. 15, no. 1, pp. 9516-, Mar. 2025, doi: 10.1038/s41598-025-92192-5.

