FixAble: Creating An Automated Scanning and Fixing Tool for Web Accessibility Compliance

Lebron Rodriguez, Jakin Mishle Bacalla, Ligaya Leah Figueroa, Ma. Rowena Solamo, Rommel Feria
University of the Philippines Diliman
Quezon City, Philippines
{lgrodriguez1|jcbacalla|llfigueroa|rcsolamo|rpferia}@up.edu.ph

Abstract

Web accessibility is crucial in establishing inclusive digital experiences for users with impairments. Despite its importance, a 2025 study by WebAIM revealed that many websites still fail to comply with the Web Accessibility Content Guidelines version 2 (WCAG 2), a global standard for web accessibility. Existing tools primarily assist in detecting accessibility issues, but often rely on manual developer intervention for remediation, limiting their practicality in large-scale applications. This research introduces FixAble, an integrated system designed to automate the detection and remediation of accessibility violations within web pages. FixAble achieved a 77.23% reduction in accessibility issues based on axe-core audits and a 68.30% reduction according to WAVE assessments. Consequently, the tool demonstrates the significant potential of scalable, efficient automated solutions in promoting web accessibility.

1 Introduction

Web accessibility is an essential practice in ensuring that digital content and online services can be used by all individuals, regardless of their disabilities. This involves designing products, services, and interfaces on the World Wide Web in such a way that eliminates barriers to access. This approach allows all users, especially those with visual, auditory, motor, and cognitive impairments, to be able to navigate and interact with content in the interweb independently and effectively. As such, this is a crucial step in establishing and promoting true inclusivity and equal participation for all web users.

To ensure the achievement of this ideal vision, the Web Content Accessibility Guidelines (WCAG) was developed as the universal set of standards by the World Wide Web Consortium (W3C) to guide and delineate principles for accessible web design. Caldwell et al. (2008) state that this comprehensive framework "covers a wide range of recommendations for making Web content more accessible," which is geared towards upholding four principles: that the Web is (1) perceivable, (2) operable, (3) understandable, and (4) robust [4]. As of 2025, web design primarily adheres to the WCAG 2.0 standards that were released in 2008, along with its subsequent improvements WCAG 2.1 and 2.2 that were published in 2018 and 2023, respectively. Developed over a 15 year period, these standards show the progressing nature of web technologies in connection to user needs.

The WCAG standards are divided into subcategories that comprise guidelines and success criteria for different aspects and elements of web content. This could be used as a foundation for the specific standards and metrics to achieve in a website's accessibility. Furthermore, the levels of conformance – defined as A, AA, AAA, with the latter being the highest – can be a metric in evaluating accessibility compliance systems [4]. Level A is deemed the minimum level of compliance, which encompasses the essential elements that make a web page, at the very least, possible to navigate for users with disabilities. This includes requirements like ensuring proper keyboard navigation, adding video captions, and reducing keyboard traps. The next level, AA, is considered acceptable compliance, which is widely used as the ideal conformance level across most web pages. This level warrants the usability of websites among the majority of web users, regardless of disability, including requirements such as the inclusion of alt text, and the implementation of proper color contrast. Lastly, the highest level is AAA, which ensures that web page navigation is easy for all users. Some of the level's requirements involve support for sign language interpretation, and the reduction of time-bound interactive content.

As the global standard, this has been incorporated into many accessibility laws across several countries. In the United States, state and local government websites are mandated to meet WCAG 2.1 Level AA compliance – a policy enacted by the Department of Justice as an addition to its existing Americans with Disabilities Act (ADA) [9]. Similarly, the European Union integrates the same compliance standards for its European Accessibility Act to ensure the accessibility of websites among both the public and private sectors [10]. Lastly, in the Philippines, the Department of Information and Communications Technology requires government websites to be accessible, citing WCAG 2.0 as the standards to be followed by these pages [8].

Although accessibility standards such as WCAG are widely promoted, many websites still fail to comply with them. In a 2025 study by WebAIM, an American non-profit organization focused on upholding web accessibility, over 50 million accessibility errors were identified among one million web pages, which averages to around 51 errors per page. These figures have decreased from previous years; however, the prevalence of such errors remain a significant concern. Users are still expected to encounter an accessibility issue for every 24 home page elements within an average of 1,257 elements per page. This becomes a concern, especially when coupled with the discovered increase in page complexity, or the number of interface elements used among these websites [18].

Moreover, 94.7% of these pages had an instance of non-compliance with respect to WCAG 2.2 standards. Among the various accessibility violations, the most frequent was low contrast text, in which 79.1% of the tested pages had such an issue. This is followed by missing alternative text for images, which was found in 55.5% of the sites. Subsequently, the next common issues were missing form input labels (48.2%), empty links (45.4%), empty buttons (29.6%), and missing document language (15.8%). Moreover, these findings have remained mostly consistent over the past few years, which further highlights the prevalence of such accessibility errors. As WebAIM stated, "Addressing just these few types of issues would significantly improve accessibility across the web" [18]. These accessibility issues constitute a majority of the violations found among web sites.

While many accessibility auditing tools are available for web accessibility testing, there is a lack of readily-available tools that make automated fixes on web interfaces on top of such error detection. For this purpose, there has been an identified need to provide an automated, reliable, and user-friendly tool to help fix web accessibility violations. The study develops FixAble, a system that automatically audits and remediates accessibility issues – providing a practical solution in improving a web page's structure. This system is designed to solve the commonly encountered errors, primarily for static landing pages for web pages and informational, textual web content.

This paper presents the design, implementation, and evaluation of FixAble, an automated tool for accessibility remediation. Section 2 establishes an understanding of the current body of literature surrounding the underlying factors and dynamics behind automated accessibility remediation. Following this, Section 3 details the architecture behind FixAble's implementation, while Section 4 discusses the particular technologies employed, the specific accessibility violations it targets, and the summary of test results. Finally, the paper culminates with Section 5, the Conclusions and Recommendations, which highlights key findings and implications for future works and studies.

2 Review of Related Literature

Numerous studies have assessed the accessibility of various web pages, particularly those of government agencies and State Universities and Colleges (SUCs) in the Philippines. Noche et al. (2024) found that all SUC websites in the Ilocos region were unable to achieve acceptable compliance, highlighting the need for stronger efforts to establish better web accessibility across SUC websites in the country [7]. This sentiment is echoed by Bokingkito et al. (2025) regarding government websites [3]. Despite achieving an average score of 82.5%, their study pointed out the need to improve digital services and visibility among these web pages, advocating for the adoption of accessible design principles, the integration of assistive technology, and the administration of regular audits.

This inadequacy is further illuminated by other studies. Salvio (2020) posits that most Philippine e-government service websites were deemed inaccessible by people with disabilities [14]. This is corroborated by Jordanoski and Nielsen (2023), who noted that low accessibility compliance levels were prevalent among government websites of different countries, regardless of their income cluster [5]. These consistent findings illustrate the insufficiency of accessibility features in public-facing platforms like government and SUC web pages. Furthermore, this underscores the importance of ensuring that web accessibility is not only mandated but actively implemented, as digital platforms serve as gateways to public services and information. Thus, despite the legal mandate to make Philippine government websites accessible, it is still necessary to include these websites in the study's scope, especially in their role for broad public access.

To aid in implementing web accessibility, numerous automated accessibility evaluation tools have been developed over time. In fact, according to the World Wide Web Consortium (W3C), more than 80 automated accessibility evaluation tools are currently listed and maintained for public use, with varying scopes and compatibility levels including WAVE, axe-core, Accessibility Insights, Lighthouse, SiteImprove, and Tenon [17]. Consequently, several studies have explored the capabilities and limitations of these tools to evaluate their efficacy in identifying accessibility violations. According to Nguyen's (2024) evaluation five prominent automated accessibility evaluation tools, WAVE emerged as the most reliable in detecting accessibility violations among the tools tested, which also included axe-core, Lighthouse, SiteImprove, and Accessibility Insights [6]. However, as Pool (2023) and Al-Ahmad et al. (2015) both highlight, these tools are often complementary, in which certain tools can flag issues that others might miss [12] [2]. These studies both suggest that leveraging multiple tools can provide a more informative and comprehensive assessment than relying on a single tool.

While automated testing tools can be beneficial in quickly identifying basic compliance issues, they also present inherent limitations. This perspective is reinforced by Vigo et al. (2013), who state that solely relying on such tools can be detrimental, as their findings tend to be incomplete, especially when dealing with dynamic content [16]. Dynamic content encompasses web elements that frequently change, are interactive, or load asynchronously without refreshing the page. Such elements pose significant challenges for automated tools, as it is difficult to capture a stable or complete snapshot of such elements for evaluation.

Beyond technical evaluations, web accessibility must also be approached from a user experience perspective. Ultimately, accessibility is an important aspect of websites because many users with impairments rely on assistive technologies to consume web content. Common assistive tools include screen readers, screen magnifiers, speech recognition tools, and other input/output aids. The effectiveness of web accessibility is significantly linked with how well websites function through these technologies – a point that is illustrated by several studies. For instance, Power at al. (2012) specifically examined the effectiveness of accessibility guidelines and testing tools among users with visual impairments. Their findings suggest the need for more reliable and consistent testing frameworks, as only half of the problems reported by users were covered by WCAG 2.0's Success Criteria. "Few developers are implementing the current version of WCAG, and even when the guidelines are implemented on websites, there is little indication that people with disabilities will encounter fewer problems" [13]. This emphasizes that effective tools and standards must not only identify basic violations, but also suggest context-sensitive improvements that extend beyond the WCAG Success Criteria.

Building on this idea behind the need for user experience in testing, Petrie and Kheir (2007) point out that accessibility and usability are neither mutually exclusive nor entirely overlapping, but rather partially intersecting [11]. These insights underline the importance of human evaluation and user input to thoroughly understand how users interact with a website, and more importantly, how well it performs when accessed with assistive technologies.

This body of literature emphasizes various essential factors behind implementing web accessibility. While there have been significant improvements in the domain through global standards, legal mandates, and automated testing tools; there are also challenges that remain, including the continued prevalence of accessibility violations, the inherent limitations of automated testing tools, and the underlying implications of usability in accessibility. These must all be considered in the development of FixAble, so that a comprehensive remediation tool can be created to encompass many user dilemmas and accessibility concerns.

3 FixAble: An Automated Accessibility Remediation Tool

FixAble operates on a purpose-built architecture tailored for the automated detection and remediation of accessibility issues found on web pages. This architecture is composed of five core components: the Input Handler, Content Loader, Accessibility Analyzer, Content Enhancer, and Output Handler. This structure allows for easy development, growth, and connection to different workflows. A diagram of this implementation is presented in Figure 3.1.

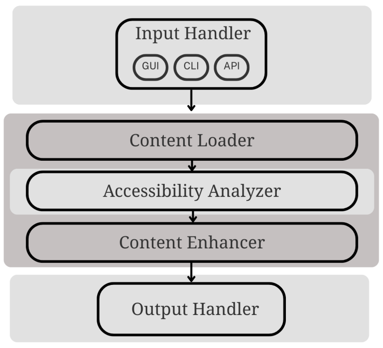


Figure 3.1: FixAble's Architecture

In the Input Handler, which receives user input in the form of a web URL through interfaces. Once input is received, it will then be passed to the Content Loader that retrieves the web content. In static pages, simple HTML fetching works properly, but in dynamic JavaScript-heavy websites, the content loader renders the page using a headless browser engine to fully capture the final Document Object Model (DOM).

Once the content is loaded, Accessibility Analyzer scans the rendered HTML to detect violations of established accessibility standards, particularly the Web Content Accessibility Guidelines (WCAG) 2.0 and 2.1. This module identifies common accessibility issues, and the output of this is a detailed list of violations.

After that, the Content Enhancer automatically fixes the problems it found. In more advanced configurations, this module can be extended to use image recognition tools to generate descriptive alternative text for images lacking appropriate descriptions.

Finally, the Output Handler compiles the enhanced content and audit results. This module presents users with a visual preview of the fixed page, a downloadable version of the corrected HTML, and an audit summary that includes both resolved and unresolved issues.

4 Discussion

In particular, FixAble is built upon a web-based implementation that is presented by the software architecture in Figure 4.1, which serves as an aid for understanding how the system's layers operate together. This diagram illustrates the components, internal structure, and operational flow that are necessary for FixAble's automated remediation. The workflow, along with the tools and technologies used, are mapped to the architecture presented in Figure 3.1.

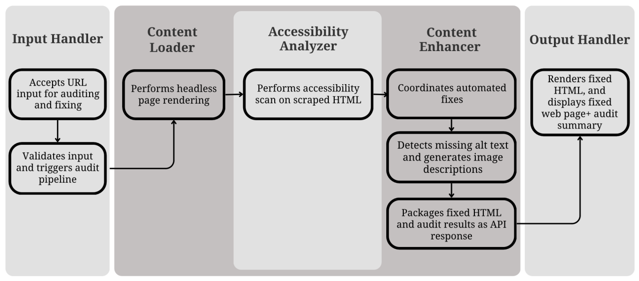

Input Handler
Content Loader
Accessibility Analyzer
Content Enhancer
Output Handler
Accepts URL input for auditing and fixing
Validates input and triggers audit pipeline
Performs headless page rendering
Performs accessibility scan on scraped HTML
Coordinates automated fixes
Detects missing alt text and generates image descriptions
Packages fixed HTML and audit results as API response
Renders fixed HTML, and displays fixed web page+ audit summary

Figure 4.1: Implementation Workflow

The operational process of the web application begins with the client-side (or frontend), which accepts a URL input from the user. This step corresponds to the Input Handler in the architecture, which is responsible for receiving user input. This is then transmitted to the server-side (or backend) for validation, audit, and remediation.

Upon receiving the URL, the backend would validate the input by attempting to fetch the specified web page, then subsequently triggering the audit pipeline upon successful validation. This input validation is handled by the Input Handler, and the fetching process is handled by the Content Loader layer, which loads and prepares web content for analysis. This pipeline begins with the rendering and scraping engine, wherein the HTML of the website through headless page rendering is scraped. This is directly mapped to the Content Loader.

After the HTML is successfully scraped, the backend then facilitates the audit using an accessibility audit engine. This tool performs a comprehensive scan of the rendered HTML to identify the accessibility issues that need to be resolved. This step is performed in the Accessibility Analyzer, which is tasked with scanning and detecting accessibility violations within the loaded content. Thereafter, the backend then proceeds to perform automated fixes based on these findings, including generating missing alt texts. This task is done through the image recognition tool, which generates descriptive alt texts for images lacking them. Automated remediation such as this is managed in the Content Enhancer layer which applies the enhancements and fixes to the content based on the audit results.

Once all fixes have been applied, the backend would reconstruct the HTML of the improved web page. The fix details are saved and cached for future reuse within the next 24 hours, where a new scan and fix are performed after this time period has lapsed. Furthermore, it performs an audit on this fixed interface to identify any remaining errors. Lastly, the fixed HTML, along with a summary of the audit results, is packaged and sent back to the frontend, which then displays the fixed web page and the audit summaries to the user. On the client-side, the Output Handler is also responsible for rendering the improved page and visualizing the audit outcomes to the user.

To implement FixAble, the web application relied on a combination of various web technologies and tools, which were chosen based on its suitability for automated accessibility remediation. Each technology is mapped to the core components of the system architecture as illustrated in Figures 3.1 and 4.1, ensuring clarity in how implementation aligns with the architectural design.

- **Vanilla JavaScript** – Used as the Input and Ouput Handler, or the frontend interface. This was specifically utilized for rendering the user interface and handling Document Object Model (DOM) manipulation. It was chosen over alternatives like Vue.js and Next.js for its minimality and direct access to DOM elements. This is particularly important for modifying HTML elements, essential for fixing and rendering web pages.
- **Express.js** – Utilized as the Content Enhancer, specifically for the backend operations. These manage server-side logic, and handle the communication between the frontend and various backend processes like input validation, accessibility audits, and automated fixing. These were selected over alternatives like Angular, Svelte, and NestJS for their compatibility with the other backend tools used by FixAble, like Puppeteer and axe-core. Express.js was chosen due to its stability, documentation support, and compatibility with asynchronous operations required by Puppeteer and axe-core APIs. The system specifically uses [1]Express.js v4.21.2.
- **Puppeteer** – Employed as the Content Loader. Puppeteer (specifically v24.3.0[2]) performs headless browser rendering, which is necessary to capture the rendered HTML of a web page. While tools like Cheerio are faster for static content, they do not render JavaScript. Furthermore, Puppeteer is also more efficient and simpler to implement than other alternatives like Selenium.
- **axe-core** – Implemented as the Accessibility Analyzer for identifying accessibility issues within a web page. This service detects violations with respect to WCAG 2.0, 2.1, and 2.2 standards. It performs audits before and after remediation, and identifies common issues such as insufficient color contrast, missing alt text, and incorrect heading levels. Although WAVE is widely regarded as the most ideal auditing tool, axe-core was preferred, as it was more suitable for web app integration. It provides structured error data, unlike WAVE, which only provides summaries. axe-core was used due to its comprehensive WCAG 2.1/2.2 support and well-documented JavaScript integration. Its output format also aligns well with FixAble's result display. The application uses axe-core v4.10.2[3].
- **Google Cloud Vision AI** – Utilized as part of the Content Enhancer, particularly for generating alternative text for images that lack descriptions. The service uses image recognition to produce descriptions of visual content. However, it is unable to process animated GIFs and videos for alt text generation.

FixAble aims to automatically remediate some of the most prevalent web accessibility issues. Some of the issues are explained below, along with the relevant WCAG 2.1 Success Criteria they violate, and a summary of the web tool's automated solution. The selection and prioritization of these accessibility issues are based on frequency data from the 2025 WebAIM Million report and our own audit results, which confirm that these problems constitute the majority of violations observed across diverse websites. By focusing on these high-frequency issues, FixAble maximizes its impact in accessibility improvement [18].

(1) **Low Contrast Issues**: Insufficient color contrast between text and backgrounds significantly hinders readability for users with visual impairments. This violates SC 1.4.3 (Contrast (Minimum)) and 1.4.6 (Contrast (Enhanced)). FixAble adjusts the text and background colors (either by darkening or lightening) to meet the required contrast ratios of 4.5:1 for normal text, and 3:1 for large text.
(2) **Missing Alternative Text**: The lack of descriptive alternative text for image and other visual content prevents screen readers from conveying information about them, violating SC 1.1.1 (Non-Text Content) and 3.3.2 (Labels or Instructions). FixAble solves this by automatically generating missing alt text using an image recognition tool.
(3) **Missing Form Labels**: This error concerns form elements that do not have explicit or visible labels, making them inaccessible to screen readers. This violates SC 1.3.1 (Info and Relationships), and this is resolved by adding visible labels or aria-label attributes to form elements.
(4) **Improper Heading Structure**: Disordered or skipped heading levels disrupt the logical outline of a web page,

[1]Express.js, v4.21.2, https://expressjs.com/
[2]Puppeteer, v24.3.0, https://pptr.dev/
[3]axe-core, v4.10.2, https://www.deque.com/axe/core-documentation/

which makes navigation difficult for screen readers. This violates SC 1.3.1 (Info and Relationships) and 2.4.6 (Heading and Labels). This is corrected by checking for skipped heading levels and adjusting them as necessary.

(5) **Missing Language Attribute**: Language attributes help screen readers interpret content language and apply the appropriate pronunciation rules. The absence of such violates SC 3.1.1 (Language of Page). FixAble appends the HTML's lang attribute with "en" for English, as most of the target websites are written in English.

(6) **Empty Interactive Elements**: Interactive elements like buttons and links without discernible text or labels cannot be effectively accessed by screen headers, violating SC 4.1.2 (Name, Role, Value). The tool addresses this by an aria-label corresponding to the element's function.

(7) **Missing Table Headers**: The lack of table headers (and corresponding text) prevents users and assistive technologies from understanding the table structure and relationships between cells, violates SC 1.3.1 (Info and Relationships). This is resolved by defaulting the first row of tables into headers.

(8) **Improper List Structure**: Incorrectly formatted lists interfere with the recognition and navigation of screen readers, violating SC 1.3.1 (Info and Relationships). This is solved by wrapping list elements within their proper parent elements.

(9) **Improper ARIA Usage and Attributes**: Incorrect ARIA values, roles, and attributes can mislead assistive technologies, which violates SC 4.1.2 (Name, Role, Value). This is corrected by adding or removing ARIA attributes and values as required.

(10) **Missing or Incorrect Landmarks**: Landmarks further aid screen reader navigation. The absence or misuse of such would violate SC 1.3.1 (Info and Relationships) and 2.4.1 (Bypass Blocks). FixAble resolves through various means, such as adding a missing <main> element, ensuring only one <header> and <footer> element in a page, and correcting <div> or <span> misusages.

(11) **Zooming Restrictions**: Disabling zoom and scale features restricts users with low vision from scaling content, which is detrimental especially on smaller screens. This violates SC 1.4.4 (Resize Text) and 1.4.10 (Reflow). FixAble modifies the viewport tag to allow user zooming and scaling.

(12) **Poor Focus Indicators**: Focus navigators help users navigating with a keyboard to interact with the web page. The inadequacy of such violates SC 2.1.1 (Keyboard) and 2.4.7 (Focus Visible). This is fixed by adding appropriate CSS styles to focus indicators.

To evaluate the performance of FixAble in remediating these issues, a sample of 100 Philippine websites were selected. The websites were selected using purposive sampling, prioritizing accessibility relevance, page load capability, and national reach. Government and SUC websites were chosen from official directories, while popular websites were selected based on platforms commonly accessed by users in the Philippines, such as news outlets, online magazines, and e-commerce services. These platforms were identified using local web traffic data from sources such as ahrefs and Semrush [1, 15].

Particularly, the websites chosen were predominantly static or semi-static, as these allowed for a complete page load. Pages that failed to load completely were omitted - a limitation will be further discussed in another section. The final dataset included 35 state universities and colleges (SUCs), 30 government web pages, and 35 popular websites. Such a selection aimed to ensure that the findings can be more generalizable across various website types and user populations in the country.

Meanwhile, popular websites were identified based on local traffic rankings from publicly available web analytics platforms such as SimilarWeb and StatCounter, focusing on domains frequently accessed by Filipino users—namely news outlets, online magazines, and e-commerce services.

For this purpose, two accessibility auditing tools were used to conduct pre and post-fix testing. The first was axe-core, the accessibility audit engine integrated in FixAble. For each website, the web app automatically generated an audit report through axe-core twice: once before applying the automated fixes, and then after the remediation process. These reports were then displayed on FixAble's interface. To further validate these results, WAVE, a widely recognized auditing tool, was also employed. Manual audits using the WAVE browser extension were done for each website, both before and after undergoing FixAble.

The primary metric for evaluating the performance of FixAble was its ability to significantly decrease the overall number of detected accessibility errors. As such, the total error counts reported by axe-core and WAVE pre and post-remediation were recorded. Beyond this, specific error instance counts (e.g., number of color contrast issues, missing alt texts) were also documented to have more granular data, which can provide deeper insight into FixAble's performance in solving accessibility issues. The workflow behind FixAble's testing process is presented in Figure 4.2 below.

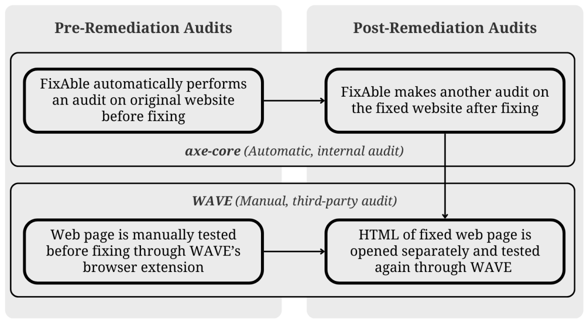

Pre-Remediation Audits
Post-Remediation Audits
FixAble automatically performs an audit on original website before fixing
FixAble makes another audit on the fixed website after fixing
axe-core (Automatic, internal audit)
WAVE (Manual, third-party audit)
Web page is manually tested before fixing through WAVE's browser extension
HTML of fixed web page is opened separately and tested again through WAVE

Figure 4.2: FixAble's Testing Process

After the implementation of the FixAble, a series of tests were conducted to assess its functionality and performance. The results are presented in the tables below, along with the key findings synthesized from these data. The challenges and limitations faced during testing would also be discussed, using them to create further insight into the tool's current capabilities and areas for future improvements.

Table 1: Overall Accessibility Audits (axe-core)

Category	Average Errors Before Fixing	Average Errors After Fixing	Total Error Improvement
SUCs	59.0	18.9	67.88%
Gov't	82.8	15.0	81.84%
Popular	93.5	19.0	79.65%
TOTAL	78.2	17.8	77.23%

Table 1 illustrates a detailed summary of the axe-core audits. According to the findings from 100 websites tested, axe-core detected a total of 7,819 accessibility violations prior to remediation, which averages to 78.2 errors per page.

For the axe-core audits in the SUCs category, the highest number of errors before fixing was detected in Bukidnon State University's (BukSU) website at 169 violations, while the highest number after fixing was identified in the web page of Eastern Visayas University (EVSU) at 66 violations. Meanwhile, the lowest number of errors before and after fixing were 8 and 2, found in the websites of Camarines Norte State College (CNSC) and Easter Samar State University (ESSU), respectively. In summary, the SUC web pages reported the fewest initial issues which averages to 59.0 errors per site, which is significantly lower comparing to the other two categories. However, these SUC websites also presented the lowest improvement rate of only 67.88%, resulting in a post-fix average of 18.9 errors.

In the government category, the Commission of Human Rights (CHR) website had the highest number of errors pre-remediation at 213, while the Department of Finance (DOF) web page had the highest number of errors post-remediation at 66. Furthermore, the lowest values belonged to the Philippine News Agency (PNA) website at 8 pre-remediation, and the Bureau of Immigration web page at 0 post-remediation. While SUC websites had the lowest improvement rate, government websites experienced the highest improvement rate with FixAble resolving 81.84% of accessibility violations which averaged 82.8 errors prior to fixing, which was further reduced to just 15.0 errors per page, marking the lowest post-fix error count across all categories.

Lastly, in the popular websites category, the buy-and-sell platform Carousell had the highest pre-fix error count at 366, while the landing page of streaming website Viu had the highest post-fix error count at 211. Conversely, the lowest error count pre-fix belonged to magazine web page Vogue at just 1 error, while the news website Daily Inquirer had no remaining errors post-fix, thus having the lowest error count post-fix. In summary, popular websites exhibited the highest number of initial errors, averaging 93.5 violations per page, which is likely due to their more complex layouts and dynamic content. Nonetheless, FixAble was able to resolve 79.5% of these violations, reducing the post-fix average to 19.0 errors per page.

Table 2: Overall Accessibility Audits (WAVE)

Category	Average Errors Before Fixing	Average Errors After Fixing	Total Error Improvement
SUCs	52.8	14.3	72.84%
Gov't	57.3	16.1	71.98%
Popular	44.1	18.2	58.77%
TOTAL	51.1	16.2	68.30%

Table 2 presents the summary of WAVE audits. In total, WAVE detected a 5,113 errors before remediation and 1,621 errors after, reflecting a slightly lower error improvement percentage of 68.30%. Notably, WAVE reported fewer issues than axe-core. As the researchers observed, WAVE categorizes issues into "errors" and "alerts" compared to axe-core which flags all violations regardless of severity. While errors encompass severe accessibility violations like missing alt text and insufficient color contrast, alerts are less critical accessibility violations that do not significantly impede website accessibility. These include issues like incorrect heading order, and redundant or suspicious links (but not empty links). In summary, WAVE reported 8,989 alerts, many of which likely correspond to errors in axe-core. Due to such a difference in WAVE's error reporting, the distribution of WAVE errors and improvement rates across categories varied slightly.

Among the WAVE audits of SUC websites, the highest error count pre-remediation was found in the web page of Sultan Kudarat State University (SKSU), while the highest error count post-remediation was found in the website of Eastern Visayas State University (EVSU). The lowest error counts were detected in the web pages of Cebu Technological University (CTU) at 8 pre-remediation, and Jose Rizal Memorial State University (JRMSU) with no errors post-remediation. In summary, SUC websites saw a 72.84% error reduction, from an average of 52.8 to 14.3 errors per page.

Among the government web pages tested, the highest number of errors pre-fix was found in PAG-ASA's website at 121 violations, while the highest number post-fix was identified in DepEd's web page at 54 violations. Similar to the findings in Table 1, the Philippine News Agency (PNA) website also had the lowest WAVE errors pre-fix at just 2. Meanwhile, the Department of Transportation (DOTr) web page had the lowest WAVE errors post-fix at just 1. In total, government websites had a 71.98% improvement, reducing the errors from 57.3 to 16.1 issues per page.

Lastly, among the popular websites tested, the GMA News website had the highest error count before fixing at 137, while BDO's landing page had the highest error count after fixing at 84. Moreover, Manila Bulletin had both the lowest error count before and after fixing at just 1 pre-fix, with no errors post-fix. Popular websites experienced the lowest error reduction rate under WAVE, with only 58.77% of errors resolved, reducing the average errors from 44.1 to 18.2. This lower success rate may be attributed to page load errors and the difficulty of automatically modifying complex and dynamic content, which will be discussed further on the next section. While this category had the lowest pre-fix error average, it recorded the highest pre-fix alert average, at 124.0 per page, significantly exceeding the 73.5 for SUCs and 63.9 for government websites. These results further highlights the complexity and accessibility challenges among commercial and highly interactive websites.

These findings demonstrate that while automated remediation can be capable for resolving a large portion of accessibility violations, specific challenges that remain, particularly with dynamic content, complex visual designs, and ARIA attributes. Such limitations reveal gaps in current automated tools, especially their difficulty with rendering asynchronous content, dismissing overlays, and fixing persistent issues like contrast and alt text errors, which must be addressed through improved strategies discussed in the following section.

Table 3: Most Frequent Accessibility Issues (axe-core)

Before		After	
Page content must be contained in landmarks	2614	Color contrast issues	322
Color contrast issues	1502	Aside must not be contained in landmarks	263
Links without discernible text	1292	Prohibited ARIA attributes	224
Images missing alt text	552	List-item related error	119
Buttons without discernible text	450	Presentational elements must be ignored	117

Table 4: Most Frequent Accessibility Issues (WAVE)

Before		After	
Very low contrast elements	2496	Very low contrast elements	1048
Linked images missing alt text	1105	Missing alt texts	211
Empty links	587	Linked images missing alt text	94
Missing alt texts	374	Multiple form labels	42
Empty buttons	150	Missing form labels	39

Among the specific accessibility issues reported by axe-core across 100 websites, the most prevalent violation identified before remediation was "Page content must be contained by landmarks," as indicated in Table 3. This issue was detected at a significantly higher frequency than other issues, at 2,614 instances. This error is particularly detrimental because page content that is not enclosed within semantic landmarks such as <main>, can interfere with assistive technologies, like screen readers, in effectively navigating and interpreting page structure. Following this, other frequently detected pre-fix issues included color contrast issues, links without discernible text, images without alt text, and buttons without discernible text. According to axe-core's post-remediation reports, FixAble was able to resolve this page content landmark error, reducing its instances by 96%. Post-fix, the next most frequent issue was insufficient color contrast at 322 reports, followed by the label "asides being contained in another landmark," "prohibited ARIA attributes", "list item-related errors", and "presentational elements must be consistently ignored."

Meanwhile, as presented in Table 4, WAVE's pre-remediation audit presented a different pattern. With the most frequent accessibility issue being the label"very low contrast elements"similar to the landmark error in axe-core's findings, this was significantly more prevalent than other issues with 2,496 instances. While FixAble was able to reduce these instances to 1,048, this issue remained the most frequent post-fix. Furthermore, other common WAVE-reported issues pre-fix included linked images without alt text, empty links, missing alt texts, and empty buttons. It is important to note beforehand that WAVE separates alt text issues into three distinct reports: "Missing alt text" for basic images and related media, "Linked images missing alt text" for media embedded within links and "Spacer images missing alt text"for images used purely for design purposes. Upon observation, after FixAble's remediation, WAVE's post-fix reports indicated a slightly similar set of frequent issues, albeit in varying order: missing alt texts, linked images missing alt text, multiple form labels, and missing form labels.

The varying success in remediating specific issues warrants further analysis as these can provide insight into FixAble's current capabilities. For one, the persistence of low contrast elements across both audits may suggest the need for better fixing for such issues. Meanwhile, certain issues like missing alt texts, inadequate form labels, and landmark issues were also frequent, suggesting the need for enhanced handling of these error types as well.

Table 5: Most Frequent Violation Categories (axe-core)

Before		After	
Landmark and Page Structure Issues	2916	Landmark and Page Structure Issues	489
Labeling Issues	1778	ARIA Issues	338
Insufficient Color and Visual Contrast	1524	Insufficient Color and Visual Contrast	336
Missing or Inadequate Alt Text	676	Incorrect Headings and Semantic Structure	204
Incorrect Headings and Semantic Structure	462	Missing or Inadequate Alt Text	141

Table 6: Most Frequent Violation Categories (WAVE)

Before		After	
Insufficient Color and Visual Contrast	2496	Insufficient Color and Visual Contrast	1048
Missing or Inadequate Alt Text	1515	Missing or Inadequate Alt Text	339
Empty Interactive Elements	737	Form Label Issues	112
Form Label Issues	179	Empty Interactive Elements	43
ARIA Issues	38	ARIA Issues	31

To provide further insight into the errors detected before and after fixing, Tables 5 and 6 indicate the most prevalent error categories from the audits of axe-core and WAVE, respectively.

From the axe-core audits as seen in Table 5, the most frequent category both before and after fixing was landmark and page structure issues. This category encompasses violations related to page content not being contained within landmarks, as well as elements like "asides", "contentinfo", and "document landmarks" that conversely must not be contained within other landmarks. This category was initially reported 2,916 times, which was then significantly reduced to just 489.

Before fixing, the subsequent most prevalent categories were: "labeling issues" including links and buttons without discernible text and form elements without visible labels, "insufficient color and visual contrast", "missing or inadequate alt text at fourth" including missing and repeated alt texts, and "incorrect heading orders" which includes empty headings, list item-related errors, empty table header text, and level-one heading errors.

Post-fix, the list of most prevalent categories remained largely consistent, which still included landmark and page structure issues, insufficient color and visual contrast, incorrect headings and semantic structures, and missing or inadequate alt text. However, new issues were added to the list such as ARIA issues; a category encompassing prohibited ARIA attributed, misused parent and children roles, and inappropriate or invalid ARIA values.

Table 6 presents the most frequent categories among WAVE audits, insufficient color and visual contrast emerged as the most prevalent category, with 2,496 instances before remediation, then reduced to 1,048 after. This finding aligns with the implications of Tables 5 and 6, particularly on the high frequency of color contrast issues. This could suggest both the widespread occurrence of this issue across websites, as well as the need for improved handling of this issue. Moreover, missing or inadequate alt texts also remained the second most frequent category, reducing from 1,515 instances pre-remediation to 339 instances post-remediation. This category includes the various alt text categories discussed earlier such as basic images, linked images, and spacer images. The next three categories for the WAVE audits also remained consistent before and after fixing, albeit in different order: empty interactive elements which reduced from 737 to 43, form label issues which reduced from 179 to 112, and lastly, ARIA issues that reduced from 38 to 31.

In summary, the findings from Tables 3 to 6 highlights insufficient color contrast, missing alt text, incorrectly used landmarks, and incorrect ARIA usages as the most prevalent and persistent set of issues. This highlights particular areas requiring further investigation into FixAble's current limitations and capabilities; particularly on the challenges posed by these persistent issues, the need for enhanced remediation strategies, and the effect of various hindrances encountered by the FixAble.

From the results presented above, it is apparent that FixAble was capable of automating the detection and resolution of accessibility issues, as it was able to achieve an overall error reduction rate of 77.23% under axe-core, and 68.30% under WAVE. This confirms its capacity to resolve a wide range of accessibility issues - especially the most common violations, such as missing alt text, insufficient color contrast, and improper landmarks. However, despite its overall success, several limitations were identified during the testing phase that impacted its performance, coverage, and accuracy.

The most prominent challenge involved FixAble's inability to consistently load or process highly dynamic, JavaScript-heavy websites. This was particularly evident in the popular website category, where many pages rely on asynchronous rendering, client-side routing, and delayed content injection. While successful for most static and slightly dynamic web pages, the use of headless browsing via Puppeteer was sometimes insufficient for capturing a fully rendered DOM snapshot of such sites, especially when combined with content protection systems, such as Cloudflare.

Because of this issue, some websites were loaded in an unstyled page format, lacking the expected CSS and interactive elements. This not only reduced the tool's ability to detect real issues, but also introduced new false positives, where new errors were added after automated fixing. A particular instance of this is the website of news agency Rappler, where styles failed to load, leading to links being unstyled, in turn causing contrast violations to increase.

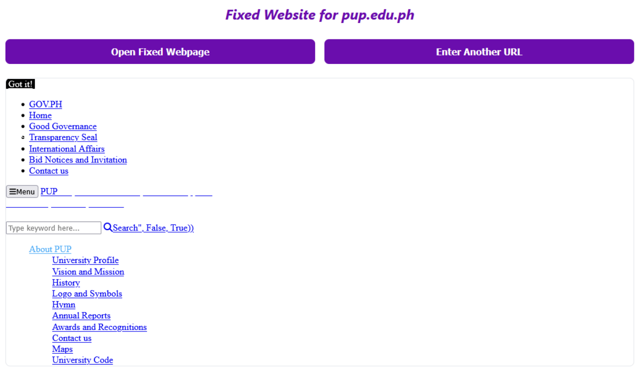


Figure 4.3: Example of unstyled page load (https://www.pup.edu.ph/)

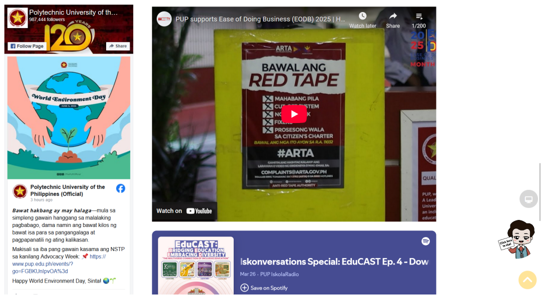
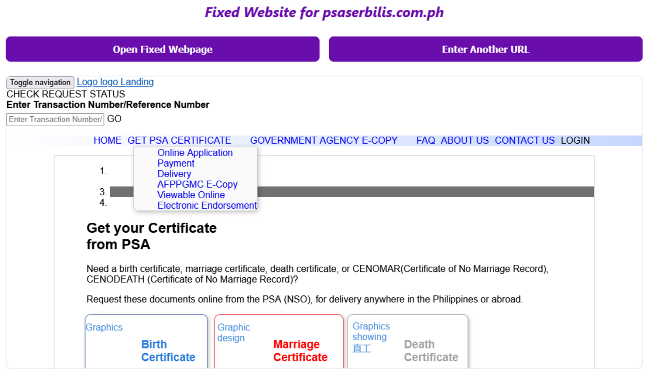

Polytechnic University of th...
World Environment Day
Polytechnic University of the Philippines (Official)
PUP supports Ease of Doing Business (EODB) 2025 | H...
BAWAL ANG RED TAPE
#ARTA
Iskonversations Special: EduCAST Ep. 4 - Dow
Watch on YouTube

Figure 4.4: Original website (https://www.pup.edu.ph/)



Fixed Website for psaserbilis.com.ph

Open Fixed Webpage | Enter Another URL

Toggle navigation Logo logo Landing
CHECK REQUEST STATUS
Enter Transaction Number/Reference Number
Enter Transaction Number/ GO

HOME GET PSA CERTIFICATE GOVERNMENT AGENCY E-COPY FAQ ABOUT US CONTACT US LOGIN

Online Application
Payment
Delivery
AFPPGMC E-Copy
Viewable Online
Electronic Endorsement

1.
3.
4.

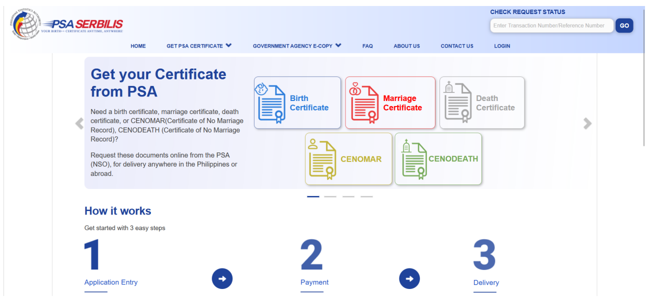
Get your Certificate from PSA

Need a birth certificate, marriage certificate, death certificate, or CENOMAR(Certificate of No Marriage Record), CENODEATH (Certificate of No Marriage Record)?

Request these documents online from the PSA (NSO), for delivery anywhere in the Philippines or abroad.

Graphics **Birth Certificate** Graphic design **Marriage Certificate** Graphics showing 直工 **Death Certificate**

Figure 4.5: Example of unstyled page load (https://psaserbilis.com.ph/)



PSA SERBILIS
YOUR BIRTH++ CERTIFICATE ANYTIME, ANYWHERE!

CHECK REQUEST STATUS
Enter Transaction Number/Reference Number GO

HOME GET PSA CERTIFICATE GOVERNMENT AGENCY E-COPY FAQ ABOUT US CONTACT US LOGIN

Get your Certificate from PSA

Need a birth certificate, marriage certificate, death certificate, or CENOMAR(Certificate of No Marriage Record), CENODEATH (Certificate of No Marriage Record)?

Request these documents online from the PSA (NSO), for delivery anywhere in the Philippines or abroad.

Birth Certificate | Marriage Certificate | Death Certificate | CENOMAR | CENODEATH

How it works

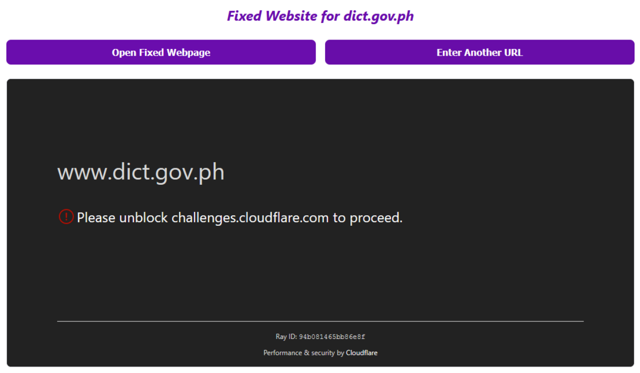
Get started with 3 easy steps

1 Application Entry → 2 Payment → 3 Delivery

Figure 4.6: Original website (https://psaserbilis.com.ph/)

Even some SUC (e.g., PUP, non-UP Diliman websites) and government websites (e.g., PSA) were affected by these issues, which show how page complexity can hinder the web application's performance.

Additionally, another key limitation was the system's inability to access login-protected content. Along with websites restricted by content protection systems, web pages that required authentication or CAPTCHA-like systems were unable to be accessed by FixAble. For this purpose, its use is restricted to public-facing websites only, such as landing pages of websites. This can exclude many portals, dashboards, or internal government web pages that may also have significant accessibility issues.



Fixed Website for dict.gov.ph

Open Fixed Webpage | Enter Another URL

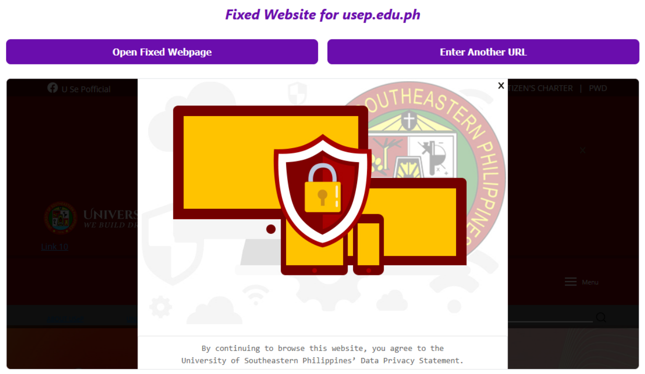
www.dict.gov.ph

Please unblock challenges.cloudflare.com to proceed.

Ray ID: 94b081465bb86e8f
Performance & security by Cloudflare

Figure 4.7: Example of restricted access (https://dict.gov.ph/)

While FixAble implemented strategies to dismiss obstructive overlays and modals windows, such as cookie banners and pop-ups, some modals remained persistent and blocked full access to the content underneath – may it be by not being able to be dismissed at all, or darkening the page content after being closed. This further affected FixAble's ability to audit and fix certain websites.



Fixed Website for usep.edu.ph

Open Fixed Webpage | Enter Another URL

×

By continuing to browse this website, you agree to the University of Southeastern Philippines' Data Privacy Statement.

Figure 4.8: Example of modal handling error (https://www.usep.edu.ph/)

Lastly, FixAble was also limited in handling media-rich content, particularly GIFs and videos. While its image recognition tool was effective in generating descriptive alt text for static images, it could not process animated or embedded media, leaving those elements unfixed. This reduces the tool's comprehensiveness when applied to multimedia-heavy websites.

5 Conclusion and Recommendations

This study attempts to develop a tool to automatically detect and remediate accessibility issues on web pages. This initiative addressed a critical gap, as it was evident through literature that many web pages remain non-compliant within WCAG 2.0 standards.

The proposed solution, FixAble, demonstrated that a tool can be developed to assist in transforming non-compliant web pages into ones that are more aligned with WCAG 2.0 standards. FixAble was able to resolve a substantial portion of common violations, achieving a 77.23% error reduction rate in axe-core audits and 68.30% in WAVE reports. These results show the tool's capacity to address frequent accessibility violations, such as missing alternative text, poor color contrast, and improper use of landmarks.

However, the tool's applicability does not extend to all types of web pages, as FixAble was found to be less capable on web pages that are highly dynamic, restricted by logins, protected by content filtering systems, or heavily reliant on multimedia elements. These limitations emphasized the challenges encountered with page rendering, blocked access, and insufficient support for rich interactive content.

While FixAble demonstrated success in fixing static and semi-static web pages, it is recommended that further research should focus on resolving the challenges faced in this study. This includes rendering and remediating dynamic content completely, resolving issues with accessing restricted websites, handling modals properly, and increasing support for alt text generation with more types of media. It is also recommended to widen the coverage of errors remediated by the tool to encompass more types of violations, such as specific issues that impede assistive technologies, and other errors that may not be tackled by automated testing tools, as mentioned by literature. Nonetheless, FixAble remains a valuable tool for enhancing accessibility on a wide range of informational, educational, and government websites.

References

[1] Ahrefs. 2025. Top Websites Ranking in Philippines, June 2025. https://ahrefs.com/websites/philippines. Date accessed: June 8, 2025.

[2] Ahmad Al-Ahmad, Ibraheem Ahmaro, and Malik Mustafa. 2015. Comparison between web accessibility Evaluation tools. https://www.researchgate.net/profile/Ahmad-Al-Ahmad-2/publication/279181549_Comparison_between_web_accessibility_Evaluation_tools/links/558ca41308ae591c19da093e/Comparison-between-web-accessibility-Evaluation-tools. *Al-Madinah Technical Studies* 1, 4 (2015).

[3] Paul Bokingkito Jr, Jerame Beloy, Jerina Jean Ecleo, Apple Rose Alce, Nenen Borinaga, and Adrian Galido. 2025. Are We Inclusive? Accessibility Challenges in Philippine E-Government Websites. https://doi.org/10.3390/informatics12020041. In *Informatics*, Vol. 12. MDPI, 41.

[4] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, Gregg Vanderheiden, Wendy Chisholm, John Slatin, and Jason White. 2008. Web content accessibility guidelines (WCAG) 2.0. https://www.w3.org/TR/WCAG20/. *WWW Consortium (W3C)* 290, 1-34 (2008), 5–12.

[5] Zoran Jordanoski and Morten Meyerhoff Nielsen. 2023. The challenge of web accessibility: An evaluation of selected government websites and service portals of high, middle and low-income countries. https://doi.org/10.1145/3614321.3614343. In *Proceedings of the 16th International Conference on Theory and Practice of Electronic Governance*. 101–110.

[6] Thai Nguyen. 2024. Evaluating Automated Accessibility Checker Tools. https://cedar.wwu.edu/wwu_honors/80. (2024).

[7] Elmar Noche, Richard Hortizuela, Carlos Manuel Abalos, and Rhenea Lizlie Viray. 2024. Comparative Web Accessibility Compliance Evaluation of the State Universities and Colleges in the Ilocos Region Philippines. https://doi.org/10.1109/ICITDA64560.2024.10809801. In *2024 9th International Conference on Information Technology and Digital Applications (ICITDA)*. IEEE, 01–07.

[8] Department of Information and Communications Technology (DICT) Philippines. 2017. Memorandum Circular No. 2017-004. https://navy.mil.ph/images/DICT-MEMO-CIRCULAR-004-2017-W3C-WEB-CONTENT-ACCESSIBILITY-GUIDELINES-WCAG-2.pdf. Date accessed: June 5, 2025.

[9] U.S. Department of Justice Civil Rights Division. 2024. Fact sheet: New rule on the accessibility of web content and mobile apps provided by state and local governments. https://www.ada.gov/resources/2024-03-08-web-rule/. Date accessed: June 5, 2025.

[10] Official Journal of the European Union. 2019. Directive (EU) 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019L0882. Date accessed: June 5, 2025.

[11] Helen Petrie and Omar Kheir. 2007. The relationship between accessibility and usability of websites. https://doi.org/10.1145/1240624.1240688. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 397–406.

[12] Jonathan Robert Pool. 2023. Accessibility metatesting: comparing nine testing tools. https://doi.org/10.1145/3587281.3587282. In *Proceedings of the 20th International Web for All Conference*. 1–4.

[13] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines are only half of the story: accessibility problems encountered by blind users on the web. https://doi.org/10.1145/2207676.2207736. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 433–442.

[14] Kristen Bhing V Salvio. 2020. Extending the evaluation on Philippine e-Government services on its accessibility for disabled person. https://doi.org/10.1109/WorldS450073.2020.9210374. In *2020 fourth world conference on smart trends in systems, security and sustainability (WorldS4)*. IEEE, 428–434.

[15] Semrush. 2025. Trending Websites in the Philippines. https://www.semrush.com/trending-websites/ph/all. Date accessed: June 8, 2025.

[16] Markel Vigo, Justin Brown, and Vivienne Conway. 2013. Benchmarking web accessibility evaluation tools: measuring the harm of sole reliance on automated tests. https://doi.org/10.1145/2461121.2461124. In *Proceedings of the 10th international cross-disciplinary conference on web accessibility*. 1–10.

[17] Web Accessibility Initiative (WAI). 2025. Web Accessibility Evaluation Tools List. https://www.w3.org/WAI/test-evaluate/tools/list/. Date accessed: June 7, 2025.

[18] WebAIM. 2025. The WebAIM Million: The 2025 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million. Date accessed: June 5, 2025.