JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 9 Number 1 | June 2024: 70-79
DOI: 10.15575/join.v9i1.1310

AI-Powered Real-time Accessibility Enhancement: A Solution for Web Content Accessibility Issues

Samir Kumar Dash
Cisco Systems (India) Pvt. Ltd., Bengaluru, Karnataka, India

Article Info

Article history:

Received February 02, 2024
Revised March 03, 2024
Accepted March 05, 2024
Available Online April 23, 2024

Keywords:

Accessibility
ARIA
Artificial Intelligence
Security
Web

ABSTRACT

The web accessibility landscape is a significant challenge, with 96.3% of home pages displaying issues with Web Content Accessibility Guidelines (WCAG). This paper addresses the primary accessibility issues, such as missing Accessible Rich Internet Applications (ARIA) landmarks, ill-formed headings, low contrast text, and inadequate form labeling. The dynamic nature of modern web and cloud applications presents challenges, such as developers' limited awareness of accessibility implications, potential code bugs, and API failures. To address these issues, an AI-enabled system is proposed to dynamically enhance web accessibility. The system uses machine learning algorithms to identify and rectify accessibility issues in real-time, integrating with existing development workflows. Empirical evaluation and case studies demonstrate the efficacy of this solution in improving web accessibility across diverse scenarios.

Corresponding Author:

Samir Kumar Dash
Cisco Systems (India) Pvt. Ltd., Bengaluru, Karnataka, India
Email: samdash@cisco.com
ORCID ID: https://orcid.org/0009-0007-8887-9265

1. INTRODUCTION

A study by webaim.org on accessibility in 2023 found that 96.3% of home pages had been found to have problems with aligning to the Web Content Accessibility Guidelines (WebAIM: The WebAIM Million - the 2023 Report on the Accessibility of the Top 1,000,000 Home Pages. 29 Mar. 2023, webaim.org/projects/million). These are only mistakes that were found automatically and are very likely to be WCAG conformance failures (Web Content Accessibility Guidelines (WCAG) 2.1). This means that the actual WCAG 2 A/AA conformance level was lower since computer testing couldn't find all possible WCAG failure types (Web Content Accessibility Guidelines (WCAG) 2.1). While, just the web accessibility evaluation is a costly and complex process due to limited time, resources, and ambiguity [1], the cost of fixing them definitely comes with a larger cost that also factors into any organization's roadmap having the goals to fix the accessibility issues of existing websites. This results in the fact that despite the recognized importance, the accessibility of the websites remains a serious challenge, making their content partially or completely inaccessible to some categories of the population [2].

Various factors influence the implementation of accessibility on the web. Research indicates that the complexity of websites, county population density, budget resources, and the percentage of the population with disabilities play significant roles in determining the level of web accessibility [3]. Additionally, adherence to WCAG is crucial for organizations to meet minimum standard accessibility guidelines and avoid costly mistakes during software and website development [4]. Furthermore, utilizing tools like accessibility databases (AD) can aid in evaluating and increasing the accessibility of public facilities, highlighting the importance of communication and iterative processes within

organizations for successful implementation [5]. Incorporating official standards like WCAG and conducting internal audits can help in identifying areas for improvement and ensuring a more accessible web environment [6].

Most accessibility challenges are attributed to four main factors: missing ARIA landmarks, ill-formed headings, low-contrast text, and form labeling. Web interaction, for most blind users, is made possible through mediation by screen readers to read on-screen text in a sequential manner [7], [8]. However, these are impacted by the fact that many websites miss the ARIA landmark and do not use native HTML5 tags for semantic support. Additionally, many have multiple ill-formed headings, no hierarchy, or are missing headings entirely. Low contrast text, below WCAG 2 AA thresh-olds, is found on 83.6% of home pages. Form labeling, which is not properly used, is a major challenge in making forms accessible to screen readers. However, despite this fact, there is little formal literature about the accessibility of web development with a screen reader [9].

Apart from the scenario where content is directly updated by the developer, many of these also further complicate the situation in dynamic websites and web/cloud applications, where content, forms data visualizations, etc., are generated dynamically through program code, where it may be caused due to the following three reasons: (1) developer may not be fully aware of the accessibility content being generated by code in the specific situation where complex views and interactions are involved; (2) any bug in the code may generate broken DOM may cause such issues; (3) API failure etc. may render some content/forms halfway. So, in any case, accessibility is severely affected.

The evolution of AI has profoundly reshaped numerous domains across industries, heralding a new era of innovation and efficiency. AI leverages vast datasets to conduct intricate analyses and derive actionable insights, offering data-driven solutions to complex problems through sophisticated models. AI sifts through large volumes of structured and unstructured data by employing advanced algorithms and identifying patterns, correlations, and anomalies that may elude human analysis.

2. METHOD

This research conducts several activities begin with literature review, problem definition, proposed solution and evaluation. In literature review, this research conducts a comprehensive review of existing literature on web accessibility [10], [11], [12], WCAG guidelines [13], [14], ARIA landmarks [15], machine learning algorithms for accessibility enhancement [16], and related topics. After clearly define the primary accessibility issues faced by modern web and cloud applications in problem definition activity, then this research describes the AI-enabled system proposed for dynamically enhancing web accessibility. The need for a dynamic solution to address web accessibility challenges that proposed in this research is AI or machine learning. Then, using experimental evaluation in the research would involve conducting controlled experiments to assess the effectiveness of the proposed AI-enabled system for enhancing web accessibility.

3. RESULT AND DISCUSSION

3.1. The Experiment Using AI

Given the expansive capabilities of AI in addressing diverse use cases through data-driven approaches, there's considerable potential for leveraging this technology to tackle accessibility issues. Recognizing this, as part of the experiment, a custom application was created, utilizing technologies like PHP, Node.js, Python, and JSON at the backend, and the client was developed using HTML5 and JavaScript. Into the development process, this application aims to proactively identify and rectify accessibility issues in real time.

3.2. The Major Accessibility Challenges to Address

As part of the experiment, the following major contributors of the webpage accessibility are identified to be addressed.

3.2.1. Missing ARIA Landmark

Many websites miss the landmark ARIA being defined to cater to the semantic need of the websites/application to support screen readers. Even many of the websites do not use the native HTML5 tags supporting the key regions using the following structure [17]:

```
<header><header>
```

```
<main></main>
<footer></footer>
<nav></nav>
```

Rather, mostly `<div>` tags are used and continue creating text blocks, and these do not use the proper structures. Also, for images, WCAG guideline says that non-text content must have a text alternative (Web Content Accessibility Guidelines (WCAG) 2.1). That text alternative must have the equivalent purpose. One of the exceptions is for decorative images, images used for formatting, or that would be invisible to everyone. These must be "implemented in a way that can be ignored by assistive technology." The accepted method is to have a null alt attribute, as a missing alt attribute would be read out. Also, from ARIA's perspective, it should have `aria-hidden=true`.

3.2.2. Missing or Ill-formed Headings

Because headings from `<h1>` to `<h6>` are the primary mechanism screen reader users use to navigate content, their proper implementation is important in the context of accessibility. In many cases, 20.1% of home pages had more than one `<h1>` tag. Many sites have ill-formed headings, i.e., no hierarchy of the headings is maintained, like under lower headings like h5 or h6, the higher headings like h2 or h3 are placed. In many cases, headings are missing totally, and instead, <p> directly contain texts with larger size font sizes as a custom code / CSS. All these give rise to challenges for screen-readers to navigate through content reliably.

3.2.3. Low Contract Text

Low contrast text, below the WCAG 2 AA thresholds, was found on 83.6% of home pages. This indicates that most web pages surveyed in the study fail to meet the basic accessibility standards regarding text readability. This is the most detected accessibility issue. ow contrast text refers to text elements on a web page that do not have sufficient contrast ratio between the foreground (text) and background colors. This lack of contrast makes it difficult for users, especially those with visual impairments, to read and comprehend the content. The Web Content Accessibility Guidelines (WCAG) 2 AA standards specify the minimum level of accessibility required for web content. One of the criteria for meeting WCAG 2 AA compliance is ensuring an adequate color contrast ratio between text and background colors. Specifically, the contrast ratio should meet certain thresholds to ensure readability for users with low vision or color deficiencies.

3.2.4. Form Labelling

From the above-mentioned research of accessibility, it is found out that 35.8% of form inputs identified were not properly labeled (either via `<label>`, `aria-label`, `aria-labelled-by`, or `title`). This is a major challenge in making forms accessible to screen readers.

3.3. Limitation of Existing Solutions

As of today, several existing solutions try to address the accessibility issue, yet those have certain limitations, as outlined below:

1. The guidelines for developing accessible content, e.g., WCAG2.0, etc. But this depends on the developer who codes for static sites. As we see, even if guidelines exist, the accessibility challenges are significant, as proved by research made globally, e.g., by webaim.org. Overall, exposure to accessibility, limitations in technology, lack of time, guideline ambiguity, and organizational structure are contributing to the non-implementation of the accessibility features during development.
2. Tools and IDEs for content creation and HTML DOM design /development focus on checking and validating the markups while the page is being made. This kind of IDE has existed for a long time, but it is still not able to solve the key issues mentioned [18]. Also, these tools do not control dynamic site content.
3. The tools to check site accessibility exist, where some of the challenges can be detected and the developer notified, but this is not real-time, and it does not automatically fix anything.

3.4. Solution Hypothesis

The proposed invention is about a solution includes a webpage renderer or HTML DOM generator that works in real time as part of the webpage and uses AI/ML to comprehend the existing state of the page and fixes each of the 4 issues identified in the previous section. The webpage includes the following two additional modules that illustrated in the Figure 1.

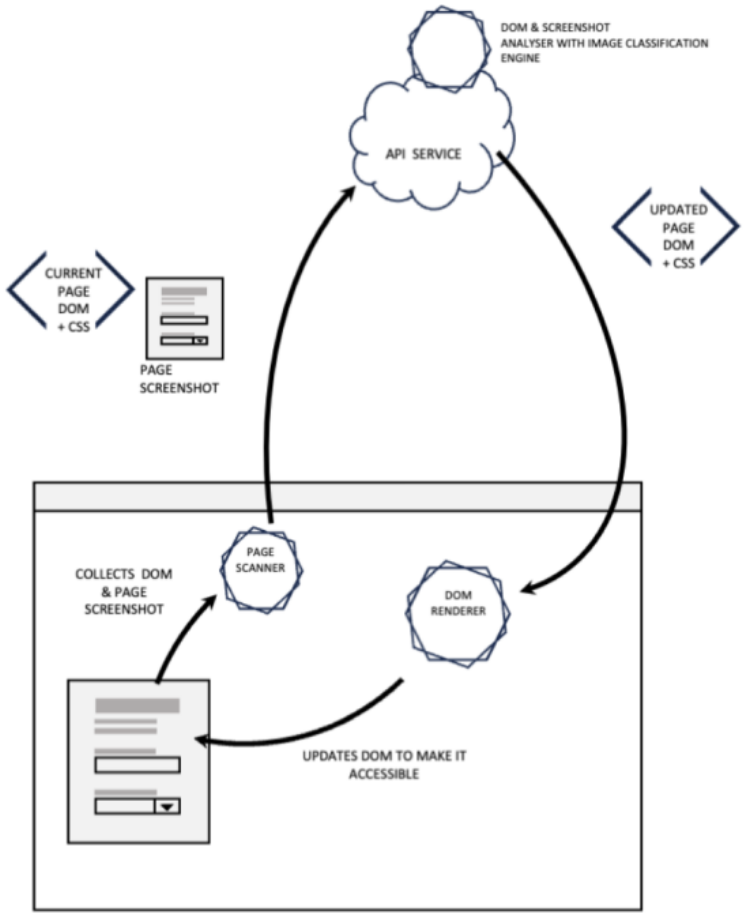

Figure 1. The 'Page Scanner' and 'DOM Renderer' are two modules that are added to the webpage via additional `<scripts>` tags.

Page Scanner is one for scanning pages and collecting the currently rendered DOM of the page and a screenshot. **DOM Renderer** is the second script, a DOM rendered that replaces some or more portions of the webpage DOM in real-time, based on what it received from the server. The solution includes a server component - an automated DOM and Screenshot analyzer tool - enabled with AI or a rule-based engine that can analyze the existing DOM and the screenshot image of the page to run pre-trained image classifier models to identify the various elements and form controls along with their co-ordinates and the final rendered state, etc. Then, there is **Information Flow,** which has four steps through which the solution workflow should be executed as follows:

1. *Step 1:* The page DOM is rendered in the browser. When the page load completes, the "Page Scanner "component/script collects the rendered DOM information.
2. *Step 2:* The "Page Scanner" component/script takes a screenshot of the page (e.g., script libraries like html2canvas can take reliable screenshots of the rendered HTML in the client)
3. *Step 3:* "Page Scanner" sends the collected DOM and screenshot of the page to the Server through the rest of the APIs.
4. *Step 4:* In the server/cloud, the AI/ML-enabled automatic analyzer component analyses the data and carries out the following four tasks:
   a. *Task 1*: Fixing ARIA

In identifying and enhancing web page accessibility, the cloud-based automated analyzer tool focuses on detecting potential ARIA landmarks, such as headers, footers, navigation bars, and main content regions. The tool analyzes the page's structure and element positioning to identify areas that can benefit from ARIA landmarks to improve screen reader navigation and user experience. Subsequently, the tool intelligently injects appropriate ARIA tags, such as "role" and "aria-label," into the HTML markup. This process effectively establishes semantic relationships between page elements and ARIA landmarks, allowing assistive technologies to navigate the page more efficiently and providing a more inclusive browsing experience for users with disabilities. For `<img>` tags that are used for decorative purposes or for non-display purposes, `aria-hidden=true` is to be used to be understood by ARIA. Null (blank) alt attributes have been established long before and will be accepted by more assistive technologies, so add a null alt tag for the `<img>` element on the page that screen readers do not read out.

b. *Task 2:* Fixing Ill-formed Headings

The Cloud analyzer component is an automated tool that is utilized to examine the webpage's structure and layout. First, the tool identifies all page elements, capturing their position coordinates and size. This data determines if any header elements need to be included or properly structured. The tool then assesses the size of text blocks and their positioning in relation to each other. Using this information, the tool identifies and segments different sections on the page. To enhance the semantic structure and improve accessibility, the tool intelligently replaces larger text sizes with H1 to H6 tags where needed and injects appropriate heading elements, e.g., `<h1>` till `<h6>`. This process ensures the proper hierarchical organization of content, leading to more accessible and search engine-friendly web pages.

c. *Task 3*: Fixing Low Contrast Text

In a similar approach, the cloud-based automated analyzer tool scans a screenshot of a webpage to extract critical visual data. The tool utilizes advanced image processing techniques to assess text density, font size, and color contrast within the captured screenshot. By analyzing these elements, the tool gains insights into the web page's readability and accessibility. Based on the data collected, the tool generates HTML and CSS code, proposing changes to the web page's DOM structure and styles. It suggests adjustments to improve text legibility, ensure appropriate font sizes, and enhance color contrast, complying with accessibility guidelines. By automating this process, web developers can efficiently implement accessibility improvements, leading to a more inclusive user experience for all visitors.

d. *Task 4:* Fixing Form Labelling

The automatic analyzer tool also addresses the issue of missing labels within web forms. The tool identifies input fields, dropdown menus, and other form elements by intelligently analyzing the web page's DOM data. It then performs an in-depth contextual analysis to understand the relationships between these elements and the surrounding content. Using this information, the tool generates descriptive labels for the form elements, filling in the gaps where labels are absent or insufficient. These new labels are meticulously crafted to provide meaningful and informative descriptions, ensuring that users with assistive technologies can accurately interpret and interact with the form. By automating the label generation process, the tool significantly streamlines the accessibility optimization process, making it easier for developers to create accessible and user-friendly web forms.

5. *Step 3:* Finally, the resultant output DOM is sent back to the page where the DOM Renderer component script, which then uses this DOM to replace the existing DOM of the page, thereby making it instantly accessible for the previously pointed out 4 accessible issues.

3.5. Experimental Design

The study involved the creation of a custom software module using PHP and JQuery, supporting Rest API calls using JSON.

3.5.1. The Tuned Model

The training data was used to train OpenAI's GPT 3.5 model to get a tuned model [19], which acted as the server component for the DOM analyzer, that can accurately generate the fixed DOM or the HTML elements. For this sample data was prepared based on the before and after state of DOMs. The high-level flow is shown in the Figure 2.

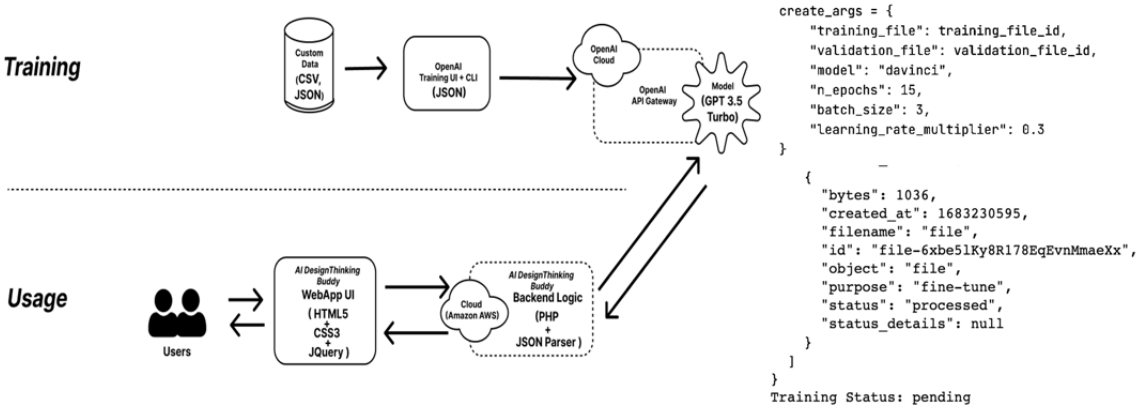

Figure 2. Sample data in JSON format was used to train the standard OpenAI GPT model

By using the OpenAI's CLIP zero-shot image classifier for image analysis to understand image type [20]. This approach allows AI to "understand" the context of the image and decide if aria-hidden=true would be applied or not.

3.5.2. The Training Data

The training data used was based on JSON format. Each set of training JSON contained pre and post-state of various scenarios of Accessibility issues. An is as follows, where the pre-state uses a DOM example with missing ARIA attribute for a <nav> element :

```
{"messages":
[{"role": "system", "content": "<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Services</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>"},
{"role": "user", "content": "Fixed Version HTML with Accessible Rich Internet
Applications (ARIA) roles and attributes"},
{"role": "assistant", "content": "<nav role="navigation">
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Services</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>"}
]}
```

3.5.1. The DOM Renderer

The HTML DOM renderer that worked in real time as part of the webpage, was created using jQuery and CSS. Once it received the output from the server through ajax, applied the output to replace the existing DOM elements with the fixed DOM.

3.6. Experiment Execution

Multiple sample webpages with basic block on HTML elements with ill-formed and missing tags were introduced was accessed over localhost and during the runtime the 'Page Scanner' and 'DOM Renderer' scripts were injected at the runtime. As per the expectation, the software analyzed the pages and updated the pages DOM with suggested fixed DOM by the AI.

In Figure 3, one sample case is shown, where the *Before* and the *After* states are provided, how it appears visually. The fixed state may look similar visually as the goal was to make minimal changes visually, and as part of the accessibility fixes, correct the ARIA, ALT tag, Heading types etc. to be fixed. With a closure look, there are some subtle differences like the margin of the heading title seems to be

increased, as the tags of earlier got changed to `<h1>` tags. Similarly, the label of the text area for message is also appeared in the fixed version, where as in the before applying fix, it was not visible.

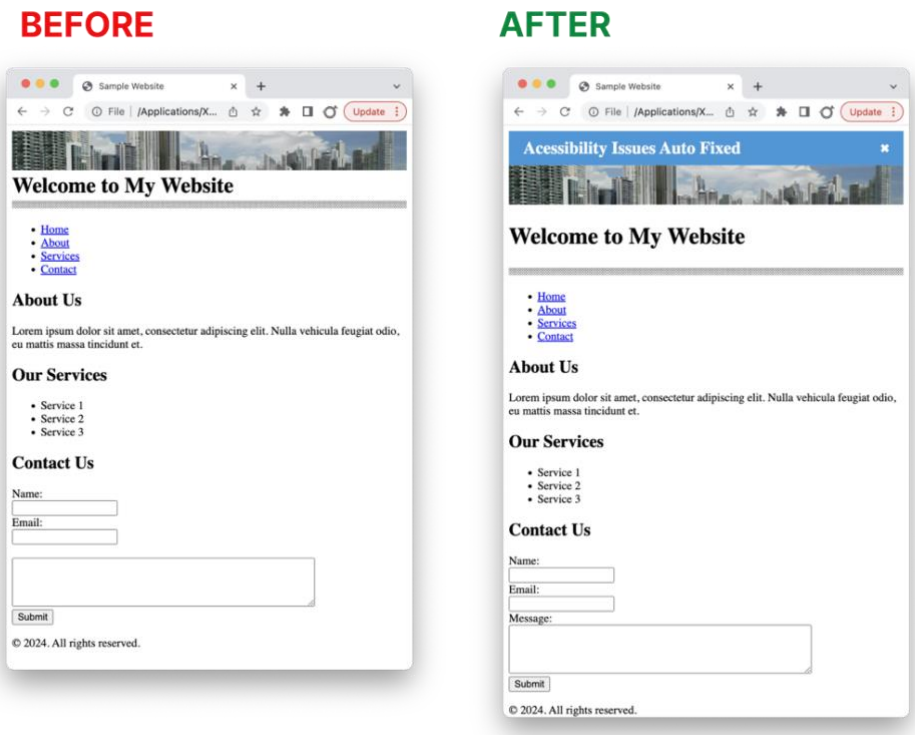

Figure 3. A sample webpage was accessed over localhost and during the runtime the scanner and renderer scripts were injected at the runtime to fix the page.

3.7. Result and Analysis

Based on the experiment ran on the sample webpages, the outcome was satisfactory. The AI powered software was able to fix the accessibility issues in scope in all sample webpages. The before and after states (fixed webpage DOM) are compared in Figures 4 and 5. The fixed DOM, has the missing ARIA landmarks were added. The decorative image with pattern was identified and a null alt attribute is added along with the `aria-hiddent="true"` attribute. The banner image got the alt attribute describing the image that was generated from the *OpenAI's* CLIP zero-shot image classifier.

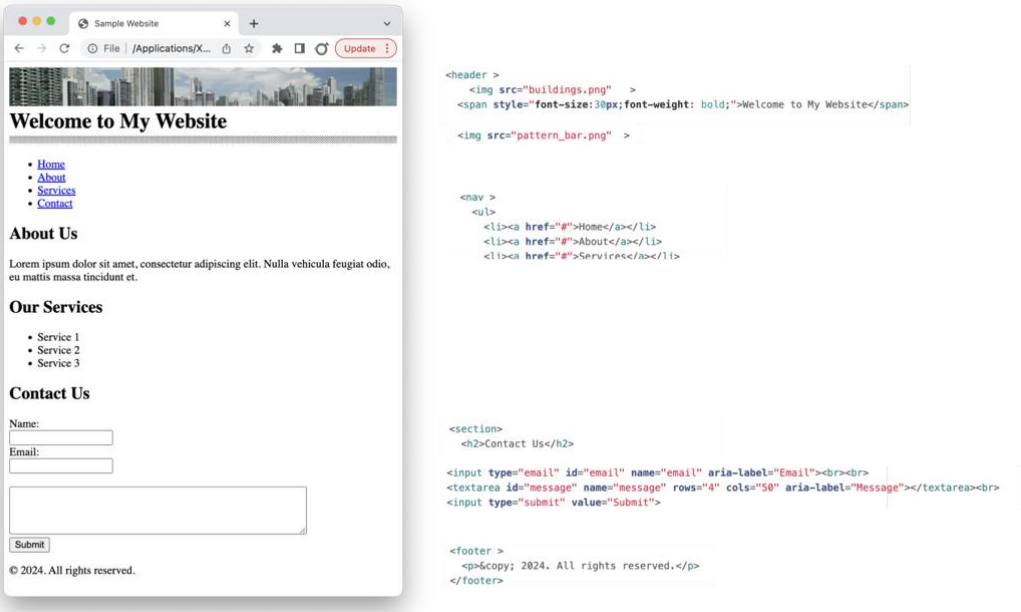

Figure 4. A sample webpage was accessed over localhost and during the runtime the scanner and renderer scripts were injected at the runtime.

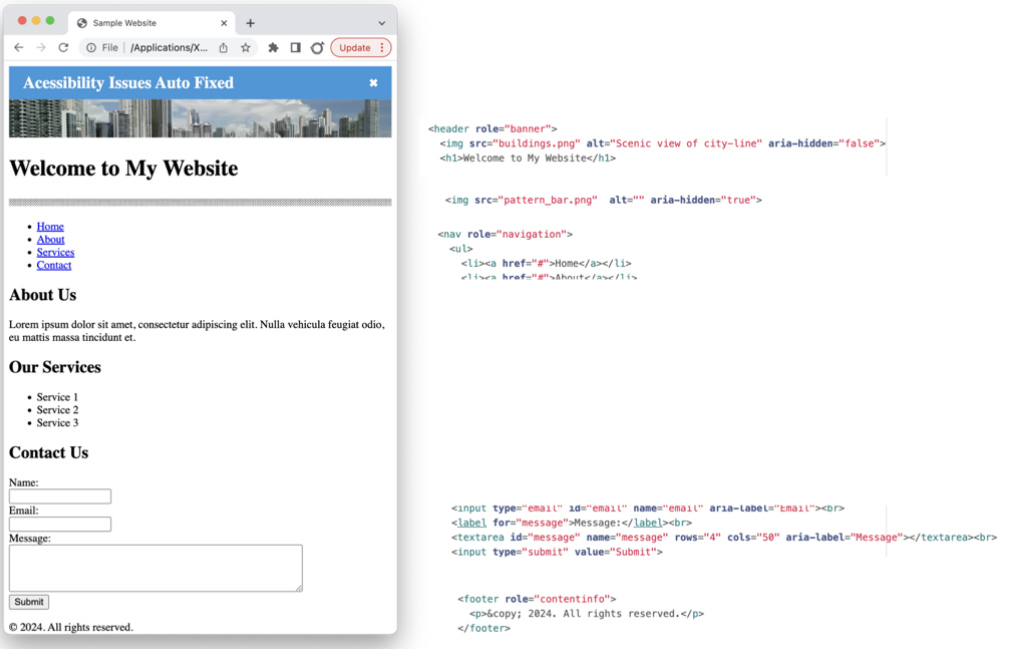

Figure 5. A sample webpage was accessed over localhost and during the runtime the scanner and renderer scripts were injected at the runtime.

6. CONCLUSION

From the experiment conducted, it is concluded that using an AI-powered system can, in real time, fix some of the major web page accessibility challenges. The current study focuses on specific accessibility issues that are mostly found across the internet, e.g., ARIA tags, missing tags, and ALT tags etc. However, further study can be conducted to extend this approach to fix issues such as tab-indexing, form elements etc. Also, by training the tuned model with a wide variety of data, the solution can be made production-grade.

REFERENCES

[1] A. Hambley, Y. Yesilada, M. Vigo, and S. Harper, "Web Structure Derived Clustering for Optimised Web Accessibility Evaluation," in *Proceedings of the ACM Web Conference 2023*, New York, NY, USA: ACM, Apr. 2023, pp. 1345–1354. doi: 10.1145/3543507.3583508.

[2] Z. Jordanoski and M. Meyerhoff Nielsen, "The challenge of web accessibility: an evaluation of selected government websites and service portals of high, middle and low-income countries," in *Proceedings of the 16th International Conference on Theory and Practice of Electronic Governance*, New York, NY, USA: ACM, Sep. 2023, pp. 101–110. doi: 10.1145/3614321.3614343.

[3] G. Carlsson, O. Jonsson, S. Olander, M. Salén, E. Månsson Lexell, and B. Slaug, "Exploration of a Web-based accessibility tool for public facilities," *Facilities*, vol. 41, no. 15/16, pp. 66–84, Dec. 2023, doi: 10.1108/F-10-2022-0132.

[4] Y. Bai, J. Grzeslo, B. Min, and K. Jayakar, "Accessibility of local government websites: influence of financial resources, county characteristics and local demographics," *Univers Access Inf Soc*, vol. 20, no. 4, pp. 851–861, Nov. 2021, doi: 10.1007/s10209-020-00752-5.

[5] N. Palani, *The Web Accessibility Project*. Boca Raton: Auerbach Publications, 2022. doi: 10.1201/9781003299431.

[6] T. W. Hostetler *et al.*, "Web accessibility trends and implementation in dynamic web applications," Feb. 2022.

[7] A. Hambley, "Empirical web accessibility evaluation for blind web users," *ACM SIGACCESS Accessibility and Computing*, no. 129, pp. 1–5, Jan. 2021, doi: 10.1145/3458055.3458057.

[8] B. Gomes, J. Rios, and K. R. H. Rodrigues, "Challenges for the Implementation of Accessible Web and Mobile Systems," 2020, pp. 138–158. doi: 10.1007/978-3-030-46130-0_8.

[9] C. Kearney-Volpe and A. Hurst, "Accessible Web Development," *ACM Trans Access Comput*, vol. 14, no. 2, pp. 1–32, Jun. 2021, doi: 10.1145/3458024.

[10] K. Wallace, "Workshop: Accessibility in Web Design," in *2022 IEEE International Professional Communication Conference (ProComm)*, IEEE, Jul. 2022, pp. 462–463. doi: 10.1109/ProComm53155.2022.00091.

[11] S. S. Macakoglu and S. Peker, "Web accessibility performance analysis using web content accessibility guidelines and automated tools: a systematic literature review," in *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, Jun. 2022, pp. 1–8. doi: 10.1109/HORA55278.2022.9799981.

[12] N. Palani, *The Web Accessibility Project*. Boca Raton: Auerbach Publications, 2022. doi: 10.1201/9781003299431.

[13] K. Boyalakuntla, A. S. M. Venigalla, and S. Chimalakonda, "WAccess -- A Web Accessibility Tool based on WCAG 2.2, 2.1 and 2.0 Guidelines," Jul. 2021.

[14] R. S. Germano and I. Frango Silveira, "WCAG-Easy Tool : A tool based in the WCAG to learn web accessibility," in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, Jun. 2022, pp. 1–6. doi: 10.23919/CISTI54924.2022.9820012.

[15] R. P. M. Fortes, H. L. Antonelli, and A. de Lima Salgado, "Accessibility and Usability Evaluation of Rich Internet Applications," in *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*, New York, NY, USA: ACM, Nov. 2016, pp. 7–8. doi: 10.1145/2976796.2988221.

[16] F. Zambrano and E. Muñoz, "Statistical machine learning methods applied in the study of web accessibility: a literature review," *Minerva*, vol. 1, no. Special, pp. 150–157, Dec. 2022, doi: 10.47460/minerva.v1iSpecial.90.

[17] J. O. Connor, *Pro HTML5 Accessibility*. Berkeley, CA: Apress, 2012. doi: 10.1007/978-1-4302-4195-9.

[18] M. M. Alnahari, J. Chakraborty, M. Mohamed, and A. Ali-Gombe, "Arabic web accessibility analysis: findings from a usability study of Arabian web developers," *Human-Intelligent Systems Integration*, vol. 4, no. 3–4, pp. 81–96, Dec. 2022, doi: 10.1007/s42454-022-00045-7.

[19] Open AI, "Fine-tuning," Open AI Documentation. Accessed: Apr. 15, 2024. [Online]. Available: https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset.

[20] S. Rahman, S. Khan, and F. Porikli, "A Unified Approach for Conventional Zero-Shot, Generalized Zero-Shot, and Few-Shot Learning," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, Nov. 2018, doi: 10.1109/TIP.2018.2861573.