# Adapting Modularized Web Applications to Web Accessibility Standards

L. Žuliček*, S. Tomić**, I. Bosnić**
* Netgen, Zagreb, Croatia
** University of Zagreb Faculty of Electrical Engineering and Computing, Zagreb, Croatia
zuliceklora@gmail.com, {sinisa.tomic, ivana.bosnic}@fer.hr

***Abstract* - This paper presents a web application designed to evaluate the accessibility of a modular Content Management System called Quilt CMS, widely used as a solution for Croatian higher education institutions. Aimed primarily at web developers, this tool is focused not on evaluating subpages as whole documents, but on modules of a Quilt CMS instance used in their context, providing cleaner analysis and easier code refinement. The modules were evaluated using AATT - Automated Accessibility Testing Tool, composed of Axe, Chrome and HTML Code Sniffer accessibility evaluation libraries. The modules' code is accessible via the newly created, customizable REST API methods, providing a solution easily extendible to other web platforms. The web accessibility of modules is evaluated according to the following accessibility guidelines: Web Content Access Guidelines (WCAG 2.0 and 2.1), Web Accessibility Initiative - Affordable Rich Internet Applications (WAI-ARIA) and Section 508. The analysis was done on one of the websites using this CMS, but is applicable in other instances due to module code sharing. The results were used to improve the source code of selected Quilt CMS modules, providing a higher level of accessibility norms' implementation.***

***Keywords* - web accessibility; content management systems; modules, W3C, WCAG, WAI-ARIA, Quilt CMS**

## I. INTRODUCTION

The basic idea of the web is to be accessible to all people, regardless of their language, location, capability, or device through which they access it. Web accessibility is key for developers and organizations looking to create high-quality applications and tools, and the web must be accessible to people of all abilities.

In order to provide a single common standard for reading and creating web content, the Web Accessibility Initiative (WAI) has published the WCAG (Web Content Accessibility Guidelines) and the WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) [1]. With the same goal, the US Congress amended the Rehabilitation Act and added Section 508 accessibility rules [2].

In this paper, we present an application created to help developers to quickly identify possible problems with accessibility of modular web applications. The application, instead of testing page by page, checks and arranges errors by modules. Modules are components used to structure specific elements (e.g. module "Repository", "Notification", etc.). Due to the division of code by modules, the application easily integrates with other instances.

## II. SPECIFICATIONS AND ASSESSMENT TOOLS FOR WEBSITE ACCESSIBILITY

In the last few years, many guidelines and rules have been published to improve web accessibility so that people with various (dis)abilities can perceive, understand, manage and communicate using the web [3].

### *A. Accessibility Guidelines*

The W3C (World Wide Web Consortium) is the main international organization for Internet standards, which has published a series of guidelines and recommendations for creating more accessible web content. The term web content generally refers to information on a web page or application, including natural information such as text, images, sounds and tags that define the structure, presentation etc.

- The WCAG (Web Content Accessibility Guidelines) became the ISO standard in 2012. WCAG 2.1. consists of 13 guidelines that are organized under 4 principles: visible, operable, understandable and robust. For each guideline, there are success criteria that can be tested, and are divided into three levels of compliance: A (lowest), AA, and AAA (highest).
- The WAI-ARIA (Accessible Rich Internet Applications) guidelines focus on dynamic content and more advanced parts of the user interface (tree structure, drag and drop, content that depends on user actions and mouse usage).

Under Section 508 of the Rehabilitation Act, the U.S. Congress requires all government agencies to provide employees and people with disabilities the same access to information. The relevant standard for websites is paragraph 1194.22, "Web-based Intranet and Internet Information and Applications" which sets out 16 rules that must be met. Some rules have similar guidelines in WCAG 1.0, and some cover success criteria in WCAG 2.0.

Furthermore, as of 22 December 2016, European Directive (EU) 2016/2102 is in force, which seeks to provide people with disabilities with better access to public service websites and mobile applications. The Directive obliges public sector bodies' websites and applications to meet certain technical accessibility standards.

### B. Accessibility Evaluation Tools

Various tools that evaluate sites according to accessibility standards, guidelines and recommendations exist, such as: DynoMapper, A11Y Compliance Platform, Accessibility Checker by CKSource, AChecker, WAVE etc. There are also tools that combine a number of tools, such as AATT[1] (Automated Accessibility Testing Tool), developed by PayPal consists of Axe, Chrome and HTML CodeSniffer API services for testing applications for compliance with accessibility standards.

- Axe[2] is an open source tool maintained by Deque Systems. It tests content according to WCAG 2.0 and WCAG 2.1. Level A and AA guidelines, WAI-ARIA guidelines and best practice rules - rules that do not necessarily comply with the WCAG success criterion, but are industry-accepted practices that improve the user experience.
- Chrome[3] is a library developed by Google and it consists of API services that check a collection of accessibility issues (includes, but is not limited to, calculating contrast ratios and suggesting colors, and retrieving and verifying ARIA attributes and states).
- HTML_CodeSniffer[4] is a client-side script that verifies HTML source code and conducts testing at all three levels of compliance with the WCAG 2.1 and U.S. law policy "Section 508".

Other tools require manual testing of each page, one at a time, and with AATT, accessibility testing is easily integrated into an existing automation package. It allows you to configure the test server and test the individual HTML code, or modules in this case.

## III. Related Work

A huge number of tools has been proposed as a solution for accessibility evaluation. At the moment, W3C offers a list of 160 tools, complying with some of the 14 accessibility guideline specifications [4]. These tools often become obsolete over time as guidelines change and the machine-readable way of providing the testing rules, such as Accessibility Conformance Testing specification is not provided. Sometimes the tools lack functionalities needed by website users or website developers. They also often do not provide the evaluation results in a standardized format, such as EARL – Evaluation and Report Language, which would help in easier integration and reuse.

An evaluation of 126 online web accessibility tools, using the W3C criteria for accessibility tools selection [5] is provided in [6]. Another evaluation assesses eight popular tools, but also proposes new methods for such evaluations [7]. The web accessibility evaluation of higher education websites was performed on 44 websites in India, using two tools: Axe and TAW. Besides the statistical analysis, the paper provides the steps for making those higher education websites more accessible [8]. The same tool, Axe, is proposed along the Tenon application, in some higher education institutions in the US [9]. In Latin America, the accessibility evaluation of WCAG 2.0 guidelines was performed on 348 main university websites, using the WAVE tool, showing the needs for better accessibility policies and directives [10].

The accessibility research presents specific issues. There is more need for DOM support, to evaluate dynamic web pages with content being created after the page loads, or to evaluate the accessibility of systems based on modules, not on subpages. One approach to solving the first problem can be the implementation of a browser plugin, such as MAUVE++ [11]. Another approach to the issue of huge websites with thousands of subpages, often based on content management systems, is clustering subpages of a web application and selecting representative pages for score calculation, to improve the evaluation speed [12]. Another well-researched topic is using alternative text for images, one of the basic elements of accessibility for blind persons. For instance, some authors develop algorithms to automatically classify alternative text as “descriptive” or “undescriptive” [13]. While these elements are important, it seems that the need for evaluating the “real usability” factors, such as ease of understanding the page structure and navigation, would be more important for people with reading disability [14].

Not much effort is made to help developers in the process of developing the accessible web applications. From the developers’ perspective, a survey on IBM web developers, from 2010, shows the need for clear explanation of the automatically detected problems, but pinpointing the location of the problem on the rendered page view and the opportunity to experience the website from the perspective of users with disabilities [15].

## IV. Application Proposal

### A. User Requirements

Existing testing tools test only one web page (one URL) and it would be very tedious for developers to test all subpages of one domain. As web pages created using Quilt CMS consist of modules, the idea of this approach is to enable accessibility testing and error reporting by modules. For accessibility evaluation, AATT (Automated Accessibility Testing Tool) API services are used that evaluate modules by the most important accessibility guidelines (WCAG, WAI-ARIA and Section 508).

The user selects multiple test modules, the pages where the modules are placed, and the test options. Testing options include selecting the tool to be tested, selecting standards and selecting the impact of the error (critical, serious and minor). They can be saved and downloaded so that the user does not have to repeat the selection next time and for easier comparison of the results.

Analysis of results consist of error names, descriptions, impact and position in the code. The results, which can include the links to websites with solutions and tips, can be downloaded as a CSV file.

[1] https://github.com/paypal/AATT
[2] https://www.deque.com/axe/
[3] https://github.com/GoogleChrome/accessibility-developer-tools
[4] http://squizlabs.github.io/HTML_CodeSniffer

### B. Technologies and Tools

React, an open source JavaScript library, is used to build the user interface; accessibility testing was implemented using the already mentioned AATT API service. To make the application as interactive and easy to use as possible, SCSS, a preprocessor scripting language that is interpreted or compiled into CSS (Cascading Style Sheets), is used to display HTML elements, in combination with the Bootstrap and FontAwesome libraries.

### C. System Architecture

To evaluate the interface, a web application for testing and analysis was created that communicates with the user via a graphical interface from a browser. To retrieve data, API (Application Programming Interface) services are used - API for retrieving data on Quilt CMS (Content Management System) modules and AATT (Automated Accessibility Testing Tool) API for assessing and testing the source HTML (HyperText Markup Language) code from other servers.

Quilt CMS is a modern platform built on open source technologies, which supports a large number of uses - from a simple modular system to advanced content management system providing many integrations with external applications. It is often used as a management system for various academic institutions in Croatia, as well as abroad.

The module is a user interface component that generates dynamic content. Given the context, it can display different types of information providing users the ability to customize the content. For example, the "Notifications" module may also exist on the "Mathematics 2" and "Parallel Programming" course pages, but will display different content (information).

To obtain information about modules (portlets), special REST API methods have been developed. Retrieving content from a web address retrieves the HTML result of the module on the requested page. HTML content contains all elements of the page (CSS, JavaScript, etc.), but does not contain elements of the design itself or other modules that can be found on the page. All methods are of the GET type and return a response in JSON format. After sending the request to download the list of modules, it is necessary to select the module for which the user wants to retrieve the HTML and the pages where the module is located, and which the user wants to test. The API will return only those pages where the user has permission to retrieve data. This solution is built to integrate with any existing test environment and is easily extended with other web platforms.

The user interface displays the application information to the user (modules, errors, analysis, testing options) and allows interaction. Business logic processes server data, sends (module codes) or receives it (errors analysis) from the AATT API service and displays it on the user interface. Special API methods retrieve information about Quilt CMS modules - the pages on which they are located and their code. The system architecture flow is shown below in Figure 1.

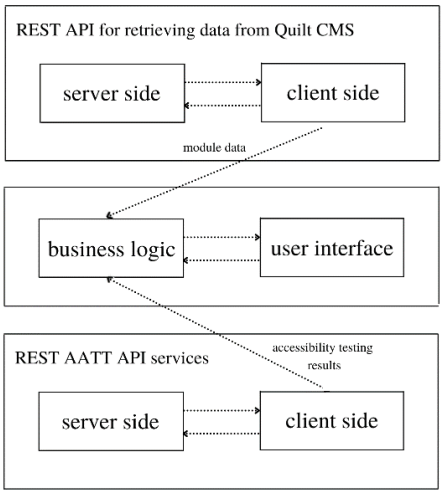


Figure 1. System architecture graph

## V. IMPLEMENTATION

The application consists of two main parts - selection of the modules to be tested and selection of the test method.

### A. Application Workflow

The application code is divided into two directories. The *api* directory contains the code needed to run the AATT API service, while the *client* directory contains the code needed to implement the user interface and call the API service methods (AATT and Quilt CMS).

When the application is launched, the user can test the two websites - the Faculty of Electrical Engineering and Computing [5] and the Study of Energy Efficiency and Renewable Energy Sources[6] .

Once a website is selected, the API service retrieves all available modules existing on it. A list of modules is displayed, and the test settings menu appears on the right. The modules list can be searched for easier selection. When the modules are selected, the API is called to retrieve all the pages on which the module is located. It initially selects the first five pages, but users can change this. The user also needs to choose the testing tool - Axe, Chrome or HTML Code Sniffer. When selecting the HTML Code Sniffer tool, the user can choose between testing according to WCAG 2.1 (performance criteria A, AA or AAA) or Section 508. Also, in this case, there is the possibility of testing errors, warnings and/or notifications.

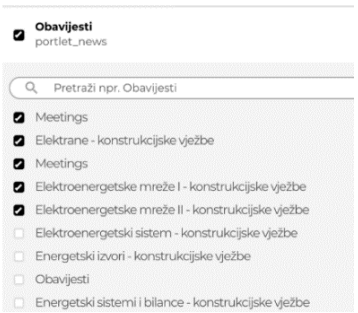


Figure 2. The list of pages where the module is located

[5] https://www.fer.unizg.hr/

[6] https://sibenik.unizg.hr/

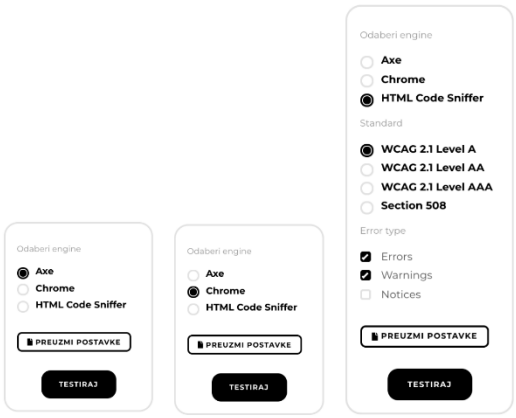


Figure 3. Testing tools (a) Axe (b) Chrome (c) HTML Code Sniffer

## B. *Accessibility Analysis*

The analysis is performed at the module level (the total number of errors on all pages selected) and there is a more detailed analysis of each selected page.

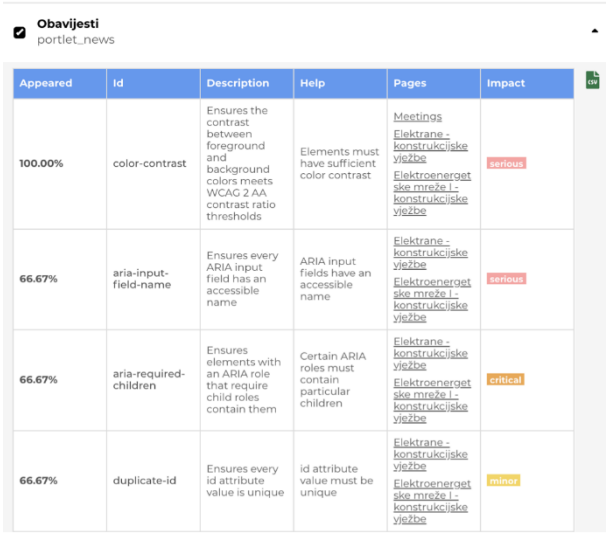


Figure 4. Example of a module analysis using the Axe tool

The Axe analysis table columns are shown in Table 1.

Chrome analysis consists of a collection of rules that check for the most common accessibility issues, which are shown in Table 2.

TABLE 1. AXE ANALYSIS TABLE COLUMNS

| | |
|---|---|
| Appeared | the number of error occurrences relative to the total number of selected pages (expressed as a percentage) |
| Id | the name of the error or accessibility item being tested |
| Description | a more detailed description of the error |
| Help | explanation of the error |
| Pages | a list of pages where the error is found |
| Impact | possible values are "serious", "critical", "minor" |

TABLE 2. CHROME ANALYSIS TABLE COLUMNS

| | |
|---|---|
| Appeared | the number of error occurrences relative to the total number of selected pages (expressed as a percentage) |
| Heading | a description of the error or accessibility item being tested |
| Pages | a list of pages where everything contains an error |
| Severity | possible values are "severe" and "warning" |

When selecting the HTML Code Sniffer test tool, the user can select the testing method, either WCAG 2.1 (all three levels) or Section 508.

WCAG 2.1. standard analysis table columns are shown in Table 3.

TABLE 3. HTML CODE SNIFFER ANALYSIS ACCORDING TO WCAG 2.1 STANDARD

| | |
|---|---|
| Appeared | the number of error occurrences relative to the total number of selected pages (expressed as a percentage) |
| Principle | WCAG 2.1. principle |
| Message | a description of the error or accessibility item being tested |
| Techniques | link to the official W3C website (https://www.w3.org/) with a technical description of how to solve the error |
| Pages | a list of pages where the error is found |
| Error level | error type, possible values are error, warning, notice |

Section 508 standard analysis table columns are shown in Table 4.

TABLE 4. HTML CODE SNIFFER ANALYSIS ACCORDING TO SECTION 508 STANDARD

| | |
|---|---|
| Appeared | the number of error occurrences relative to the total number of selected pages (expressed as a percentage) |
| Message | a description of the error or accessibility item being tested |
| Rule | link to official site Section 508 |
| Pages | a list of all pages where the error is found |
| Error level | error type, possible values are error, warning, notice |

For the user to be able to compare the results after changing the source code, results of the module analysis are available in CSV format for download. Downloading the analysis results is possible only at the module level - for each module analysis result there is one CSV file.

The user does not have to choose the same options every time for comparison with previous analyzes; the application enables the user to download test settings and reuse them. The file saves: the URL of the website being tested, the testing tool, the standard (WCAG, Section 508), whether error printing, warnings and notifications have been selected and tested modules (each module ID and the ID of each corresponding page selected for testing). When a user wants to test the same modules and pages, it is possible to load them before selecting a domain.

## VI. EVALUATION

The application consists of two main parts - selection of the modules to be tested and selection of the evaluation method.

This paper evaluates the website of the University of Zagreb, Study Program of Energy Efficiency and Renewable Energy Sources and the most frequently used modules. The modules and their number of occurrence on the pages are listed in Table 5.

### A. Most Frequent Errors

After checking all selected modules, the most common errors are:

1. *This form does not contain a submit button, which creates issues for those who cannot submit the form using the keyboard. Submit buttons are INPUT elements with type attribute "submit" or "image", or BUTTON elements with type "submit" or omitted/invalid* - The code does not contain a send button, which makes it difficult for those to fill in the form with the keyboard
2. *Link elements can only be located in the head section of the document* - <link> elements may only be inside the <head> element
3. *Duplicate id attribute value* - There are several of the same unique identifiers (attribute id)
4. *Anchor element found with a valid href attribute, but no link content has been supplied* - element <a> missing content (text)
5. *Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio threshold* - It is necessary to ensure the contrast between the content of the element and the background so that it satisfies the WCAG 2 AA contrast ratio threshold
6. *Ensures every id attribute value of active elements is unique* - it is necessary to ensure that each unique identifier (attribute id) is really unique
7. *Align attributes* - The align attribute is not used in HTML5 (used until version 4)
8. *Tables should have appropriate headers* - Table header cells must be marked with <th>, and data cells with <td> for tables to be accessible
9. *Ensures role attribute has an ap- propriate value for the element* - It is necessary to ensure that the role attribute has the appropriate value

TABLE 5. THE MOST COMMONLY USED MODULES

| Module | The number of pages on which it appears |
|---|---|
| Employee portfolios | 1135 |
| Notifications | 952 |
| Repository | 485 |
| Poll | 476 |
| Course info | 475 |

### B. Testing Results and Error Correction

Table 6 presents the results of the initial and final testing of modules. The columns state how many pages were tested and how many errors each tool found.

After checking the modules, it was necessary to correct the errors, but as can be seen from Table 6 some errors cannot be corrected successfully. The specifics of each error are listed below and the success of their corrections is explained.

1. This error occurs on the form (element <form>) located at the top of the page and it is hidden (display: "none"). It is used for internal actions and if corrected, it will cause malfunction of the website.
2. This error occurs only because the HTML code of the entire portlet is retrieved (including the <head> with the <body> element), while the tool assumes that retrieve consist only the content of the <body> element and considers the <link> elements to be outside the <head> and this error does not actually appear on the webpage
3. This error is mostly corrected, but in the Notifications module some elements are generated by the content management system so it cannot be corrected, while in the Poll module the cause of this error is an external library which also cannot be changed
4. This error requires that link element (<a>) must contain title attribute, but in the case of Notifications module that element was created using a custom function that does not receive such a parameter
5. This error could be corrected when it came to the smaller parts of the module, but sometimes it was a matter of changing the main color of the website and the change would completely disrupt the design.

TABLE 6. RESULTS OF TESTING MODULES

| errors | Axe | Chrome | WCAG A | WCAG AA | WCAG AAA | Section 508 | Total |
|---|---|---|---|---|---|---|---|
| Employee portfolios | | | | | | | |
| before | 1 | 2 | 5 | 5 | 8 | 3 | 24 |
| after | 0 | 1 | 1 | 1 | 2 | 1 | 6 |
| Notifications | | | | | | | |
| before | 3 | 1 | 4 | 5 | 6 | 2 | 21 |
| after | 3 | 0 | 4 | 4 | 5 | 2 | 18 |
| Repository | | | | | | | |
| before | 1 | 0 | 3 | 3 | 4 | 1 | 12 |
| after | 0 | 0 | 1 | 1 | 2 | 1 | 5 |
| Poll | | | | | | | |
| before | 2 | 2 | 2 | 3 | 4 | 0 | 13 |
| after | 1 | 1 | 2 | 2 | 3 | 0 | 9 |
| Course Info | | | | | | | |
| before | 1 | 1 | 3 | 3 | 4 | 2 | 16 |
| after | 1 | 0 | 1 | 1 | 2 | 0 | 5 |

6. This error is corrected by adding an identifier attribute, but this is not possible when this attribute is generated using the content management system.
7. This error was corrected every time it was found because it was only necessary to adjust the way the tables were styled according to HTML 5.
8. This error was corrected every time it was found because it was necessary to adjust the appearance of the tables and add the appropriate <*th*> and <*td*> elements.
9. This error appeared because of a role = "tablist", code that is included through an external library and cannot be changed.

## VII. Discussion

As Chrome and HTML Code Sniffer show which element contains an error, and HTML Code Sniffer clear instructions based on Section 508 guidelines, on what needs to be fixed together with error location, and links to debugging techniques, some errors were very easy to correct. However, the Axe tool only prints the name and description of the error, and it was sometimes very difficult to find out where the error occurred.

On the other hand, there are parts of code included through external libraries that cannot be modified. Also, the non-accessible module content is often provided by the user. In addition, the website design itself can be a source of errors, which is also not easy nor desirable to change. This shows that it is important to think about accessibility, especially if the code will be shared among websites or it is the created content will be available to everyone.

## VIII. Future Work

As the user enters content via Quilt CMS, currently the application cannot determine which part comes from the user and which from the web page layout code. In the future it would be useful to make an analysis of the original template used, not just the final html code of the module.

Also, the application currently offers testing using three different tools and a separate analysis for each. Joint testing and analysis would make the job easier for the web developers using the application, because there is currently a repetition of individual errors and duplication of results.

Finally, a third direction for future implementation could be mobile application testing. With the increasing progress of mobile devices, users more often access and use the websites over mobile devices. It is necessary to enable equal access to people with disabilities, and for that reason it is necessary to provide accessibility testing on mobile implementations also.

## IX. Conclusion

To help people with disabilities to participate more actively on the web, the European Commission has adopted the Web Content Accessibility Guidelines 2.0, AA compliance level, published by the Web Accessibility Initiative. In addition to this globally recognized standard, this web application also tests content according to WAI-ARIA and Section 508 guidelines.

Unlike existing tools, this application not only tests the HTML code of a given URL, but also tests the modules on all pages on which it is located, with four different types of testing. After testing and correcting errors from test results, this application not only displays errors, but also prints on exactly which elements are located and provides information on techniques on how to fix them, giving more detailed and efficient analysis. This application could easily be integrated with other content management systems based on modules/portlets, provided that the REST API for accessing the HTML code of the modules is created. This tool enabled simpler accessibility evaluation compared to existing tools; 50% of detected errors were corrected.

## Literature


[1] "W3C - Accessibility." https://www.w3.org/standards/webdesign/accessibility (accessed Jun. 01, 2020).

[2] "U.S. General Services Administration. IT accessibility laws and policies - U.S. general services administration." https://www.section508.gov (accessed Jun. 01, 2020).

[3] "European Commission - Web Accessibility policy," Accessed: Jun. 01, 2020. Available: https://ec.europa.eu/digital-single-market/en/web-accessibility.

[4] "W3C - Web Accessibility Evaluation Tools List." https://www.w3.org/WAI/ER/tools/ (accessed Jun. 01, 2020).

[5] "W3C - Selecting Web Accessibility Evaluation Tools." https://www.w3.org/WAI/test-evaluate/tools/selecting/ (accessed Jun. 01, 2020).

[6] C. Timbi-Sisalima, C. I. M. Amor, S. O. Tortosa, J. R. Hilera, and J. Aguado-Delgado, "Comparative analysis of online Web accessibility evaluation tools," 2016.

[7] S. G. Abduganiev, "Towards automated web accessibility evaluation: a comparative study," *Information Technology and Computer Science*, vol. 9, pp. 18–44, 2017, doi: 10.5815/ijitcs.2017.09.03.

[8] A. Ismail and K. S. Kuppusamy, "Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites," *Journal of King Saud University - Computer and Information Sciences*, 2019, doi: 10.1016/j.jksuci.2019.03.011.

[9] Z. W. Taylor, "Web accessibility: Not just for tech experts anymore," *Disability Compliance for Higher Education*, vol. 23, no. 9, 2018, doi: 10.1002/dhe.30416.

[10] P. Acosta-Vargas, T. Acosta, and S. Lujan-Mora, "Challenges to assess accessibility in higher education websites: A comparative study of Latin america universities," *IEEE Access*, vol. 6, 2018, doi: 10.1109/ACCESS.2018.2848978.

[11] G. Broccia, M. Manca, F. Paternò, and F. Pulina, "Flexible Automatic Support for Web Accessibility Validation," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. EICS, 2020, doi: 10.1145/3397871.

[12] J. Mucha, M. Snaprud, and A. Nietzio, "Web page clustering for more efficient website accessibility evaluations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9758, doi: 10.1007/978-3-319-41264-1_35.

[13] M. G. Olsen, M. Snaprud, and A. Nietzio, "Automatic checking of alternative texts on web pages," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6179 LNCS, no. PART 1, doi: 10.1007/978-3-642-14097-6_68.

[14] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda, "Accessibility designer: Visualizing usability for the blind," 2004.

[15] S. Trewin, B. Cragun, C. Swart, J. Brezin, and J. Richards, "Accessibility challenges and tool features: An IBM Web developer perspective," 2010, doi: 10.1145/1805986.1806029.