Proceedings of the 4th International Conference on Computing and Data Science (CONF-CDS 2022)
DOI: 10.54254/2755-2721/2/20220571

Automated Accessibility Evaluation of Mobile Applications on Initial Stages of Design and Development

Lyba Kashif, CM Nadeem Faisal, and Toqeer Mahmood

Department of Computer Science, National Textile University, Faisalabad-37610, Pakistan

lybakashif@gmail.com

Abstract. The growth of users' reliance on smartphones increases as new features and functions are continuously added to these technologies. Accessibility is an essential characteristic of information architecture in interface design research. There is a need to focus on accessibility to minimize the efforts, cost, and time during mobile application design and development. Most of the applications available on the App/Play Store lack the accessibility guidelines during development because implementing these aspects requires sufficient time and resources. Especially start-up companies normally ignore accessibility because it requires additional resources. However, perceivable design may increase profits, downloads, and reputation. Thus, the designer should ensure that their applications are perceivable and usable to satisfy the individuals. The underlying research intends to develop an automated accessibility analysis and design evaluation tool for mobile applications. Accordingly, the researchers developed a solution as a plugin for Figma. Further, the authors selected seven important techniques from the WCAG 2.1 success criterion. Finally, the proposed solution evaluates the designed screen (frame) and generates results in the report form. The proposed automated solution based on the success criterion helps the application's developers and is also useful for related stakeholders by filling the gap in the existing design related tools.

Keywords: accessibility, automatic evaluation, information architecture, interface design, universal access

1. Introduction

With the increase of mobile usage, mobile applications are also growing [1]. More and more organizations and start-ups want to develop mobile applications to solve specific problems. The success of a mobile application depends on how well the application is developed [1]. It can have a positive impact on people if it is: useful, usable, desirable, findable, accessible, and credible [2]. All these characteristics are essential for any famous and successful application. Every characteristic has its own set of rules and standards that are required to meet for it to survive in the market.

User Interfaces (UI) are components on digital screens to interact with a computer, website, or mobile applications. It is crucial to improve the user experiences and is generally categorized into multimedia, touch, and voice-based interfaces [3]. A practical and successful application comprises an easy-to-use and perceivable interface that helps users achieve their desired goals [3-5]. Thus, a balanced color contrast, responsiveness, and appropriate design strategies establish effective communication between

DOI: 10.54254/2755-2721/2/20220571

the user and the system [2]. The aspects used to create an engaging and accessible environment may include organization, layout, typography, appropriate color contract, symmetrical visual arrangement, and ease and convenience to navigate [2].

One of the critical reasons for the failure of mobile applications available on the App/Play Store is because the designer ignores the incorporation of the accessibility and usability standards due to lack of awareness, low budgets, and fewer development resources. Since COVID 19 pandemic, mobile application downloads are increased. In this situation, we believe the utilization of technologies (e.g., social, health and fitness, gaming, entertainment, food, etc.) is more likely to be raised gradually.

The mobile application designs must be personalized as per individuals' preferences**.** This comparative environment forces the designers to improve customers' experience to retain them. The individuals wish to have more engaging and accessible applications [6]. However, the design community ignores accessibility guidelines because of a lack of awareness, low budget, and fewer resources to use assistive technologies. As more people are using mobile applications than ever before, designers must change their designs if they like to survive in this growing market.

1.1. Accessibility

Accessibility refers to universal access to the contest of a website [6]. Accessibility ensures equal easy access to information and provides standard guidelines to improve the quality of interaction. The concept of accessibility is derived from software equality. For instance, if a person is in a wheelchair, he should not be stopped or excluded from entering any public place because he is disabled. You might have separate wheelchair ramps or elevators for disabled or older adults in public places. In the same way, if a person is visually impaired, he cannot be deprived of accessing a website or mobile application. The product developers must adapt to accessibility standards while planning and developing the applications for targeted markets [7]. Some countries have proper legislation to deliver accessible content in the digital environment.

According to World Health Organization [8], over 1 billion people live with disabilities. These numbers are considerably increasing due to poor health services in remote areas. Moreover, everyone experiences a disability at some point in their life. Now accessibility guidelines and standards are available to design accessible content [4]. These guidelines make a system more accessible for people having different disabilities such as visual impairment, hearing impairment, mobility impairment, and cognitive impairment [4].

Designing as per accessibility standards not only helps the disability but also improves the usability and users' experiences with the technologies [7, 9]. Therefore, WCAG guidelines are essential, and conforming to these guidelines is crucial while developing the technologies [10]. Still, the available tools and web extensions do not fully configure for accessibility. Moreover, it is also recommended to evaluate accessibility as early as possible during the planning, design, and development phases. Because determining accessibility-related issues in the early design, phase minimizes the product's development cost and time. So, it is always better to consider accessibility evaluation from the start of the project. Mainly it helps to minimize the cost of developing accessible content.

1.2. W3C

W3C stands for World Wide Web Consortium [11]. It is an international community that develops web standards together, including member organizations, the public, and staff [11]. The mission of W3C is to lead the web to its full potential [12]. W3C has two importantly emphasize the followings (1) Web for All, so everyone should be able to use the web and related contents regardless of hardware, software, culture, language, geographical locations, or physical and mental abilities, and (2) Web is everything - People should be able to access the web anywhere like smartphones, desktops, watches, personal digital assistants, or embedded systems, etc. [4].

DOI: 10.54254/2755-2721/2/20220571

1.3. WCAG

Web Content Accessibility Guidelines is a list of standard checks developed by the W3C process, which includes individuals from all around the world intending to develop a standard for web accessibility that fulfills the needs of everyone on an international level [11, 12]. Content accessibility means the information available on the web page should be accessible and perceivable. Information can be text, multimedia, sounds, hyperlinks, code, and markups [13]. The WCAG requirements are known as success criteria. There are two versions of the WCAG document available publicly on the W3C website named WCAG 1.0, 2.0 (2.1 also added recently) with backward compatibility [13], which further contains conformance levels from level A to AAA, which defines the complexity of success criteria [12, 14]. WCAG requirements are mainly for Web content developers, web authoring tool developers, accessibility evaluation tool developers, and others who need web or mobile accessibility standards [14].

1.4. Layers of guidance

There are four principles in WCAG, including perceivable, operable, understandable, and robust which are further segregated into assessment matrices for communication technologies. These techniques fall into two categories, i.e., sufficient and advisory techniques. The relationship of these layers can be visualized in Figure 1.

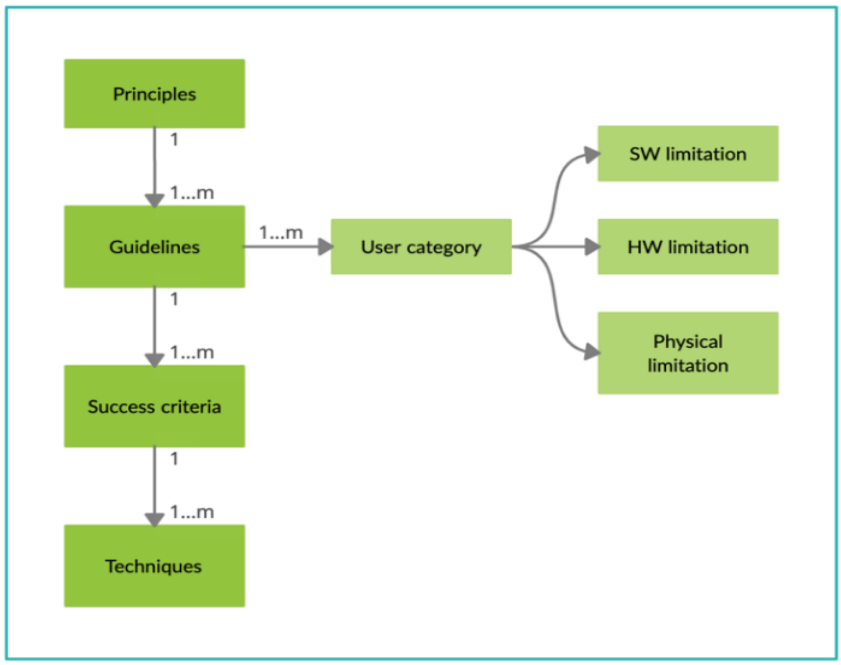

Figure 1. Layers of guidelines in WCAG [21].

1.5. Accessibility Evaluation Systems

Evaluation is a systematic way of determining a subject's worth, significance, or merit according to well-defined standards [15]. Accessibility evaluation means evaluating a system according to accessibility standards to check whether it meets the specified guidelines and generate a report accordingly [15]. Accessibility evaluation systems are the automated tools, software programs, or plugins/services used to evaluate a system, which ultimately helps the accessibility evaluation team reduce manual evaluation [15]. These tools help determine how much the system is accessible. Accessibility evaluation tools can be divided into two categories, i.e., available tools that help to evaluate most of the WCAG guidelines and special tools that can check only a specific guideline, for example, an online tool or extension that tests the color accessibility or text spacing accessibility only through the system [16].

This research is based on automating the accessibility evaluation for mobile applications, specifically during the design stage using WCAG 2.1 guidelines. Further, the WCAG 2.1 has three conformance

DOI: 10.54254/2755-2721/2/20220571

levels A, AA, and AAA. In underlying research, the authors adopt these guidelines to develop and automate an accessibility tool to determine the flaws on the design level to minimize the developers' efforts, time, and development cost [17]. Otherwise, hiring an accessibility expert during development is expensive because the developers are not fully aware of implementing these guidelines.

2. Related work

Some automated tools (i.e., AChecker, WAVE) are available to evaluate accessibility in conformance to WCAG 2.0 and 2.1 for web and mobile applications [3, 18-20]. These tools automate a few WCAG 2.0 techniques for mobile application accessibility testing [19]. Most of these tools work on source codes or evaluate applications directly after deployment. Al-Khalifa et al. [21] designed and implemented a semi-automatic accessibility evaluation system for evaluating Arabic websites based on Level A WCAG 2.0. Different results were observed using this tool compared to other evaluation tools available online. This system returned two accessibility problems, i.e., errors and warnings. In addition, this system generates two types of reports, i.e., summary report and detailed report. While working on this system, the author mentions the limitations and difficulties faced that limit the work, and hence, they called it a semi-automated accessibility evaluation system. This system does not work on websites that use dynamic URL generation. It is crucial to make the application accessible for visually impaired people with blindness, low vision, or color blindness.

Eler et al. [22] examined that manually checking accessibility properties is time challenging, due to which this conference paper proposes a tool that generates automated accessibility tests. In this paper, they presented a tool named MATE that automatically explores the application by applying tests for accessibility, mainly for visual impairments. The test generator analyzes the screens/activities and identifies the issues or flaws. A test case is generated each time a flaw is identified, and this test case always fails until the flaw is resolved. In this paper, they applied the MATE tool to more than 70 open-source android applications. They examined that the MATE tool's results are better than other available accessibility frameworks or tools. Five accessibility properties focused in this paper are explored at run time, and the test always fails until the flaw is fixed or resolved. The automated test generation technique has the potential to improve the results or add more accessibility properties to this tool.

Myriam et al. [23] developed a mobile web accessibility evaluation tool prototype using WCAG 1.0 and MWBP (Mobile Web Best Practices) to help developers create accessible interfaces. This paper transformed a flexible tool named EvalAccess into the mobile web accessibility tool EvalAccess MOBILE. It is a web service. This tool retrieves HTML code first and analyzes it to detect all the HTML elements, then requests accessibility information of the detected elements, evaluates the website, and returns errors and warnings to the accessibility reports manager. Giovanna et al. [23] argue that automating the web accessibility validation needs to evolve. This paper presented a tool named MAUVE++ to provide open and flexible support for potential issues. MAUVE++ provides a double view of the results, i.e., code-oriented and graphical views. This presented tool can provide both single page and multipage validations and report in different formats and for other types of users according to WCAG 2.1 guidelines. Vargas et al. [24] proposed to use WCAG 2.1 guidelines through automated and manual review methods. After the analysis, suggestions for the mobile application developers were provided so that mobile application accessibility could be improved in the development stage. These tools evaluated the websites as per the WCAG guidelines. But still, consistent efforts are required to improve the capability of these tools by incorporating new standards because of instant changes in the communication technologies.

3. Methodology

This research aims to analyze and possibly automate WCAG 2.1 techniques in the initial application design stage to minimize the time and cost spent resolving accessibility issues. Figma is an online design and prototyping tool, and it allows developers to make their plugins that assist UI/UX designers in designing applications and developing prototypes. A plugin for Figma was developed to automate the selected techniques to see how automated accessibility evaluation can be done in the design stage. The

DOI: 10.54254/2755-2721/2/20220571

plugin evaluates seven text, color, button, and hyperlink techniques. The developed plugin evaluates one frame at a time. Usually, a screen is referred to as a frame in Figma, but it can vary from designer to designer. This plugin effectively evaluates the design if it is based on frame nodes rather than rectangle, group, or component nodes.

3.1. System analysis and design

Studying the accessibility guidelines before designing and implementing the system was necessary. It helped select the techniques that could be automated in the design phase and discover the limitations of the techniques that could not be automated on the design level using Figma plugin API. As the first step in this plugin development, we have focused on text, color, button, and links related to WCAG 2.1 techniques [5, 19, 26].

*Analyzing Techniques for Automatic Evaluation***.** Out of all the available techniques, it was required to find which techniques can apply automated evaluation using the Figma Plugin Developer API. Therefore, we have implemented seven techniques from 2 accessibility principles to develop the plugin in this research. All other techniques were not considered because they required complex and enhanced procedures for automation. The authors adopted the techniques (i.e., G18, G145, G179, G183, G207, C36, and target size) related to Web Content Accessibility Guidelines (WCAG) success criterion as standards techniques to develop an accessibility automated tool for the identification of design issues during the development process [5, 19, 26]. These techniques are generally related to visual aspects such as the use of color, resizing text, contrast, contrast ratio, and text spacing [5, 19, 26, 27].

*Pseudocode for selected techniques***.** An evaluation algorithm was designed for each selected technique by converting the test procedures provided by w3c. The algorithm is written in the form of pseudocode, and then it is converted into code. The pseudocode for technique G183 is shown in **Figure 2**. This technique G183 specifies that "using a contrast ratio of 3:1 with surrounding text provides additional visual cues on focus for links or controls where color alone is used [26]." It includes the following test procedures (1) "locate all instances where color alone is used to convey information about the text"; (2) "check that the relative luminance of the color of the text differs from the relative luminance of surrounding text by a contrast ratio of at least 3:1"; (3) "check that pointing to the link causes a visual enhancement"; (4) "check that moving keyboard focus to the link causes a visual enhancement [25]."

Expected results - Check that test procedures 2, 3, and 4 are true (since this is a semi-automated technique, the plugin will check only #2 as the other two required complex procedures to be performed.

```
procedure CHECKG183TECHNIQUE
    nodes = figma.currentPage.selection[0].findAll(getTextNodesWithFills)
    g183IssuesList = []
    textBlockColorsArr = []
    textBlockDecorArr = []
    Read the value
    for let i = 0, len = nodes.length; i is less than len; i++ do
        s = nodes[i]
        for let j = 0, len = s.characters.length; j is less than len; j++ do
            sColor = s.getRangeFills(j, j + 1)[0].color
            r = Math.round(sColor.r * 255)
            g = Math.round(sColor.r * 255)
            b = Math.round(sColor.r * 255)
            textDecoration = s.getRangeTextDecoration(j, j + 1)
            if textDecoration !== "NONE" then
                g183AddTextBlockDecoration(textBlockDecorArr, textDecoration)
        if textBlockColorsArr.length is greater than 0 then
            foregroundColor = getRGB(textBlockColorsArr[0].rgbColor)
            backgoundColor = getRGB(textBlockColorsArr[1].rgbColor)
            contrast = calculateContrast(foregroundColor, backgoundColor)
            textScore = getContrastScores(contrast).text
            if textScore == "FAIL" then
                if textBlockDecorArr == [] then
                    g183IssuesList.push('error text')
```

Figure 2. Showing pseudocode as an example of one automated technique in the developed plugin.

Design Patterns. Designing a usable and accessible interface for plugin users is important to use the plugin with ease. After examining various tools and interfaces, the results were designed to report the user about issues per selected screen.

DOI: 10.54254/2755-2721/2/20220571

Result types. In this plugin, accessibility problems are indicated in the form of errors which the user must correct to pass all the implemented techniques for the currently selected screen.

Report generation. This plugin generates a summary report at this stage. The summary report gives high-level information about the accessibility problems on the selected screen. **Figure 3** shows several errors that occurred against each technique implemented under a specific success criterion; shows a screenshot of the report generated by the plugin as it is run on a screen

Providing guidance. The generated report on plugin UI shows the success criteria description and mentions the technique number to let the user quickly find it on the w3c.org site.

Evaluation depth. This plugin evaluates one screen at a time. Selecting one screen at a time allows the user to view the relevant errors and fix them quickly and easily.

Plugin implementation. A duplicate of the original screen is generated and evaluated for each selected technique. The identified errors are sent to the plugin interface in which the user is shown a list of errors about the selected screen along with a summary report where all errors are logged in a table view of all the screens.

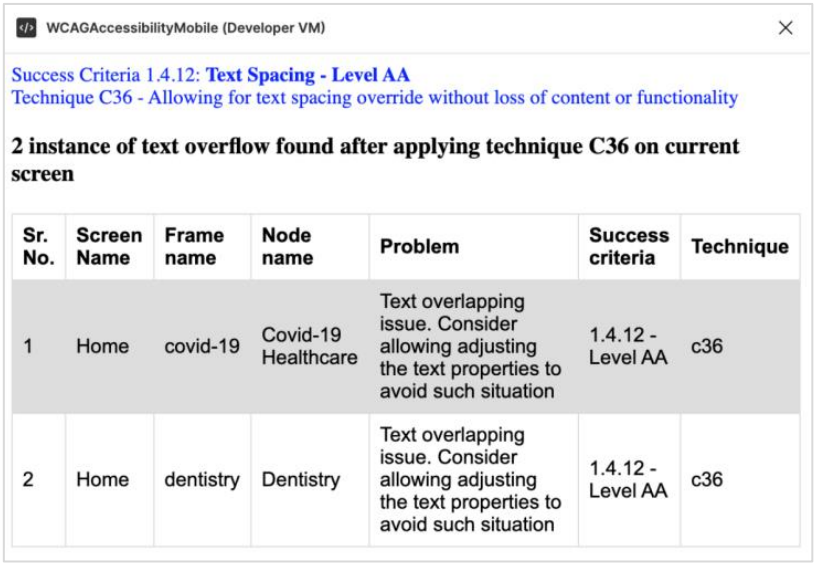

Figure 3. A summary report shows errors on a specific screen with its node and frame name so that the designer can easily identify and fix them.

3.2. Experiment and results

To test the plugin, we took a freely available design on the Figma community and applied the automated techniques on one screen at a time. To get the best results from the plugin, the user must follow some suggestions before running the plugin. The suggestions are as follows:

Naming the nodes. The node name must start with a relevant prefix or suffix, e.g., button in case of a button, or icon in case the node represents an icon in the design

Frame as a parent. For any node in the design, it is recommended to have a frame as its direct parent; it is because a frame node has more properties for analysis than other node types, which helped us to identify that a frame node as a parent would serve the best results to get most out of the plugin

3.3. Plugin test results

An example of plugin results for the technique C36 is shown in **Figure 4** shows that text overlap occurred after c36 was applied to this screen, whereas, **Figure 3** shows the generated report after the c36 evaluation.

DOI: 10.54254/2755-2721/2/20220571

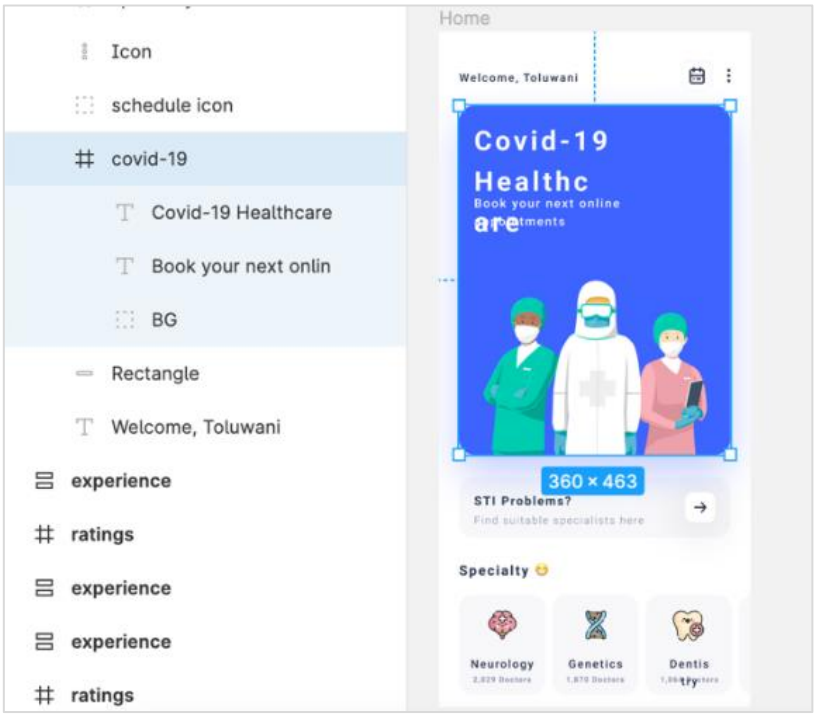

Figure 4. C36 applied on a mobile screen, on the left side, the covid-19 block can be seen as selected currently, and on the right side, the relevant duplicate frame is selected on the page. The symbol # is used to represent a frame node in Figma. Since the covid-19 node is a frame, any text that overflows or overlaps can be determined within this frame's children only.

4. Conclusion and results

In underlying research, we developed a plugin for Figma to automate a few techniques from WCAG 2.1 conformance to evaluate the accessibility of mobile application design. The objective of this research was to automate a few guidelines. Initially, the authors analyzed guidelines that could be applied to mobile application design; accordingly, we observed a lot of limitations or scenarios that needed some predefined checks to ensure accessibility. It is observed from the experimentation that the plugin works well when the direct parent of the node is a frame. Hence, it identifies issues more effectively.

Furthermore, it is also important to mention that the naming convention of the nodes plays a vital role in determining the correct type of node to perform a relevant technique evaluation. The developed plugin recognizes errors after evaluating the selected screens according to WCAG 2.1 criterion. It generates a report which contains details about the error and the exact location of the error. Those evaluations provided us with complete feedback and details having the strengths and weaknesses for future improvements, as there are still many limitations in this plugin at the initial level. There is a potential to implement success criteria techniques by defining the rules of designing standards through automation. There are possibilities for the future, like implementing the techniques on component nodes. A component node can contain a well-defined name, and it is re-usable. So, suppose UI Kits start designing in a standard way to name the components well, such as identifying a text input. For example, let's make a design component in Figma to include all the text input field requirements according to the required Material UI input field anatomy and name all the variations of input field components correctly. This plugin can be improved to a level where many WCAG 2.1 success criteria related to input fields can be automated. Another worth mentioning point for future work would be to allow the user/designers to give the plugin their naming convention. It will take the plugin to the next level of automation, where it can evaluate the designs according to the dynamic naming of the nodes.

DOI: 10.54254/2755-2721/2/20220571

References

[1] Faisal, C.M.N., Fernandez-Lanvin, D., De Andrés, J. and Gonzalez-Rodriguez, M, "Design quality in building behavioral intention through affective and cognitive involvement for e-learning on smartphones," *Internet Research*, **30(6)**, 1631-1663 (2020).

[2] Faisal, C. N., Gonzalez-Rodriguez, M., Fernandez-Lanvin, D., & de Andres-Suarez, J. Web design attributes in building user trust, satisfaction, and loyalty for a high uncertainty avoidance culture. *IEEE Transactions on Human-Machine Systems*, **47(6)**, 847-859 (2016)

[3] Abascal, J., Arrue, M., & Valencia, X. Tools for web accessibility evaluation. In Web Accessibility (pp. 479-503). *Springer, London* (2019)

[4] Nuñez, A., Moquillaza, A., & Paz, F. Web accessibility evaluation methods: a systematic review. In International Conference on Human-Computer Interaction (pp. 226-237). Springer, Cham (2019).

[5] Safi, Q. G. K., Nawaz, T., Shah, S. M. A., and Mahmood, T., "Intelligent device independent ui adaption for heterogeneous ubiquitous environments," *IJCSNS*, **vol. 11**, p. 75, 2011.

[6] Acosta-Vargas, P., Salvador-Ullauri, L., Jadán-Guerrero, J., Guevara, C., Sanchez-Gordon, S., Calle-Jimenez, T., Lara-Alvarez, P., Medina, A. and Nunes, I.L., July. Accessibility assessment in mobile applications for android. In International Conference on Applied Human Factors and Ergonomics (pp. 279-288). Springer, Cham (2019).

[7] Yeliz Yesilada, Giorgio Brajnik, Markel Vigo & Simon Harper. Exploring perceptions of web accessibility: a survey approach, Behaviour & Information Technology, 34(2), 119-134 (2015).

[8] World Health Organization, "Disability and health," 2021. [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/disability-and-health.

[9] Petrie, Helen, and Omar Kheir. "The relationship between accessibility and usability of websites." In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 397-406. 2007.

[10] Arch, Andrew. "Web accessibility for older users: successes and opportunities (keynote)." In Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibility (W4A), 1-6. (2009).

[11] Petrie, H., & Bevan, N. (2009). The Evaluation of Accessibility, Usability, and User Experience. The universal access handbook, 1, 1-16 (2009).

[12] Sohaib, O., & Kang, K. E-commerce web accessibility for people with disabilities. In Complexity in Information Systems Development (pp. 87-100). Springer, Cham (2017).

[13] Wilson, A., & Crabb, M. W3C Accessibility guidelines for mobile games. *The Computer Games Journal*, **7(2)**, 49-61 (2018).

[14] Ismailova, R. Web site accessibility, usability, and security: a survey of government websites in Kyrgyz Republic. *Universal Access in the Information Society*, **16(1)**, 257-264 (2017).

[15] Ismail, A., & Kuppusamy, K. S. Accessibility of Indian universities' homepages: An exploratory study. Journal of King Saud University-Computer and Information Sciences, **30(2)**, 268-278 (2018).

[16] Aizpurua, A., Harper, S., & Vigo, M. Exploring the relationship between web accessibility and user experience. *International Journal of Human-Computer Studies*, **91**, 13-23 (2016).

[17] Sinha, A. (2020, September). Web Accessibility Analysis on Government of India Websites based on WCAG. In 2020 IEEE International IoT, Electronics and Mechatronics Conference (IEMTRONICS) (pp. 1-7). *IEEE*.

[18] Acosta, T., Acosta-Vargas, P., Zambrano-Miranda, J., & Lujan-Mora, S. Web Accessibility evaluation of videos published on YouTube by worldwide top-ranking universities. *IEEE Access*, **8**, (2020).

[19] Yan, S., & Ramachandran, P. G. The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing (TACCESS)*, **12(1)**, 1-31 (2019).

DOI: 10.54254/2755-2721/2/20220571

[20] Kurt, S. Moving toward a universally accessible web: Web accessibility and education. Assistive Technology (2018).
[21] Al-Khalifa, H. S. WCAG 2.0 Semi-automatic Accessibility Evaluation System: Design and Implementation. *Comput. Inf. Sci.*, **5(6)**, 73-87 (2012).
[22] Eler, M. M., Rojas, J. M., Ge, Y., & Fraser, G. Automated accessibility testing of mobile apps. In 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST) (pp. 116-126). IEEE (2018).
[23] Arrue, M., Vigo, M., & Abascal, J. Automatic evaluation of mobile web accessibility. In Universal Access in Ambient Intelligence Environments (pp. 244-260). Springer, Berlin, Heidelberg (2007).
[24] Broccia, G., Manca, M., Paternò, F., & Pulina, F. Flexible automatic support for web accessibility validation. Proceedings of the ACM on Human-Computer Interaction, 4(EICS), 1-24, (2020).
[25] Acosta-Vargas, P., Salvador-Ullauri, L., Jadán-Guerrero, J., Guevara, C., Sanchez-Gordon, S., Calle-Jimenez, T., ... & Nunes, I. L. Accessibility assessment in mobile applications for android. In International Conference on Applied Human Factors and Ergonomics (279-288). Springer, Cham (2019).
[26] W3C, https://www.w3.org/WAI/WCAG21/Understanding/, last accessed 2022/01/22.
[27] Kulkarni, M. Digital accessibility: Challenges and opportunities. *IIMB Management Review*, **31(1)**, 91-98 (2019).