Web Accessibility Testing Using Robotic Process Automation and Artificial Intelligence

Vidhi Srivastava
Computer Science Department
Nordakademie
Elmshorn, Germany
vidhi.srivastava@nordakademie.de

Hans-Werner Sehring
Computer Science Department
Nordakademie
Elmshorn, Germany
https://orcid.org/0009-0008-3016-6868

***Abstract*—Digital transformation is still an ongoing process that changes how certain tasks are carried out in society. Not only is reshaping processes for the digital domain challenging, it has to be done in a way that includes everyone in social interaction, regardless of their perceptual, cognitive, or physical abilities. Designing processes and their user interfaces with digital participation in mind is crucial to avoiding a digital divide. Web interfaces still are perhaps the most important kind of user interface. Criteria have been defined to evaluate the accessibility of a service. However, many online services remain inaccessible to people with disabilities. Government websites have been required to be accessible for quite some time. Recent laws require companies to fulfill the same set of accessibility criteria. Therefore, testing user interfaces with respect to accessibility has become a mandatory part of website development. However, standard user interface testing is usually labor-intensive and cannot be fully automated. Thus, new approaches to user interface testing that focus on accessibility need to be researched. This paper proposes using Robotic Process Automation enhanced by Artificial Intelligence for accessibility testing. Although some automation tools have begun to incorporate artificial intelligence, a more fine-tuned approach to this kind of testing seems appropriate. Initial experiments were conducted on a limited set of test cases. The results suggest that automated web accessibility testing may be more effective with this approach.**

***Keywords*—Artificial intelligence, Software engineering, Software quality, Software testing**

I. Introduction

The *digital transformation* increasingly shapes all areas of life. Although the digitization of all kinds of interactive processes has obvious benefits, there is a risk that some groups of people are left behind. In the past, digital service offerings did not always fully consider the needs of people with disabilities or people of age. *Digital participation*, therefore, must be considered whenever digital services are designed. This is particularly true for interactive digital services with a User Interface (UI).

Requirements for accessible user interfaces are defined by standards. The *Web Content Accessibility Guidelines* (WCAG) are in widespread use for web page design. There are more than 80 criteria in the current version 2.2 of the guidelines that cover structural aspects of content as well as presentation characteristics.

These criteria are applied internationally to regulations, in an increasing number of countries on a legal basis. Since 2005, all public websites in Germany have been required to adhere to defined criteria. The *European Accessibility Act* (EAA) and the German *Barrierefreiheitsstärkungsgesetz* (Accessibility Support Law, BFSG) mandate that web content be accessible according to the WCAG. Starting in June 2025, commercial websites in Germany will also be required to meet the WCAG criteria. Consequently, usability testing must now be incorporated into software engineering processes to include accessibility criteria.

However, accessibility testing is resource-intensive and, in practice, largely manual. Traditional automated testing based on test scripts is inadequate in this area because it is unreliable [1]. As with any type of UI testing, testing the fulfillment of WCAG criteria is labor-intensive.

Automated UI testing turns out to be problematic for function tests already, especially for the web domain [2]. One main obstacle is the fragility of test cases. These are closely linked to the technical structure of UIs, for example by using the framework *Selenium* to identify components of a dialog. Test scripts will stop functioning when the structure of a UI changes due to a graphical or programming change.

As an alternative to test scripts, *Robotic Process Automation* (*RPA*) is employed to automatically operate UIs in tests. RPA tools are typically customizable low-code environments that allow automating the use of UIs. Several RPA tools are offered as commercial products.

Artificial Intelligence (*AI*), particularly *Generative AI* (*Gen AI*), receives a lot of attention. This is because it offers new solution strategies for existing problems that are designed in a less algorithmic way.

This paper investigates whether RPA in combination with Gen AI can improve automated accessibility testing. Using two benchmark websites, we compare three approaches: purely RPA-based testing, RPA-based testing enhanced with AI, and existing accessibility tools (Axe-core, WAVE, Lighthouse). In particular, we compare the number of accessibility problems that can be detected (precision and recall), the number of WCAG criteria that can be addressed by each approach, and their efficiency. We used a subset of the WCAG criteria in our first experiment. Results show that AI-enhanced RPA provides higher precision and coverage than both standalone RPA and established tools, although false positives and limitations in simulating subjective user experiences remain challenges.

The remainder of this paper is organized as follows. In Section II, we introduce the context of accessibility regulations and laws. In our case, we particularly refer to German law. Section III presents approaches for accessibility testing of web pages. The first experiment with AI-enhanced RPA we conducted is presented in Section IV, and the test results are discussed in Section V. The paper concludes in Section VI with a summary and an outlook for future work.

II. Rules and Legal Basis

There is no uniform definition of the term *accessibility*. Essentially, it refers to the ability of all people regardless of their capabilities, to access and use physical and digital systems. Some definitions focus on buildings, while others extend to digital channels, social interactions, and different forms of communication.

A. Definitions

The United Nations Convention on the Rights of Persons with Disabilities states that no individual measures or special rules should be applied to achieve accessibility. Rather, it should be applied naturally in all areas of life so that people with disabilities can have equal rights to participate in them. It defines it as a cross-cutting principle, meaning that it should be treated as a fundamental requirement that rules and regulations have to fulfill.

In Germany, there is a law called *Behindertengleichstellungsgesetz* (*Disability Discrimination Act*, BGG), this has been taken as a central definition for accessibility across all platforms. This law defines accessibility as the comprehensive availability and usability of a physical and digital environment.

B. Laws and Regulations

The European Union established *Directive (EU) 2019/882* in 2019. This directive defines the accessibility requirements that products and services must fulfill to be considered accessible. This rule was introduced as *European Accessibility Act*. Under this act, all digital products and services from public administrations in the EU must be accessible to people with disabilities. Many countries, including Germany, have incorporated this law into their national legislation.

Germany has had the BGG since 2002. This law obliges all public administrations to make their applications accessible to everyone. In 2021, the *Barrierefreiheitsstärkungsgesetz* (*Accessibility Enhancement Act*, BFSG) was amended to include the requirements of the European Accessibility Act. This act requires not only public administration companies, but also private sector companies, to make their digital applications accessible to all. Digital service offerings, such as e-books and online shops, must be accessible to everyone. The law took effect on 28. June 2025.

Article 8 of *EU Directive (EU) 2016/2102* requires all member states of the European Union to produce a report every three years on the accessibility of websites and mobile applications of public sector bodies. Therefore, websites are regularly tested for accessibility.

C. Accessibility Criteria

The WCAG are guidelines that define how applications should be designed to be accessible to people with disabilities. The *Web Accessibility Initiative* (WAI) of the *World Wide Web Consortium* (W3C) developed these guidelines. The WCAG criteria are based on four fundamental principles known as the *POUR* principles:

Perceivable	User interfaces must be perceivable by users.
Operable	Users must be able to operate the user interfaces and navigation.
Understandable	The available information and operation should be understandable.
Robust	The content has to be interpreted reliably by user agents, including assistive technologies.

Based on these four principles, there are currently more than 80 WCAG criteria that need to be checked in order to assess the accessibility of a web application.

III. Automated UI Testing Approaches

In this section, we highlight known testing approaches. In practice, however, none of these approaches is sufficient for accessibility testing in isolation [1]. For this reason, accessibility testing typically involves manual testing.

However, manual testing is not ideal for a variety of reasons. With incremental development techniques, such as agile approaches, retesting every change is part of the process. In continuous deployment scenarios, all tests must be automated. As previously mentioned, manual tests performed by users are subjective and, therefore, imprecise and not repeatable.

Extensive manual testing is usually impractical due to the sheer volume of test cases. This number stems from the number of channels through which one can interact (web browsers, mobile apps), multiplied by the number of dialogues (web pages, etc.) and dialog states. In the case of accessibility tests, the number of test cases increases further with the addition of the WCAG criteria and testing perspectives (e. g., types of disabilities).

In modern software development approaches that use agile methodology and continuous integration and deployment, tests must be regularly reapplied. Overall, manual testing is not feasible.

A. Test Automation

Since manual testing is not an option for the various reasons stated above, software development processes strive for test automation.

UI testing is typically based on test scripts that recognize UI elements, test data that are used as user input, and descriptions of expected responses. In the case of web application testing, the identification of UI elements is based on the web page structure represented in the *Document Object Model* (DOM). The aforementioned Selenium framework provides this identification.

Web application tests are feasible because of this direct identification of UI elements and the easy way to enter data through the DOM and HTTP requests. However, the test cases that describe interactions are fragile because they are strongly connected to the structure and implementation of a UI [2], [3]. Implementation details that are used in test scripts include the UI programming (HTML), the interaction programming (client and eventually server side), and the interaction protocols (URLs).

Using such details closely couples tests and implementation, limiting reusability. Therefore, a large number of test scripts must be developed for the many test cases (see above).

Furthermore, such tests cannot judge the representation of a web page as it is perceived by a user. This is particularly important for accessibility testing.

There are various products for UI test automation on the market, and also one for automated accessibility tests [4]. These tools differ in the accuracy with which they detect accessibility issues [5]. Therefore, automated tests alone are not sufficient to rate the WCAG conformity of a website [6] and must be combined with manual tests in practice.

B. Robot Process Automation

RPA is an approach to process automation in rule-based repetitive business processes. Software robots, called *bots*, are used to operate interactive software at the same level as a human user. Bots enable the automation of individual process steps. An *orchestrator* component controls the bots and drives a business process.

RPA tools have the ability to simulate user interactions, such as mouse clicks and keyboard input. They do so by interacting with the UI elements, not by using source code.

There are two types of software bots: *Attended bots* are activated before the process is started. *Unattended bots* start the process themselves. The type of bot used depends on the process being automated.

Common functionality of RPA solutions are the customization of bots, modeling of business processes, process execution, and monitoring and maintenance.

RPA products generally include a low-code development environment, making the implementation process for an RPA bot relatively straightforward.

RPA can be used to drive UI test execution in order to avoid the problems with test automation tools. Improvements include independence from UI programming, the ability to use the presentation of a UI rather than its code, and the ability to provide input based on the state of the process and activities. Thus, RPA bots can be used to implement accessibility tests that detect issues that static test scripts do not reveal [7].

RPA is not in widespread use for accessibility testing. However, some tools, such as the software *UIPath* that we use in our experiments, have started adding support for WCAG tests.

C. Dynamic Generation of Test Cases Through AI

In computer science, approaches that are not based on classical algorithms are subsumed under the term AI. Gen AI in particular receives a lot of attention.

Many AI approaches are include *learning* to handle input that is not known in advance. This allows applications to adapt to changing environments. Deep learning [8] enables advanced learning capabilities.

Gen AI has a tremendous impact on the way how AI is integrated into software tool setups. The proposal of the Transformer architecture proposed by Vaswani et al. [9] marks a milestone in tool development.

The rapidly increasing adaptation of (Gen) AI technology led to the development of various software libraries, frameworks, and toolkits that ease the development of AI-based applications. The availability of pre-trained models as used in the Transformer architecture further adds to the availability of AI technology for software development even without in-depth knowledge in AI.

This leads to the idea of integrating Gen AI into test tools for test case generation [10]. First AI-based approaches have already been successfully applied to accessibility testing [11]. Additionally, ML approaches with specifically trained models have been applied [12], [13], [14]. Additionally, reports on the application of computer vision to accessibility testing are available [10].

In our approach, we integrate RPA with AI technology by implementing software bots as AI applications.

IV. Experimental Setup

We designed a test process in an RPA tool and implemented software bots with the help of different AI tools. The main goal is to save the effort of applying multiple tests, in particular a combination of manual and automated tests, by increasing the accuracy of the detection of accessibility issues of websites.

In order to evaluate the approach according to these goals, we applied it on two sample websites. For comparison, we also tested the websites using other approaches.

A. Scope of the Experiment

Out of the numerous WCAG criteria, five have been selected for experimentation. We chose to constrain experiments to this small subset in order to have a feasible setup that allows getting first results.

We selected the five WCAG criteria that are most often violated by websites according to a report issued by the German *Bundesanstalt für Materialforschung und -prüfung* (Federal Institute for Materials Research and Testing):

1) missing or insufficient alternative texts (“alt texts”) of images
2) sufficiently high contrast
3) missing labels of form fields and input elements
4) consistent navigation structure and heading hierarch
5) non-accessible keyboard navigation

B. Solution Alternatives

We designed an experiment to compare five testing approaches:

RPA+AI	using an RPA tool integrated with AI-driven website analysis
RPA-only	using an RPA tool with custom code for accessibility tests
Tools	for web accessibility testing, in particular the products Axe-core, Lighthouse, and WAVE

Tests for the criteria listed in the previous section are implemented in each of the five approaches. RPA workflows execute navigation flows, while AI modules generate alternative texts, analyze contrasts, validate labels, detect language inconsistencies, and simulate keyboard navigation.

RPA+AI is the approach under investigation. We use the product UIPath as the RPA framework and a collection of available libraries for the AI part. Both the RPA framework and the libraries we use for the Gen AI setup are developed in Python so that the ability to technically integrate is technically given. The libraries used are *PyTorch*, which runs transformer models, and *Scikit-Image*, which is used for image analysis. Pre-trained models were retrieved from the Hugging Face repository.

Functionality for the tests of the five selected WCAG criteria has been developed using this technology stack.

- Alternative texts of images are checked using *BlipProcessor*.
- Contrast checks are performed using Scikit-image and further image analysis libraries.
- Labels in web forms are checked using a BERT model.
- Keyboard navigation on a web page is checked using Selenium. It does not require the use of AI.
- Checks for the correct declaration of a web pages localization (language) are performed with the help of a model based on the multilingual XLM-RoBERTa (papluca/xlm-roberta-base-language-detection).

For comparison, we tested with a purely RPA-based approach (experiment RPA-only) and three accessibility testing tools. For RPA-only we used UIPath to have the same product as in the RPA+AI case. In this tool a workflow with functions to test the five selected criteria has been implemented. Technically, they are based on Selenium to simulate user interactions.

As Tools, we use established accessibility products: *Axe-core*, *WAVE*, and *Lighthouse*. Axe-core is an open source JavaScript library. WAVE is a tool from WebAIM with a graphical user interface. Lighthouse is an open source tool from Google. These three products have been chosen for the experiments in this paper because they are the most frequently used accessibility testing tools according to a market study [15]. Furthermore, they are also used in comparative experiments found in the literature [16]. Since studies indicate that these three tools have different characteristics [17], we chose to include all three of them in the comparison.

C. Experimental Test Objects

Two benchmark sites from Deque University with known accessibility issues are tested:

- Deque Workshop Demo [18]
- Deque Booking Demo [19]

V. Results

In this section, we present the results of the first experiments. We use the following metrics for measurement:

- number of WCAG issues detected
- the kinds of issues that are detected
- coverage of WCAG criteria in detected issues
- number of false positives and false negatives

In accordance with general quality standards such as ISO/IEC 9126-1:2001 and ISO/IEC 25010:2011, we evaluated the precision, coverage, and efficiency of the approaches with respect to the two selected sample websites and the five selected WCAG criteria.

Table I summarizes the results of testing the two sample websites with the five approaches. Additionally, the first line of the table shows the results of a manual test. We consider these results the benchmark against which to measure the different approaches. As outlined above, some of these numbers may be subjective.

A. Precision

The approaches differ significantly in their ability to detect accessibility issues. Testing Tools recognize fewer than 30% of the accessibility issues on demo websites. These results align with previous literature [5], [20].

This means that Tools find the smallest number of issues in nearly all cases, the test for inconsistent navigation on the University Demo website being the only exception where at least one of the tools finds more accessibility problems. The test for missing alternative texts shows nearly equal numbers for all approaches on the Broken Workshop website. In all other cases, RPA-based approaches are more successful, partially with six to sixty times higher detection numbers.

The numbers from RPA-only are higher than those from our AI-assisted setup. However, this is due to a high number of duplicate problem reports by RPA-only. These originate partially from problems that are detected repeatedly in different dialogues and dialog states and are, therefore, counted multiple times. For instance, the number of contrast problems on the workshop demo site is more accurate in the RPA+AI case. In general, RPA-only exhibits a high number of false positives that RPA+AI does not encounter.

B. Detection Rates

In the experiment, Tools have a coverage of only two of the five kinds of WCAG criteria. The RPA-based approaches perform much better and detected problems of all five kinds. None of the approaches can detect problems of all kinds equally well, though.

Every experiment shows that out of the five chosen WCAG criteria, contrast issues are the most frequently detected kind

TABLE I
EXPERIMENTAL RESULTS

	Errors found on the workshop demo site [18]					**Errors found on the booking demo site [19]**				
Test approach	**missing alternative texts for images**	**contrast**	**missing form labels**	**keyboard navigation**	**language checker**	**missing alternative texts for images**	**contrast**	**missing form labels**	**keyboard navigation**	**language checker**
manual	8	13	website does not contain forms	9	ok	5	49	15	20	ok
RPA-only	8	46		18	ok	8	182	30	62	ok
RPA+KI	9	19		18	ok	4	63	23	59	ok
Axe-dev-core	8	0		0	0	0	0	0	0	0
Lighthouse	8	1		0	0	0	1	0	0	0
WAVE	8	22		42	0	0	33	1	1	0

TABLE II
SUMMARY OF THE RESULTS FROM THE EXPERIMENTS

Approach	**Coverage (WCAG criteria)**	**Precision**	**Efficiency**
RPA-only	High (5/5)	Medium	Low
RPA+AI	High (5/5)	High	Medium
Tools (avg.)	Low (2–3/5)	Medium	High

of problem, followed by keyboard navigation and form labels. This is consistent with the findings of [15].

In the workshop demo site, problems with keyboard navigation and form labels are not detected at all by some tools because they are caused by dynamic conditions. WAVE is able to recognize such problems, but not as well as the RPA-based approaches. RPA-based tools that operate on the dynamically rendered web pages can interact with the forms and thus find the accessibility problems.

C. Efficiency

The three accessibility Tools performed tests quickly. This aligns with the results reported in the literature. RPA-only had longer execution times for the tests. This may be partly due to our programming, but it is also a result of all tests being executed extensively without applying any heuristics.

Our experimental RPA+AI setup has a runtime behavior that lies in-between the other approaches. Execution times are considerably shorter than those of RPA-only.

D. Quantitative Comparison

Table I counts the results that we obtained from our experiment. In comparison, the RPA-based approaches receive better results than the UI testing tools.

RPA-only tends to report false positives, leading to the higher numbers in Table I. For example, RPA-only detected 182 missing alternative texts on the booking demo site, where RPA+AI found 63, with 80 rated as semantically useful in prior studies. Thus, alternative text generation was successful in most cases.

There are different sources of false positives. For example, missing alternative texts for images are reported for every image by RPA-only because it has no means of distinguishing between content images that need alternative descriptions and decorative graphics that do not need them. The RPA tool is based on Selenium to interface with web pages. Selenium detects some problems in the Document Object Model that have no visible effect and are thus falsely reported as problems.

Using AI to interpret web pages increases precision by avoiding false positives. AI allows for interpretation of web pages similar to that of a human. Through contextual "understanding" of the content, a Transformer model can differentiate between meaningful and decorative images, identify form fields with missing labels, distinguish dialog states and intermediate loading animations, and more.

Table II summarizes the results and provides ratings. Therefore, our result is that, at least for this limited set of tests, the combination of RPA and AI performed best in terms of precision and coverage, and performed well in terms of efficiency.

VI. CONCLUSION AND FUTURE WORK

The paper concludes with a summary and an outlook on future work.

A. Conclusion

Website accessibility is becoming a legal obligation in an increasing number of countries. For this reason, usability testing that emphasizes accessibility requirements is becoming increasingly important for companies. The WCAG criteria are widely adopted to evaluate accessibility.

However, existing UI test tools are insufficient for use in isolation because they cannot cover all WCAG criteria. Promising approaches based on RPA and (Gen) AI are emerging. In this paper, we present our initial experiments combining RPA and Gen AI. RPA drives the testing process and AI approaches augment software bots.

Initial experimental results on a subset of the WCAG criteria indicate that combining the technologies improves testing precision compared to using only one technology. The improvement was measured in terms of both test precision and coverage of WCAG criteria.

In the field of web accessibility testing, these results suggest that a combination of Gen AI and RPA designed specifically for this purpose has advantages over commercial products that combine these two technologies.

Therefore, to fulfill their legal obligations, companies should consider using a combination of RPA and Gen AI.

B. Outlook

Future work will have to address further WCAG criteria to achieve a more complete evaluation of the approach. Finally, all criteria must be included. Among other aspects, it will be interesting to experiment with input/output devices that usually require manual testing, such as the correct display on Braille readers or hands-free operation.

One of the strengths of the approach presented in this paper seems to lie in the ability to judge presentations of UIs, not only the code. Additional semantic interpretations, such as judging the suitability of alternative texts of images, may further increase the amount of accessibility barriers found.

Developments in the area of AI might render the RPA component of the approach unnecessary since agentic AI can operate websites itself. In that case, the idea of bots implemented with different AI-based strategies can be abandoned. Instead, sufficiently detailed descriptions of the WCAG criteria can be fed into a large language model, and agentic AI can possibly derive a test strategy.

Acknowledgment

The authors would like to thank Nordakademie for the opportunity to publish this work. We would also like to thank our colleagues at CGI and numerous customers for the many fruitful discussions on this topic.

References

[1] V. Regec, "Comparison of automatic, manual and real user experience based testing of accessibility of web sites for persons with visual impairment," *Journal of Exceptional People*, vol. 1, no. 6, pp. 117–123, 2015.

[2] R. Coppola, E. Raffero, and M. Torchiano, "Automated mobile UI test fragility: An exploratory assessment study on android," in *Proceedings of the 2nd International Workshop on User Interface Test Automation*, ACM, 2016, pp. 11–20.

[3] M. Nass, E. Alégroth, and R. Feldt, "Why many challenges with GUI test automation (will) remain," *Information and Software Technology*, vol. 138, 2021.

[4] W3C WAI, *Web accessibility evaluation tools list*, Accessed: Sep. 21, 2025, Jun. 2025. [Online]. Available: https://www.w3.org/WAI/test-evaluate/tools/list/.

[5] M. Vigo, J. Brown, and V. Conway, "Benchmarking web accessibility evaluation tools: Measuring the harm of sole reliance on automated tests," in *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A 2013)*, ACM, 2013, pp. 1–10.

[6] I. S. Baazeem and H. S. Al-Khalifa, "Advancements in web accessibility evaluation methods: How far are we?" In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2015)*, 2015, pp. 90–95.

[7] T. Bostic, J. Stanley, J. Higgins, D. Chudnov, J. Brunelle, and B. Tracy, "Automated evaluation of web site accessibility using a dynamic accessibility measurement crawler," *arXiv preprint arXiv:2110.14097*, 2021, Accessed: Sep. 21, 2025. [Online]. Available: https://arxiv.org/abs/2110.14097.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[9] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, *et al.*, "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[10] J.-M. López-Gil and J. Pereira, "Turning manual web accessibility success criteria into automatic: An LLM-based approach," *Universal Access in the Information Society*, vol. 24, no. 1, pp. 837–852, 2025.

[11] K. Chemnad and A. Othman, "Digital accessibility in the era of artificial intelligence—bibliometric analysis and systematic review," *Frontiers in Artificial Intelligence*, vol. 7, 2024.

[12] K. S. Fuglerud, T. Halbach, I. Utseth, and A. U. Waldeland, "Exploring the use of AI for enhanced accessibility testing of web solutions," Stud Health Technol Inform, pp. 453–460, 2024.

[13] S. K. Mandava, "AI-powered accessibility: Using machine learning to detect and correct accessibility gaps in web interfaces," *International Journal of Medical Informatics*, vol. 13, no. 3, pp. 9196–9214, 2024.

[14] P. K. Gollapudi, "Deep learning-enhanced accessibility compliance automation for web-based," *World Journal of Advanced Research and Reviews*, 2024.

[15] WebAIM, *Survey of web accessibility practitioners #3 results*, WebAIM Project Report, Accessed: Sep. 21, 2025, Jan. 2021. [Online]. Available: https://webaim.org/projects/practitionersurvey3/.

[16] T. Nguyen, "Evaluating automated accessibility checker tools," Western Washington University, Tech. Rep. WWU Honors College Senior Projects. 805. 2024, Accessed: Sep. 21, 2025. [Online]. Available: https://cedar.wwu.edu/wwu_honors/805.

[17] T. Frazão and C. Duarte, "Comparing accessibility evaluation plug-ins," in *Proceedings of the 17th International Web for All Conference*, ACM, 2020, pp. 1–11.

[18] Deque University, Accessed: Sep. 21, 2025. [Online]. Available: https://broken-workshop.dequelabs.com/.

[19] Deque University, Accessed: Sep. 21, 2025. [Online]. Available: https://dequeuniversity.com/demo/dream.

[20] M. Duran, *What we found when we tested tools on the world's least-accessible webpage*, Blog post, *Accessibility in government* (GOV.UK), Accessed: Sep. 21, 2025, Feb. 2017. [Online]. Available: https://accessibility.blog.gov.uk/2017/02/24/what-we-found-when-we-tested-tools-on-the-worlds-least-accessible-webpage/.