A PA11Y BASED APPROACH TO WEB ACCESSIBILITY EVALUATION

(WEB ERİŞİLEBİLİRLİĞİ DEĞERLENDİRMESİNE PA11Y TABANLI BİR YAKLAŞIM)

by

Burcu Kalkan, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Supervisor: Asst. Prof. Dr. Reis Burak ARSLAN

May 2025

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, **Asst. Prof. Dr. Reis Burak Arslan**, for his invaluable guidance, patience, and encouragement throughout the course of this thesis. His insightful feedback and constant support have been crucial to the successful completion of this study.

I am sincerely thankful to my beloved husband, **Muhammed Kalkan**, whose unwavering support, love, and understanding have carried me through the most challenging times of this academic journey. His belief in me has been a source of strength and motivation.

To my precious son, **Ahmet Ömer Kalkan**, who brought light and joy to my days during this long and demanding process thank you for your endless love, even when I couldn't always be present.

I am also grateful to my dear parents for their lifelong support and prayers. Their encouragement and sacrifices laid the foundation for all my achievements.

May 2025
Burcu KALKAN

TABLE OF CONTENTS

LIST OF SYMBOLS

CORS	**:** Cross-Origin Resource Sharing
WAVE	**:** Web Accessibility Evaluation Tool
WCAG	**:** Web Content Accessibility Guidelines
W3C	**:** World Wide Web Consortium
WAI	**:** Web Accessibility Initiative
HTTP	**:** Hypertext Transfer Protocol
URL	**:** Uniform Resource Locator

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In today's world, the impact of digital technologies on social life is increasing, and access to information is predominantly facilitated through digital platforms. This development necessitates that publicly available websites be accessible to all individuals. Ensuring equal digital rights for people with disabilities is not only an ethical responsibility but also a legal obligation. Website accessibility evaluation processes are generally carried out using automated testing tools. However, existing tools have significant limitations, particularly in analyzing dynamically generated content. They often produce false positives and fail to provide comprehensive assessments.

In this thesis, an accessible evaluation system has been developed based on the open-source tool Pa11y, offering Turkish language support and visual feedback mechanisms. The system features a Node.js-based backend and a React.js-based frontend, enabling users to visually detect and interactively correct accessibility issues such as insufficient color contrast, missing alternative texts, and unlabelled form elements. With proxy support, the system overcomes CORS restrictions and can analyze both static and dynamic content. Furthermore, it includes features such as saving analysis history, performing comparative evaluations, and exporting reports in PDF/JSON formats.

Applied tests and user feedback indicate that the system produces more accurate and consistent results compared to existing tools. In particular, it successfully analyzes dynamic content that popular tools like WAVE fail to process properly. The system’s Turkish localization and visual feedback features significantly enhance user experience. This study represents a concrete contribution to improving the accessibility of digital services in Turkey and lays the groundwork for future developments in compliance with WCAG 2.2 and beyond.

ÖZET

Günümüzde dijital teknolojilerin toplumsal yaşamdaki etkisi artmakta, bilgiye erişim dijital platformlar aracılığıyla sağlanmaktadır. Bu durum, kamuya açık web sitelerinin tüm bireyler için erişilebilir olmasını zorunlu kılmaktadır. Özellikle engelli bireylerin dijital ortamlarda eşit haklara sahip olması, etik bir sorumluluk olmanın ötesinde yasal bir gereklilik haline gelmiştir. Web sitelerinin erişilebilirliğini değerlendirme süreci genellikle otomatik test araçlarıyla yürütülmektedir. Ancak mevcut araçlar, özellikle dinamik içeriklerin analizinde sınırlılıklar barındırmakta, hatalı bildirimler üretmekte ve kapsamlı değerlendirme açısından yetersiz kalmaktadır.

Bu tez kapsamında, açık kaynaklı Pa11y aracı temel alınarak, Türkçe dil desteğine ve görsel geri bildirim mekanizmalarına sahip, özelleştirilebilir bir erişilebilirlik değerlendirme sistemi geliştirilmiştir. Node.js tabanlı arka uç ve React.js ile geliştirilen ön yüz sayesinde sistem, kullanıcıya renk kontrastı, alternatif metin eksikliği gibi hataları görsel olarak sunmakta ve düzenleme imkânı sağlamaktadır. Proxy desteğiyle CORS sınırlamaları aşılmakta, hem statik hem dinamik içerikler analiz edilebilmektedir. Ayrıca sistem, analiz geçmişi tutma, kıyaslama yapma ve PDF/JSON dışa aktarma gibi özellikler de sunmaktadır.

Yapılan testler ve kullanıcı geri bildirimleri, sistemin mevcut araçlara kıyasla daha doğru ve kullanıcı dostu olduğunu ortaya koymuştur. Bu çalışma, Türkiye'deki dijital hizmetlerin erişilebilirliğini artırmaya yönelik önemli bir katkı sunmaktadır.

1. INTRODUCTION

User Experience (UX) is a multidisciplinary field focused on improving the overall experience users have while interacting with a product, system, or service. It encompasses various dimensions such as usability, accessibility, aesthetics, emotions, and satisfaction. User experience is not limited to the functionality of a product it also involves users' emotional responses, personal meanings, and social interactions (Hassenzahl & Tractinsky, 2006). A core component of user experience is web accessibility, which aims to ensure that web content, sites, and digital tools are accessible, understandable, and usable by everyone regardless of physical, cognitive, visual, auditory, or technological limitations. This concept includes not only individuals with permanent disabilities but also older adults, users with low-spec devices, those with limited internet access, and people with temporary impairments. Web accessibility, as an extension of the right to access information, seeks to establish equality for individuals in the digital environment. In today's world, where technology is integrated into every aspect of life, digital accessibility has become a prerequisite for social participation across a wide range from education and healthcare to public services and the private sector. In this context, providing an accessible web experience should be regarded not merely as catering to a specific user group, but as a universal design approach that enhances the level of digital inclusivity for society as a whole.

Web accessibility is not only an ethical obligation but also a mandatory practice area supported by legal regulations at both national and international levels. Regulations such as the European Accessibility Act adopted by the European Union, the Americans with Disabilities Act (ADA) in the United States, and Turkey's Accessibility Monitoring and Auditing Regulation clearly define the responsibilities of digital service providers regarding accessibility. These laws cover both public institutions and the private sector,

mandating that digital content, software, and user interfaces be designed to be accessible. Furthermore, accessibility is not just a legal obligation but also a strategic advantage. Accessible web design improves user experience, enables access to a broader audience, increases user satisfaction, and builds brand loyalty in the long run. For example, accessibility solutions such as screen reader-friendly content, keyboard-navigable interfaces, subtitled media, and optimized contrast do not only benefit individuals with disabilities but also offer a more intuitive and effective digital experience for all users. Therefore, accessibility reminds us that technological advancements have not only a technical dimension but also a social one (W3C, n.d.-a).

The most comprehensive and systematic standard for web accessibility at the international level is the Web Content Accessibility Guidelines (WCAG) developed by the World Wide Web Consortium (W3C). WCAG is based on four core principles for producing accessible digital content: perceivable, operable, understandable, and robust. These principles aim to ensure that all users can perceive and interact with digital content smoothly through various devices, assistive technologies, or browsers. WCAG's different compliance levels (A, AA, AAA) offer developers a gradual accessibility roadmap (W3C, n.d.-b). However, full compliance with these guidelines often requires technical expertise, manual inspection processes, and extensive time planning. Especially in large-scale projects, as the number of pages increases, ensuring accessibility through manual inspection becomes significantly more challenging. Therefore, the use of automated analysis tools is of great importance in order to accelerate accessibility evaluations and facilitate compliance with standards. However, most existing tools fall short in areas such as localized language support, user guidance, and visual interaction—often leaving users with only technical error messages. This situation leads to developers misinterpreting errors and struggling to take corrective actions (W3C, n.d.-c).

In this study, based on some of the fundamental limitations of Wave, one of the commonly used automated accessibility testing tools, the need to develop an alternative system has been identified. Tools like Wave often fail to correctly interpret DOM structures dynamically generated with JavaScript, resulting in false positives. For example, even when images on some websites contain alternative text (alt attribute), Wave may fail to detect them and issue warnings such as "Linked image missing alternative text." Additionally, due to CORS (Cross-Origin Resource Sharing) restrictions, analysis cannot

be performed on client-side rendered content, or general error messages such as “page not accessible” may occur. These situations lead to unnecessary time loss and misdirection for developers. Based on these issues, an accessibility evaluation system has been developed within the scope of this thesis, built on the open-source Pa11y tool, with support for Turkish characters, visual feedback, and a user-friendly and interactive interface. Built with Node.js and React.js, the system not only reports accessibility errors but also allows users to fix issues like contrast errors directly through the interface. Moreover, the system can export analysis results in PDF and JSON formats, store analysis history for tracking, and bypass CORS limitations with proxy support to conduct broader testing. Thus, the developed tool functions not only as a technical analysis system but also as an interactive accessibility guide.

2. LITERATURE REVIEW

In the early days of the internet, digital content was more accessible to individuals with disabilities; however, over time, the increasing use of media elements driven by developers' aesthetic concerns has made these contents more complex (Carter & Markel, 2001). These developments have introduced new challenges regarding accessibility. In this context, it is emphasized that web design must consider accessibility principles. Adopting an approach based on universal design principles during the design process holds the potential to offer more inclusive solutions for a broader range of users.

Universal design provides a framework that enables the creation of products that can be accessed and used by individuals with diverse abilities. Nevertheless, the term *accessibility* more specifically refers to adjustments that ensure individuals with disabilities can access digital systems without barriers (Henry, Zahra, & Brewer, 2014). Considering that one in seven people globally lives with some form of disability (World Health Organization [WHO], 2020), making websites accessible to everyone has become a necessity.

Barriers to accessibility are not limited to physical impairments; cognitive limitations, perceptual difficulties, motor skill impairments, low-tech devices, or the use of mobile devices without mouse/keyboard support can also hinder access (Friedman & Bryen, 2007). However, this indicates that accessibility efforts benefit not only individuals with disabilities but all users. Indeed, a study conducted with participants who had no disabilities found that users completed 19% more tasks and did so 8% faster on websites with higher compliance to WCAG 2.0 standards (Schmutz, Sonderegger, & Sauer, 2016). This finding demonstrates that accessibility standards generally improve user experience.

Therefore, accessibility principles must be applied systematically in digital design. For these practices to become widespread, there is a need for the development of

internationally recognized standards. Web accessibility is an approach that aims to ensure digital content is equally accessible and usable by everyone. In this context, web accessibility evaluation tools are used to determine the level of accessibility of websites, test their compliance with standards, and guide developers. Accessibility evaluation is generally conducted through three main methods: automated testing, manual inspection, and user testing (Abascal et al., 2019). Each of these methods has its own advantages and disadvantages, and it is commonly recommended that they be used in combination for effective evaluation.

Automated tools provide significant time and cost advantages by performing fast and comprehensive scans, particularly on large-scale websites. These tools are especially useful in the initial phase to check whether accessibility standards (e.g., WCAG) are being applied. Abascal, Arrue, and Valencia (2019) examined the main features of web accessibility evaluation tools and noted that although tools like AChecker offer various advantages, they are not sufficient for accessibility evaluation on their own. This is because such tools can generally detect only programmatically identifiable issues. This limitation is clearly emphasized by the W3C as well. In a guideline published by Abou-Zahra et al. (2017), it is stated: "We cannot automatically check all accessibility problems. Human judgment is required. In some cases, evaluation tools may produce incorrect or misleading results. Web accessibility evaluation tools can only assist in the evaluation; they cannot determine accessibility by themselves."

Centeno et al. (2006) approached web accessibility evaluation tools from a broader perspective by comparing various tools such as Bobby, Tawdis, and WebXACT. The study emphasized that while automated test tools are useful, they do not cover all accessibility-related problems. On the other hand, manual evaluation methods require more expertise and are time-consuming. Therefore, the study highlighted the need to integrate automated testing tools with manual evaluation. It was also suggested that different types of disabilities require different evaluation methods, and new tools to be developed should be specifically designed for certain disability groups.

In studies conducted on web accessibility evaluation tools, various tools have been compared in terms of their success in identifying accessibility issues, ease of use, and

cost. In a study by Timbi-Sisalima and colleagues, several online tools such as AChecker, Accessibility Valet, Cynthia Says, Siteimprove Accessibility Checker, and WAVE (Web Accessibility Evaluation Tool) were evaluated. The study noted that each tool is useful for accessibility evaluations, yet each has its own strengths and weaknesses. These findings confirm a reality that has also been emphasized in previous studies: while accessibility evaluation tools are helpful, relying solely on them is not sufficient. For an effective and comprehensive evaluation, the integration of manual assessment methods and real user feedback is of great importance. Therefore, it is not possible for a single tool to cover all accessibility issues; each tool focuses on different issues and analyzes them using different methods (Timbi-Sisalima et al., 2016).

In a study conducted by Brophy and Craven, the importance of web accessibility was examined in detail, particularly in ensuring that individuals with disabilities can access digital content. The authors emphasized that accessibility is not merely a technical issue but also involves aspects of policy and societal awareness. Technically, it is stated that websites should be designed and developed in accordance with certain standards; meanwhile, at the institutional and societal levels, accessibility should be adopted as a core value. In addition, technical elements such as screen readers, text size adjustments, and contrast settings were noted to facilitate accessibility, while structuring, labeling, and organizing web content directly contribute to accessibility (Brophy & Craven, 2007).
Another study focusing on the challenges in the field of web accessibility addressed various technical, design, and usability issues related to providing accessible digital content. Among these issues were missing or incorrect labeling, inconsistent layout structures, insufficient color contrast, keyboard navigation problems, and incompatibility of media content. Furthermore, it was emphasized that understanding and implementing WCAG 2.0 standards requires developers to possess a certain level of technical knowledge. While applying these standards can be time-consuming, difficulties were also noted in allocating necessary resources (time, money, human power) for accessibility. All of these challenges reveal the practical issues encountered in studies based on accessibility standards (Abuaddous et al., 2016).

In the study by Rysavy and Michalak, the characteristics of accessibility tools and the barriers encountered when designing accessible web products were analyzed. The study

emphasized that conducting accessibility tests is challenging and that the most time-consuming stage of the process is testing and developing alternative solutions. Moreover, the importance of providing a checklist for issues detected by automated tools, along with explanations for these issues, was highlighted. Participants noted that experiencing a website under a specific disability scenario and identifying problems through the end-user version of the page is critically important in accessibility evaluations. It was concluded that existing tools are largely unclear, complex, and incomplete, and insufficient in meeting standards (Rysavy & Michalak, 2020).

Despite the clear advantages of web accessibility, many organizations still do not prioritize this issue, as shown by various studies. Interviews conducted with web developers shed light on the reasons behind this. Some developers have limited knowledge of what accessibility entails and mistakenly believe that an accessible website must not include graphics or multimedia. Additionally, there is a widespread perception that ensuring accessibility requires high cost and time. However, studies have shown that implementing accessibility in web applications can often be achieved at low cost, and that these costs are gradually decreasing thanks to technological advancements (Carter & Markel, 2015).

The evaluation of the validity and reliability of the Web Content Accessibility Guidelines (WCAG) is also an important topic addressed in the literature. Studies conducted in this context examine the impact of existing guidelines on user experience and their adequacy in terms of validity (the ability to accurately identify real issues) and reliability (the ability to yield similar results when the same page is evaluated at different times or by different individuals). However, it is observed that direct analyses focusing specifically on the validity and reliability of WCAG guidelines are limited. Among the methods used in accessibility evaluations are compliance reviews, user testing, subjective assessments, and automated scanning techniques. Nonetheless, due to the lack of sufficient standardization, uncertainties may arise regarding which accessibility issues are identified and how they are assessed (Brajnik, 2009).

Tools developed to evaluate web accessibility also occupy a significant place in the literature. Among the Chrome extensions widely used by developers are Lighthouse, WAVE, aXe, Accessibility Insights, Arc Toolkit, TotalValidator, ACCESS Assistant, and

Tenon Check. The features, usability, and evaluation outputs of these tools have been comparatively analyzed. Tests conducted on the ten most popular websites selected according to Alexa rankings revealed that none of these tools fully cover all WCAG success criteria. This highlights the necessity of supplementing automated evaluation tools with manual methods (Frazão & Duarte, 2020).

The effectiveness of Automated Web Accessibility Evaluation Tools (AWAETs) has also been extensively studied. In studies conducted using eight common and free accessibility evaluation tools AChecker, Cynthia Says, EIII Checker, MAUVE, SortSite, TAW, Tenon, and WAVE the issues detected by these tools were compared to those identified by experts through manual evaluations. The results showed that the tools can produce both false positives and omissions. None of these tools fully meet all WCAG 2.0 criteria. Therefore, it is stated that using automated tools in conjunction with manual methods, particularly in the early stages of development, yields more effective results. Additionally, the studies discussed the various advantages and disadvantages of existing tools and offered suggestions for their improvement. For example, while some tools have complex user interfaces, others fail to report accessibility issues in a sufficiently explanatory manner (Abduganiev, 2017).

Overall, the literature examines web accessibility tools from various perspectives, such as technical competence, ease of use, coverage, and accuracy. However, it is a common conclusion that these tools alone are not sufficient, and that accessibility evaluations should be supported with manual tests and user feedback. Furthermore, ensuring accessibility is considered not merely a technical issue but also a value that must be embraced at the level of institutional policy. Increasing developers' knowledge of accessibility, designing tools in a more user-friendly way, and making standards more accessible are of critical importance for the widespread adoption of web accessibility (Heim, 2000).

In the study conducted by Alsaeedi (2020), several recommendations were made to enhance the effectiveness of web accessibility tools. In this context, accessibility evaluation tools were categorized into two main types: manual and automated. It was emphasized that while automated tools have the advantage of quickly and easily identifying certain accessibility issues, they may fall short in detecting all accessibility

problems. On the other hand, manual evaluation methods can offer more detailed and comprehensive analyses, but these methods require expertise and time. The researcher recommends combining both manual and automated evaluation methods to achieve more effective results.

In the case study conducted using the WAVE and Siteimprove tools, it was observed that the WAVE tool, in particular, failed to detect critical accessibility problems such as the lack of component highlighting during keyboard navigation. These findings are significant in terms of revealing common barriers based on accessibility principles and the limitations of evaluation tools.

In this context, the WAVE tool was especially found to struggle with correctly analyzing dynamically generated content, leading to false positive notifications such as “Linked image missing alternative text.” Moreover, its inability to function properly in the face of access restrictions like CORS prevents comprehensive evaluations. Applied tests showed that in cases where WAVE was insufficient, the Pa11y-based system conducted accurate and error-free analyses. User feedback also indicated that the system's support for the Turkish language and its visually rich feedback mechanisms significantly enhanced user experience.

The system developed as part of this thesis not only contributes to the open-source community but also represents a concrete step toward improving the accessibility of digital services in Turkey.

3. ANALYSIS OF WAVE (WEB ACCESSIBILITY EVALUATION TOOL)

WAVE (Web Accessibility Evaluation Tool), developed by WebAIM (Web Accessibility In Mind) at Utah State University, is one of the most widely used and accessible tools for evaluating the accessibility of web content. Its primary objective is to assist developers, designers, and content creators in identifying and addressing accessibility issues in accordance with the Web Content Accessibility Guidelines (WCAG).

3.1 Technical Functionality and Evaluation Scope

WAVE analyzes the HTML structure and CSS styles of a web page to detect potential accessibility issues. The tool scans for a wide array of issues including:

- Missing alternative text (alt text),
- Improper heading structure,
- Incorrect form labeling,
- Inadequate color contrast,
- Lack of landmark elements,
- Links lacking context, among others.

WAVE categorizes its findings into five main classifications:

- Errors – definite accessibility issues (e.g., missing alt attributes).
- Alerts – potential problems that may require manual verification.
- Features – correctly implemented accessibility elements.
- Structural Elements – page structures such as headings and regions.
- ARIA usage – accessibility support through ARIA attributes.

This classification system allows users to distinguish between confirmed issues and informative elements, helping prioritize remediation efforts.

3.2 User Interface and Visual Feedback

One of WAVE's most significant advantages is its intuitive interface, which offers in-page visual feedback. It overlays icons directly on the evaluated page, enabling users to instantly locate problematic elements. This feature is particularly beneficial for novice developers or those new to accessibility practices.

WAVE is available both as a web-based tool and a browser extension (notably for Chrome), which allows for the evaluation of local or authenticated content that the online version may not access.

3.3 Limitations and Criticism

Despite its widespread use, WAVE has some limitations:

- Limited support for dynamic content: Content rendered via JavaScript may not be fully analyzed.
- False positives/negatives: For example, WAVE may incorrectly flag linked images with sufficient context as missing alt text.
- CORS restrictions: Cross-origin resources may not be fully analyzed due to browser security policies.
- Limited keyboard accessibility testing: While WAVE highlights structural issues, it cannot fully emulate real-world keyboard-only navigation scenarios.

These limitations have been widely acknowledged in the literature, with many researchers emphasizing that WAVE should be used primarily as a preliminary diagnostic tool, supplemented by manual testing and user feedback (Alsaeedi, 2020; Abduganiev, 2017).

3.4 Academic and Practical Relevance

WAVE is frequently referenced in academic studies and is often employed in evaluating public and educational institution websites. It is especially effective in identifying issues relevant to WCAG 2.1 criteria and provides valuable insights during early development and quality assurance phases.

Comparative studies have shown that while tools such as Pa11y or Axe may offer more extensive automation or CI/CD integration, WAVE stands out for its user-friendly visual interface and immediate feedback, making it a preferred choice for non-specialist developers.

In conclusion, WAVE serves as a powerful and accessible entry point for web accessibility evaluation. However, given its technical limitations, it should not be used in isolation. To ensure comprehensive accessibility compliance, WAVE outputs should be verified through manual audits, user testing, and cross-tool validation. Used appropriately, WAVE can play a crucial role within a broader, well-informed web accessibility strategy.

4. WCAG GENERAL DEFINITION AND STANDARDS

The Web Content Accessibility Guidelines (WCAG) are a set of international standards developed to ensure that digital content is accessible to everyone. These guidelines were created by the Web Accessibility Initiative (WAI) under the World Wide Web Consortium (W3C), with the primary aim of facilitating access to information for individuals with disabilities (W3C, 2018).

WCAG is structured around four fundamental principles, which are designed to ensure that digital content is accessible and usable by all users (Caldwell et al., 2008):

- **Perceivable:** Web content must be presented in ways that users can perceive through their senses. For example, providing alternative text for images and captions for audio content are practices that fall under this principle.
- **Operable:** User interfaces must be operable by all users. This includes supporting keyboard navigation, accommodating time limits, and ensuring that links are accessible.
- **Understandable:** The content and interface must be understandable and predictable. This requires clear and consistent language, as well as proper labeling of form fields.
- **Robust:** Content must be robust enough to be interpreted reliably by a wide variety of user agents, including current and future assistive technologies.

WCAG defines three levels of accessibility conformance: Level A, Level AA, and Level AAA. These levels help classify the accessibility of web content (W3C, 2018):

- **Level A (Minimum Conformance):** This level includes the most basic web accessibility features. Meeting this level indicates that a website removes

- significant barriers for users with disabilities and is generally considered the starting point for accessibility implementation.
- **Level AA (Mid-Range Conformance):** This level includes more comprehensive criteria and ensures accessibility for a broader range of users. It is the most commonly targeted level by public institutions and enterprises, and legal regulations in many countries mandate conformance at this level (Henry, 2015).
- **Level AAA (Highest Conformance):** This level represents the highest degree of accessibility, aiming to meet the needs of all users. However, achieving this level of compliance is not always feasible for all web content and is typically reserved for specific cases.

WCAG criteria encompass not only technical requirements but also ethical and social responsibility aspects. These guidelines serve as a critical reference for digital content creators and software developers, promoting both individual access and broader digital inclusion (Kelly et al., 2005).

Currently, WCAG 2.1 is the most widely adopted version, which includes additional criteria addressing mobile accessibility, low vision, and cognitive disabilities. Moreover, as of 2023, a draft version of WCAG 2.2 has been published, with the goal of further enhancing user-friendly digital experiences (W3C, 2023).

5. OVERVIEW AND FUNCTIONALITY OF PA11Y

Pa11y is an open-source accessibility testing tool designed to evaluate websites and web applications for compliance with accessibility standards such as the Web Content Accessibility Guidelines (WCAG) 2.0 and 2.1. It serves as a crucial resource for developers, designers, and testers aiming to create inclusive digital experiences, particularly for individuals with disabilities.

5.1 Technical Capabilities and Operation

Pa11y performs automated accessibility audits by analyzing web pages that utilize HTML, CSS, and JavaScript. It identifies accessibility violations such as missing alternative text, improper heading structures, low color contrast, and insufficient ARIA labeling. It simulates how users—particularly those using assistive technologies—interact with content, and it flags components that might cause barriers.

The tool operates by running predefined test cases aligned with WCAG standards (e.g., WCAG 2.1 Level AA) against the page's markup and structure. The results are then compiled into structured reports that:

- List the identified accessibility issues,
- Specify the location and nature of each issue, and
- Provide guidance and references on how to resolve the problems.

5.2 Integration and Automation

One of Pa11y's strengths lies in its flexibility and integration capabilities. It can be used:

- As a command-line interface (CLI) tool,

- As part of continuous integration (CI) pipelines,
- Embedded into broader development workflows and testing frameworks. This adaptability enables developers to include accessibility checks within the development lifecycle, promoting a shift-left approach to accessibility that emphasizes early detection and resolution of issues.

5.3 Practical Significance and Advantages

Pa11y is particularly valuable for teams seeking to automate accessibility testing and generate repeatable, consistent results. Unlike tools that focus primarily on visual feedback (e.g., WAVE), Pa11y excels in headless environments and can test dynamic content rendered with JavaScript, making it ideal for modern web applications built with frameworks such as React, Angular, or Vue.js.
Furthermore, Pa11y can be configured to run scheduled audits across multiple URLs, enabling regular monitoring of accessibility status over time. This helps organizations maintain compliance and track improvements.

5.4 Contribution to Inclusive Web Development

By embedding Pa11y into their workflows, development teams can proactively identify and remediate accessibility barriers before they reach end users. This not only aligns with legal and ethical responsibilities but also enhances usability and overall user experience for all.

In sum, Pa11y is a robust, developer-friendly solution that complements manual testing efforts and supports the integration of accessibility practices into agile development cycles.

5.5 PA11y Dashboard Review and Comparative Evaluation with the Developed System

Web accessibility evaluation tools stand out with their technical capacities and usage scenarios tailored to different user profiles. In this context, Pa11y is a powerful tool that offers flexible and customizable solutions thanks to its open-source and developer-

oriented structure. Operating via a command-line interface, Pa11y analyzes page accessibility in accordance with WCAG standards and yields more reliable results, particularly for modern web pages with dynamic content.

The Pa11y Dashboard, which builds upon Pa11y's capabilities by providing a more holistic and sustainable structure, allows for automatic analysis of multiple web pages at regular intervals, stores historical data, and visualizes results in a graphical format. This structure offers significant value for enterprise-level accessibility monitoring and quality assurance processes.

However, in this study, rather than directly utilizing the Pa11y Dashboard, the developed system aims to offer a different user experience by customizing Pa11y's core scanning engine. The primary reason for this choice lies in the Pa11y Dashboard's limited visual interface and lack of guidance functions aimed at end users. While the Dashboard provides technical analysis data, it does not facilitate the interpretation of results or allow users to directly act on the identified issues.

In contrast, the system developed within the scope of this thesis goes beyond the automatic analysis and historical tracking features of Pa11y Dashboard by integrating functionalities such as visual feedback, a user-friendly interface, real-time issue correction, Turkish character support, and interactive error fixing. For example, users can directly correct contrast errors through the interface and export the analysis results in PDF or JSON formats. Additionally, the system enables broader testing through proxy support, which helps bypass CORS restrictions.

The rationale behind designing such a system stems from a careful analysis of the limitations found in existing tools. One of the most widely used tools in the field, WAVE, while well-known for its accessibility focus, also shows fundamental shortcomings. WAVE often struggles to accurately analyze DOM structures dynamically generated by JavaScript, leading to misleading false positives for developers. For instance, it may issue warnings such as "Linked image missing alternative text" even when the image's alt attribute is correctly defined. Additionally, due to CORS restrictions, WAVE may fail to analyze certain pages entirely or produce vague errors such as "page not accessible."

These issues result in wasted development time and misunderstandings of accessibility problems.

Therefore, the developed system leverages Pa11y's technical accuracy while addressing WAVE’s shortcomings in user experience and guidance capabilities. One of the innovative aspects of this study is the transformation of Pa11y Dashboard’s data-centric structure into an interactive and assistive interface within the developed system.
In conclusion, the system is not merely a tool for instant analysis, but also a solution that offers interactive suggestions, guides developers, and is open to future extensions such as dashboard-like modules. In this regard, the system presents a comprehensive, sustainable, and accessible solution by reinterpreting the infrastructure strength of Pa11y Dashboard and the user-centric design philosophy of WAVE.

6. WEB ACCESSIBILITY TOOL DEVELOPMENT

This study consists of two main components: the backend component, which performs accessibility analyses, and the frontend component, which provides the user interface. These components are discussed in detail under the following subheadings.

6.1 Backend Development Process

This section provides a detailed explanation of the backend development process of the web accessibility evaluation tool. The backend is responsible for receiving URL requests from users, performing accessibility checks on the corresponding web pages, processing the results, and returning them to users in a structured format. Additionally, MongoDB has been integrated to offer persistent data storage.

6.1.1 Technologies Used

In the backend development of the project, several technologies and tools were utilized:

- **Node.js** was used as the server-side JavaScript runtime environment, enabling the execution of JavaScript code outside the browser.
- **Express.js** served as the framework for configuring the HTTP server and defining API endpoints.
- **Pa11y**, an open-source tool, was employed to perform accessibility audits based on WCAG 2.1 standards.
- **Axios** was used as an HTTP client to retrieve HTML content from external URLs for analysis.
- **MongoDB** (optional) was considered as a NoSQL database solution to enable the permanent storage of accessibility audit results.

- **CORS** (Cross-Origin Resource Sharing) was configured to manage requests originating from different domains, ensuring secure and controlled communication.

6.1.2 System Architecture

The backend application is structured as an Express.js server. It has two main functionalities:

- **/proxy endpoint**: Fetches the HTML content of a given URL, enriches it with necessary scripts and style files, rewrites relative paths dynamically, and serves it to the client.
- **/pa11y endpoint**: Performs an accessibility test on a given URL using Pa11y and returns the results categorized as notice, warning, or error in JSON format.

Within the /proxy endpoint, all source paths such as src, href, and url() in the retrieved HTML content are dynamically updated to ensure they work correctly on the client side. This is done via the processResponse function. Additionally, the necessary scripts for accessibility analysis are automatically injected into the <head> tag of the HTML content.

Through the /pa11y endpoint, the URL provided by the user is tested using the Pa11y library. The accessibility evaluation is conducted according to the WCAG 2.1 AA standard. The results are categorized as follows:

- **Error**: Critical violations of WCAG standards.
- **Warning**: Potential accessibility issues.
- **Notice**: Informational messages provided to the user.

The results are organized using the processPa11yResults function and returned to the client in a structured format.

WCAG rule codes (e.g., wcag2aa.Principle1.Guideline1_1.1_1_1.H37) are converted into meaningful descriptions for the user. This is handled by the explainWCAGCode function. The codes are interpreted based on the four fundamental principles of WCAG (Perceivable, Operable, Understandable, Robust) and their associated guidelines.

The system includes infrastructure to store audit results in MongoDB. Using the Pa11yResult model, the tested URL and its corresponding results can be saved in the database for future analysis.

6.1.3 Automatic Accessibility Testing with Pa11y

In this system, an open-source tool called Pa11y is utilized to assess the accessibility of web pages. Pa11y automates the process of evaluating web pages against the Web Content Accessibility Guidelines (WCAG) 2.1 standards, ensuring that the pages comply with accessibility best practices. Pa11y checks the page's HTML and other resources for accessibility issues, such as missing alt attributes, improper heading structures, or poor color contrast.

The URL information provided by the user is passed to Pa11y via the /pa11y endpoint. After performing the audit, Pa11y categorizes the issues found into three main categories: Error, Warning, and Notice. The results are then returned in a structured format to the client.

6.1.4 How Pa11y Works

Pa11y uses a set of predefined accessibility rules based on WCAG 2.1 guidelines. It checks for different violations, including but not limited to:

- Missing alternative text for images
- Incorrect heading hierarchy
- Forms that are not labeled correctly
- Color contrast issues

The test process works as follows:

- User provides a URL to the backend service via the /pa11y endpoint.
- The backend uses Pa11y to run a series of checks on the webpage, capturing all relevant accessibility issues.
- The results are categorized into Errors, Warnings, and Notices, and returned to the client in JSON format for further analysis.

6.1.5 Example Output from Pa11y

After running Pa11y on a page, the output is typically in the form of a JSON object, containing details about each issue found. Here is an example of what the output may look like:


```
{
  "url": "https://example.com",
  "issues": [
    {
      "type": "error",
      "message": "Missing alt attribute on <img> element",
      "context": "<img src='logo.png'>",
      "selector": "img",
      "code": "WCAG2AA.Principle1.Guideline1_1.1_1_1.H37"
    },
    {
      "type": "warning",
      "message": "Low color contrast on text and background",
      "context": "<p style='color:#aaaaaa;'>Some text here</p>",
      "selector": "p",
      "code": "WCAG2AA.Principle1.Guideline1_4.1_4_3.G144"
    },
    {
      "type": "notice",
      "message": "Consider providing a skip to content link",
      "context": "<nav><a href='#maincontent'>Skip to main content</a></nav>",
      "selector": "a",
      "code": "WCAG2AA.Principle2.Guideline2_4.2_4_1.H44"
    }
  ]
}
```

6.1.6 Interpreting the Output

Each issue is categorized by **Error**, **Warning**, or **Notice**, based on the severity of the accessibility problem:

- **Error**: These are critical violations of WCAG guidelines. If not addressed, they could severely impact the accessibility of the web page.
- **Warning**: These issues are potential accessibility problems that may not directly violate WCAG guidelines but could lead to a poor user experience if left unresolved.
- **Notice**: These are minor suggestions that could improve the accessibility or usability of the site, but they are not violations.

In the output, each issue is identified by:

- **Message**: A description of the issue.
- **Context**: A snippet of the HTML code where the issue was found.
- **Selector**: The CSS selector used to identify the element.
- **Code**: A WCAG rule code that explains the specific guideline being violated.

6.1.7 Explaining WCAG Codes

Each issue in the output is accompanied by a WCAG code, such as WCAG2AA.Principle1.Guideline1_1.1_1_1.H37. These codes can be difficult to interpret without understanding the WCAG 2.1 guidelines. Therefore, a custom function called explain WCAGCode is developed to make these codes more understandable. This function decodes the WCAG rule and provides an easy-to-understand explanation for each violation:

For example, the code WCAG2AA.Principle1.Guideline1_1.1_1_1.H37 refers to:

- **Principle 1 (Perceivable)**: The content must be presented in ways that users can perceive.
- **Guideline 1.1 (Text Alternatives)**: Provide text alternatives for any non-text content.
- **Rule 1_1_1**: All non-text content must have a text alternative that serves the same purpose.
- **H37**: The image must have an alt attribute for accessibility.

This explanation is presented to the user in a more understandable format to ensure that the issue can be addressed properly.

6.1.8 Proxy Service and HTML Manipulation

To ensure the correct and consistent functioning of accessibility tests, a special proxy service is employed instead of directly calling the URL requests from the client. This service operates via the /proxy endpoint, fetching the relevant HTML content from the web address specified by the user.

The following operations are performed on this content:

- The source paths (such as src, href, url()) in the HTML page are scanned and dynamically rewritten. This process ensures that relative paths work seamlessly on the client side.
- The processResponse function carries out this transformation.
- Additionally, the necessary script files required for the accessibility analysis by Pa11y are automatically injected into the page's <head> tag.

This structure allows the system to deliver content securely and in a controlled manner, ensuring the necessary conditions for accessibility analysis are met instead of directly serving external URLs.

6.2 Frontend Development Process

In this study, modern web technologies were utilized in the front-end development process. At the core of the application lies the React.js library, which was developed by Facebook and is widely adopted by a large developer community today. Thanks to its component-based architecture, React enables the development of user interfaces in a more modular, reusable, and maintainable manner. Each visual or functional unit (e.g., input form, analysis result panel, iframe preview) has been handled as a separate component. This approach has significantly improved the application's readability and ease of maintenance.

For data exchange and communication with the server, the Axios library was chosen. Axios simplifies the use of HTTP methods and has been used in this project to retrieve Pa11y analysis results from the server.

A component-based architecture has been adopted in the project structure, where each functionality is organized into separate component files. This architecture not only reduces code repetition but also allows team members in collaborative projects to work independently on specific components.

Additionally, to ensure the security of URLs entered by users and to prevent external websites from being loaded directly into the interface, a proxy mechanism was implemented. Thanks to this mechanism, external sites are loaded into the iframe not directly, but through redirection via the backend server. This provides an effective solution both in terms of security and for overcoming CORS (Cross-Origin Resource Sharing) restrictions.

6.2.1 Workflow of the Developed Web-Based Application

The developed web-based application performs an accessibility evaluation of a website address (URL) provided by the user. The URL entered by the user is utilized both for a visual preview (via an iframe) on the user interface and for an accessibility test performed in the backend. This process follows a series of well-defined steps:

6.2.2 URL Entry and Validation

Upon launching the application, the user is prompted to enter the URL of the website they wish to analyze. This step initiates user interaction and continues as follows:

- Once the user submits a URL, the input is processed by a function named handleUrlSubmit.
- The function checks whether the entered URL begins with either the http:// or https:// protocol. This validation ensures proper routing and establishes a secure connection with the target website.

- If the entered URL does not conform to a valid format, an error message is displayed on the interface, informing the user of the invalid input. This prevents the initiation of the analysis process for incorrect or malformed URLs.

6.2.2.1 Accessibility Testing via Pa11y

If a valid URL is provided, the application automatically forwards this address to the backend service for accessibility evaluation:

- The frontend sends a GET request to the /pa11y endpoint, passing the user-provided URL as a parameter.
- On the backend, the integrated Pa11y tool visits the specified web page and performs an accessibility analysis based on the Web Content Accessibility Guidelines (WCAG).
- The results of this analysis are returned to the frontend in JSON format.
- The returned data is categorized into three severity levels: errors, warnings, and notices, allowing for a comprehensive assessment of the accessibility level of the analyzed webpage.

6.2.2.2 Visualization of the Results

The results received from the backend are displayed to the user in a clear and visually organized manner:

- Custom React components, such as WCAGCriteria, AuditScore, and IssueDetails, are used to present and detail the results of the Pa11y analysis.
- Through these components, users can easily identify which WCAG criteria are satisfied and which are violated.
- A function called calcScore() processes all errors, warnings, and notices to calculate an overall accessibility score. This score provides a numerical indication of the website's accessibility level, giving users a general assessment. The calculation uses a weighted formula:

```
const calcScore = () => {

  return (
```

```
        pa11yResult.errorCount * 3 +
        pa11yResult.warningCount * 1 +
        pa11yResult.noticeCount * 0.5
      ).toFixed(2);
    };
```

In this formula:

- **Each error** contributes **3 points**,
- **Each warning** contributes **1 point**,
- **Each notice** contributes **0.5 points**.

The higher the score, the **less accessible** the page is considered, with a lower score indicating better accessibility. This weighted scoring helps prioritize critical issues over minor ones when assessing overall accessibility health.

6.2.2.3 Website Preview

In addition to the accessibility analysis, a visual preview of the analyzed website is provided:

- The entered URL is not directly embedded into an iframe. Instead, it is routed through a backend endpoint named /proxy, which acts as an intermediary between the external site and the frontend.
- This proxy mechanism helps bypass CORS (Cross-Origin Resource Sharing) restrictions and prevents potential security risks associated with direct external content loading.
- As a result, the user is able to view a processed version of the analyzed webpage within the application interface.

6.2.3 Component Communication and State Management

The developed application is structured using a modular, component-based architecture, in which each functionality or visual element is encapsulated within its own React component. This approach not only enhances code reusability and readability but also simplifies debugging and testing processes. The communication and coordination between these components form the foundation of the application's interactive behavior.

6.2.3.1 Modular Component Structure

The application consists of several key components, each with specific responsibilities:

- **AuditHeader**
  This component is responsible for displaying the audit result summary, including the page title and the overall accessibility score. It provides a concise overview of the analysis outcome and helps orient the user within the context of the evaluated web page.

- **Splitter**
  This component manages the layout of the interface, particularly allowing the user to dynamically adjust the width of the left and right panels. Through interactive resizing, the user can prioritize either the accessibility results or the website preview area, enhancing usability and flexibility.

- **WCAGCriteria, AuditScore and IssueDetails**
  These components collectively handle the visualization of the Pa11y accessibility test results.

  - WCAGCriteria displays the relevant WCAG success criteria and highlights which ones have been met or violated.
  - AuditScore computes and shows a numeric accessibility score based on the number and severity of issues.

- IssueDetails provides a detailed breakdown of each detected accessibility issue, categorized by type (error, warning, notice), along with contextual information such as HTML element references and descriptions.

6.2.3.2 Central Coordination via the App Component

All the individual components are orchestrated within a central parent component named <App />. This root component serves as the main controller that manages both data flow and UI state across the entire application.

React's useState hooks are extensively utilized within the App component to handle:

- **UI states**, such as the visibility of panels, loading spinners, or error messages.
- **Data states**, including the user-entered URL, the fetched accessibility results, the calculated audit score, and the iframe source.

By maintaining state at the top-level component (App), data can be passed down to child components as props, ensuring a unidirectional data flow — a core principle of React. When user interactions or external data updates occur, the relevant state is updated in the App component, which then triggers a re-render of the affected child components with the latest data.

This communication model ensures synchronization across the user interface and allows for a responsive and dynamic experience.

6.2.4 Performance and Usability

The developed application includes various performance and usability enhancements aimed at improving the user experience and ensuring efficient use of system resources. These improvements are designed to cover both the technical infrastructure and the user interface.

6.2.4.1 Lazy Loading Implementation

The application uses an <iframe> component to present a preview of the webpage being analyzed. This component is configured according to the lazy loading principle, meaning that the content is only loaded when it becomes necessary (e.g., after the user submits a URL). This approach provides several benefits:

- The initial load time of the application is reduced,
- Unnecessary network traffic and memory usage are prevented,
- Potential performance bottlenecks on the browser are minimized.

This method contributes significantly to performance, especially on networks with limited bandwidth or on devices with lower hardware capabilities.

6.2.4.2 Loading Process and Feedback Mechanism

When a user submits a URL and initiates the analysis, the application displays informative messages regarding both the accessibility test and the webpage preview process. These messages include:

- “Loading website preview...”: Displayed until the content is fully loaded into the iframe via the server, indicating to the user that the website is being fetched.
- “Analyzing accessibility...”: Displayed while the Pa11y tool performs the accessibility evaluation in the background, informing the user that the analysis is in progress.

Such messages allow the user to clearly see that the system is responsive and actively processing the request. Without these visual indicators, the user might assume the system is frozen or malfunctioning. Feedback mechanisms like these are critically important for ensuring a positive user experience (UX).

6.2.4.3 Contribution to User Experience

The technical implementations described above enable a smooth and intuitive interaction process for users. In particular, since accessibility analysis may take some time, providing real-time feedback enhances user satisfaction. Additionally, clearly indicating that the system is working in the background helps prevent users from repeatedly initiating the process out of impatience.

In this context, the application has been developed and optimized in accordance with both technical efficiency (performance) and user-centered design (usability) principles.

7. APPLICATION

This section presents the developed web accessibility evaluation tool from the user interface perspective. The application offers a simple and intuitive experience for users to perform accessibility checks on web pages. Each subsection explains different components and functionalities of the application, starting from the entry page to the display of evaluation results.

7.1 Application Entry Page

When the application is launched, users are greeted with a clean and user-friendly home screen (Figure 7.1). This screen allows the user to enter the URL of the website they want to analyze for accessibility compliance. After entering a valid website address into the input field in the center, the user can start the analysis process by clicking the **"Analyze Website"** button located on the right.

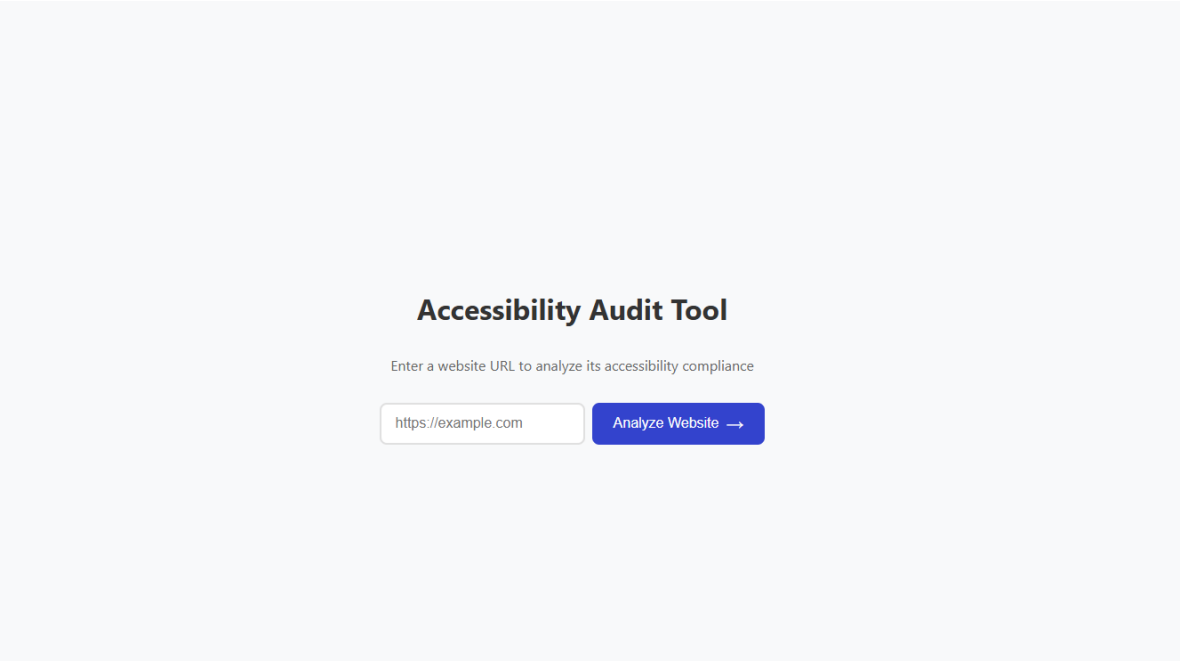

Figure 7.1 Entry Page

This page aims to provide a simple entry point without overwhelming the user with complex options. The explanatory text on the screen helps users quickly understand the main purpose of the tool: the phrase "Enter a website URL to analyze its accessibility compliance" clearly conveys the intended function of the application.

7.2 Left Panel – Accessibility Analysis Summary and Scoring System

The left panel is designed as a structural component that presents both a summary of accessibility analysis results and a detailed breakdown of errors based on specific criteria. The primary aim of this panel is to provide users with a quick overview of the website's general accessibility status while also offering access to detailed insights based on defined principles and guidelines.

At the top of the panel, the accessibility score—calculated using a weighting formula based on the severity of error types—is displayed. Just below the score, a colored indicator shows whether the score falls into the "Good", "Caution", or "Critical" category. This allows users to understand the overall accessibility condition without needing to dive into technical details.

The lower section lists the identified issues categorized according to WCAG 2.1 principles. Each main heading (e.g., Robust, Perceivable, Operable, Understandable) expands to show the related issues and the WCAG guideline numbers associated with them. Each item is color-coded based on its type (Error, Warning, Notice).
Interactive features of the panel include:

- When a user clicks on any issue listed in the left panel, the corresponding element is automatically brought into view in the right panel and highlighted with a blinking indicator.

- Likewise, when a user clicks on the marker in the right panel, the corresponding issue in the left panel is selected and the panel scrolls to that specific entry.

- Additionally, filter options at the top-left corner allow users to display only specific types of issues. For instance, selecting only "Errors" will hide "Warnings" and "Notices" from the panel, enabling the user to focus on the most critical problems first.

This interactive structure enhances the analysis process by enabling users to quickly locate the source of the issue, making the process more intuitive and efficient.

7.2.1 Distribution According to WCAG 2.1 Criteria

In this panel, issues are categorized according to the Web Content Accessibility Guidelines (WCAG) 2.1 standard and summarized as follows:

- **Errors**: 35 items
- **Warnings**: 166 items
- **Notices**: 863 items

Additionally, a "*What is WCAG?*" link is provided, allowing users to access further information about WCAG. This element is intended to raise awareness and encourage users to better understand the guiding principles.

7.2.2 Audit Score

Scoring systems aim to provide a quick and comparable overview of a website's accessibility level. This enables tracking of progress over time or across different pages. Harper and Yesilada (2008) emphasize that such numerical indicators are effective in raising awareness and supporting decision-making processes for stakeholders during development. However, they also warn that these scores may obscure the complexity of accessibility issues.

Vigo, Brown, and Conway (2013) point out that many accessibility evaluation tools assess detected issues at the same level of severity without distinguishing their importance. This may mislead developers regarding which problems require urgent attention. Research advocates for scoring systems to be transparent, explainable, and weighted according to their impact on users.

The WAVE tool does not provide a direct numerical score; instead, it enhances user awareness through categorized issue lists and visual annotations.

However, one of the main problems in existing systems is that all detected issues are treated with equal importance, which prevents developers from having a meaningful way to prioritize them. Moreover, even when some tools generate a score, they do not explain how this score is calculated, making the scoring process non-transparent. As a result, developers may struggle to understand what needs to be fixed and may perceive the score merely as an outcome. When issues are not weighted according to their impact, minor errors can disproportionately affect the accessibility score, while the impact of critical errors may be overlooked.

In this context, a custom scoring system has been developed in this thesis, based on the data provided by the Pa11y tool. This system calculates the accessibility level not only by the number of issues but also through a formula that incorporates weights determined by the impact of different types of issues on accessibility, offering more meaningful insights and prioritization for necessary interventions:

- **Score = (Error × 3) + (Warning × 1) + (Notice × 0.5)**

This formula takes into account the qualitative effect of different issue types on user experience, aiming to evaluate accessibility not only quantitatively but also qualitatively. Based on the resulting score, accessibility is categorized into three levels:

- 0–200: Good (Accessible)
- 201–400: Caution (Needs Attention)
- 401 and above: Critical (Severely Inaccessible)

These score thresholds are not directly derived from an academic study but are structured based on the approaches adopted by industry tools such as AudioEye (2021) and AccessibilityChecker.org (2022). These tools prioritize critical issues with higher weights in accessibility scoring, guiding users not only by the number of issues but also by their impact on the user experience.

The rationale behind these thresholds is as follows:

- **0–200:Good**

  This range corresponds to a maximum of approximately 66 critical issues (66 × 3 = 198) or combinations of less severe ones. It indicates that the system is generally accessible and only minor fixes are required.

- **201–400:Caution**

  This level suggests noticeable accessibility issues, though the overall user experience is not yet severely impaired. It helps users focus on areas needing prioritized intervention. This aligns with AccessibilityChecker.org's "where should I start?" approach (AccessibilityChecker.org, 2022).

- **401+:Critical**

  This range implies that accessibility issues are at a critical level, potentially preventing users from accessing essential information. It parallels AudioEye's principle that "critical issues should significantly lower the score" (AudioEye, 2021).

In recent years, researchers have proposed more advanced scoring models. For example, Alnanih, Pedell, and Burdett (2016) developed a weighted scoring model that considers both the WCAG compliance levels (A, AA, AAA) and the contextual importance of the violated element. This model seeks to measure not just the number of issues but also their impact on accessibility.

More recent studies have attempted to associate accessibility scores with machine learning models to predict perceived accessibility by users (Kelly et al., 2020). These models combine automatic test data with user feedback to offer a more holistic evaluation. In conclusion, while accessibility scoring systems are valuable tools for assessment and monitoring, the literature agrees that scores alone are insufficient. They should be supplemented by manual audits and user testing. Future developments aim to create more context-aware and user-centered scoring systems.

In this context, the accessibility scoring method implemented in this study represents an experimental model inspired by weighting principles in the literature. It aims to make Pa11y outputs more meaningful and actionable, ultimately bridging the gap between technical analysis and user experience-based evaluation.

7.2.3 Issue Type Filter

To enhance user experience, the panel includes three filter buttons:

- Errors
- Warnings
- Notices

These filters enable users to view only specific types of accessibility issues, allowing them to focus on areas of interest or concern.

7.2.4 Issue Distribution by Accessibility Principles

The identified issues are also categorized based on the WCAG accessibility principles, as outlined below:

- Principle: Robust
    - Compatible guideline: 4 issues identified
- Principle: Perceivable
    - Adaptable guideline: 3 issues
    - Distinguishable guideline: 9 issues
    - Text Alternatives guideline: 6 issues
- Principle: Operable
    - Navigable guideline: 14 issues

This structure provides a semantic classification of issues, showing not only numerical data but also indicating which core accessibility principles are being violated. This helps users identify problematic areas more easily and guides them in prioritizing improvements during the development process.

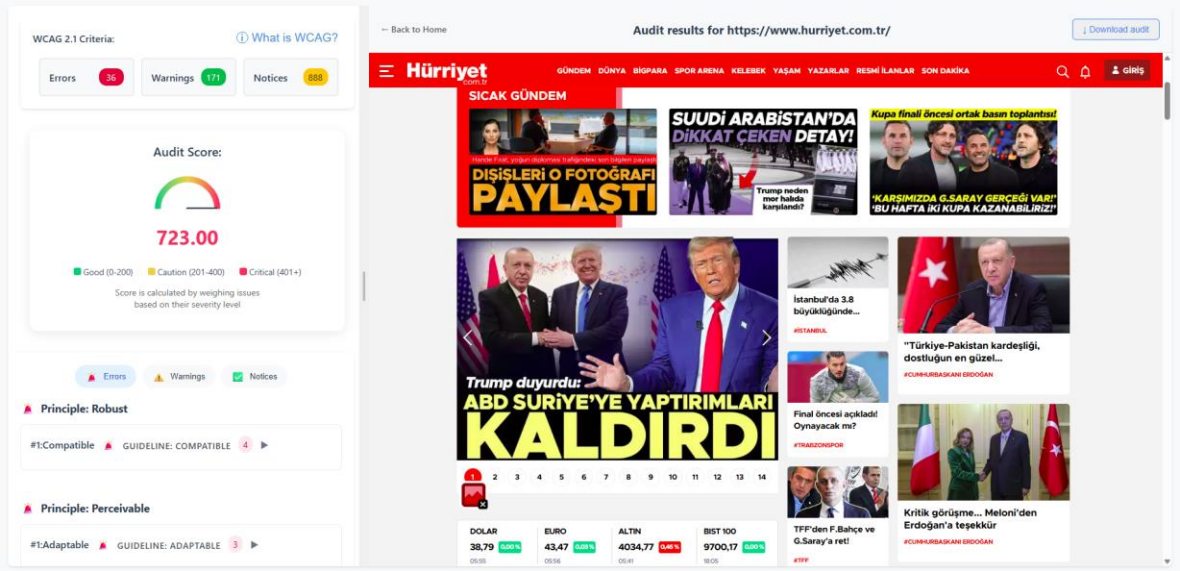

Figure 7.2 Accessibility Analysis Summary and Scoring System

7.3 Right Panel – Live Website Preview

The right panel provides an interactive live preview of the analyzed website. At the top of the panel, the analyzed URL is clearly shown, and in the top-right corner, a "Download Audit" button allows the user to export the audit results as a report file.

One of the most significant features of this panel is the visual highlighting of the issues detected by the system. When a user clicks on an issue from the left panel:

- The right panel automatically scrolls to the corresponding element on the page.
- The element is visually emphasized with a blinking marker, helping users to easily identify it within the visual context of the website.

Similarly, if the user clicks the marker directly in the right panel, the corresponding issue in the left panel is highlighted and brought into focus. This two-way synchronization provides users with both navigational support and a faster path to troubleshooting.

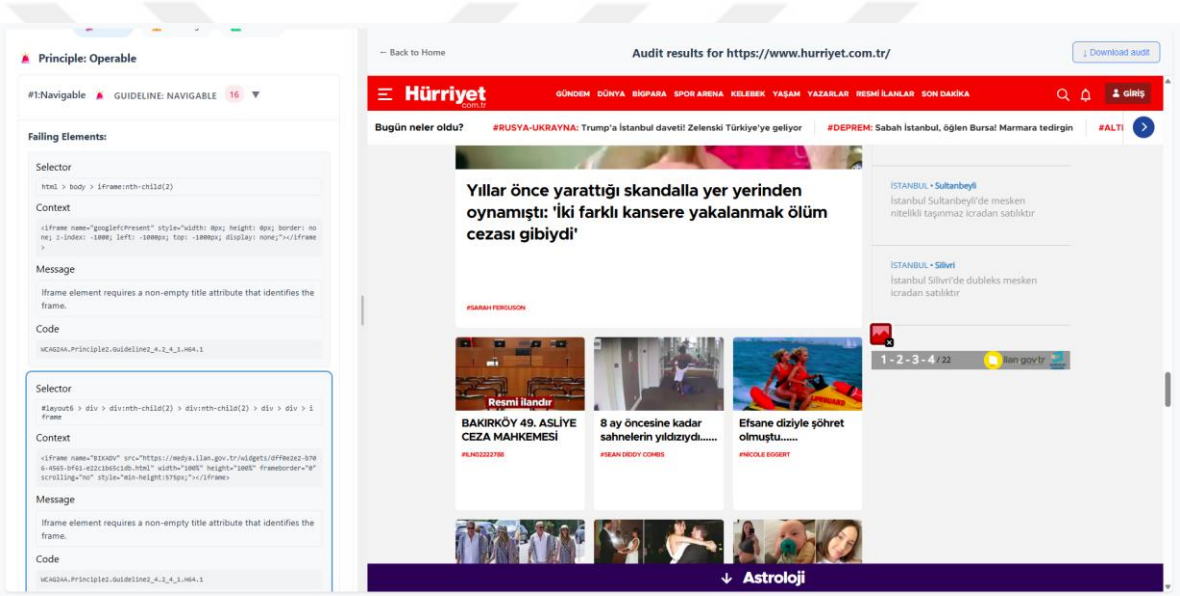

Figure 7.3 Website Preview

7.4 Contrast Adjustment Screen and Interactive Error Resolution

Web accessibility is an important factor in ensuring equal access to digital content for all users. In this context, the appropriateness of contrast ratios is a fundamental component of visual accessibility. The developed tool not only detects contrast errors directly but also provides users with the ability to correct these errors interactively. Contrast errors occur when the contrast between a background and foreground color of an element does not meet accessibility standards. These types of errors severely limit accessibility, especially for individuals with low vision.

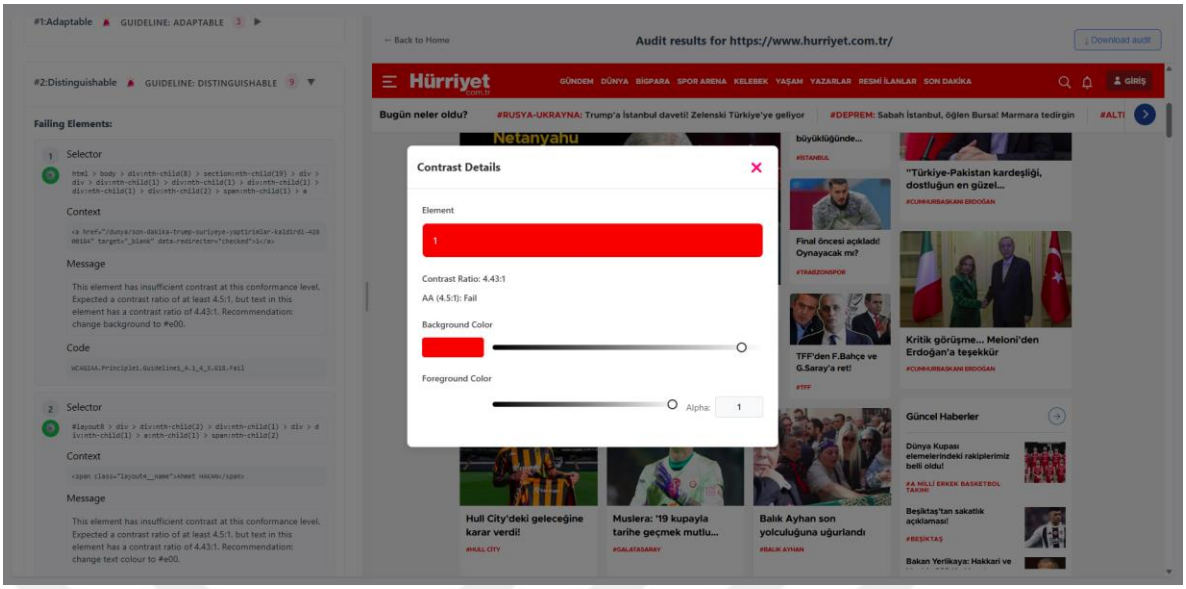

Figure 7.4 Contrast Adjustment Screen

Each element's contrast error is visually highlighted in the interface by displaying the contrast ratio. If a contrast error is detected, the user is directed to an editing screen through an icon next to the relevant element to examine and adjust the error. This screen presents the user with detailed contrast ratio information and shows the calculations of the background and foreground colors.

In the modal window, the user is presented with the contrast ratio, AA rating, and a detailed visual preview of the colors. When the contrast ratio is calculated, it dynamically appears to the user, providing a visual indicator for error analysis. This feature allows the user to quickly adjust the colors, and with visual feedback, makes it easier to identify errors.

The user can dynamically change the background and foreground colors to fix the contrast error. Interactive tools such as color pickers and range sliders are provided for both colors. These tools allow the user to adjust the colors manually, input hex values directly, or modify color tones using the slider. Additionally, the alpha (opacity) value can also be controlled, allowing for more precise adjustments of the contrast ratio by adjusting the opacity levels of the colors.

This interactive adjustment interface is a feature that is typically not found in other accessibility tools, enabling users to directly modify the styles and accessibility standards of website elements. This feature helps developers and users quickly detect and fix contrast errors, taking an important step in enhancing accessibility.

The developed interactive contrast adjustment interface not only enables the detection and analysis of contrast errors but also facilitates the quick resolution of these errors. Dynamic changes made with color and opacity values allow users to optimize accessibility while visual and interactive feedback helps effectively resolve errors. Such a feature can be considered an important innovation that makes web accessibility tools more effective and user-friendly.

7.5 Reporting

In web design, accessibility is crucial for improving the user experience and making websites more accessible for users with disabilities. In this context, the tool you developed provides significant functionality for auditing and reporting the accessibility of web pages. When the "Download Audit" button on the interface is clicked, the system performs accessibility tests on the webpage and generates a report in JSON format that can be downloaded. This report enables users to evaluate the page's compliance with accessibility standards and make the necessary corrections. The report is divided into three main categories: Errors, Warnings, and Notices. Each category addresses different levels of accessibility issues and highlights areas that need improvement.

The downloaded JSON report provides a detailed view of the accessibility test results for the webpage. Web developers and designers can use this report to identify areas where accessibility improvements are needed. Additionally, each item in the report clearly specifies the problematic elements and the necessary steps to fix them. By doing so, websites can be made compliant with accessibility standards, reaching a broader audience.

7.6 Tool Comparison

One of the widely used tools for web accessibility analysis, WAVE, may encounter errors or fail to complete the analysis when analyzing some modern websites. An example of

this situation is the website https://www.sephora.fr. When the WAVE tool attempts to analyze this site, it generates a technical error and fails to provide results.

This issue is primarily caused by the analysis being blocked due to dynamically generated content using JavaScript, custom components, or various client-side security measures. Such situations highlight the limitations of tools like WAVE.

However, the accessibility analysis tool I developed was able to successfully analyze the https://www.sephora.fr website, identifying errors according to the WCAG 2.1 criteria and providing a detailed report. This success demonstrates that my tool is more compatible with modern web technologies and capable of conducting a more comprehensive analysis.

The following images show a comparative view of the error screen produced by the WAVE tool and the successful analysis report from my developed tool for the same website:

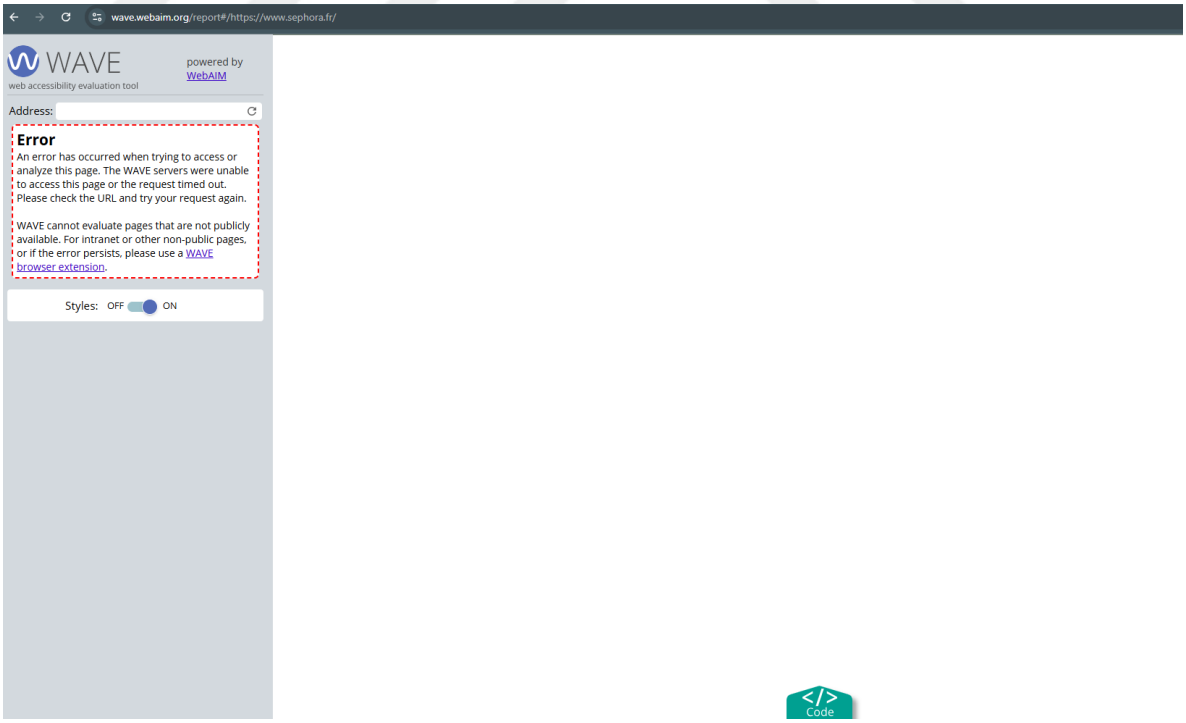

Figure 7.5 Error Screen Produced by the WAVE Tool

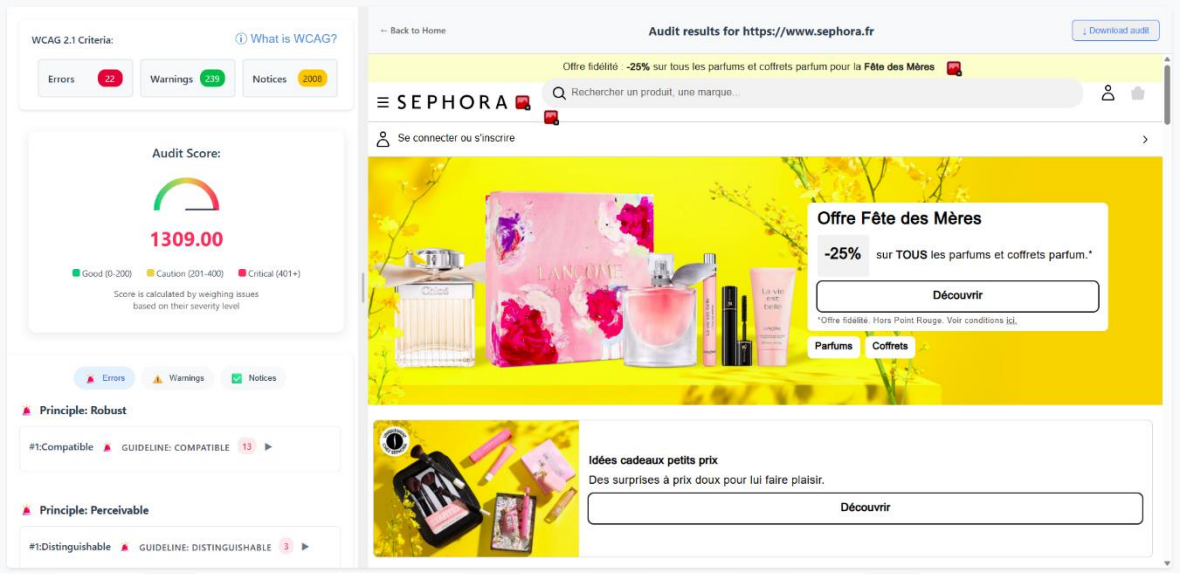

Figure 7.6 Accessibility Report Generated by the Developed Tool Without Error

This comparison not only demonstrates the ability of the developed tool to conduct analysis based on accessibility criteria, but also its capacity to overcome technical obstacles. The work presented in this thesis proves that, by understanding the limitations of existing tools, more advanced solutions can be designed.

8. WEB ACCESSIBILITY RESEARCH AND COMPARATIVE ANALYSIS

There are significant methodological differences among accessibility evaluation tools. In this study, Lighthouse, WAVE, and a custom-developed tool (My Tool) were used, each adopting distinct approaches in analyzing accessibility. For instance, Lighthouse, developed by Google, focuses on dynamic criteria such as interaction, performance, and user experience on mobile devices, identifying accessibility issues primarily from a user interaction perspective. In contrast, WAVE (Web Accessibility Evaluation Tool) conducts more code-level evaluations through static HTML elements, semantic structure, and ARIA labels, thus offering a more structural analysis.

As a result, notable differences in the number and types of errors identified by the tools were observed, even when analyzing the same website. For example, in the analysis of the CNN website regarding WCAG 2.1 criterion SC 2.5.8 (Target Size Minimum), Lighthouse reported 60 errors, whereas WAVE and My Tool either reported significantly fewer errors or none at all. This illustrates that the interpretation, measurement, and reporting methods of WCAG criteria differ substantially across tools.

Rather than viewing these differences as shortcomings, they should be considered complementary. Such diversity enables a more comprehensive and multidimensional evaluation. The My Tool developed in this study aims to offer more inclusive and balanced results in the future by integrating the analytical logic derived from multiple tools into a multi-approach model.

Moreover, some tools tend to produce false positive results. For example, in WAVE's analysis of the Hürriyet website, 108 errors were reported under the warning "linked image missing alternative text." However, manual inspection revealed that the majority of these images actually contained alternative text, indicating that the system

misinterpreted some label structures and reported inaccurate errors. This highlights that accessibility testing should not rely solely on automated tool outputs and should be supplemented with manual checks when necessary. Doing so can help minimize both false positives and false negatives, leading to more reliable assessments.

In conclusion, the fact that one tool reports more errors than another does not necessarily mean it is more accurate or effective. A high number of reported issues may sometimes stem from false positives. Therefore, when comparing tools, it is important to consider not only quantitative data but also the accuracy, context, and relevance of the findings.

8.1 Accessibility Analysis of Galatasaray University Turkish Page

Table 8.1 Accessibility Analysis of Galatasaray University Turkish Page

Tool	Accessibility Criterion	Error Count
Lighthouse	SC 1.4.3: Contrast (Minimum) (Level AA)	9
	SC 1.1.1: Non-text Content (Sufficient)	1
	SC 2.4.4: Link Purpose (In Context) (Sufficient)	1
	SC 2.4.9: Link Purpose (Link Only)	2
	SC 2.4.3: Focus Order	2
	SC 2.5.8: Target Size (Minimum) (Level AA)	34
WAVE	Missing alternative text	1
	Linked image missing alternative text	2
	SC 2.4.4: Link Purpose (In Context) (Level A)	2
	Missing form label	3
	Empty link	3
	SC 1.4.3: Contrast (Minimum) (Level AA)	14
My Tool	SC 3.2.2: On Input (Level A)	3
	SC 4.1.2: Name, Role, Value (Level A)	8
	SC 1.4.3: Contrast (Minimum) (Level AA)	5
	SC 1.1.1: Non-text Content (Level A)	1

In the accessibility analysis of the Turkish homepage of Galatasaray University, the most common issue was found under SC 2.5.8: Target Size (Minimum) criterion (34 errors). This indicates that the clickable areas targeted, especially for mobile users, are not large

enough. Contrast issues were also commonly observed across all tools (Lighthouse: 9, WAVE: 14, My Tool: 5). This reveals that readability is low for users with color blindness or visual impairments. Additionally, fundamental accessibility problems such as missing alternative texts, empty links, and missing form labels were also reported. These findings indicate that the website contains important areas that need improvement in terms of accessibility.

8.2 Accessibility Analysis of the Le Monde Website

Table 8.2 Accessibility Analysis of the Le Monde Website

Tool	Accessibility Criterion	Error Count
Lighthouse	SC 1.1.1: Non-text Content (Level A)	1
	SC 4.1.2: Name, Role, Value (Level A)	4
	SC 1.3.1: Info and Relationships (Level AA)	4
	SC 1.4.3: Contrast (Minimum) (Level AA)	59
WAVE	SC 1.1.1: Non-text Content (Level A)	5
	SC 2.4.4: Link Purpose (In Context) (Sufficient)	4
	SC 1.4.3: Contrast (Minimum) (Level AA)	25
My Tool	SC 1.4.3: Contrast (Minimum) (Level AA)	18
	SC 1.1.1: Non-text Content (Level A)	4
	SC 4.1.2: Name, Role, Value (Level A)	4
	SC 2.4.1: Bypass Blocks (Level A)	1
My Tool-axe	SC 1.4.3: Contrast (Minimum) (Level AA)	18
	SC 1.1.1: Non-text Content (Level A)	4
	SC 4.1.2: Name, Role, Value (Level A)	4
	SC 2.4.1: Bypass Blocks (Level A)	1

The accessibility analysis conducted on the Le Monde news portal reveals that the most common issues are related to contrast deficiencies. A total of nearly 150 contrast violations were identified across all tools (Lighthouse: 59, WAVE: 25, My Tool: 18, axe: 56). This suggests that individuals with visual impairments or those using the site under low light conditions may have difficulty reading the content.

Additionally, missing alternative text for non-text content, particularly identified by WAVE and My Tool, was prominently reported. This deficiency makes it harder for screen reader users to access the content. Moreover, the SC 4.1.2: Name, Role, Value

criterion was violated by all four tools, indicating that user interface components are not correctly defined and are incompatible with accessibility technologies.

Finally, the SC 2.4.1: Bypass Blocks criterion, reported only by "My Tool," highlights the absence of structural mechanisms that would allow users to bypass repetitive content. This deficiency can lead to time wastage and difficulty in navigation, especially for keyboard users.

Overall, the Le Monde website shows a significant need for accessibility improvements, particularly in the areas of visual contrast, alternative text, and structural labeling.

8.3 Accessibility Analysis of the CNN Website

Table 8.3 Accessibility Analysis of the CNN Website

Tool	Accessibility Criterion	Error Count
Lighthouse	SC 4.1.2: Name, Role, Value (Level A)	1
	SC 2.5.8: Target Size (Minimum) (Level AA)	60
	SC 1.2.2: Captions (Prerecorded) (Level A)	1
	SC 1.2.4: Captions (Live) (Sufficient)	1
WAVE	Missing alternative text	5
	Spacer image missing alternative text	3
	Missing form label	1
	Empty form label	6
	Empty link	5
My Tool	SC 2.4.1: Bypass Blocks (Level A)	14
	SC 4.1.2: Name, Role, Value (Level A)	6
	SC 4.1.1: Parsing (Level A)	2
	SC 1.1.1: Non-text Content (Level A)	7
	SC 1.3.1: Info and Relationships (Level A)	1
My Tool-axe	SC 1.4.3: Contrast (Minimum) (Level AA)	9
	SC 4.1.2: Name, Role, Value (Level A)	5
	SC 2.4.4: Link Purpose (In Context) (Sufficient)	1
	SC 1.2.2: Captions (Prerecorded) (Level A)	2

When analyzing the CNN website for accessibility, the most significant issues identified were the target size deficiencies (SC 2.5.8) found by Lighthouse. This criterion was violated 60 times, indicating that users, especially those using touch screens, might face difficulty interacting with small or overly close interface elements.

Missing alternative text (reported by WAVE and My Tool, totaling around 15) reduces the accessibility of visual content for screen readers, negatively affecting the experience of users with visual impairments. Additionally, empty form labels (6) and empty links (5), as reported by WAVE, create significant barriers to navigation, impairing the semantic integrity of the page.

The SC 4.1.2: Name, Role, Value criterion was violated by all tools (totaling 17 times), suggesting that interface components are not properly labeled and may cause interaction issues with accessibility technologies.

The lack of Bypass Blocks (SC 2.4.1) reported by My Tool (14 occurrences) is also noteworthy. This indicates that there is no mechanism to allow users to skip repetitive navigation blocks on each page transition.
Moreover, caption deficiencies for video content (SC 1.2.2 and 1.2.4) were reported by Lighthouse and axe, creating a significant accessibility barrier for users with hearing impairments and showing that multimedia content is not fully inclusive.

Finally, contrast deficiencies were reported by axe (9 occurrences). This poses a problem for users with low vision, as it reduces the readability of text. In general, the CNN website shows substantial violations of accessibility criteria related to interactive areas, visual content, and video materials. Therefore, it can be concluded that a comprehensive improvement process is required to make the site compliant with WCAG standards.

8.4 Accessibility Analysis of the Hürriyet Website

Table 8.4 Accessibility Analysis of the Hürriyet Website

Tool	Accessibility Criterion	Error Count
Lighthouse	SC 1.4.3: Contrast (Minimum) (Level AA)	14
	SC 2.4.1: Bypass Blocks	2
	SC 4.1.2: Name, Role, Value (Level A)	4
	SC 2.4.4: Link Purpose (In Context)	4
	SC 2.5.8: Target Size (Minimum) (Level AA)	174
WAVE	Linked image missing alternative text	108
	Spacer image missing alternative text	2
	Missing form label	1

	Empty button	2
	Very low contrast	13
My Tool	SC 4.1.2: Name, Role, Value (Level A)	4
	SC 1.4.3: Contrast (Minimum) (Level AA)	12
	SC 4.1.1: Parsing (Level A)	0
	SC 2.4.1: Bypass Blocks (Level A)	11
	SC 1.1.1: Non-text Content (Level A)	6
	SC 1.3.1: Info and Relationships (Level A)	2
My Tool-axe	SC 1.4.3: Contrast (Minimum) (Level AA)	42
	SC 4.1.2: Name, Role, Value (Level A)	2
	SC 2.4.4: Link Purpose (In Context)	14

Accessibility tests performed on the Hürriyet website show that the most frequent violations are related to missing alternative text for linked images. The “linked image missing alternative text” error reported 108 times by the WAVE tool indicates that the alt attribute of img tags, which is crucial for screen reader users, is missing.

However, detailed manual review found that in many cases, these images actually have alt tags. This suggests that the WAVE tool may produce incorrect evaluations in some HTML structures or content rendered with specific JavaScript. Therefore, this issue can be considered a false positive caused by the tool.

Building on these technical issues, custom algorithms were developed in your own accessibility tool to analyze such errors more accurately. By properly analyzing the relationship of the alt text of the linked image in the DOM tree, and considering the context that WAVE missed, this tool provided more accurate results. This demonstrates the tool’s important contribution not only to automatic error detection but also to filtering false positives.

In addition, the SC 2.5.8 (Target Size) error reported by Lighthouse 174 times indicates that the touch areas (buttons, links, etc.) on the page do not meet the minimum interaction size, which is a significant accessibility issue for mobile users.

Contrast issues were repeatedly detected by different tools (Lighthouse: 14, WAVE: 13, axe: 42). This suggests that the Hürriyet website contains low-contrast text in various areas, which decreases readability for users with visual impairments.

The SC 4.1.2: Name, Role, Value errors reported by My Tool and axe indicate that interactive elements on the page are not properly understood by screen readers and that the necessary information is not being passed to accessibility technologies.

Overall, it appears that the Hürriyet website requires various improvements in terms of accessibility standards, and it is also important to question the accuracy of the tools being used. The tool you developed fills a significant gap in the industry and enables more reliable results in accessibility assessments.

8.5 Accessibility Analysis of the The Guardian Website

Table 8.5 Accessibility Analysis of the The Guardian Website

Tool	Accessibility Criterion	Error Count
Lighthouse	SC 4.1.2: Name, Role, Value (Level A)	1
	SC 4.1.1: Parsing (Level A)	1
WAVE	Multiple form labels	2
	Very low contrast	4
My Tool	SC 4.1.1: Parsing (Level A)	3
My Tool-axe	SC 1.4.3: Contrast (Minimum) (Level AA)	250
	SC 2.1.3: Keyboard (No Exception) (Level AAA)	1
	SC 4.1.1: Parsing (Level A)	1

The Guardian website is generally in good structural condition, but the following issues were identified:

- Contrast issues represent a significant accessibility barrier (especially according to axe results).
- Parsing and labeling errors can make it difficult for assistive technologies to interpret page content correctly.
- Keyboard accessibility issues pose a fundamental barrier for users who rely solely on the keyboard.

These findings indicate that the website has several deficiencies in both Level A and Level AA criteria according to the WCAG 2.1 guidelines.

8.6 Accessibility Findings by Website

The error counts based on the WCAG (Web Content Accessibility Guidelines) criteria determined by the tools used for each website are presented in the table below:

Table 8.6 Accessibility Website Results

Website	Lighthouse	WAVE	My Tool	My Tool-axe
https://gsu.edu.tr/tr	49	25	17	63
https://www.lemonde.fr/	68	34	27	60
https://www.cnn.com	63	24	30	17
https://www.hurriyet.com	198	126	35	58
https://www.theguardian.com	2	6	3	252

As seen in the table, hurriyet.com.tr has the highest number of accessibility errors, while gsu.edu.tr shows a more accessible structure with the lowest error count.

8.7 General Distribution According to WCAG Success Criteria

The types of errors detected by the tools are grouped according to the WCAG criteria and presented in the table below:

Table 8.7 General Distribution According to WCAG Success Criteria

WCAG Success Criterion	WAVE	Lighthouse	My Tool
SC 1.4.3: Contrast (Minimum)	63	147	169
SC 2.5.8: Target Size (Minimum)	N/A	268	N/A
SC 4.1.2: Name, Role, Value	31	31	31
SC 1.1.1: Non-text Content	20	20	20
SC 2.4.4: Link Purpose	7	7	7

WCAG Success Criterion	WAVE	Lighthouse	My Tool
SC 1.3.1: Info and Relationships	7	7	7
SC 4.1.1: Parsing	6	6	6
SC 2.4.1: Bypass Blocks	26	26	26

According to these results, the most common accessibility issue is related to SC 1.4.3: Contrast (Minimum). Texts with insufficient color contrast, in particular, create significant accessibility problems for individuals using screen readers.

8.8 Galatasaray University Website Language-Based Test Results

The Galatasaray University website was also tested in Turkish, English, and French:

Table 8.8 Galatasaray University Website Language-Based Test Results

Language Version	SC 3.2.2	SC 4.1.2	SC 1.4.3	SC 1.3.1	SC 1.1.1
https://gsu.edu.tr/tr	3	6	5	2	1
https://gsu.edu.tr/en	3	6	8	2	1
https://gsu.edu.tr/fr	3	6	8	2	1

It was observed that the language differences only created variations in contrast errors. Accessibility issues, other than contrast, showed consistency across languages. This indicates that the overall structure of the site has similar accessibility features independent of the language.

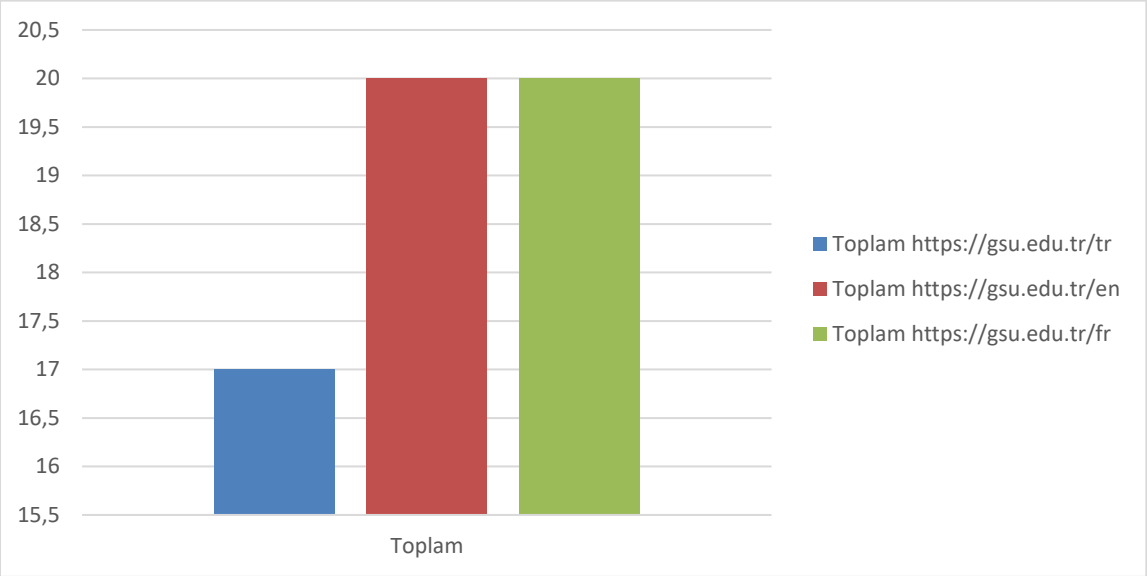

Figure 8.1 Galatasaray University Website Language-Based Test Results

8.9 Accessibility Test Page (index.html)

As part of this study, a dedicated HTML test page (index.html), presented in Appendix A, was created in order to better analyze the accessibility issues identified on the Hürriyet website. This page aims to replicate the accessibility deficiencies observed on the Hürriyet homepage—such as those related to images, links, buttons, and form elements—by recreating similar structures. In this way, the impact of these issues was tested in an isolated environment using accessibility evaluation tools and screen readers, enabling more accurate assessments. This approach, free from the complex interactions of the actual web page, was intended to improve the reliability of the analysis.

The index.html file is included in the appendices of this thesis and has been used as a reference in accessibility evaluations.

8.10 Accessibility Test Results Comparison

In this study, accessibility tests were conducted on various news websites, university portals, government institutions, and commercial platforms both in Turkey and internationally. The tests involved a comparative analysis between the widely-used

WAVE accessibility evaluation tool and a custom-developed accessibility testing tool (referred to as *My Tool*). While some web pages could be successfully analyzed by both tools, others could not be tested by WAVE due to technical limitations. In such cases, only the results obtained from the custom-developed tool were presented.

Sample Comparison:

Table 8.9 Accessibility Test Results Comparison

Website	WAVE Error Count	My Tool Error Count
https://www.hurriyet.com.tr/	99	6
https://www.mynet.com/	34	20
https://onedio.com/	Not testable	269 errors, 241 warnings, 1415 notices
https://www.trendyol.com/	Not testable	160 errors, 265 warnings, 1323 notices
https://eksisozluk.com/	Not testable	338 errors, 156 warnings, 439 notices
https://www.yemeksepeti.com/	Not testable	12 errors, 140 warnings, 221 notices
https://ibb.istanbul/	Not testable	15 errors, 38 warnings, 365 notices
https://www.turkiye.gov.tr/	Not testable	11 errors, 97 warnings, 176 notices
https://www.marketingturkiye.com.tr/	Not testable	26 errors, 96 warnings, 400 notices
https://www.yeniakit.com.tr/	Not testable	24 errors, 180 warnings, 1154 notices

Website	WAVE Error Count	My Tool Error Count
https://www.yenisafak.com/	Not testable	239 errors, 332 warnings, 1270 notices
https://www.fotomac.com.tr/	Not testable	85 errors, 324 warnings, 1425 notices
https://www.takvim.com.tr/spor	Not testable	227 errors, 144 warnings, 511 notices

The ability of the custom-developed tool to successfully complete tests on pages that WAVE could not access demonstrates its significant contribution to accessibility evaluations.

Sites not testable by either tool: n11.com, sahibinden.com, migros, hepsiburada, etstur, tesla.com, ciceksepeti.com, oley.com, bilyoner.com, misli.com, several university websites, and government platforms.

As a result of these analyses, the flexibility, analysis scope, and error detection capabilities of the custom-developed accessibility tool stand out, particularly for modern websites with dynamic content or those restricted from third-party resource analysis due to specific access policies.

However, some limitations of the tool were also observed. For pages with heavy visual content or complex DOM structures, mapping identified errors to their corresponding elements on the user interface with visual references is not always feasible. While the WAVE tool often provides such visual markers, the custom-developed tool is currently limited in this regard. This limitation restricts the ability to display errors alongside their page locations, but does not affect the core analytical functionality of the tool.

Future work will focus on enhancing the user experience by introducing visual element mapping (element highlight) and interactive reporting features to provide more intuitive and visually contextualized analysis results.

8.11 Encoding and Accessibility Issues of Turkish-Specific Characters

Automated accessibility evaluation tools provide a fast way to assess the compliance of web content with WCAG standards. However, neglecting basic elements such as language specification can lead to significant accessibility problems, especially in agglutinative languages like Turkish (W3C, 2018).

WCAG 2.1 Guideline 3.1.1 “Language of Page” requires the default language of each web page to be programmatically defined. For example, if the <html lang="tr"> tag is missing, screen readers may pronounce the content in English, resulting in loss of meaning and a poor user experience.

When Turkish-specific characters (ç, ğ, ş, ö, ü, ı) are not properly encoded, screen readers may either mispronounce them or skip them entirely. An analysis by Scope (2023) revealed that some special characters were read aloud as nonsensical phrases such as “mathematical sans-serif t.”

To prevent such issues, the use of universal character encodings such as Unicode (UTF-8) is of critical importance. In addition, these characters must be correctly processed in elements that directly convey information to the user, such as alt texts, button labels, and form descriptions.

Henry et al. (2014) emphasize that achieving accessibility requires a holistic approach that includes technical components such as character encoding, semantic structure, and a consistent interface.

8.12 Comparative Evaluation with WAVE Tool

While the WAVE tool provides a detailed analysis of static HTML and ARIA markup, it presents certain limitations in terms of user interaction. For example:

In WAVE, when a user clicks on an issue, the page scrolls to the relevant element. However, clicking on the same issue multiple times causes the page to scroll back to the top repeatedly, which negatively affects the user experience.

Furthermore, WAVE displays all types of issues simultaneously, resulting in a visually overwhelming interface where markers can overlap and clutter the page.
In contrast, the developed system offers users the ability to filter and view only the selected issue types. This helps reduce visual clutter and makes it easier for users to focus on specific categories of problems.

In summary, the developed analysis panel contributes not only to technical accuracy but also to ease of use through its interactive user experience, flexible filtering, two-way synchronization, and visual feedback features. These enhancements provide a more effective and user-friendly environment for conducting accessibility evaluations.

9. CONCLUSION

In this thesis study, the importance of making websites accessible to all users, especially individuals with disabilities, has been emphasized, and a system has been developed to address the limitations of existing automated accessibility evaluation tools. It has been observed that current tools are insufficient in providing reliable analyses, particularly for modern web pages containing dynamic content and JavaScript-based structures. For instance, the widely used WAVE tool sometimes generates false alerts (such as “Linked image missing alternative text”) even when components are correctly configured, or it fails to complete the analysis process. This can lead developers to be misled and result in insufficient improvements in terms of accessibility.

In this context, the developed accessibility evaluation system was built using the open-source Pa11y tool as its foundation, and it was implemented with Node.js and React.js technologies. The system not only reports accessibility issues but also offers visual feedback mechanisms and interactive interfaces that allow users to directly intervene in the analysis results. Moreover, the system has been customized to support Turkish character compatibility, enabling local users to perform more accurate and efficient evaluations.

Although Pa11y Dashboard was not directly used in the implementation, the core tool Pa11y was integrated into the system architecture. Given this foundation, it is technically feasible to extend the system in the future by incorporating dashboard-like features such as scheduled multi-page analysis, historical tracking of results, and graphical visualizations, similar to what Pa11y Dashboard offers. This indicates that the system is not only functional in its current state but also flexible and expandable for future needs.

As a result of applied testing, the system was found to provide more accurate error detection and was capable of successfully analyzing content dynamically generated by JavaScript. Features such as proxy support to overcome CORS restrictions, export options in PDF/JSON format, and the ability to store historical analysis records have expanded the system's usability and offered developers greater flexibility.

The developed system complements the shortcomings of existing accessibility evaluation tools, offering developers and user experience experts a more reliable, flexible, and user-friendly alternative. It has evolved into a comprehensive solution that not only serves as an analysis tool but also guides users with a visual, interactive structure that encourages correction, thereby contributing to digital inclusivity.

Future developments of the system will focus on enhancing compatibility with up-to-date standards such as WCAG 2.2, as well as expanding functionality to support accessibility evaluations for mobile applications. Moreover, the integration of AI-powered modules that generate automatic correction suggestions is planned in order to provide a more intelligent, personalized, and user-oriented evaluation process. These improvements will make the system more dynamic and capable of supporting inclusive design at scale.

Another important direction is conducting comprehensive usability testing with diverse user profiles, including accessibility experts, non-technical users, and individuals with various disabilities. This user-centric approach will provide deeper insights into the usability of the system, enabling refinements that ensure it remains practical, inclusive, and easy to adopt across different skill levels and accessibility needs.

In addition to technical and usability-related goals, upcoming work will address legal and institutional integration. Specifically, the Presidential Circular No. 2025/10, published in the Official Gazette of the Republic of Türkiye on June 21, 2025 (No. 32933), introduces a national web and mobile accessibility monitoring framework. Under this regulation, a commission led by the Ministry of Family and Social Services will oversee accessibility compliance and award qualifying institutions the right to display an Accessibility Logo for two years.

The Circular also mandates that a wide range of public and private institutions including ministries, universities, municipalities, public enterprises, professional chambers, private education institutions, and service providers in transportation and e-commerce sectors must ensure full accessibility compliance within 1 to 2 years, using the A-Level Web Accessibility Checklist and aligning with WCAG guidelines.

In this regard, the developed system is well-positioned to support such mandates and could be extended with features such as:

- Official accessibility scoring mechanisms (e.g., star-based ratings),
- Compliance report generation aligned with national requirements,
- Administrative dashboards for long-term monitoring and institutional reporting.

Such enhancements would not only broaden the system's adoption among public institutions but also strengthen its role as a strategic digital accessibility tool aligned with national policy, contributing to the broader mission of equal access and non-discrimination.

Ultimately, future work will aim to ensure that the system evolves into a scalable, adaptable, and policy-compliant solution that empowers all stakeholders from developers to policymakers in building a more accessible digital ecosystem.

REFERENCES

Abascal, J., Arrue, M., & Valencia, X. (2019). Tools for web accessibility evaluation. In M. Vigo & Y. Yesilada (Eds.), Web accessibility: A foundation for research (pp. 479–503). Springer.

Abascal, J., Arrue, M., & Valencia, X. (2019). Assessing web accessibility: Evaluation methods and tools. In M. Vigo & Y. Yesilada (Eds.), Web accessibility: A foundation for research (pp. 143–164). Springer. https://doi.org/10.1007/978-1-4471-7440-0_7

Abduganiev, A. (2017). Web accessibility evaluation tools: A comparative study [Master's thesis, Malmö University]. DiVA Portal. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1143576

Abou-Zahra, S., Brewer, J., & Cooper, M. (2017). Web accessibility evaluation tools: Overview and limitations. World Wide Web Consortium (W3C). https://www.w3.org/WAI/eval/tools/

Abou-Zahra, S., Steenhout, N., & Keen, L. (Eds.). (2017). Selecting web accessibility evaluation tools. https://www.w3.org/WAI/test-evaluate/tools/selecting/

AccessibilityChecker.org. (2022). How we calculate accessibility scores. https://www.accessibilitychecker.org/blog/accessibilitychecker-score/

Alnanih, L., Pedell, S., & Burdett, J. (2016). A weighted scoring model for assessing web accessibility. In K. Miesenberger, C. Bühler, & P. Penaz (Eds.), Computers helping people with special needs (pp. 193–200). Springer. https://doi.org/10.1007/978-3-319-41267-2_28

Alsaeedi, A. (2020). Comparing web accessibility evaluation tools and evaluating the accessibility of webpages: Proposed frameworks. Information, 11(1), 40.

Alsaeedi, F. (2020). Enhancing the effectiveness of web accessibility evaluation tools [Doctoral dissertation, University of Southampton]. ePrints Soton. https://eprints.soton.ac.uk/445399/

AudioEye. (2021). Accessibility score methodology deep dive. https://audioeye.medium.com/accessibility-score-methodology-deep-dive-4e405e0f923c

Bennefield, R. (1997). Technology and the disabled. American Demographics, 19(6), 44–49.

Brajnik, G. (2009, October). Validity and reliability of web accessibility guidelines. In Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility (pp. 131–138). ACM.

Brophy, P., & Craven, J. (2007). Web accessibility. Library Trends, 55(4), 950–972.

Caldwell, B., Cooper, M., Reid, L. G., & Vanderheiden, G. (2008). Web Content Accessibility Guidelines (WCAG) 2.0. W3C Recommendation. https://www.w3.org/TR/WCAG20/

Carter, J., & Markel, M. (2001). Web accessibility for people with disabilities: An introduction for web developers. IEEE Transactions on Professional Communication, 44(4), 225–233. https://doi.org/10.1109/47.968110

Carter, J., & Markel, M. (2015). Web accessibility for people with disabilities: An introduction for web developers. In Writing and speaking in the technology professions: A practical guide (pp. 484–492).

Centeno, C., Kloos, C. D., Gaedke, M., & Weitzel, A. (2006). Comparative analysis of web accessibility evaluation tools: A tool for selecting the most appropriate one for a given need. In Proceedings of the 13th International Conference on Computers Helping People with Special Needs (ICCHP 2006) (pp. 778–785). Springer. https://doi.org/10.1007/11788713_118

Centeno, V. L., Kloos, C. D., Fisteus, J. A., & Álvarez, L. Á. (2006). Web accessibility evaluation tools: A survey and some improvements. Electronic Notes in Theoretical Computer Science, 157(2), 87–100.

Cumhurbaşkanlığı. (2025, Haziran 21). Web Siteleri ve Mobil Uygulamaların Erişilebilirliği Hakkında Genelge (Genelge No: 2025/10). Resmî Gazete (Sayı: 32933).

Frazão, T., & Duarte, C. (2020). Comparing accessibility evaluation plug-ins. In Proceedings of the 17th International Web for All Conference (pp. 1–4). ACM. https://doi.org/10.1145/3371300.3383340

Friedman, M. G., & Bryen, D. N. (2007). Web accessibility design recommendations for people with cognitive disabilities. Technology and Disability, 19(4), 205–212. https://doi.org/10.3233/TAD-2007-19404

Harper, S., & Yesilada, Y. (2008). Web accessibility: A foundation for research. Springer.

Hassenzahl, M., & Tractinsky, N. (2006). User experience – A research agenda. Behaviour & Information Technology, 25(2), 91–97.

Heim, J. (2000). Locking out the disabled. PC World, 18(9), 181–185.

Henry, S. L. (2015). Just Ask: Integrating accessibility throughout design. W3C WAI.

Henry, S. L., Zahra, F. M., & Brewer, J. (2014). The role of accessibility in a universal web. W3C. https://www.w3.org/WAI/intro/accessibility.php

Kelly, B., Phipps, L., & Howell, C. (2005). Implementing a holistic approach to e-learning accessibility. ALT-J, 13(1), 69–78.

Kelly, B., Sloan, D., Phipps, L., Petrie, H., & Hamilton, F. (2020). Accessibility 2.0: Next steps for web accessibility evaluation. Journal of Web Engineering, 19(1), 65–82.

Schmutz, S., Sonderegger, A., & Sauer, J. (2016). Implementing recommendations from web accessibility guidelines: Would they also provide benefits to nondisabled users? Human Factors, 58(4), 611–629. https://doi.org/10.1177/0018720816640963

Timbi-Sisalima, C., Amor, C. I. M., Otón, S., Hilera, J. R., & Aguado-Delgado, J. (2016). Comparative analysis of online web accessibility evaluation tools. In Proceedings of the 15th International Conference on Computers Helping People with Special Needs (pp. 136–143). Springer.

Vigo, M., Brown, J., & Conway, V. (2013). 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 '13). ACM. https://doi.org/10.1145/2461121.2461124

W3C. (n.d.-a). W3C Web Accessibility Initiative (WAI). https://www.w3.org/WAI/

W3C. (n.d.-b). Web Content Accessibility Guidelines (WCAG). https://www.w3.org/WAI/standards-guidelines/

W3C. (n.d.-c). Web Accessibility Evaluation Tools List. https://www.w3.org/WAI/ER/tools/

WebAIM. (n.d.). WAVE: Web Accessibility Evaluation Tool. Utah State University. https://wave.webaim.org/

World Health Organization. (2020). Disability and health. https://www.who.int/news-room/fact-sheets/detail/disability-and-health

APPENDICES

Appendix A: Accessibility Test Page (HTML Example)

Below is an example page created to test various HTML usage scenarios in terms of accessibility. This example has been used to evaluate how different HTML elements such as visual content, forms, links, and buttons behave in the context of accessibility.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Accessibility Test Page - Multilang</title>
  <style>
    .hidden-display { display: none; }
    .hidden-visibility { visibility: hidden; }
    .decorative { width: 50px; height: 50px; }
  </style>
</head>
<body>
  <h1>Web Accessibility Test Page</h1>

  <!-- Invisible image with accessible alt text -->
  <a href="https://example.com">
    <img src="https://placekitten.com/112/112" alt="Anasayfaya git / Go to homepage" class="hidden-visibility">
  </a>
   <!-- Visible image with descriptive alt text -->
```

```
<a href="https://example.com">
  <img src="https://placekitten.com/112/112" alt="Anasayfaya git / Go to homepage">
 </a>
 <!-- Image using only data-src (problematic for accessibility) -->
 <a href="https://example.com/news" title="Click to read">
  <img data-src="https://via.placeholder.com/150" alt="Haber görseli / News image" title="Click to read">
 </a>

 <!-- Repeated content leading to the same link -->
 <a href="/news/123" title="News headline">
  <img src="https://via.placeholder.com/150" alt="Haber başlığı / News headline">
 </a>
 <a href="/news/123" title="News headline">
  <span>Haber başlığı / News headline</span>
 </a>

 <!-- Image link without alt text -->
 <a href="https://example.com" title="Go to image">
  <img src="https://via.placeholder.com/150" alt="">
 </a>

 <!-- Second image defined using data-src -->
 <a href="https://example.com" title="Example image">
  <img data-src="https://via.placeholder.com/150" alt="Örnek görsel / Example image">
 </a>

 <!-- Multilingual examples -->
 <a href="https://example.com">
  <img src="https://placekitten.com/101/101" alt="Gizli Görsel / Hidden Image" class="hidden-display">
 </a>
```

```
  <a href="https://example.com">
    <img src="https://placekitten.com/102/102" alt="İstanbul’un logosu – Görsel bağlantı / Istanbul logo – Image link">
  </a>
  <a href="https://example.com">
    <img src="https://placekitten.com/103/103" alt="Görünmez görsel / Invisible image" class="hidden-visibility">
  </a>
  <a href="https://example.com">
    <img src="https://placekitten.com/104/104" alt="Invisible image" class="hidden-visibility">
  </a>
  <a href="https://example.com">
    <img src="https://placekitten.com/111/111" alt="アクセシビリティテスト用画像 / Image for accessibility test (JP)" class="hidden-visibility">
  </a>
  <a href="https://example.com">
    <img src="https://placekitten.com/105/105" alt="Image cachée / Hidden image (FR)" class="hidden-visibility">
  </a>
  <a href="https://example.com">
    <img src="https://placekitten.com/106/106" alt="Imagen de prueba para accesibilidad / Accessibility test image (ES)" class="hidden-visibility">
  </a>
  <a href="https://example.com">
    <img src="https://placekitten.com/112/112" alt="Verstecktes Bild / Hidden image (DE)" class="hidden-visibility">
  </a>

  <!-- Decorative image -->
  <img src="https://placekitten.com/107/107" alt="" role="presentation" class="decorative">
```

```
<!-- Informative image -->
<img src="https://placekitten.com/108/108" alt="Bu sadece bilgi veren bir görsel / Informative image only">

<!-- Image links conveying meaningful content -->
<a href="https://example.com">
 <img src="https://placekitten.com/109/109" alt="Çalışmalarımız hakkında bilgi almak için tıklayın / Click to learn more about our work.">
</a>
<a href="https://example.com">
 <img src="https://placekitten.com/110/110" alt="Gizli erişim / Hidden access" aria-hidden="true">
</a>
<a href="https://example.com">
 <img src="https://placekitten.com/111/111" alt="アクセシビリティテスト用画像 / Image for accessibility test (JP)">
</a>
<a href="https://example.com">
 <img src="https://placekitten.com/112/112" alt="Verstecktes Bild / Hidden image (DE)" class="hidden-display">
</a>

<!-- Buttons -->
<button style="display:none;">Tıklanamaz </button>
<button style="visibility:hidden;">Görünmeyen buton </button>
<button aria-hidden="true">Ekran okuyucuya kapalı </button>
<button style="display:none;">İçeriği Göster / Show Content</button>

<!-- Headings -->
<h2 class="hidden-display">Gizli Başlık (display:none) / Hidden Heading</h2>
<h2 class="hidden-visibility">Gizli Başlık (visibility:hidden) / Hidden Heading</h2>
<h2 aria-hidden="true">Gizli Başlık (aria-hidden) / Hidden Heading</h2>
```

```
<!-- Paragraphs -->
<p class="hidden-display">Gizli Paragraf (display:none) / Hidden Paragraph</p>
<p class="hidden-visibility">Gizli Paragraf (visibility:hidden) / Hidden Paragraph</p>
<p aria-hidden="true">Gizli Paragraf (aria-hidden) / Hidden Paragraph</p>

<!-- Links -->
<a href="#" class="hidden-display">Gizli Link (display:none) / Hidden Link</a>
<a href="#" class="hidden-visibility">Gizli Link (visibility:hidden) / Hidden Link</a>
<a href="#" aria-hidden="true">Gizli Link (aria-hidden) / Hidden Link</a>

<!-- Form elements -->
<h2>Form Field Accessibility Test</h2>

<!-- Properly labeled field -->
<label for="ad1">Adınız (doğru örnek) / Your Name (correct example):</label>
<input type="text" id="ad1" placeholder="Adınızı yazın / Enter your name">

<!-- Unlabeled field -->
<input type="text" id="ad2" placeholder="Eksik etiketli input / Input without label">

<!-- Hidden input with label -->
<label for="ad3">Gizli Ad (label var, input gizli) / Hidden Name (label exists, input is hidden):</label>
<input type="text" id="ad3" style="display:none;" placeholder="Gizli input / Hidden input">

<!-- Hidden input (visibility:hidden) -->
<input type="text" id="ad4" style="visibility:hidden;" placeholder="Etiketsiz ve gizli input / No label, hidden input">

<!-- Input hidden from screen readers -->
<label for="ad5">Adınız (aria-hidden) / Your Name (aria-hidden):</label>
```

```
  <input type="text" id="ad5" aria-hidden="true" placeholder="Ekran okuyucuya kapalı / Hidden from screen reader">

  <!-- Textarea without label -->
  <textarea placeholder="Etiketsiz textarea / Textarea without label"></textarea>

  <!-- Textarea with label -->
  <label for="mesaj1">Mesajınız / Your Message:</label>
  <textarea id="mesaj1" placeholder="Bir mesaj yazın... / Write a message..."></textarea>

  <!-- Textarea hidden from screen readers -->
  <label for="mesaj2">Gizli Mesaj Alanı / Hidden Message Field:</label>
  <textarea id="mesaj2" aria-hidden="true">Gizli metin / Hidden text</textarea>

</body>
</html>
```