الجمهورية الجزائرية الديمقراطية الشعبية
The People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جامعة محمد بوضياف بالمسيلة
University Mohamed Boudiaf of M'sila

كلية الرياضيات والإعلام الآلي
Faculty of Mathematics and Informatics

قسم الإعلام الآلي
Department of Computer Science

Domain: Mathematics and Computer Science

Thesis Presented to Fulfill the Partial Requirement
for **Master's Degree** in Computer Science

Specialty: Information Systems and Software Engineering

Prepared By: Thabit Seyf Eddine ZOUAK and Ayoub TAHARI

Supervised By:

Dr. Mahmoud BRAHIMI

ENTITLED

Development of an Automated UX Heuristic Evaluation System Based on Nielsen's Guidelines

Jury Members

Nourreddine CHIKOUCHE	President
Mahmoud BRAHIMI	Supervisor
Seddik BOUGHRARA	Examiner

Academic Year 2024/2025

Dedication

To those who instilled in me a love for knowledge and stayed up countless nights for my sake,
To those whose prayers were my strength at every step,
To my beloved mother, the symbol of kindness and sacrifice,
And to my dear father, the beacon of wisdom and unwavering support...
I dedicate this thesis in deep gratitude for your endless love and sacrifices—words cannot truly express it.

To my dear siblings, who shared with me both the struggles and the joys of the journey,
Your support was a constant motivation, and your smiles were a source of strength.
Thank you from the bottom of my heart.

To everyone who had a positive impact on my academic path,
I dedicate this work as the fruit of a shared effort and unforgettable perseverance.

Zouak Thabit Seyf Eddine

To those who have always been a source of strength and inspiration,
To those who never ceased to give and pray,
To my loving mother, the symbol of unconditional love,
And to my dear father, my role model and greatest supporter...
I dedicate this work as a token of eternal gratitude and heartfelt appreciation.

To my beloved brothers and sisters,
My companions on this journey and a constant source of joy and encouragement,
This work is dedicated to you in recognition of your sincere support and love.

To everyone who supported me, even with a kind word,
This work is for you — a fruit of perseverance, shared effort, and sincere intentions.

Tahari Ayoub

Acknowledgment

First and foremost, we express our deepest gratitude to ALLAH for granting us the strength, patience, and perseverance to reach this milestone. We would like to sincerely thank Mr. Brahimi Mahmoud, our esteemed supervisor, for his valuable guidance, continuous support, and insightful feedback throughout the development of this project. His encouragement and constructive remarks played a crucial role in the progress and quality of our work. We also extend our heartfelt thanks to everyone who supported us during this journey, our families, friends, and colleagues for their unwavering encouragement and belief in us.

ملخص

يهدف هذا المشروع إلى تصميم وتنفيذ أداة آلية تقوم بإجراء تدقيق لتجربة المستخدم (UX) على مواقع الويب، بالاعتماد على تقنيات مثل Selenium تعتمد الأداة على القواعد العشر لنيلسن في تقييم واجهات الاستخدام، مثل وضوح حالة النظام، التطابق مع العالم الواقعي، تحكم المستخدم وحريته، الاتساق، ومنع الأخطاء، وغيرها. تشمل وظائف الأداة اكتشاف عناصر واجهة المستخدم غير المتوافقة مثل الأزرار، رسائل الخطأ والتنقل، إلى جانب تحليل إمكانية الوصول، وتوليد تقارير تلقائية تتضمن نتائج التدقيق، مع تقديم توصيات UX قائمة على المعايير المعتمدة. تسهم هذه الأداة في تسهيل عمليات التدقيق وتحسين جودة تجربة المستخدم بشكل منهجي وفعال.

الكلمات المفتاحية: واجهة المستخدم، قواعد نيلسن، تدقيق، تجربة المستخدم.

Abstract

This project aims to design and implement an automated tool that performs user experience (UX) audits on websites, relying on technologies such as Selenium. The tool evaluates user interfaces based on Nielsen's ten usability heuristics, including visibility of system status, the match between the system and the real world, user control and freedom, consistency, and error prevention, among others. Its functionalities include detecting non-compliant UI elements such as buttons, error messages, and navigation components, in addition to analyzing accessibility, generating automated audit reports, and providing UX recommendations based on established standards. This tool contributes to streamlining the audit process and enhancing the quality of user experience systematically and effectively.

Keywords: User Interface, Nielsen's Heuristics, Audit, User Experience.

Résumé

Ce projet vise à concevoir et à mettre en œuvre un outil automatisé permettant d'effectuer des audits de l'expérience utilisateur (UX) sur des sites web, en s'appuyant sur des technologies telles que Selenium. L'outil évalue les interfaces utilisateur selon les dix heuristiques d'utilisabilité de Nielsen, notamment la visibilité de l'état du système, la correspondance entre le système et le monde réel, le contrôle et la liberté de l'utilisateur, la cohérence, la prévention des erreurs, entre autres. Ses fonctionnalités incluent la détection des éléments d'interface non conformes tels que les boutons, les messages d'erreur et les composants de navigation, ainsi que l'analyse de l'accessibilité, la génération de rapports d'audit automatisés, et la fourniture de recommandations UX basées sur des standards établis. Cet outil contribue à rationaliser le processus d'audit et à améliorer la qualité de l'expérience utilisateur de manière systématique et efficace.

Mots-clés : Interface utilisateur, Heuristiques de Nielsen, Audit, Expérience utilisateur.

Table of Contents

CHAPTER 3: Technical Development of the Automated UX Audit System

List of Figures

List of Tables

General Introduction

In today's digital age, where technology is constantly reshaping how individuals interact with information, services, and one another, the need for digital systems that are practical, user-friendly, and human-centered has become more critical than ever. Human-machine interaction (HMI) is now increasingly pervasive, with profound and multi-dimensional implications across various domains. Users are no longer passive consumers of technology; instead, they are active participants who demand intuitive interactions, rapid response times, and seamless user experiences.

In response to these evolving expectations, the field of Human-Computer Interaction (HCI) has emerged as a pivotal discipline within software engineering and user experience (UX) design. HCI seeks to bridge the gap between human cognitive behavior and technological functionality by designing systems that align with users' needs, expectations, abilities, and limitations.

However, ensuring usability in digital interfaces remains a challenging task, particularly when it comes to evaluating interfaces efficiently and objectively. Traditional usability evaluation methods often require significant human effort, time, and expertise, making them difficult to scale or automate. This highlights the need for solutions that can assess usability based on established principles in a more automated and systematic way.

Within this context, our project focuses on developing an automated UX heuristic evaluation system grounded in Nielsen's usability principles. Jakob Nielsen's contributions—particularly his formulation of the Ten Usability Heuristics for User Interface Design—have provided both practical and research-driven guidelines for assessing and improving digital interfaces. These heuristics—such as visibility of system status, consistency, and standards, recognition rather than recall, and error prevention—have become foundational tools used by designers and developers worldwide to ensure that usability is integrated from the earliest stages of system development.

To achieve this objective, we divided our work into three main chapters. The first chapter presents Nielsen's body of work and examines in detail his ten usability heuristics, highlighting their theoretical underpinnings and practical relevance. The second chapter is dedicated to the design phase of the proposed system, including the choice of architecture, data flow, and evaluation criteria. Finally, the third chapter focuses on the implementation of the automated evaluation system, which relies primarily on the Selenium framework to conduct web interface audits efficiently and effectively.

CHAPTER 1

Jakob Nielsen's Ten Usability Heuristics

1.1 Introduction

This chapter discusses the theoretical basis of Human-Machine Interaction (HMI) and how it is becoming more critical in the design of digital systems that are both functional and user-centered. It talks about usability as an essential part of interface design and stresses the necessity for systematic ways to test it. In this context, the chapter centers on Jakob Nielsen's Ten Usability Heuristics, which have emerged as a prominent paradigm for detecting and addressing usability challenges. We look at each heuristic to see how it may help improve the user experience and how it can help people in both academic and professional settings evaluate interfaces.

1.2 Introduction to Human-Machine Interaction

Human-machine interaction (HMI) is becoming more common, and the effects are profound and wide-ranging. This unique track on "Evaluation of Human-Machine Interaction" is meant to be a time to share, talk about, and think about ideas, problems, and chances related to this issue. In a growing number of applications, people (users) and computers (moderation systems) are interacting with each other. There is an increasing body of research addressing various issues, including the implications of defining human-machine interactions as 'human to machine,' the contexts in which this distinction is pertinent, and its influence on the design of user-centric interactions; the meaning of 'interactivity' in this context; the cognitive, perceptual, and affective pathways involved; and how these factors can be integrated into the design and assessment of human-machine interactions [21]. There are many additional difficulties that have to do with cyber safety. Safety is having safe and secure interactions with machines, including things, software, systems, and so on. It covers protecting data, mapping threats and attacks, and finding bad behavior. On the other side, security has to do with trustworthiness, setting the right expectations, and dealing with any unforeseen effects of systems that were not appropriately created or with bad intentions. In addition, responsibility, obligation, openness, and accountability are essential in all interactive contexts. There are a lot of new variables (who interprets the consequence of a machine's sentience and with what accountability) and issues that

come up that aren't present in regular human-to-human connection settings. How do we make sure that all of this is taken into account in an evaluation? Introducing HMI means new problems, but it also means new chances. Machines don't have the same limits as people. They can look at vast amounts of data in milliseconds and find patterns that no one has ever seen before. Using this can help us make more relevant arguments, learn new things, meet new people, and finally have a deeper grasp of the issue area.

1.3 Overview of Nielsen's Usability Heuristics

Nielsen's Usability Heuristics are a set of principles that guide usability in human-machine interaction. These heuristics encompass various aspects of usability, including the visibility of system status, the alignment between the system and the real world, user control and freedom, and the importance of avoiding inadvertent trapping. In what fellow, we present the ten Nielsen's Usability Heuristics with their utility on the HMI design.

1.3.1 Visibility of System Status

The first and maybe most important of Nielsen's heuristics says that users should always know what's going on in a system through timely and suitable feedback. This approach makes sure that users never feel confused or unsure about what's going on with their engagement with an interface. The system should clearly and constantly let users know what is going on so they can be sure that their actions are being processed and recognized. For easy tasks, this feedback should be quick, and for more complicated tasks that require time to do, it should be progressive.

The progress bar that shows up when you upload files to Google Drive or Dropbox is a great illustration of this heuristic in action. When a user uploads a big file, the system doesn't just go away into a black hole of doubt. Instead, it presents a visual progress bar that shows how much has been done, how much time is left, and occasionally even the upload speed. This regular input gives consumers peace of mind that the system is running properly and helps them decide whether to wait for it to finish or do something else. Modern web browsers also show this idea by showing loading indicators, spinning wheels, and progress bars when sites are loading. This makes sure that users know their request is being processed and not that the system has stopped or failed. [1][2]

1.3.2 Match Between System and the Real World

The second heuristic is that digital interfaces should employ language that the user understands and follows real-world rules instead of utilizing system-oriented language or abstract ideas. This principle says to utilize terms, phrases, and ideas that people are already acquainted with in their daily lives. It also says to display information in a way that makes sense and follows the way things work in the actual world. The interface should employ what users already know and have experienced to make the digital world feel natural and predictable.

Email programs like Gmail and Outlook are great examples of this idea since they employ recognizable metaphors from older mail systems. Users arrange their communications by putting them in “folders,” putting undesirable emails in a “trash” or “recycle bin,” and “composing” new messages using words that are very similar to how they would handle physical mail. People who know what a physical mailbox is will know what an “inbox” is right away. “Sent items” and “drafts” folders work like real-life ways to keep track of correspondence. E-commerce sites like Amazon also use real-world purchasing metaphors, such as “shopping cart," “checkout,” and “wishlist,” which are similar to how people purchase in person. These well-known ideas make it easier for consumers to understand and use the digital world by not having to acquire new words or ways of thinking [1][2].

1.3.3 User Control and Freedom

This heuristic understands that people often make mistakes or change their minds when they utilize a system. So, interfaces need to include explicit "emergency exits" that let users get out of undesired circumstances without having to go through lengthy processes. The system should let users undo and redo actions, which will give them the confidence to try new things without worrying about making mistakes that can't be fixed. This idea gives people power by letting them choose how they want to experience things instead of pushing them down set pathways.

Microsoft Word and Google Docs are great examples of this theory since users can undo and redo numerous activities with Ctrl+Z (or Cmd+Z on Mac) and Ctrl+Y. With this feature, users may try out different formatting, content modifications, and editing without worrying about making mistakes that can't be fixed. Many modern apps also employ breadcrumb navigation algorithms that show users where they are in a hierarchy and make it easier to get back to prior levels. E-commerce sites show this idea by including "Continue Shopping" buttons on checkout pages that enable customers to go back to exploring without losing their cart contents and "Save for Later" choices that let users take products out of their basket without losing them forever. [3] [1]

1.3.4 Consistency and Standards

The fourth heuristic says that users shouldn't have to guess if various words, circumstances, or actions indicate the same thing in an interface or different portions of a system. By making sure that comparable things act in similar ways and that established rules are followed, consistency makes it easier to learn and lessens the cognitive burden. This means that an application must be consistent with itself and with the rules and standards of the platform and the industry. Apple's iOS interface is an excellent example of this idea because all of its native apps employ the same design principles. The back button is always at the top-left corner of displays, swipe motions work the same way in all programs, and the tab bar at the bottom of apps always works the same way. Once users learn how to use one iOS app, they may use that expertise to use other iOS apps right away. Web browsers also keep their interface elements the same: the address bar is always at the top, bookmarks are usually stored in identical menu structures, and navigation buttons (back, forward, refresh) always show up in the same places. This consistency also applies to keyboard shortcuts. For example, Ctrl+C for copy and Ctrl+V for paste operate the same way in almost all programs, no matter who created the software or what it does. [2] [2] [4]

1.3.5 Error Prevention

This heuristic stresses the necessity of innovative design that stops issues from developing in the first place rather than only giving appropriate error signals when they arise. The optimal user experience is one where errors are avoided by carefully designing the interface, setting clear limits, and giving users helpful advice that leads them to success. This proactive approach makes things less frustrating and makes the user experience smoother and more efficient.

Form design for websites for online banking or shopping is an excellent illustration of how to avoid mistakes. These forms frequently have real-time validation that examines what users write as they type it. This means that invalid inputs, including email addresses that aren't in the correct format, passwords that don't satisfy criteria, or phone numbers that don't have enough digits, are highlighted right away. Many forms also include input masks that automatically format data as users write. For example, they could add parentheses and hyphens to phone numbers or slashes to date fields. Calendar widgets for picking a date stop users from entering erroneous dates, and dropdown boxes for picking a region or state stop users from making spelling mistakes or using acronyms that aren't conventional. Credit card forms frequently include capabilities that automatically figure out what kind of card it is

based on the number pattern and change the length of the needed security code to match. This avoids common input mistakes before they happen [5] [6].

1.3.6 Recognition Rather Than Recall

Because human memory is limited and doesn't always work, interfaces should make things, actions, and choices visible instead of making users recall things from one portion of the interface to another. Users should be able to identify activities and information instead of needing to remember them. This theory recognizes that it is considerably easier for people to recognize something than to remember it.

This notion is well illustrated by modern online browsers, which provide features like autocomplete in search boxes and form fields, bookmarks that show website names and symbols for simple recognition, and browsing history that shows previously viewed sites with titles and thumbnails. When users start typing in a search box, the browser shows a dropdown selection of past searches and recommendations. This lets users find and choose their intended query instead of having to remember and enter it exactly. Password managers like LastPass and 1Password are good examples of this since they save login information and fill it in automatically when users visit websites. This means that users don't have to remember complicated passwords while still keeping their accounts safe. E-commerce sites use this idea in features like "recently viewed items," "frequently purchased products," and visual shopping carts that show product images next to their names. This makes it easier for users to remember their choices without having to remember specific product details or catalog numbers [6] [7].

1.3.7 Flexibility and Efficiency of Use

This heuristic takes into account that users have diverse degrees of experience and different ways they want to engage with technology. Interfaces should include many methods to do things so that both new and experienced users may utilize them. This will let experienced users work faster while still helping those who need more help. The system should provide shortcuts and customization choices that make it easier for regular users to use without making it harder for new users to get around. Adobe Photoshop is an excellent example of this approach since it lets users complete most jobs in several ways to fit their needs and ability levels. Beginners can use all the features through well-organized menu systems with clear labels and categories, while advanced users may use a lot of keyboard shortcuts to speed up their work. The program also lets you change the toolbars and

workspaces to fit your needs or the needs of a particular activity. For example, users may go to the crop tool by clicking on the toolbar, going to the Image menu, or using the "C" key. They can also choose which tools they want to see on their toolbar based on how they usually operate. Modern text editors like VS Code also have command palettes that let advanced users instantly access any function by entering its name. For less experienced users, they still have classic menu systems. The program also lets users customize it a lot by using themes, extensions, and keyboard shortcuts to make their work environment as efficient as possible. [7]

1.3.8 Aesthetic and Minimalist Design

The eighth heuristic emphasizes that interfaces should not contain irrelevant or rarely needed information, as every extra unit of information competes with relevant information and diminishes its relative visibility. Good design focuses on essential elements and eliminates visual clutter that can distract users from their primary goals. This principle advocates for a clean, purposeful design that guides users' attention to the most essential elements and actions.

Google's search homepage has become the iconic example of minimalist design in web interfaces. The page features primarily white space with the Google logo, a single search input field, and two simple buttons ("Google Search" and "I'm Feeling Lucky"). This stark simplicity eliminates distractions and focuses users' attention entirely on the primary task of entering search queries. The design deliberately avoids the cluttered approach of early search engines that filled their homepages with news, advertisements, and multiple search categories. Similarly, Apple's product pages demonstrate aesthetic minimalism by using abundant white space, high-quality product photography, and carefully chosen typography to highlight key product features without overwhelming visitors with excessive technical specifications or promotional content. The clean design allows the product to be the hero of the page while providing essential information in a digestible, visually appealing format that guides users naturally through the decision-making process.[1] [8]

1.3.9 Help Users Recognize, Diagnose, and Recover from Errors

When errors occur, even when steps have been taken to stop them, the system should show error messages that are clear, don't use technical codes, point out the exact problem, and offer helpful remedies. Instead of just saying what went wrong, error messages should assist users in finding a solution. This philosophy recognizes that mistakes happen and aims to make the recovery process as easy and informative as feasible.

Modern online apps like GitHub have well-designed error handling that gives users clear, actionable error notifications when they run into issues. When a user tries to access a repository that doesn't exist, GitHub doesn't just show a generic "404 Error" page. Instead, it shows a friendly message that says the page couldn't be found, suggests checking the URL for typos, and gives links to search for repositories or go back to the homepage. The error page keeps the site's branding and appearance, which makes the error feel less startling and more like part of the whole experience. In the same way, form validation problems on well-designed websites tell you precisely what you need to fix. Instead of just saying "Invalid input," good error messages tell users exactly what's wrong ("Password must contain at least eight characters, including one uppercase letter and one number") and often show examples of acceptable formats. This lets users fix their mistakes quickly and learn the system requirements for future interactions. [4] [9]

1.3.10 Help and Documentation

Ideally, interfaces should be straightforward to use without any help or documentation. However, it may be essential to include help and documentation that is easy to find, focuses on the user's goal, identifies specific actions to take, and isn't too big. Good help systems are built into the interface, searchable, and based on the user's context. They are not separate, long manuals that users have to read on their own. The documentation should be able to guess what users will need and offer help when they need it without getting in the way of the main workflow.

Slack is an excellent example of how to integrate help and documentation. Its contextual help system and complete yet easy-to-use support resources are great examples. The app has in-line help tooltips that pop up when users hover over interface components. These tooltips explain how things work without making users go to separate help sites. When users are adding new features like channels or integrations, Slack gives them step-by-step instructions right on the UI, emphasizing the buttons and fields that are important as they go through the setup process. The site also has a help center that you can search through. It has articles that are focused on specific tasks and contain screenshots, video demonstrations, and code samples as needed. Also, Slack uses progressive disclosure in its help system. It starts with short explanations and gives users the option to "Learn more" if they need more information. This way, users who only need basic help won't be overwhelmed, and users who need more detailed help will still get it. [7]

1.4 The Importance of Heuristics in HCI

Nielsen's usability heuristics are very important for how people and computers interact because they provide you with a systematic way to test and improve user interfaces across a wide range of platforms and applications. These concepts are basic rules that assist in connecting human psychology with technology. They make sure that digital systems are built with human strengths and weaknesses in mind. These heuristics are an essential starting point for first-year HCI students who want to learn how people interact with technology and what makes their experiences good or bad. The heuristics condense years of study and hands-on knowledge into useful rules that can be used right away in design and assessment processes. This makes them very useful tools for both new and seasoned professionals.

The continued significance of these heuristics in the swiftly changing technology environment illustrates their intrinsic link to human cognitive functions and behavioral trends that persist despite technological progress. These concepts still work for making user-centered designs, whether you're looking at typical desktop software, a mobile app, a voice interface, or new technologies like augmented reality systems. For students just starting in HCI, learning these heuristics gives them a strong theoretical base while also helping them improve their design critique and assessment abilities. Students who understand these concepts may think methodically about problems with user experience and come up with solutions that put people's needs ahead of technological limitations. This helps them build a user-centered attitude, which is vital for success in technology design and development jobs [8].

1.5 Conclusion

In this chapter, we have presented Nielsen’s ten usability heuristics as a practical and widely adopted framework for evaluating user interfaces. These principles go beyond theory, offering concrete guidance for creating intuitive and compelling digital experiences. In the next chapter, we will present the design phase of our work concerning the proposition of an automated evaluation tool based on Nielson’s guidelines.

CHAPTER 2

Conceptual Design and Methodology of the UX Audit Tool

2.1 Introduction

This chapter aims to present the overall design of the system that fulfills the project requirements and demonstrates how to transform functional requirements into technical models that facilitate the programming and implementation process. The design phase plays a crucial role in ensuring clarity of functions and relationships between system components, providing a solid foundation for development.

System design is fundamental to successful software development as it bridges the gap between requirements analysis and actual implementation. It provides developers with a clear roadmap of how the system should be structured, how components interact with each other, and how data flows throughout the application.

This chapter presents an overview of the models used in design, particularly Unified Modeling Language (UML) diagrams, to illustrate workflow and system architecture. These visual representations help stakeholders understand the system's behavior and structure before implementation begins, reducing development time and potential errors.

2.2 System Modeling Using UML

Unified Modeling Language (UML) serves as the standard notation for modeling software systems. It provides various diagram types that capture different aspects of system design, from user interactions to internal structure and behavioral patterns. In this project, we utilize three primary UML diagrams to model our web analysis system comprehensively.[9]

2.2.1 Use Case Diagram

The Use Case Diagram represents the functional requirements of the system by illustrating the interactions between actors (users) and the system's functionalities. It provides a high-level view of what the system does without detailing how it accomplishes these tasks.

This diagram is particularly valuable for stakeholders as it presents the system's capabilities in an easily understandable format. It helps identify the main functionalities that the system must provide and establishes the scope of the project. [11].

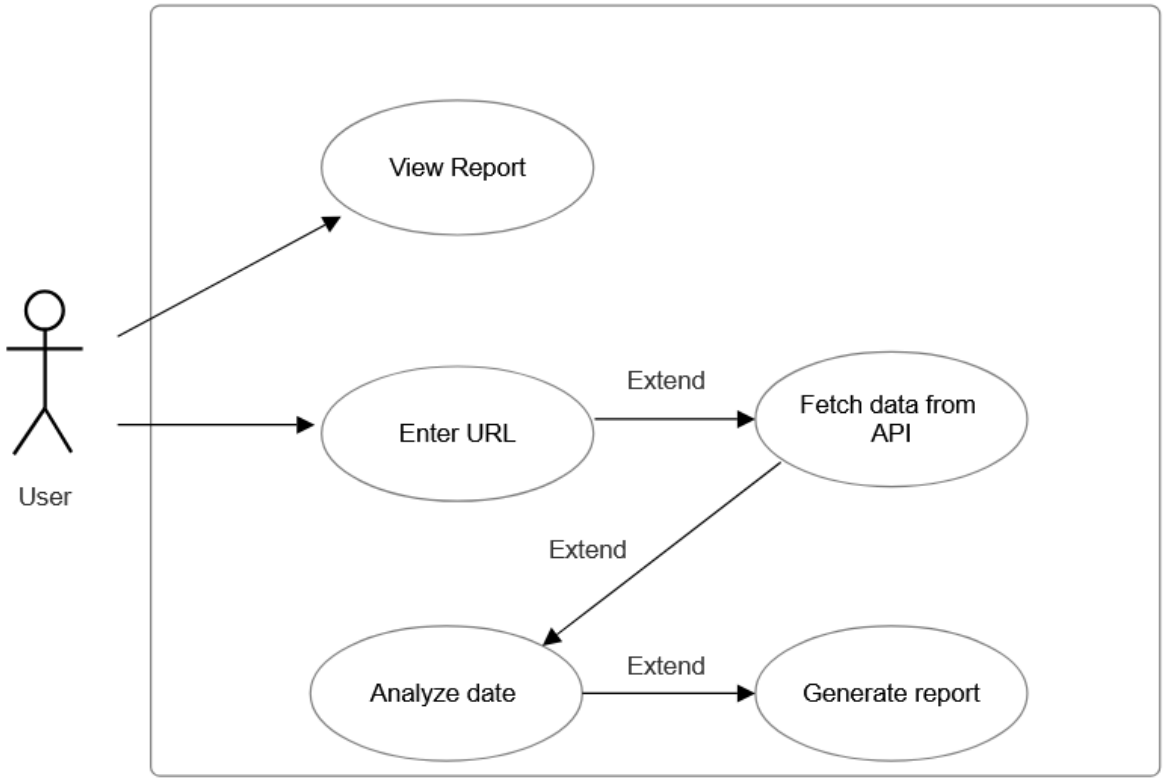


Figure 2.1 Use case diagram to represent the functional requirements

- Enter URL: The primary actor (user) initiates the process by inputting a website URL that they wish to analyze. This represents the entry point for all system operations

- Fetch Data from API: Once the URL is provided, the system automatically retrieves relevant data from external APIs. This process is transparent to the user but essential for gathering the information needed for analysis.

- Analyze Data: The system processes the retrieved data using various analytical algorithms to extract meaningful insights about the target website.

- Generate Report: Based on the analysis results, the system compiles a comprehensive report that presents the findings in a structured and readable format.

- View Report: The user can access and review the generated report through the system's interface, completing the analysis workflow.

2.2.2 Class Diagram

The Class Diagram illustrates the static structure of the system by showing classes, their attributes, methods, and the relationships between them. This diagram is crucial for understanding the system's architecture and serves as a blueprint for implementation.

In object-oriented design, classes represent the building blocks of the application. Each class encapsulates related data and behavior, promoting code reusability and maintainability. The relationships between classes define how they collaborate to achieve the system's objectives.

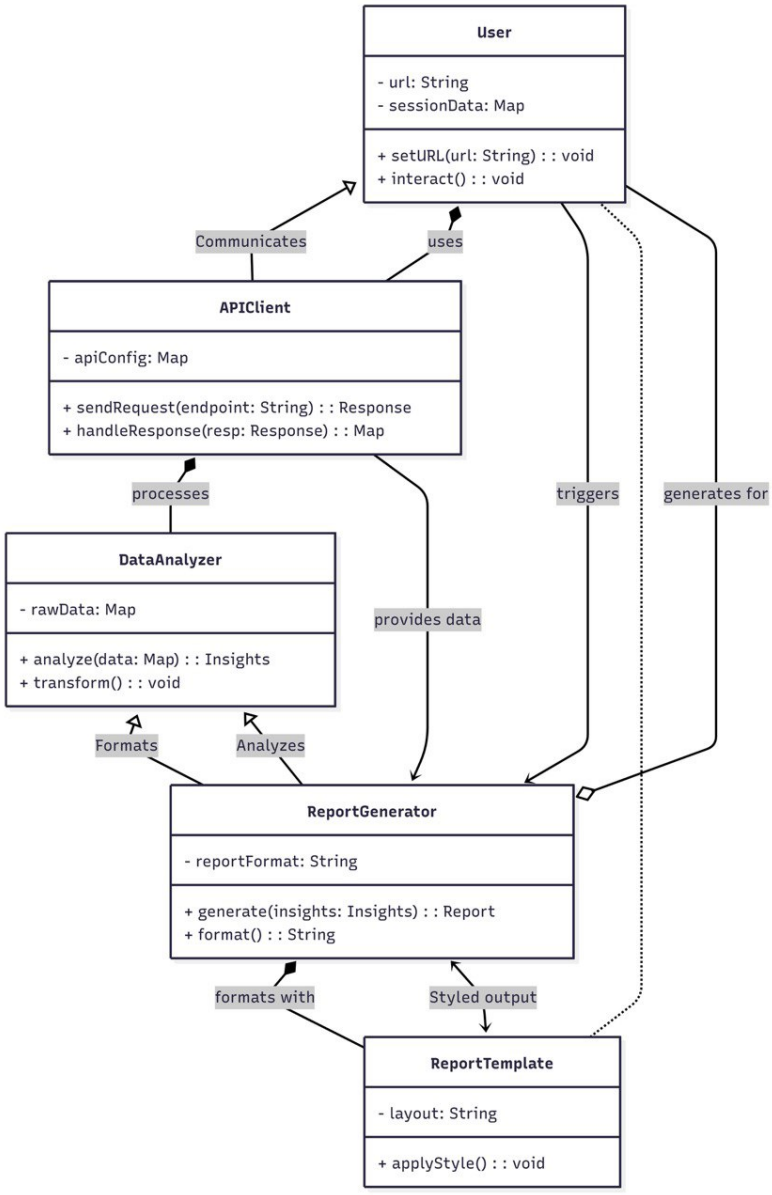


Figure 2.2 The Class Diagram illustrates the static structure

Class Analysis:

- User Class: Represents the system user with attributes for storing the input URL and methods for interacting with the system. This class manages user-related operations and maintains user session data Represents the system user with attributes for storing the input URL and methods for interacting with the system. This class manages user-related operations and maintains user session data. It serves as the entry point for initiating tasks like API communication and report generation, ensuring user-specific data is passed correctly through the system.

- APIClient Class: Handles all external API communications. It encapsulates the logic for making HTTP requests, handling responses, and managing API-specific configurations. This separation of concerns ensures that API-related changes don't affect other system components. It acts as an intermediary between the user and data layers, abstracting away the complexity of external service interactions.

- DataAnalyzer Class: Responsible for processing and analyzing the data retrieved from APIs. It implements various analytical algorithms and transforms raw data into meaningful insights. This class can be extended to support different types of analysis, such as statistical processing, pattern recognition, or AI-based interpretations. It is a central logic component that turns raw responses into usable, decision-support information.

- ReportGenerator Class: Takes the analysis results and formats them into a user-friendly report. It handles the presentation logic and ensures that the output is well-structured and easily interpretable. This class separates the concern of data visualization and formatting from core processing, allowing custom report templates or export formats such as PDF, HTML, or JSON to be implemented easily.

- ReportTemplate Class: Defines the structure and layout of the final report. It includes attributes like layout styles, themes, or formatting rules, and provides methods to apply consistent styling to the report content. The ReportTemplate class works closely with ReportGenerator, offering a reusable and customizable way to present analytical results, improving maintainability and consistency of the output.

Class Relationships:

The relationships between classes demonstrate the flow of data and control throughout the system. The User class initiates requests that flow through the APIClient to the DataAnalyzer, and finally to the ReportGenerator, creating a clear chain of operations.

• The APIClient class inherits from the User class, meaning it reuses and extends the functionalities of User.

• The User class has a composition relationship with APIClient, indicating that each User contains and manages an APIClient instance.

• The APIClient class composes a DataAnalyzer instance, tightly coupling analysis with API data flow.

• The ReportGenerator class inherits from DataAnalyzer, enabling it to reuse analysis logic while focusing on formatting and presentation.

• The ReportGenerator class uses the User class through aggregation, as it generates reports specifically for users but does not own them.

• Both APIClient and User classes have associations with ReportGenerator, as they either trigger or provide input for report creation.

• The ReportGenerator class composes a ReportTemplate, which defines the structure and layout of the generated report.

• A dependency exists between ReportTemplate and User, indicating that templates may refer to user-specific information during formatting.

• There is also a bidirectional association between ReportGenerator and ReportTemplate, as they both interact to produce the final report..

2.2.3 Sequence Diagram

The Sequence Diagram illustrates the dynamic behavior of the system by showing how objects interact over time. It provides a detailed view of the message flow between different components during the execution of a particular use case.

This diagram is handy for understanding the temporal aspects of system behavior and helps identify potential bottlenecks or optimization opportunities in the interaction flow.

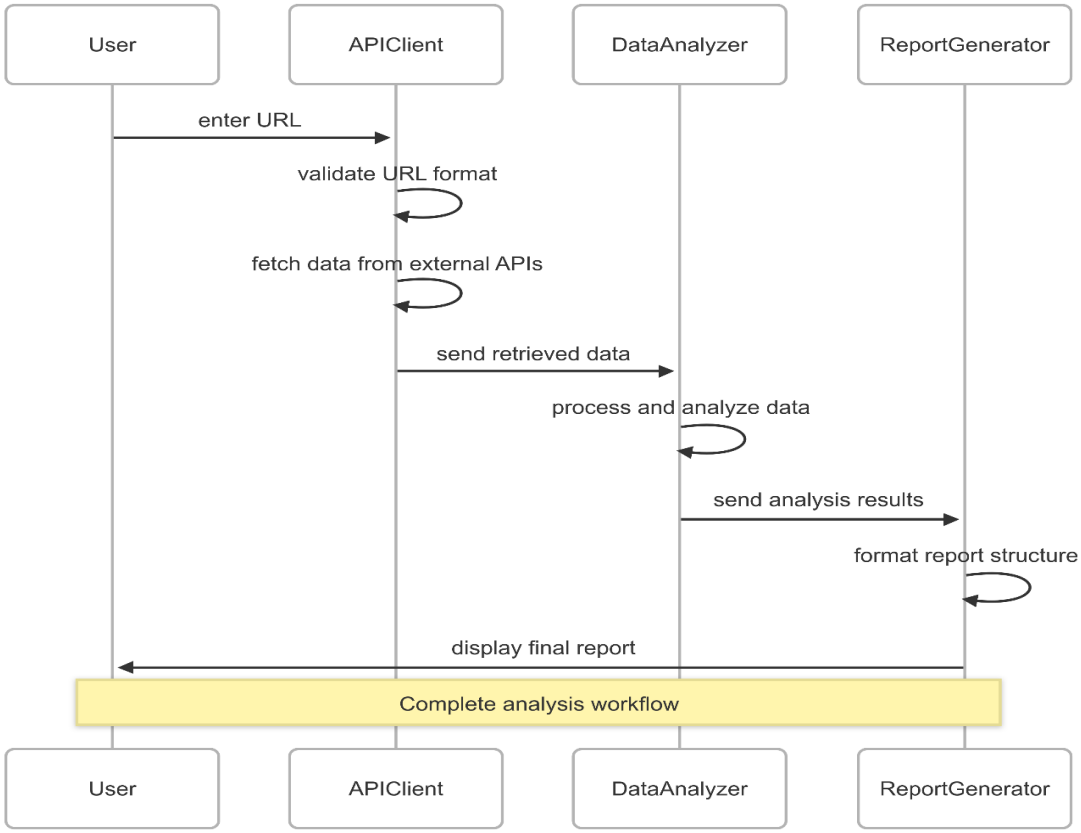


Figure 2.3 Sequence Diagram illustrates the dynamic behavior

Sequence Analysis:

The diagram shows the step-by-step interaction between system components:

1. The user initiates the process by entering a URL
2. The APIClient receives the request and fetches data from external APIs
3. Retrieved data is passed to the DataAnalyzer for processing
4. Analysis results are forwarded to the ReportGenerator
5. The final report is presented to the user

This sequence ensures proper error handling and data validation at each step, maintaining system reliability and user experience.

2.3 User Interface Design (IHM)

The user interface design places a strong emphasis on delivering an intuitive, user-friendly, and highly efficient experience for individuals interacting with the web analysis system. The goal is to ensure that users, regardless of their technical expertise, can easily navigate the system and perform tasks without confusion or extensive training. To achieve this, the interface is designed with simplicity in mind, featuring clean layouts, clear labels, and logical workflows that guide users through each step of the analysis process. At the same time, it incorporates powerful and comprehensive tools that allow more advanced users to conduct in-depth data analysis, customize parameters, and generate detailed reports. This balance between simplicity and functionality ensures that the interface meets the needs of both novice users seeking fundamental insights and experienced analysts requiring advanced capabilities. Additionally, visual aids such as charts, graphs, tooltips, and responsive feedback enhance user engagement and understanding, making the overall experience both productive and enjoyable. [12]

2.3.1 Interface Components

Input Interface: The primary interface consists of a clean, minimalist design featuring a prominent input field for URL entry. The interface includes validation feedback to ensure users enter properly formatted URLs and provides clear instructions for system usage.

Loading Interface: During data retrieval and analysis, users see a progress indicator that provides feedback on the current operation status. This interface includes estimated completion times and the ability to cancel long-running operations if necessary.

Report Display Interface: The results are presented through a structured, responsive layout that adapts to different screen sizes. The interface organizes information hierarchically, with summary statistics prominently displayed and detailed metrics available through expandable sections.

2.3.2 Technical Implementation

The user interface is implemented using Next.js, a React-based framework that provides server-side rendering capabilities and optimized performance. Next.js offers several advantages for this project:

- Server-Side Rendering (SSR): Improves initial page load times and SEO performance
- Built-in Routing: Simplifies navigation between different application views
- API Routes: Enables backend functionality within the same framework
- Optimized Bundle Splitting: Reduces JavaScript payload and improves performance

The styling is implemented using modern CSS techniques, including CSS Grid and Flexbox for responsive layouts. The interface follows accessibility guidelines to ensure usability for users with diverse needs. [12]

2.3.3 User Interaction Flow

Users interact with the system through a three-step process:

1. Input Phase: Users enter the target URL and configure analysis parameters
2. Processing Phase: The system provides real-time feedback during data collection and analysis
3. Results Phase: Users can explore the generated report, export data, or initiate new analyses

2.4 Data Flow

The data flow within the system follows a linear progression from user input to final report generation. Understanding this flow is crucial for identifying potential optimization points and ensuring data integrity throughout the process.

2.4.1 Data Flow Stages

Input Validation: User-provided URLs undergo validation to ensure they are properly formatted and accessible. This stage prevents errors in subsequent processing steps and provides immediate feedback to users.

Data Retrieval: The system makes authenticated requests to various APIs to gather comprehensive information about the target website. This includes technical metadata, performance metrics, and structural analysis.

Data Processing: Retrieved data is normalized and processed through analytical algorithms. This stage transforms raw API responses into meaningful metrics and insights.

Report Generation: Processed data is formatted into a structured report with visual elements, charts, and detailed explanations. The report is optimized for both digital viewing and printing. [14]

2.4.2 Data Quality Assurance

The system implements multiple layers of data validation to ensure accuracy and reliability:
- Input sanitization prevents malicious data injection
- API response validation ensures data completeness
- Cross-reference verification confirms data consistency
- Error handling mechanisms provide graceful failure recovery [15].

2.5 Special Design Considerations

2.5.1 Security Considerations

Data Privacy: The system handles potentially sensitive website information, requiring careful attention to data privacy and security. All data transmissions use HTTPS encryption, and temporary data is securely disposed of after processing.
API Security: External API communications are secured through authentication tokens and rate limiting to prevent abuse. The system implements proper error handling to avoid exposing sensitive information in error messages.
Input Validation: All user inputs undergo rigorous validation to prevent injection attacks and ensure system stability.
[15] [16]

2.5.2 Performance Considerations

Asynchronous Processing: The system utilizes asynchronous programming patterns to handle multiple API requests concurrently, reducing overall processing time.
Caching Strategy: Frequently accessed data is cached to reduce API calls and improve response times. The caching system includes proper invalidation mechanisms to ensure data freshness.
Resource Optimization: Next.js provides built-in optimization features including image optimization, code splitting, and compression that enhance system performance.[16]

2.5.3 Scalability Considerations

Modular Architecture: The system is designed with modular components that can be independently scaled or modified without affecting other parts of the application.

Extensible Analysis Framework: The Data Analyzer class is designed to accommodate new analysis types and data sources through a plugin-like architecture.
Database Integration: While the current implementation uses in-memory processing, the architecture supports integration with databases for persistent storage and advanced querying capabilities.

2.6 Conclusion

This chapter has presented a comprehensive design for the web analysis system, covering both structural and behavioral aspects through UML modeling. The design emphasizes modularity, maintainability, and user experience while addressing critical concerns such as security and performance.
The use of Next.js as the implementation framework provides a solid foundation for building a responsive, performant web application that meets modern development standards. The modular architecture ensures that the system can evolve to meet changing requirements and incorporate new analytical capabilities.
The design phase establishes clear guidelines for implementation, reducing development risks and ensuring that the final product meets stakeholder expectations. The next chapter will detail the implementation and testing phases, demonstrating how this design translates into a functional application.

CHAPTER 3

Technical Development of the Automated UX Audit System

3.1 Introduction

This chapter presents a comprehensive overview of the system development process, including the working environment, adopted tools, development phases, and the final outputs through visual representations and fundamental code structures. The implementation phase represents the practical realization of the theoretical framework established in the previous chapters, transforming conceptual designs into a functional web analysis system.

3.2 Development Environment

3.2.1 Software Tools

The development of this web analysis system required a carefully selected set of software tools to ensure efficiency, reliability, and maintainability. The following table outlines the primary software components utilized:

Table 3.1 Software used Table

Tool	Version	Purpose
Visual Studio Code	Latest	Primary code editor and development environment
Node.js	18.x	JavaScript runtime environment for server-side operations
Axios	1.x	HTTP client library for API data retrieval
Mermaid.js	10.x	Diagram generation and flowchart visualization
HTML5/CSS3/ES6	Latest	Frontend technologies for user interface development
Chart.js	4.x	Data visualization and chart generation

These tools were selected based on their compatibility, community support, and ability to handle the specific requirements of web data analysis and reporting.[10][11][12]

3.2.2 Hardware Specifications

The development was conducted on a system with the following hardware specifications:

Table 3.2 Hardware Specifications

Component	Specification
Processor	Intel Core i5-8400 2.8GHz
Memory	8GB DDR4 RAM
Storage	256GB SSD
Operating System	Windows 10 Professional
Network	Broadband internet connection (50 Mbps)

These specifications provided adequate performance for development, testing, and data processing operations throughout the implementation phase.

3.3 Implementation Phases

3.3.1 Data Retrieval from API

The first critical phase involved establishing a robust connection with various web APIs to retrieve analytical data. This process required implementing secure authentication mechanisms and handling different response formats. The system was designed to accommodate multiple API endpoints, ensuring flexibility in data source selection.

The implementation focused on creating asynchronous data fetching mechanisms that could handle large datasets without blocking the user interface. Error handling procedures were integrated to manage network timeouts, invalid responses, and API rate limiting.[15][17].

3.3.1 Data Processing and Analysis

Following successful data retrieval, the system implements sophisticated data processing algorithms to analyze web content, performance metrics, and user engagement patterns. This phase involved developing custom parsing functions to extract meaningful insights from raw data.

The analysis engine incorporates multiple analytical approaches, including statistical analysis, pattern recognition, and trend identification. Data validation procedures ensure accuracy and reliability of processed information before generating reports.[19][20].

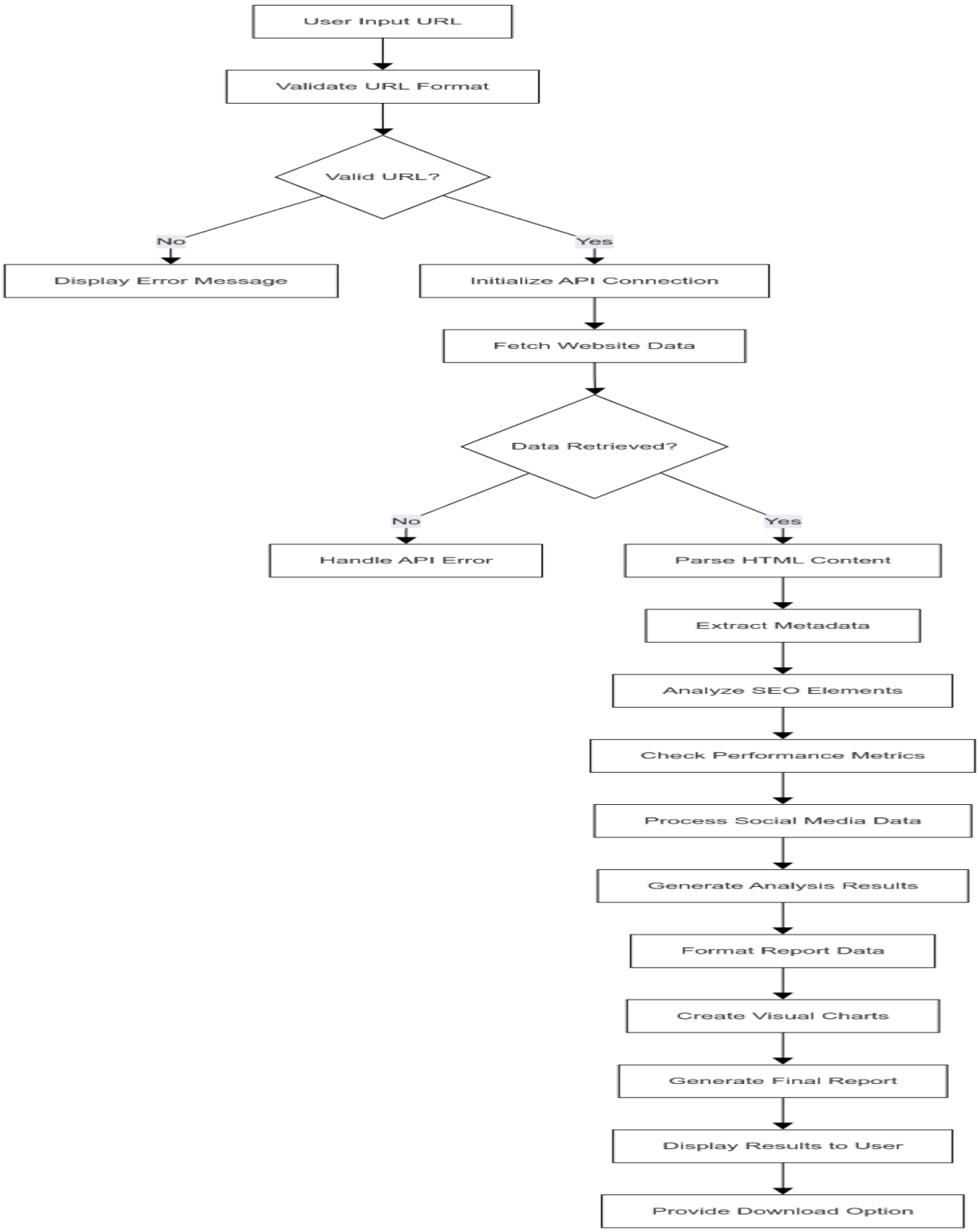


Figure 3.1 Data Processing and Analysis

3.3.2 Report Generation System

The report generation module transforms processed data into comprehensive, professionally formatted documents. This system supports multiple output formats, including HTML and PDF, allowing users to choose their preferred format based on specific requirements.

The reporting engine incorporates dynamic template generation, ensuring consistent formatting while accommodating variable data structures. Advanced styling options provide visually appealing presentations with interactive elements for enhanced user experience [17] [18].

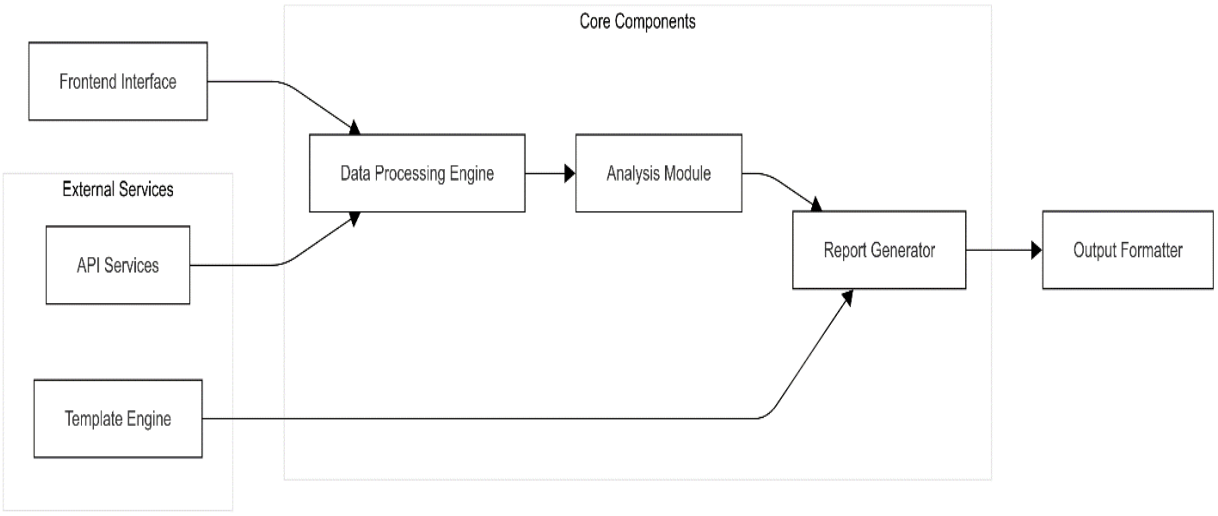


Figure 3.2 System Architecture Diagram(core components)

3.4 Graphical User Interface

The user interface design prioritizes simplicity and functionality, providing an intuitive experience for users of all technical levels. The interface follows modern web design principles, incorporating responsive design elements to ensure compatibility across different devices and screen sizes.

The main interface features a clean, minimalist design with clearly defined sections for input, processing status, and results display. Navigation elements are strategically positioned to guide users through the analysis process seamlessly.

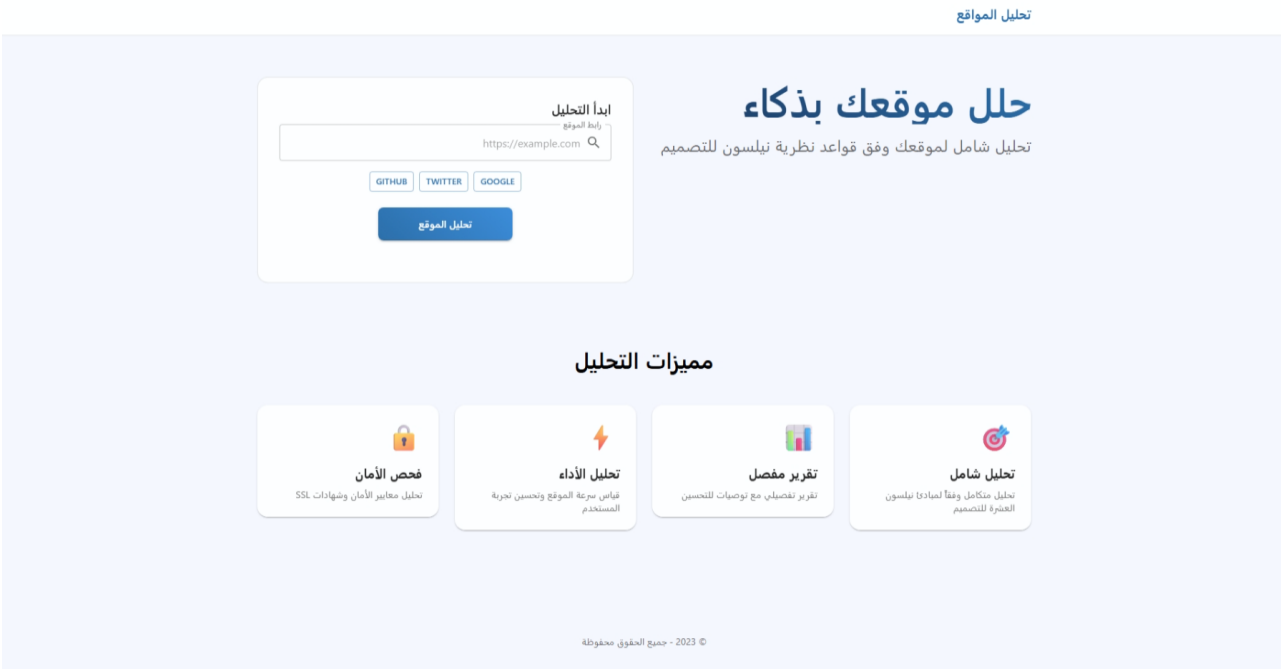
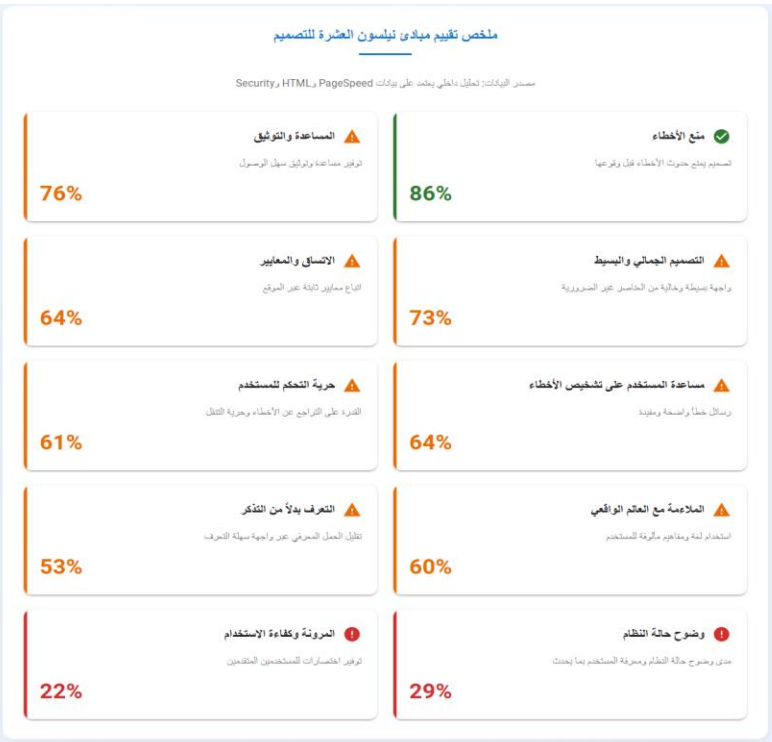
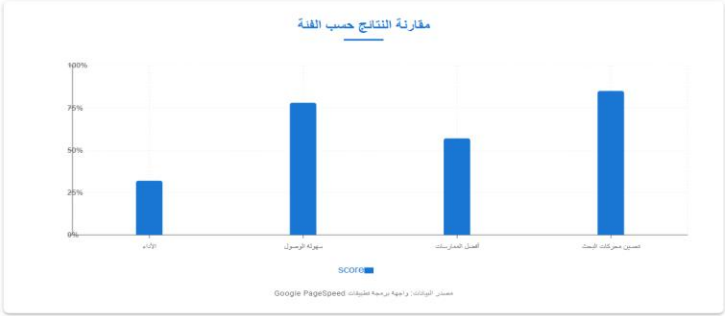
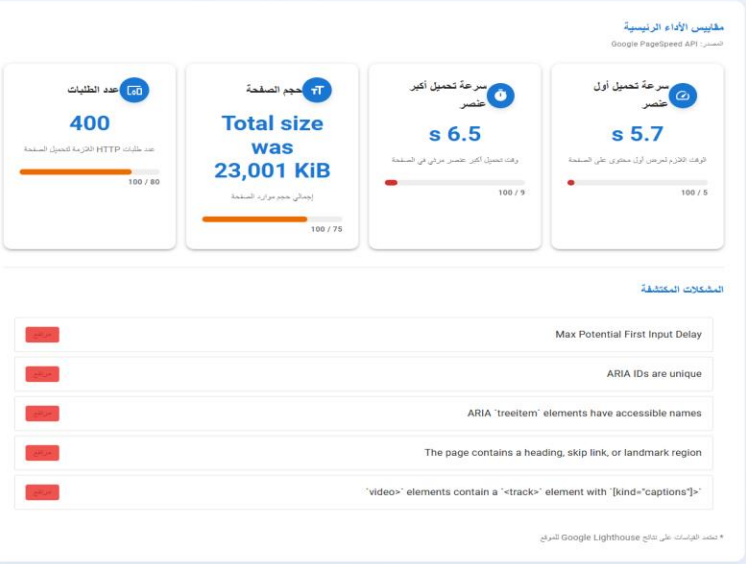


Figure 3.3 Home Page

The system provides real-time feedback during analysis operations, displaying progress indicators and status updates to keep users informed throughout the process. Results are presented in an organized, hierarchical structure with expandable sections for detailed information.


ملخص تقييم مبادئ نيلسون العشرة للتصميم
مصدر البيانات: تحليل داخلي يعتمد على بيانات PageSpeed وHTML وSecurity
منع الأخطاء
تصميم يمنع حدوث الأخطاء قبل وقوعها
86%
المساعدة والتوثيق
توفير مساعدة وتوثيق سهل الوصول
76%
التصميم الجمالي والبسيط
واجهة بسيطة وخالية من الحاسر غير الضرورية
73%
الاتساق والمعايير
اتباع معايير ثابتة عبر الموقع
64%
مساعدة المستخدم على تشخيص الأخطاء
رسائل خطأ واضحة ومفيدة
64%
حرية التحكم للمستخدم
القدرة على التراجع عن الأخطاء وحرية التنقل
61%
الملاءمة مع العالم الواقعي
استخدام لغة ومفاهيم مألوفة للمستخدم
60%
التعرف بدلاً من التذكر
تقليل الحمل المعرفي عبر واجهة سهلة التعرف
53%
وضوح حالة النظام
مدى وضوح حالة النظام ومعرفة المستخدم بما يحدث
29%
المرونة وكفاءة الاستخدام
توفير اختصارات للمستخدمين المتقدمين
22%


مقارنة النتائج حسب الفئة
100%
75%
50%
25%
0%
الأداء
سهولة الوصول
أفضل الممارسات
تحسين محركات البحث
score
مصدر البيانات: واجهة برمجة تطبيقات Google PageSpeed


مقاييس الأداء الرئيسية
المصدر: Google PageSpeed API
سرعة تحميل أول عنصر
s 5.7
الوقت اللازم لعرض أول محتوى على الصفحة
100 / 5
سرعة تحميل أكبر عنصر
s 6.5
وقت تحميل أكبر عنصر مرئي في الصفحة
100 / 9
حجم الصفحة
Total size was 23,001 KiB
إجمالي حجم موارد الصفحة
100 / 75
عدد الطلبات
400
عدد طلبات HTTP اللازمة لتحميل الصفحة
100 / 80
المشكلات المكتشفة
Max Potential First Input Delay
ARIA IDs are unique
ARIA `treeitem` elements have accessible names
The page contains a heading, skip link, or landmark region
`video>` elements contain a `<track>` element with `[kind="captions"]`
* تعتمد القياسات على نتائج Google Lighthouse للموقع

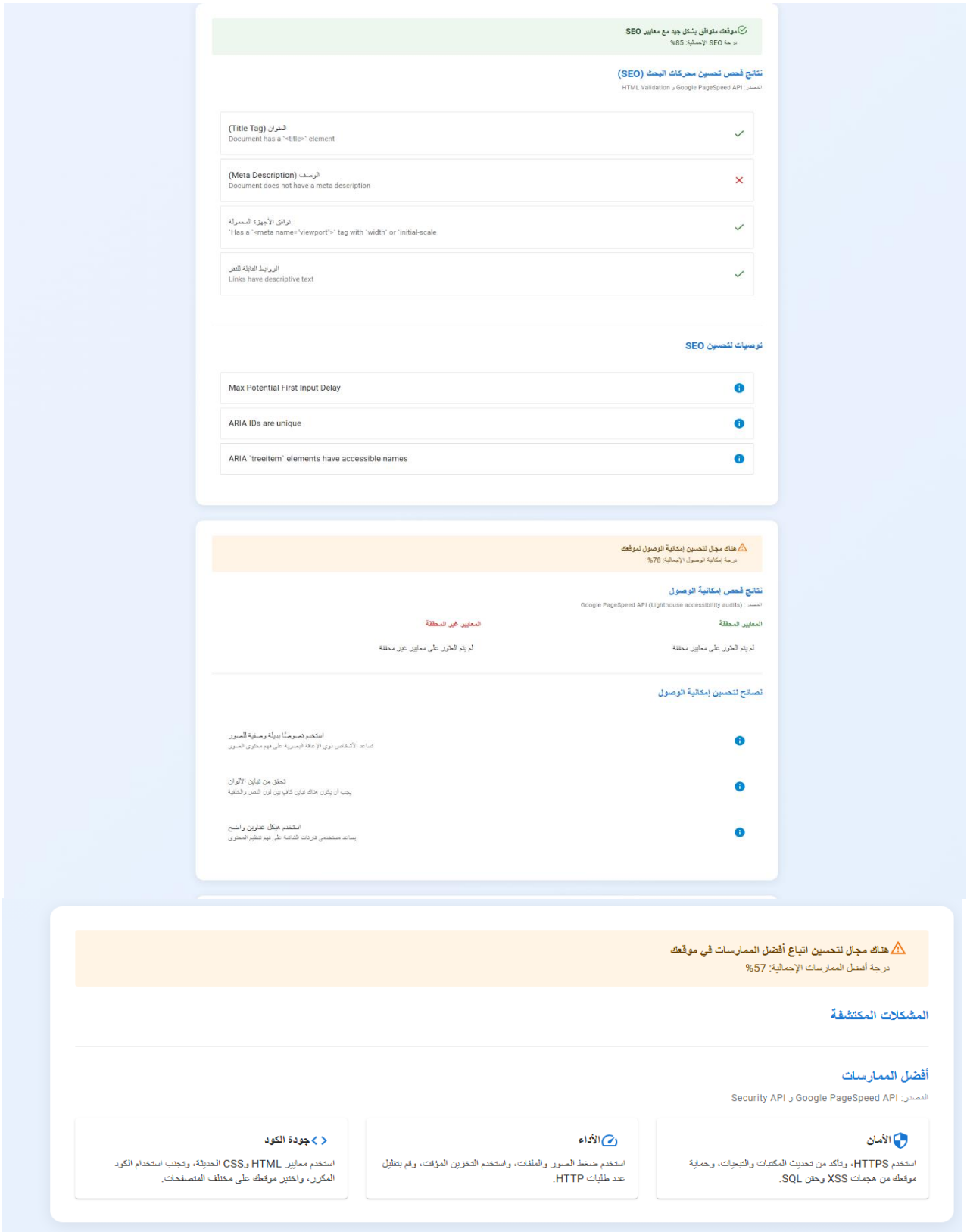


Figure 3.4 Report page

3.5 Challenges and Solutions

3.5.1 API Integration Challenges

During development, several challenges emerged related to API connectivity and data consistency. Some third-party APIs experienced intermittent availability issues, requiring the implementation of fallback mechanisms and retry logic. Rate limits imposed by certain services necessitated the development of request queuing systems to manage API calls efficiently.

3.5.2 Report Formatting Issues

Initial report generation encountered formatting inconsistencies across different browsers and devices. This challenge was addressed through extensive testing and implementation of cross-platform compatible CSS frameworks and responsive design principles.

3.5.3 Performance Optimization

Extensive dataset processing initially caused response delays, particularly when analyzing complex websites with extensive content. Performance optimization involved implementing data chunking, asynchronous processing, and caching mechanisms to reduce processing time and improve user experience.

3.6 Conclusion

This chapter has presented the comprehensive implementation process of the web analysis system, from initial development environment setup through final user interface deployment. The systematic approach adopted ensured that all planned features from Chapter 2 were successfully implemented and tested.

The development process demonstrated the effectiveness of the chosen technology stack and methodological approach. All primary objectives outlined in the planning phase have been achieved, with additional enhancements implemented to improve system reliability and user experience.

The resulting system provides a robust, user-friendly platform for web analysis and reporting, successfully meeting the requirements established in the project specification. The implementation phase confirms the viability of the proposed solution and establishes a solid foundation for future enhancements and scalability improvements.

General Conclusion

This project was a success because it combined theoretical ideas from human-computer interaction with the practical needs of online system development. This study has demonstrated that the application of Jakob Nielsen's usability heuristics during the system design and implementation stages results in significant enhancements in system usability, accessibility, and efficiency through a focus on user-centered design.

Jakob Nielsen's ten heuristics were used as a guide to design an interface that is both useful and easy to understand. For example, clear loading indicators and real-time feedback kept the system state visible, while form validation and input sanitization made sure that errors didn't happen. The idea of user control and freedom was shown through features like the ability to undo activities and controls that were clearly labeled. Consistency and standards were also followed throughout all parts of the interface and operations. These rules guided every choice, from the design of the user interface and the way reports were presented to the way the system was built and how users might use it.

The finished solution lets users do complicated things with little effort to learn, such getting data from APIs, looking at website performance indicators, and making extensive reports. This ease of interaction is a direct outcome of using tried-and-true usability concepts to build the system. The project has also included current development techniques like asynchronous data processing, flexible layouts, and progressive enhancement. This makes sure that the system is still fast, scalable, and able to work with different devices and user demands.

This study also helped us understand the real-world problems that come up when making solutions for real people to utilize. During the implementation phase, there were problems with API limits, inconsistent data formatting, cross-browser compatibility, and UI responsiveness. However, by following Nielsen's rules, notably those about mistake recovery, recognition over recall, and support and documentation, the team was able to get through these problems and make the final product more sturdy and dependable.

This project has effects that go beyond its direct technical contributions for students, teachers, developers, and designers. For students, it gives them a real-world example of how to use HCI ideas in development. It gives designers and developers a way to make usability a part of the software lifecycle. The project may be used by teachers and schools to show how important it is to make usability a priority from the start of system development.

The system sets up a strong base for future improvements. Some possible improvements are adding artificial intelligence to make analytics smarter, adding support for other languages to make

the software more accessible, adding cloud storage and collaborative tools, and making mobile versions of the app. These additions would make the system much more valuable and easy to use in a variety of situations, such as teaching, research, digital marketing, and performance audits.

In conclusion, this study confirms that Jakob Nielsen's usability guidelines are still valuable and relevant today. The system not only meets its functional goals, but it also gives users a better experience by putting them at the center of every decision, from conceptual design to final execution. This study emphasizes the essential principle that systems intended for human use must be constructed with a comprehension of human cognition, behavior, and interaction—principles that are key to usability engineering and human-centered design.

As the digital world keeps changing, future studies might look into how to apply more advanced usability testing approaches, such as machine learning-driven heuristics detection, real-time user behavior monitoring, and automated accessibility compliance checks. Also, doing usability tests with real users in different settings would provide us with helpful information that we could utilize to improve and confirm the system. Getting UX designers, developers, and HCI researchers to work together across fields will be necessary for pushing the limits of what automated evaluation tools can do.

References

[1] Nielsen, J. (n.d.). 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. Retrieved June 11, 2025, from https://www.nngroup.com/articles/ten-usability-heuristics/

[2] Nielsen, J. (n.d.). How to Conduct a Heuristic Evaluation. Nielsen Norman Group. Retrieved June 11, 2025, from https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

[3] Interaction Design Foundation. (n.d.). Jakob Nielsen. Retrieved June 11, 2025, from https://www.interaction-design.org/literature/author/jakob-nielsen

[4] UX Planet. (2019). Usability Heuristics Applied to Mobile App Design. Retrieved June 11, 2025, from https://uxplanet.org/usability-heuristics-applied-to-mobile-app-design-cf6b6573d4d1

[5] UX Collective. (2018). Nielsen Norman Heuristics Explained with UI Examples. Retrieved June 11, 2025, from https://uxdesign.cc/nielsen-norman-heuristics-explained-ux-ui-design-5452dc243f3d

[6] Adobe XD Ideas. (2020). What is Usability in UX Design? Retrieved June 11, 2025, from https://xd.adobe.com/ideas/process/user-testing/usability-heuristics-user-experience/

[7] Smashing Magazine. (2018). A Practical Guide to Usability Heuristics. Retrieved June 11, 2025, from https://www.smashingmagazine.com/2018/02/practical-guide-usability-heuristics/

[8] Cheng, K. (2021). How Jakob Nielsen's 10 Heuristics Help Usability Design. Medium. Retrieved June 11, 2025, from https://medium.com/@karenxcheng/how-jakob-nielsens-10-heuristics-help-usability-design-64f478a3c5c4

[9] W3C – Web Accessibility Initiative. (n.d.). Accessibility Principles. Retrieved June 11, 2025, from https://www.w3.org/WAI/fundamentals/accessibility-principles/

[10] Vercel. (n.d.). Next.js Documentation. Retrieved June 11, 2025, from https://nextjs.org/docs

[11] Mermaid. (n.d.). Mermaid Documentation: Generation of Diagrams and Flowcharts. Retrieved June 11, 2025, from https://mermaid.js.org/

[12] Chart.js. (n.d.). Official Documentation – Chart.js. Retrieved June 11, 2025, from https://www.chartjs.org/docs/latest/

[13] Axios. (n.d.). Axios – HTTP Requests Made Easy. Retrieved June 11, 2025, from https://axios-http.com/docs/intro

[14] React. (n.d.). Getting Started – React Docs. Retrieved June 11, 2025, from https://reactjs.org/docs/getting-started.html

[15] Google Developers. (n.d.). PageSpeed Insights API v5. Retrieved June 11, 2025, from https://developers.google.com/speed/docs/insights/v5/get-started

[16] WebAIM. (n.d.). WAVE API Documentation. Retrieved June 11, 2025, from https://wave.webaim.org/api/

[17] W3C. (n.d.). Markup Validation Service – W3C HTML Validator. Retrieved June 11, 2025, from https://validator.w3.org/nu/

[18] W3C. (n.d.). CSS Validation Service – W3C CSS Validator. Retrieved June 11, 2025, from https://jigsaw.w3.org/css-validator/

[19] URLScan.io. (n.d.). URLScan.io API Documentation. Retrieved June 11, 2025, from https://urlscan.io/docs/api/

[20] Screenshot Machine. (n.d.). Website Screenshot API. Retrieved June 11, 2025, from https://www.screenshotmachine.com/api.php

[21] A. Carrera-Rivera, D. Reguera-Bakhache, F. Larrinaga, and G. Lasa, “Exploring the transformation of user interactions to Adaptive Human-Machine Interfaces,” 2023.