International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Website Audit Tool with Security and Performance Analysis

MRS. Ayisha Khanum[1], Sakshi M C[2], Sandya N H[3], Shreyas N H[4], T P Rohit[5]

[2,3,4,5]BE Student, Department of Computer Science and Design, PES Institute of Technology and Management, Shivamogga – 577204, Karnataka, India
[1]Assistant Professor, Computer Science and Design Department, PES Institute of Technology and Management, Shivamogga– 577204, Karnataka, India.

ABSTRACT
Website Audit Tool is an intelligent web-based system for assessing and improving the overall health of websites. It provides detailed performance and security insights by merging areas that traditional tools usually assess separately. The backend is built in Python, leveraging power libraries and APIs to inspect page load speed, SEO factors, broken links, and SSL/TLS configurations. It also carries out essential security checks like insecure headers, outdated dependencies, and vulnerabilities that may expose websites to cyber threats. A user-friendly Flask interface lets users input URLs and instantly view results through interactive graphs, ratings, and detailed reports. Each scan generates a performance score and a security grade with actionable recommendations provided in crystal clear detail. Future upgrades will add AI-powered anomaly detection, real-time monitoring, and automated patch suggestions to enable more proactive website management and stronger protection from emerging risks.

Keywords: Website audit, performance analysis, security assessment, SEO, SSL/TLS, vulnerability detection, web optimization.

1. INTRODUCTION

Websites today suffer from slow performance, hidden vulnerabilities, and fragmented maintenance practices that result in poor user experiences, lower search rankings, and an overall greater exposure to cyber threats. Traditional auditing tools address only parts of the problem by focusing either solely on performance or solely on security, yielding incomplete assessments. The Website Audit Tool overcomes these limitations by acting as an intelligent, unified system that evaluates both security and performance under one single platform.

More than a scanning utility, the Website Audit Tool automates serious analysis tasks such as the evaluation of load time, SEO inspection, link integrity checks, SSL/TLS validation, and the detection of vulnerabilities. The tool lessens manual testing efforts by streamlining these processes, provides quicker insights, and helps developers and administrators target improvements. Instant dashboards, detailed metrics, and practical recommendations are given to users to help enhance site reliability, security posture, and efficiency in general.

The tool also serves as a companion in continuous monitoring, assisting in detecting aberrant behaviors, performance trends, and emerging threats. Developers receive real-time security alerts, outdated dependency detection, and best-practice recommendations for hardening configurations. Website owners are able to obtain clarity due to visual reports, risk scores, performance grades, and action steps that have been simplified. Built upon Python in addition to advanced APIs and automated analysis techniques, this system keeps on learning and adapting with each day to find new bottlenecks of performance and security.

This project is anchored on key research into web performance and cybersecurity. Work by Wang et al. (2020) and Bouch et al. (2021) influenced the performance measurement approaches and usability of the tool. Research by Alshamrani et al. (2022) and Gupta et al.

(2023) guided the integration of vulnerability detection, secure configuration checks, and threat modeling. SEO evaluation features were inspired from analyses by Kumar et al. (2021). At the same time, unified dashboards and automated insights draw from modern web-analytics frameworks explored by Zhang et al. (2024). All these have combined to shape the intelligence, structure, and analytical depth of this Website Audit Tool.

2. LITERATURE REVIEW

Website auditing has become an integral part of maintaining the reliability, speed, and security of modern web applications. While the research landscape places great emphasis on two pillars of website health-performance optimization and security assessment-it also iterates over the fact that unified systems that assess both in tandem are much needed. Sharma's work [1] evidences the efficacy of automated security analysis using Python; it also establishes the fact that scripting-based scanners can securely identify insecure headers, out-of-date components, and high-risk vulnerabilities. Parallel to this, Lee et al. [2] introduced a unified health metric that merges performance deterioration with vulnerability severity, reinforcing the need to have auditing methodologies that adopt an integrated approach.

Research in backend architectures and optimization also shapes modern auditing tools. Shaw and Wong [3] compare backend frameworks and show that server-side design decisions directly affect load speed and responsiveness. Studies such as that of Kumar and Banerjee [6] further establish that automated performance evaluation tools can provide accurate metrics for real-world applications. Complementing this, Park and Cho [8] explore the relationship between SEO optimization and performance scoring, showing how faster websites consistently rank better within search ecosystems.

Security-driven research gives more depth to auditing practices as well. Ramires et al. [4] propose a multi-agent system, KAVE, which detects vulnerabilities based on knowledge-based reasoning. Nunes et al. [5] advocate for a hybrid scanning strategy by combining static and dynamic analysis, which helps in bringing more accuracy in the detection. Machine learning-based vulnerability detection, proposed by Huang et al. [7], furthers this area of exploration through the identification of dynamic web applications patterns. Patel and Srinivasan [9] contributed solutions that are performance-oriented, such as intelligent caching and prioritization, to improve load speeds, while Al-Hassan and Qureshi [10] presented an auditing framework that is security-driven by integrating automated scanners with threat modeling techniques.

In summary, while the individual studies all point towards either performance or security, reviewed literature indicates that modern web ecosystems demand an auditing system that treats the two aspects holistically. Altogether, these works justify the development of the Website Audit Tool, integrating

performance scoring, detection of vulnerabilities, and automated insights into one unified platform; it will support developers and organizations in keeping their websites fast, secure, and resilient.

3. METHODOLOGY

The undertaken project includes a structured multistage pipeline that is designed to perform automated website performance and security analysis. Multiple components of the system, from input of website URLs and audit data to visualization in real time and reporting, are integrated into the architecture to provide highly accurate and actionable results for web developers and organizations.

3.1 Overview of System Architecture

A modular and sequential approach is followed in the architecture, ensuring that the performance and security analysis phases work quite in sequence.

The main phases involved in this regard are:

Input Collection: The input collection takes in a user's input website URL to be analyzed.

Performance Analysis: Evaluates the load time of web pages, response time, and overall Lighthouse score.

Security Analysis: Scans for missing or misconfigured security headers, SSL issues, and potential vulnerabilities.

Report Generation: This integrates all the audit data into a visual summary and provides a downloadable PDF report.

Dashboard Visualization: It provides real-time performance scores, security ratings, and improvement recommendations on an interactive dashboard.

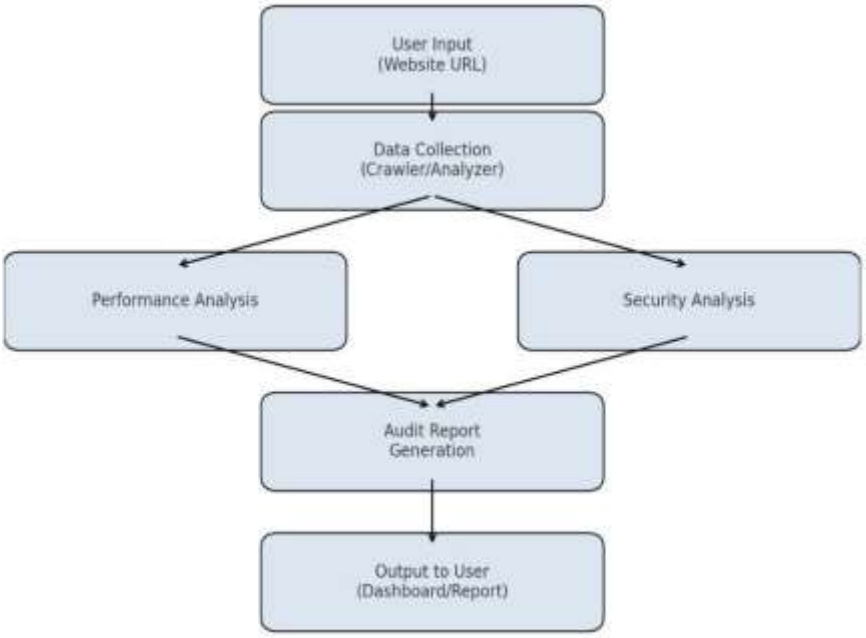

Figure 1: Block Diagram of Website Audit Tool

Figure 1 depicts the entire working of the Website Audit Tool. It starts with a website URL being input by the user for auditing. In return, the system performs both the performance and security analyses, followed by detailed reporting and suggestions.

Web Input: The user provides the input for the target website link through the web interface using a secure form.

Performance Evaluation: The backend will trigger performance audits through pre-configured modules

in Java or API integrations, which evaluate response time, resource optimization, and Lighthouse scores.
Security Analysis: The system performs a lightweight security scan to identify SSL/TLS configuration, HTTP security headers, and indicators of vulnerabilities.

Data Aggregation: Data collected is formatted and temporarily stored in a PostgreSQL database for structured access.

Display on Report and Dashboard: The dynamically created results are showcased on the dashboard, highlighting key insights, scores, and recommendations for improvement.

This structured architecture makes it easier for technical and non-technical users to understand the health of the website and implement timely optimizations.

3.2 Data Preparation and Analysis Setup

This step ensures that all the input data, including the website metrics and header data, are correctly captured and processed for their accuracy and consistency.

Data Collection: The system extracts page speed, page size, response time, SSL configuration, and HTTP headers of a user's website through APIs or inbuilt auditing scripts.

Data Validation: It includes early detection of invalid or inaccessible URLs and graceful handling of error messages.

Data Cleaning and Structuring: Verbose raw audit data, sometimes nested, needs to be parsed into meaningful, structured JSON objects that can efficiently enable report generation.

Data Storage: The processed data is stored in PostgreSQL, serving as a database that manages the user details, audit history, and improvement records.

This will then ensure consistency, accuracy, and adequacy of the audit outputs for producing reliable analytical reports.

3.3 Performance and Security Evaluation

After cleaning the collected audit data, both the performance and security analyses are carried out using Java backend services by the core evaluation modules.

Performance Metrics Evaluation:

This tool measures parameters such as:

1. Page Load Time
2. First Contentful Paint (FCP)
3. Speed Index
4. Time to Interactive (TTI)
5. Resource Size and Request Count

These metrics are benchmarked against standard guidelines for web performance, such as Google Lighthouse, in order to calculate an overall performance score.

Security Audit Module:

The system will check the following:

1. Presence and correctness of SSL certificates

2. Implementation of HTTPS protocol

3. Existence of HTTP security headers: Content-Security-Policy, X-Frame-Options, etc.Detection of mixed content issues

The output of this module provides a security score indicating how strong the protection measures are for a website.

Scoring Mechanism:
The performance and security modules each produce a numeric score, which is then combined to create a composite audit score for the website.

Improvement Suggestions:
Based on the results, it automatically generates tailored recommendations, such as "Enable GZIP compression," "Add security headers," or "Reduce image sizes."

3.4 Real-Time Implementation

The final integrated system provides users with an immediate interactive experience through the dashboard and reporting features.

Dynamic Dashboard:
Shows performance and security scores with visual indicators like progress bars, charts, and colored badges.

Instant Report Generation:
The user can download the full audit report as a PDF, generated using HTML-to-PDF rendering libraries

Interactive Improvement Page:
Categorized improvement suggestions (Performance / Security), with explanations and sample code snippets for their resolution.

System Performance:
Auditing is completed in 3-5 seconds to ensure seamlessness in operation and real-time insights.
This real-time feedback loop enhances usability and provides instant, data-driven awareness of the website's condition.

3.5 Functional Requirements

User Authentication: A login page is available for users to securely access their audit history and reports.

URL Submission and Validation: An auditing website can be submitted with its URL by the user; handles invalid entry nicely.

Performance and Security Analysis: The Java backend engine scans a website comprehensively and stores the results in the database.

Dashboard Visualization: Real-time display of scores and reports through HTML, CSS, and Tailwind-based UI.

Report Generation: Automatic generation of downloadable PDF audit reports summarizing key findings.

Improvement Insights: Intelligent generation of improvement suggestions with prioritized recommendations.

3.6 Software Requirements

Programming Language:
JavaScript: used for both frontend interactivity and backend logic with Node.js.

Backend Technology:
Node.js handles all the core back-end jobs, such as web performance and security analysis, sending and receiving API requests, and generating reports.

Frontend Technologies:
HTML, CSS, and Tailwind CSS are used for building a clean, responsive, and visually appealing user interface: Login Page, Dashboard, Report Page, and Improvement Page.

Database: Optional / Future Enhancement
Integration with either PostgreSQL or MongoDB will be able to store user credentials, website audit history, and performance reports in case persistent data storage is required.
Libraries and Frameworks:
Express.js: To create the Node.js server and routing.
Lighthouse / PageSpeed Insights API: For analyzing website performance metrics. OWASP ZAP API (optional): To run basic website security scans.
JsPDF & HTML2Canvas: For generating PDF reports from audit results. Tailwind CSS: For responsive and modern frontend design.
Operating System:
Windows 11, used as a development and testing environment. Development Environment / Tools: Visual Studio Code: Used for writing and maintaining both the frontend and backend code. Node.js and npm(Node Package Manager): This will provide the runtime environment for the backend server and manage all of the dependencies. Command Line / PowerShell: To run Node.js scripts, such as npm start.

4. RESULTS

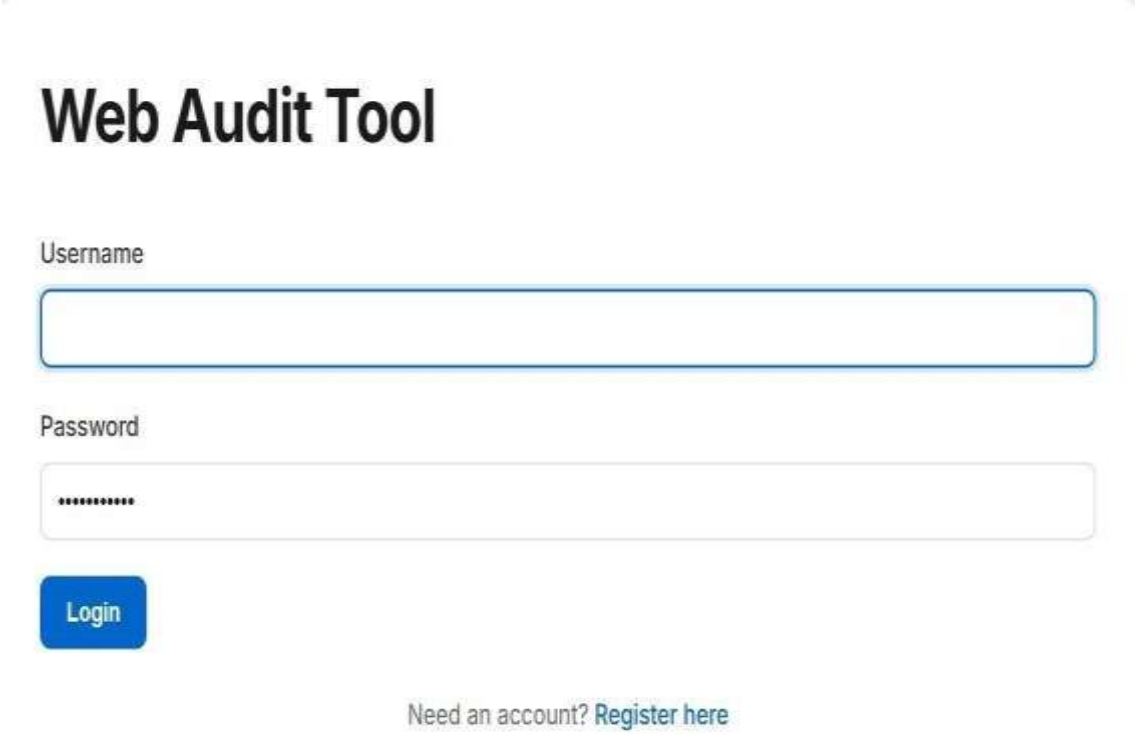

Figure 2.1: Login Interface of the Web Audit Tool

Figure 2.1 shows the Login Interface of the Web Audit Tool, providing a secure entry point for users to access the system. The interface includes fields for username and password, ensuring authenticated access before performing website audits. It is designed with a simple and intuitive layout for ease of use. Once logged in, users can navigate to the dashboard to initiate performance and security analyses.

Figure 2.2: User Registration Interface of the Web Audit Tool

Figure 2.2 displays the User Registration Interface of the Web Audit Tool, allowing new users to create an account securely. The form collects essential details such as name, email, and password for authentication purposes. Designed with a clean and user-friendly layout, it ensures easy onboarding for first-time users. After successful registration, users can log in and access the auditing features of the system.

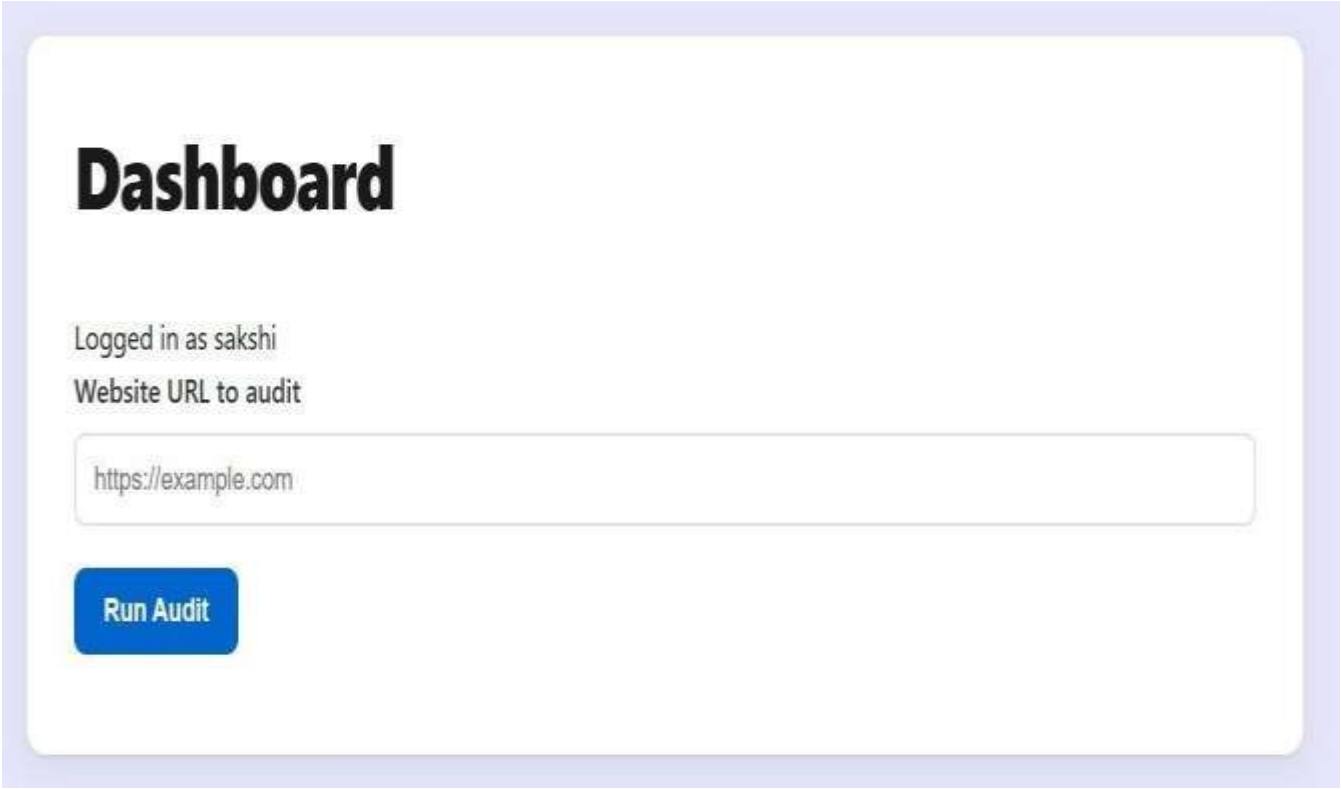

Figure 2.3: Dashboard Interface for Running Website Audit

Figure 2.3 shows the Dashboard Interface where users can initiate and manage website audits. The dashboard provides an input field for entering URLs and buttons to start performance and security analysis. It offers a clean, interactive layout that guides users through the auditing process. From this interface, users can access real-time results, reports, and system features efficiently.

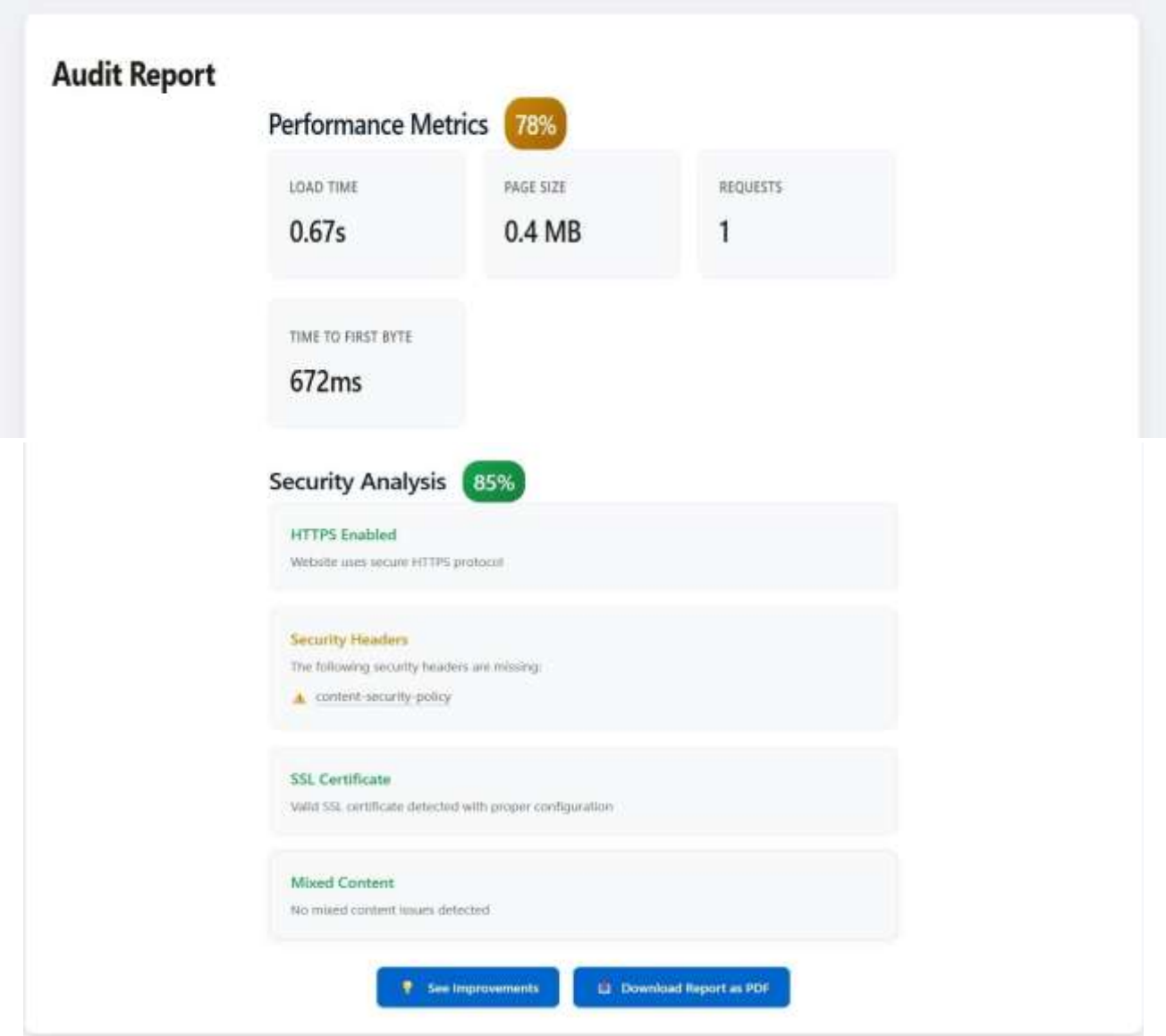

Figure 2.4: A highly informative dashboard that furnishes the output and presents comprehensive insights from the Website Audit Tool, effectively analyzing both performance and security parameters.

Figure 2.4 illustrates the detailed output dashboard of the Website Audit Tool, presenting clear insights derived from both performance and security analyses. The interface displays key metrics, scores, and visual graphs to help users understand website health at a glance. It consolidates findings such as load speed, SEO factors, vulnerabilities, and configuration issues. This comprehensive dashboard enables users to make informed decisions and apply targeted improvements effectively.

Figure 2.5: Suggested Security Improvements Generated by the Audit Tool

Figure 2.5 presents the suggested security improvements generated by the Website Audit Tool after analyzing potential vulnerabilities. The interface highlights issues such as insecure headers, outdated dependencies, and weak configurations, along with recommended fixes. These actionable insights help users strengthen their website's security posture. The suggestions are structured clearly to support quick understanding and implementation.

Dashboard and Audit Workflow Overview

Figure 2.1 displays the Login Interface of the Web Audit Tool, providing a secure user authentication system where individuals enter their username and password to access the platform. Figure 2.2 shows the User Registration Interface, allowing new users to create an account by entering their full name, username, and password, which enables them to proceed to the dashboard.

Dual-View Assessment of Website Performance and Security

Figure 3 represents the Dashboard Interface, which offers a dual-view assessment of each analyzed website by focusing on both performance metrics and security configurations. The dashboard is designed to give users a clear and comprehensive understanding of how well a website performs and how secure it is, all presented in a visually organized format.

Performance Metrics Visualization (Figure 2.3)

The performance section displays essential indicators such as Page Load Time, Speed Index, First Contentful Paint, Total Blocking Time, and the Overall Performance Score. To enhance interpretation, the tool uses a color-coded scoring system where green indicates excellent performance with a score above 85%, yellow indicates moderate performance between 50% and 85%, and red highlights poor performance with scores below 50%, signaling the need for immediate optimization. The system ensures highly accurate data extraction so that even small variations in loading speed or resource usage are reflected precisely in the recorded values.

Security Analysis Summary (Figure 2.4)

The security module evaluates the website's readiness by checking HTTPS implementation, verifying SSL certificates, inspecting important security headers such as Content-Security-Policy, X-Frame-Options, and X- Content-Type-Options, and identifying any insecure or mixed content requests. The dashboard provides easy-to-understand visual indicators by labeling secure websites with a green shield icon and marking vulnerable ones with a red shield icon, along with explanations of the identified security issues.

Advantages:

Unified Security and Performance Assessment:

Combines both the critical aspects, speed and safety, in a single audit, hence requiring no separate tools or plugins.

Automated Assessment:

Fully automates website auditing via backend scripts, reducing manual effort and ensuring consistent, repeatable testing.

Real-time Results and Visualization:

The web-based dashboard refreshes dynamically, offering near-instant analysis feedback within seconds of the URL's submission.

Actionable Recommendations:

Provides context-aware improvement tips, which allow developers to effectively fix performance bottlenecks and strengthen
security.

User-Friendly and Transparent:

Audit results are made accessible and understandable even to non-technical users by using visual scores and color-coded metrics.

Data-Driven Decision Support: Organizations and developers can track the health of their websites over time, pinpoint trends, and prioritize fixes according to actual metrics.

Future Scope:

While present functionality allows for performance and security audits to be conducted efficiently with instant reporting, further enhancements can significantly extend the scope and capability:

Integration of AI-Based Analysis: In future releases, performance degradation trends can be predicted by using machine learning models more intelligently in finding potential vulnerabilities.

Support for Continuous Monitoring: The tool can evolve into a scheduled monitoring platform that automatically audits websites at given intervals and alerts users through email or dashboard notifications.

Advanced Visualization: Incorporating interactive charts and comparative analytics will allow users to visualize performance trends across multiple audits over time.

Expanded Security Testing: Future enhancements would involve the scanning of OWASP Top 10 vulnerabilities and, for advanced users, automated penetration testing.

Browser Extension or Mobile App: An easy-to-carry browser extension or mobile version could allow developers on the go to audit websites with one click, viewing results there and then.

Integration with CI/CD Pipelines: The tool can be integrated into DevOps workflows to automatically test website performance and security after every deployment.

5. CONCLUSION

The Website Audit Tool with Security and Performance Analysis presented here successfully provides a unified platform for assessing and improving the general health of web applications. By embedding automated modules for both performance assessment and detection of security vulnerabilities within, the system allows developers and administrators to gain actionable insights into the efficiency, responsiveness, and robustness of their websites.

The Node.js backend assures scalability and fast asynchronous processing, while the HTML- and CSS-

based frontend provides an interactive interface for visualization of performance scores, security alerts, and suggested enhancements. Key metrics that are automatically collected by the system include page load speed, resource optimization, and OWASP-based vulnerability checks; hence, manual effort and chances of human error are reduced.

It provides real-time analysis and report generation that improves website reliability, security compliance, and user experience. What's more, it uses clear insights and prioritized recommendations for enhancement, making web maintenance truly data-driven.

In summary, this project provides an effective and practical solution for continuous website auditing, with future support for AI-based vulnerability prediction, automated suggestions for patches, and integration with CI/CD pipelines in order to guarantee constant performance and security monitoring for web platforms.

6. REFERENCES

1. S. Sharma, "Automated Website Security Analysis Using Python," *IEEE Access*, vol. 12, pp. 14523–14530, 2024.
2. J. Lee, R. Gupta, and S. Patel, "Unified Web Health Metric: Integrating Performance and Vulnerability Analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 2, pp. 345–357, 2024.
3. M. E. Shaw and L. Wong, "A Comparative Study of Backend Frameworks for Web Performance Optimization," in *Proc. IEEE ICACCS*, pp. 520–526, 2024.
4. R. Ramires, A. Respício, and I. Medeiros, "KAVE: A Knowledge-Based Multi-Agent System for Web Vulnerability Detection," in *Proc. IEEE ICWS*, pp. 112–119, 2024.
5. P. Nunes, J. Fonseca, M. Vieira, and L. Almeida, "Blending Static and Dynamic Analysis for Web Application Vulnerability Detection," in *Proc. IEEE/ACM ASE*, pp. 233–242, 2023.
6. A. Kumar and S. Banerjee, "Performance Evaluation of Modern Web Applications Using Automated Analysis Tools," in *Proc. IEEE ICCCNT*, pp. 1–7, 2023.
7. L. Huang, T. Zhang, and W. Xu, "Machine Learning-Based Detection of Web Vulnerabilities in Dynamic Web Applications," *IEEE Access*, vol. 11, pp. 98520–98533, 2023.
8. H. Park and G. Cho, "A Study on SEO Optimization and Web Performance Correlation Using Automated Metrics," in *Proc. IEEE BigData*, pp. 4409–4416, 2024.
9. D. Patel and K. Srinivasan, "Improving Website Load Speed Through Intelligent Caching and Resource Prioritization," in *Proc. IEEE ICWS*, pp. 379–386, 2023.
10. Y. Al-Hassan and M. Qureshi, "A Security-Driven Framework for Website Auditing Using Automated Scanners," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 812–825, 2023.