Tribhuvan University

Academia International College

Final Year Project Report

On

"Vulnerability Scanning Website"

[CSC 412]

Under the supervision of

"Mr. Ananda Adhikari"

Submitted by
Kanchan Bhujel (T.U. Exam Roll No. 26491/077)

Sandip Khadka (T.U. Exam Roll No. 26514/077)

Rupak Pathak (T.U. Exam Roll No. 26510/077)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Academia International College

Final Year Project Report

On

"Vulnerability Scanning Website"

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Kanchan Bhujel (T.U. Exam Roll No. 26491/077)

Sandip Khadka (T.U. Exam Roll No. 26514/077)

Rupak Pathak (T.U. Exam Roll No. 26510/077)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that this project prepared under my supervision by Kanchan Bhujel(2491/077), Sandip Khadka (26514/077), Rupak Pathak (26510/077) entitled "Vulnerability Scanning Website" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology

……

Mr. Ananda Adhikari

Project Supervisor

HOD/Program Coordinator

Department of Computer Science and Information Technology

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Kanchan Bhujel, Sandip Khadka, Rupak Pathak entitled "Vulnerability Scanning Website" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…… Mr. Ananda Adhikari Supervisor Academia Int'l College	…… Mr. Bishwas Mathema HOD/Program Coordinator Academia Int'l College
…….. External Examiner Academic designation	…….. Internal Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We are very grateful to Academia International College for the opportunity and facilitation in terms of resources to undertake this project as part of our curriculum. The favorable environment and facilities at the college greatly contributed to enabling us to do the project efficiently.

A special word of thanks goes to our supervisor, Mr. Bishwas Mathema (HOD/Program Coordinator, Academia International College), for continuous guidance, support, and feedback provided throughout the process. We are indeed grateful for his mentorship, which gave us an opportunity to develop further in learning and apply the theoretical knowledge practically.

We would like to extend our deepest gratitude to all, including families, friends, colleagues, and lecturers who made this journey smoother. Your support through constant encouragement, valuable advice, and steady support really helped us to reach our destination well within the specified timeframe. This venture was an educational venture, which was enriching, too, and for that, we are truly grateful.

Thank You,

Kanchan Bhujel (T.U. Exam Roll No. 26491/077)

Sandip Khadka (T.U. Exam Roll No. 26514/077)

Rupak Pathak (T.U. Exam Roll No. 26510/077)

Abstract

The Vulnerability Scanning Website is a portal meant to find and fix security bugs contained in a website. Using the BFS for crawling and Random Forest algorithms to detect vulnerabilities, it goes on crawling over the structure of the website, including such common security weaknesses in SQL Injection and Cross-site Scripting/XSS, among others. It then presents this information in comprehensive, yet understandable, reports of the identified vulnerabilities, their criticality, and suggested mitigations. The website is developed on Python, Django, and a MySQL database to ensure efficiency in data management and safety in user authentication. In essence, the major aim will be to provide a trusted, automated security scanning service that would assist users in securing their websites against cyber threats, minimizing security risks, and encouraging safety online. The site bridges technology and cybersecurity, thus enabling the owner to secure his website and keep his online environment safe.

Keywords: vulnerability scanning, BFS, Random Forest, Django, Python, MySQL, cybersecurity, website security, vulnerability detection.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

DFD	Data Flow Diagram
ER	Entity Relationship
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
SQL	Structured Query Language
VS Code	Visual Studio Code
XSS	Cross-site scripting
CSRF	Cross-site request forgery

Chapter 1: Introduction

1.1 Introduction

A website vulnerability scanning system **'WebSecurity'** is a software tool that is designed to analyze a website for security weaknesses and potential threats. Although websites are essential for business interactions and transactions, they are also more vulnerable to cyberattacks. To steal data, gain unauthorized access, or disrupt services, cybercriminals actively seek vulnerabilities.

Web vulnerabilities are flaws in the security of a website that can be taken advantage of by hackers to disrupt the operation, safety, and trustworthiness of a web application. Frequently found vulnerabilities are SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and weak authentication systems. The effects of these vulnerabilities can be serious, resulting in data leaks, monetary losses, legal issues, and harm to a company's image. It is essential to constantly monitor and evaluate websites for potential vulnerabilities to reduce these risks. Manual code reviews and penetration testing, for example, are effective, but they frequently take a long time and require a lot of resources. For businesses that frequently update their websites or rapidly implement new features, these strategies may not be practical . [1]

Website **'WebSecurity'** our project, provides an effective vulnerability scanning system that seamlessly integrates into development workflows to address these issues. Organizations can quickly identify and address potential weaknesses from this system, which enables timely and comprehensive security assessments while keeping up with the rapid pace of modern web development.

1.2 Problem Statement

In an era where websites serve as critical platforms for communication, commerce and information dissemination, ensuring their security is paramount. Despite advancements in web security, vulnerabilities in websites continue to pose significant risks, including data breaches, unauthorized access, and service disruptions. Recent data reveals that over 40% of websites contains critical vulnerabilities, potentially exposing sensitive user information and damaging organizational reputations. [2]

Current vulnerability assessment tools offer basic scanning capabilities but often fall short in several key areas. Many existing systems has:

1. **Lack Comprehensive Coverage:** They do not detect all types of vulnerabilities, such as those related to complex web applications or new and emerging threats.
2. **Generate High False Positives:** They often produce numerous false positives, leading to wasted resources and reduced trust in the results.
3. **Not Clear Report:** The results may be difficult to interpret, lacking actionable insights for remediation.

The consequences of undetected vulnerabilities are severe, ranging from financial losses and legal liabilities to loss of customer trust and competitive advantages. There is a pressing need for a more robust and comprehensive system that not only identifies vulnerabilities with higher accuracy but also provides actionable recommendations for remediation. [3]

This project aims to develop an advanced system '**Websecurity**' for identifying website vulnerabilities that address the limitations of current tools. The new system will incorporate cutting-edge scanning technique, reduce false positives, offer customizable assessments, and provide clear reports. By enhancing the ability to detect and address vulnerabilities effectively, this initiative seeks to improve overall web security and protect both individual and organizations from potential cyber threats.

1.3 Objective

- To create a system that scans websites on its own to find security flaws.
- To make sure that the vulnerabilities being reported are real and fixable.
- To create in-depth, easy-to-understand reports that describe the vulnerabilities found, their severity and provide the solution.

1.4 Scope and Limitations

1.4.1 Scope

The goal of the "**WebSecurity**" website is to provide an all-inclusive, solution for scanning website and generating report for security flaws in websites. The project's scope include following scope:

- Website Scanning:

  The system will scan websites for vulnerabilities, including SQL injection, cross-site scripting, and many others that are dangerous to a website's security.
- Reporting of Vulnerability:

  It will flag only actual vulnerabilities, and the system must avoid false positives. This shall be presenting facts useful in the remediation of identified problems.
- Comprehensive Reporting:

  The system will then give comprehensive reports on the vulnerabilities found, their severity.

1.4.2 Limitations

The project has significant limits in spite of its all-encompassing approach:

- Limited Detection: The system may miss advanced security threats requiring manual inspection or specialized tools.
- False Positives: Some vulnerabilities are incorrectly flagged and require user verification.
- Dependency: Scanning depends on the website's accessibility and data quality at the time of crawling.
- Limited scope : Mainly focusing on common vulnerabilities-newer threats might not be picked up immediately.
- Performance Impact: Scanning might slow down large or complex sites, both for the user's site and the system.

1.5. Development Methodology

We use the Agile methodology in our project to ensure an efficient and flexible approach to developing the "**WebSecurity**" website Agile emphasizes iterative development, breaking the project into manageable sprints, each focused on delivering specific features such as user authentication, vulnerability scanning, and detailed reporting. Sprints therefore provide us with the ability to continuously improve the system, take up challenges quickly, and adapt to the evolving requirements or new vulnerabilities. This keeps our development dynamic and aligned with user expectations. [4]

Inside each sprint, there are phases: planning, designing, coding, testing, and reviewing. We define the goals and tasks of the sprint during planning. Designing and development implement features, after which testing takes place to make sure that it functions properly. Regular reviews and retrospectives help the team assess progress, resolve issues, and improve the system. This keeps us focused on creating the most user-friendly system with optimum performance, such as fast scanning speed and actionable reports.

Agile is critical to this project because of its emphasis on collaboration and feedback. Team members and stakeholders can contribute at all stages, ensuring that the system meets real needs and incorporates insights. For example, feedback on vulnerability report clarity or scanning accuracy can be addressed in the next sprint. This adaptive process is going to deliver improvements in product quality and timeliness while maintaining the flexibility needed for challenges and new features. Agile will help us evolve the "**WebSecurity**" website into a dependable tool for improved web security. [4]

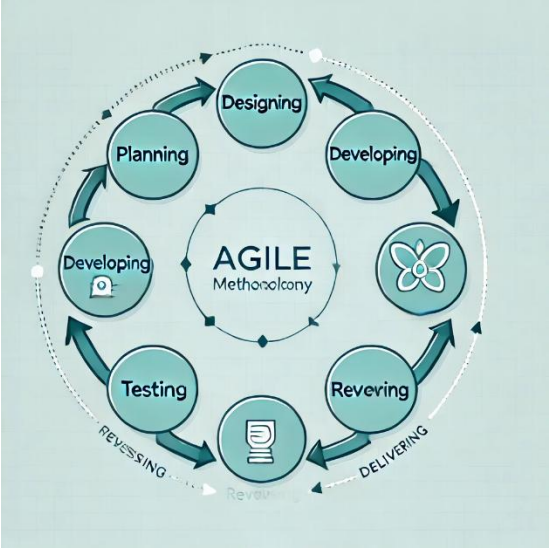


Figure 1.1: Agile Methodology Life cycle

1.6. Report Organization

This report is structured into six key chapters, each focusing on a specific part of the project to ensure a logical and organized flow of information. The chapters are as follows:

1. Introduction

This chapter introduces the "Web Security" website, which means securing the websites from SQL injection and cross-site scripting. It also presents the objectives of the project, the challenge that this project addresses, and the benefits of the proposed system.

2. Background Study and Literature Review

This chapter reviews the existing tools like Qualys, HostedScan, and ImmuniWeb; it also shows the limitations of these tools, including incomplete coverage, high false positives, and unclear reporting. It also discusses some key concepts, algorithms, and technologies used in the system, such as Breadth-First Search and Agile methodology.

3. System Analysis

This chapter presents the system requirement specification, both functional and non-functional, as proposed in the proposal. It also covers the feasibility study on technical, operational, and economic viability with a clear understanding of the foundation on which the system is based.

4. System Design

This chapter elaborates on the high-level design of the system, including DFDs, flowcharts for crawling URLs, and the working mechanism of the proposed system for vulnerability scanning. It also explains how each component contributes towards the achievement of objectives set forth in the project.

5. Implementation and Testing

This chapter covers the development process, coding, testing phases, and tools involved in the process, such as OWASP ZAP or Burp Suite. It also discusses the testing methodologies applied to ensure accurate detection of vulnerabilities and overall system reliability.

6. Conclusion and Future Recommendations

This chapter summarizes the results of the project, showing how the system will improve the deficiencies in existing tools. It also points out some suggestions for future improvements: the integration of machine learning, expansion of compatibility, and real-time alerts.

Chapter 2: Background Study and Literature Review

2.1 Background Study

Web security is crucial in this modern era. Websites are needed for communication, e-commerce, and information exchange. Despite new technology, websites remain vulnerable to numerous cyberattacks. SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) are popular vulnerabilities that are used most frequently by attackers for unauthorized access, stealing sensitive data, or ending services. The implications of such weaknesses can be severe. They may involve loss of money, loss of reputation, litigation, and loss of user trust. [5]

Traditional vulnerability detection techniques like penetration testing and manual code audits are certainly effective but time-consuming and require specialized skills. In dynamic environments where websites are being updated frequently, these methods are simply not practical for companies to undertake. To counter these issues, automated vulnerability scanning tools have been developed that provide solutions not only faster but also scalable. But existing tools fall significantly short in several areas including low vulnerability coverage, excessive false-positives, and a lack of actionable data in their resulting reports. [6]

In the context of Nepal web security systems is still at a nascent stage. Most organizations depend on international tools like Qualys, HostedScan , ImmuniWeb or basic manual testing. However, most of these tools are highly cost-intensive, complex, and difficult to access, hence impractical for small to medium-sized organizations in Nepal. The absence of localized, affordable, and automated solutions has created a big void in the market. This shows that a system like **WebSecurity** is necessary to address the specific needs of Nepali businesses with user-friendly, scalable, and affordable solutions for web vulnerability detection. [7]

In this project we will apply efficient crawling mechanisms, such as the Breadth-First Search algorithm. Besides, **WebSecurity** overcomes the shortcomings of existing tools by providing clear reports with minimal false positives.. This approach not only strengthens security but also simplifies the process of vulnerability detection for organizations and developers in Nepal.

2.2. Literature review

Quite extensive research and development relating to Web security have been carried out. Theoretical and practical aspects of the detection and mitigation of vulnerabilities have been considered. Research in Web application security points out that attacks are gradually becoming sophisticated, hence the need for proactive measures. For instance, The necessity of web security was acknowledged as early as the late 1990s, when websites started to become essential for communication, business, and data sharing. The complexity of websites made them vulnerable to attacks like SQL Injection and Cross-Site Scripting (XSS), which were both recognized as important attack vectors in early web applications. This made it clear that taking preventative action was necessary to secure websites. [8]

Several prominent tools, such as Qualys, HostedScan, and ImmuniWeb, have been widely used in the field of web security. Qualys is a cloud-based solution founded in the year 1999 for extended security and compliance solutions. It scans more than 9 billion data points every month for vulnerabilities such as SQL injection, cross-site scripting (XSS), and software being outdated. The central dashboard makes it very easy to manage the scan and generates detailed compliance reports. But it is difficult to learn, requires a complex setup, and is costly; therefore , it may not be ideal for small organizations , which also includes in Nepal. [9]

HostedScan is a lightweight, web-based vulnerability scanner that was founded in Seattle, Washington, in 2021. It targets small-scale use and scans about 10,000 websites every month. The most common vulnerabilities are scanned by this tool, which sends the categorized results to users by email in a simple interface. According to the Skybox Security report, a new CVE (Common Vulnerability and Exposures) surfaces every 17 minutes. Therefore, it's imperative to regularly scan your IT infrastructure to identify and address vulnerabilities proactively before threat actors can exploit them, and this is where HostedScan can help. While easy to use, HostedScan lacks the advanced features, detailed customization options, and integration capabilities for CI/CD workflows that make it less effective for larger-scale applications and less appealing for local organizations in Nepal. [10]

ImmuniWeb was introduced in mid-September 2019 in Geneva Switzerland, which combined automated scanning for vulnerabilities with manual testing to provide a hybrid solution. It runs more than 5 million application scans every year and provides prioritization

of vulnerabilities by severity using AI-driven tools. The platform is very strong in actionable insights with user-friendly dashboards. On the other hand, ImmuniWeb is relatively expensive for smaller businesses, has limited support for zero-day vulnerabilities detection, and hence is beyond the reach of organizations in Nepal. [11]

Chapter 3: System Analysis

3.1. System Analysis

System analysis is a very important phase in the "**WebSecurity**" website development, which includes the identification of the functional and non-functional requirements of the system, existing tools analysis, and feasibility analysis of the proposed system. This will ensure that the limitations of the present vulnerability scanning tools are overcome with the new system and that it meets the expectations of the users and operational requirements.

3.1.1. Requirement Analysis

The requirement analysis is a vital phase of the "**WebSecurity**" website development wherein the functional and non-functional requirements of the system are defined. This will assure that the proposed system will fall in line with the objectives and will eliminate the limitations identified in the existing tools. Based on the analysis from the project proposal, the following are the requirements:

I. Functional Requirements:
The system considered following functional requirements:

- **Vulnerability Scanning:** This system will scan websites to highlight vulnerabilities such as SQL Injection, Cross Site Scripting XSS, and poor authentication mechanisms.
- **Detailed Reporting:** This will develop comprehensive reports outlining the identified vulnerabilities, their severity levels, and the steps to remediate them.
- **User Management:** The system will include user authentication features, enabling secure login and registration for new users.

II. Non-Functional Requirements:
The system considered following non-functional requirements:

- **Optimized Scanning Speed**: The system will ensure fast scanning without compromising accuracy.
- **High Accuracy**: It will minimize false positives and negatives for reliable vulnerability detection.

- **User-Friendly Interface:** The system will feature an intuitive interface for easy navigation and usability.
- **Seamless Integration**: It will integrate with CI/CD pipelines to enable continuous security monitoring.

3.1.2. Feasibility Analysis

i. Technical

Web security vulnerability scanning will be thoroughly evaluated in our technical feasibility study to guarantee the resilience and security of our chosen technology stack. This entails assessing the suitability and efficacy of a variety of security scanning tools, such as OWASP ZAP or Burp Suite, as well as locating any integration issues with our current systems. Additionally, we will determine the skills and training our development team needs to effectively manage and mitigate security risks and address potential technical issues like false positives and system performance. We want to make sure that our technology stack can effectively defend against security threats and vulnerabilities by thoroughly investigating these aspects.

ii. Operational

In the operational feasibility study, the system's impact on existing workflows will be considered. This involves evaluating the new system's integration into existing pipelines for continuous integration and continuous deployment (CI/CD) as well as how security teams and developers can use it effectively. The study will look at how well the system works with the processes that are already in place, look for any potential problems or improvements, and make sure that it makes everything work better and more efficiently. This includes looking at how the system integrates security measures into development workflows, how it simplifies deployment procedures, and how it works with established CI/CD practices to make the transition go as smoothly as possible and improve operational performance .

iii. Economic

In a study on the project's economic feasibility, we weigh the costs and benefits. This includes weighing the potential savings from preventing security breaches as well as the initial development costs and ongoing maintenance costs. By looking at these things, we can figure out if the expected financial benefits and overall return on investment make the investment worthwhile. Our decision-making process regarding the project's viability and 6 sustainability is guided by this comprehensive evaluation, which helps us comprehend its economic impact.

iv. Schedule

To show the project's timeline, which includes important milestones and deliverables, a Gantt chart is created. To make sure that everyone involved is aware of the project's progress, the chart provides a visual representation of the schedule. The phases of the project, specific tasks, their start and end dates, and dependencies between tasks are all detailed in my Gantt chart, which can be found below. The project's progress can be tracked, resources can be effectively managed, and the team can be kept informed about upcoming events and deadlines with this chart.

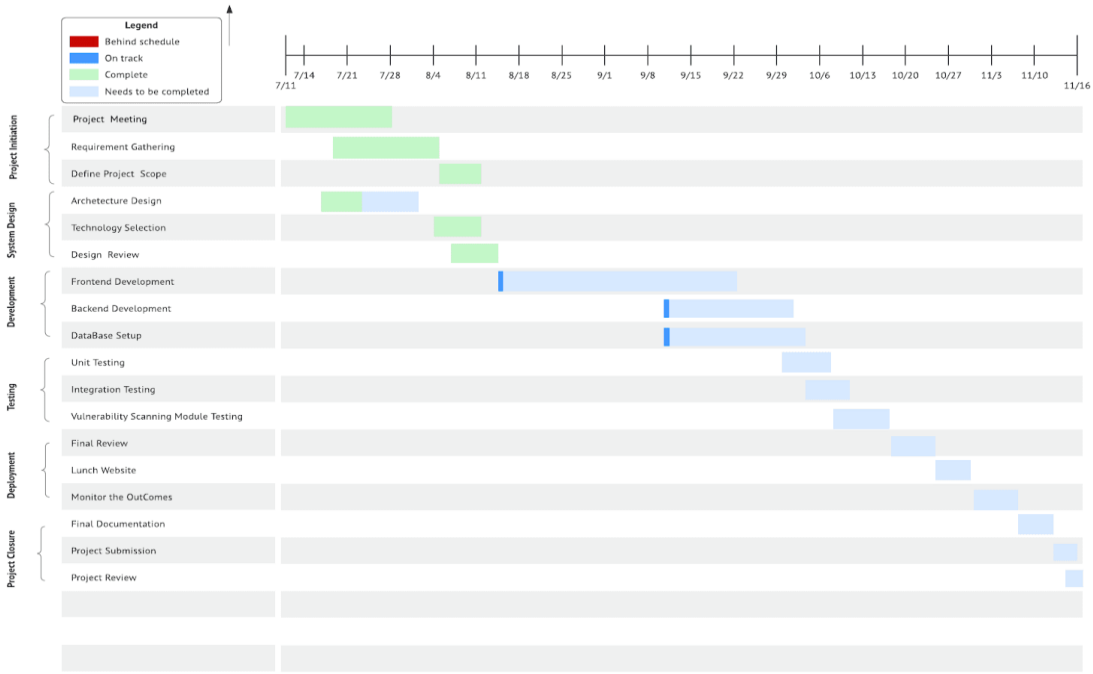


Figure 3.1: Gantt Chart

3.1.3. Data Modeling using ER Diagrams

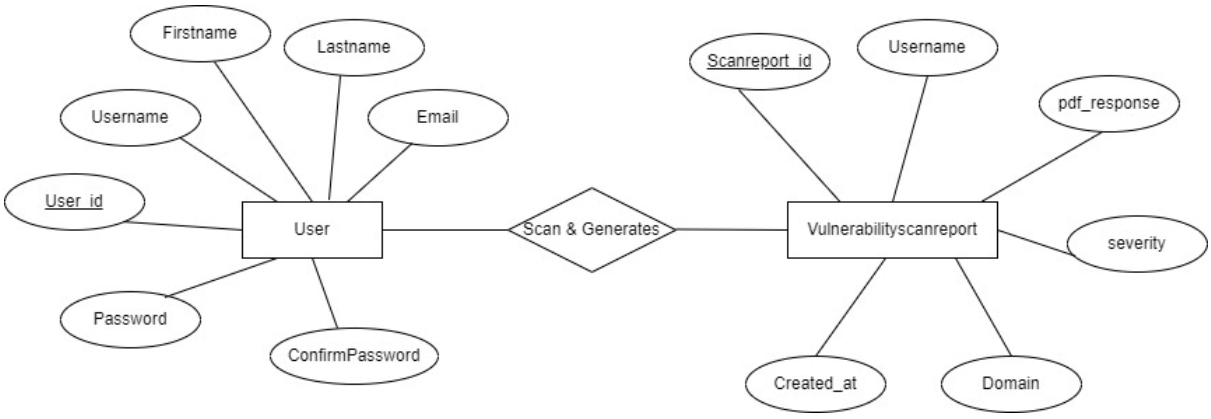


Figure 2.2: ER Diagram

3.1.4. Process Modeling using DFD

DFD Level 0

The Web Security Project's DFD Level 0 provides a high-level overview of the system by illuminating its most important processes and interactions with external entities. It demonstrates the flow of data between the system and external sources, defining the boundaries and primary functions without providing specific details about internal processes.

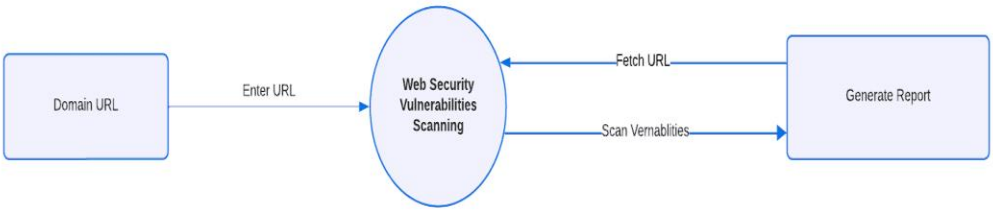


Figure 3.3: DFD level 0

DFD Level 1

The Web Security Project's DFD Level 1 provides a comprehensive view by subdividing high-level processes into sub-processes. It demonstrates the interplay between data storage, the URL queue, and the web crawler. The diagram explains how data is processed and managed within the system by showing how it flows between sub-processes like URL management, vulnerability scanning, and threat detection.

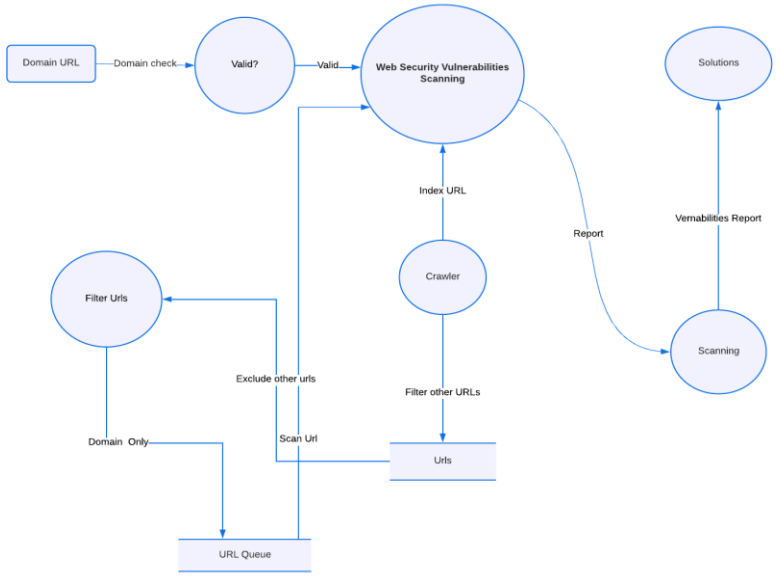


Figure 3.4: DFD Level 1

DFD Level 2

This is the Web Security Project's DFD Level 2, which provides a more comprehensive view by further breaking down the Level 1 sub-processes. It explains how vulnerability scanning, scan reports, and user login history checks work with the URL queue and data storage in detail. The flow of data between these parts is shown in the diagram. It shows how the system finds vulnerabilities, makes reports, and keeps track of login history.

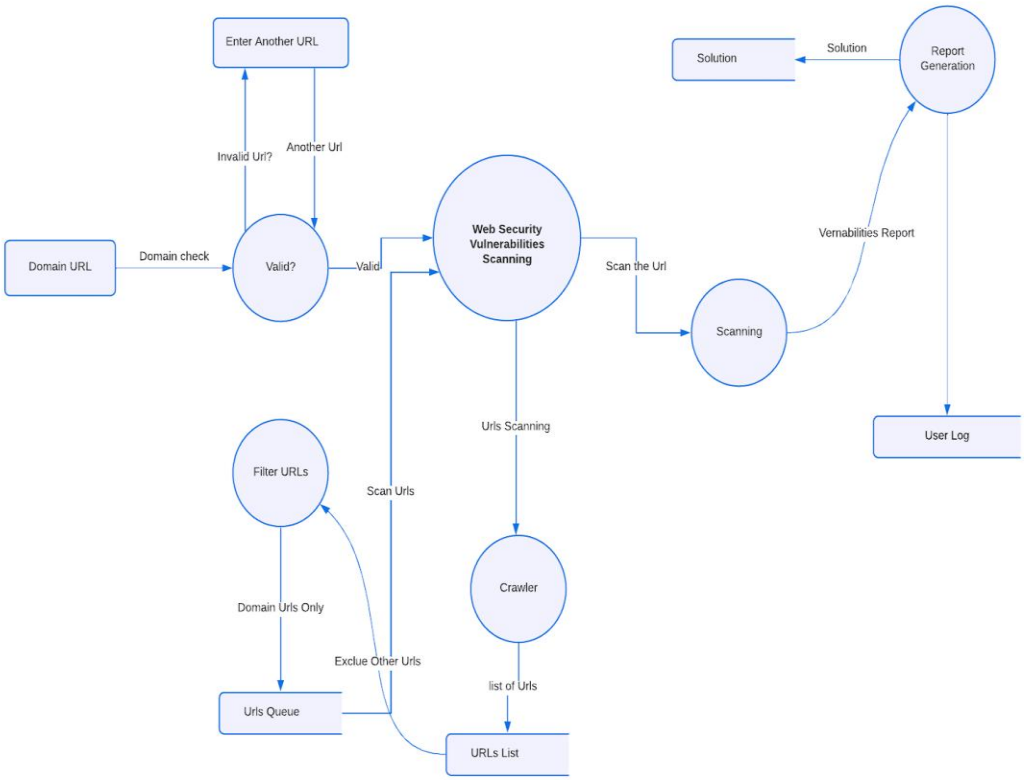


Figure 3.5: DFD Level 2

Chapter 4: System Design

4.1 Design

WebSecurity is designed for efficient, easy-to-use scanning of vulnerabilities in web applications. It will be helpful for organizations to identify security weaknesses like SQL injection, cross-site scripting, and poor authentication mechanisms, making an organization's website vulnerable to different types of cyberattacks. It aspires to accurate detection of vulnerabilities, very detailed and continuous monitoring for safer and more secure web environments.

4.1.1 Architectural Design

The architecture design of the **WebSecurity** project follows a 3-tier architecture made up of the client, crawler server, and database. The client layer can be thought of as the representation or user interface where users can log in, sign up, and submit website URLs for scanning through a web browser. The crawler server is the processing layer, handling the crawling of submitted URLs, analysis of web pages for vulnerabilities, and interaction with the various vulnerability detection tools. Finally, the database layer securely stores user credentials, submitted URLs, scan results, and generated reports for efficient retrieval and management of the data for subsequent usage. This kind of architecture would ensure scalability, modularity, and efficiency in the communication of its components.

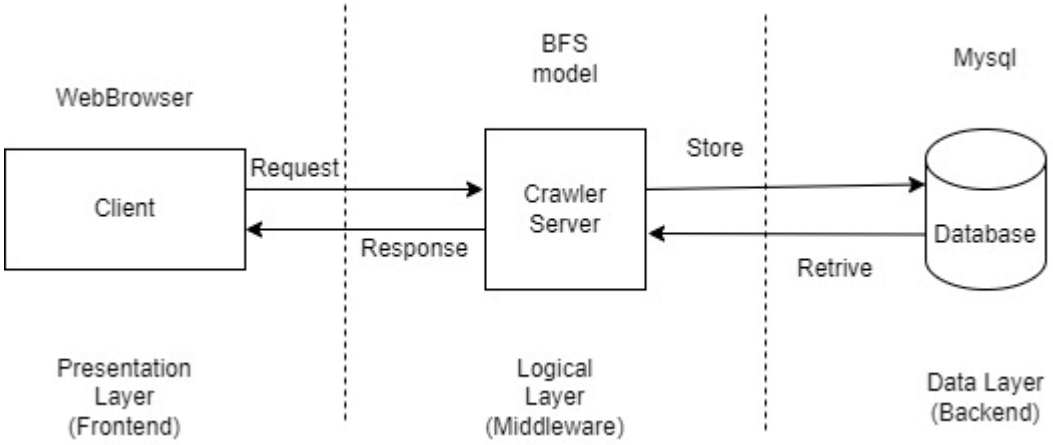


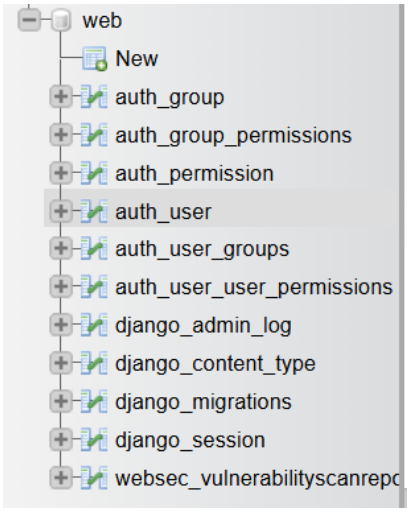
Figure 4.1: Tier Architecture Design

4.1.2 Database Design

WebSecurity uses MySQL to design the database for storing and managing information about the users, the URLs of the websites submitted, the vulnerabilities detected, and the reports generated. It ensures secure storage of user credentials, maintains relationships between users and their scan history, and enables efficient data retrieval for seamless scanning, reporting, and user interaction. MySQL provides reliability, scalability, and ease of integration with the system.

User

Id
Username
Firstname
Lastname
Email
Password
Confirm Pssword



websec_vulnerabilityscanreport

Id
Username
pdf_response
severity
Created_at
Domain

4.1.3 Forms and Interface Design

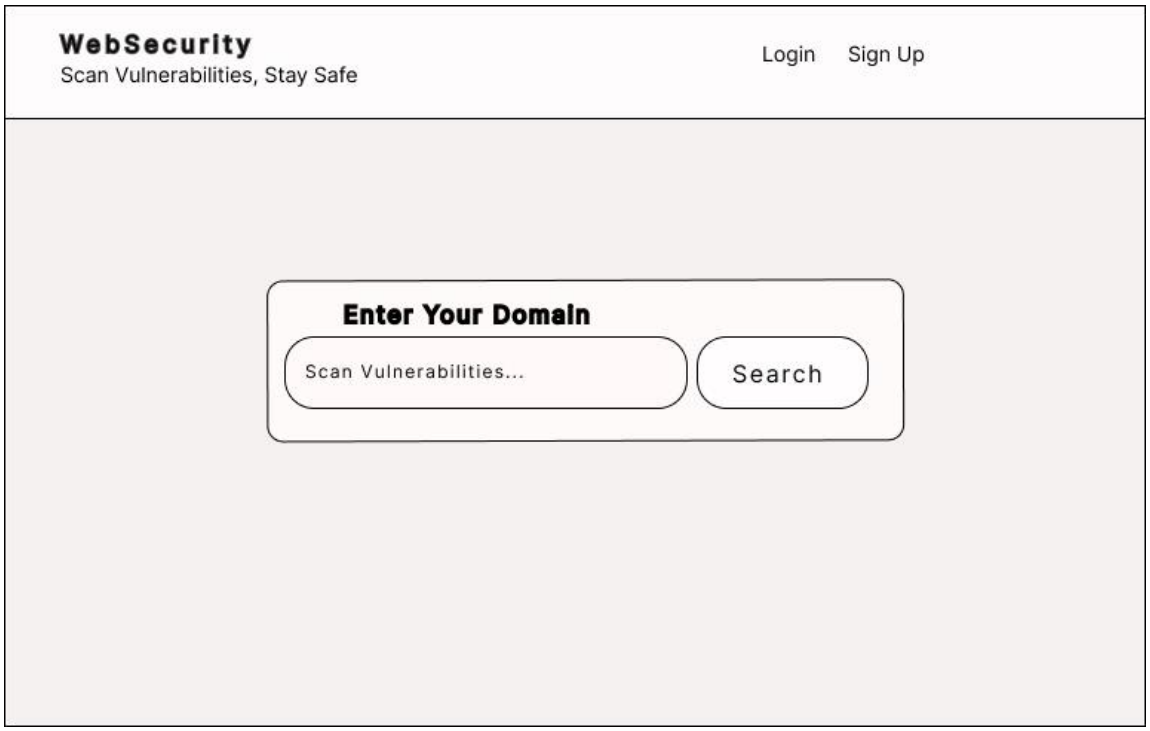


Figure 4.2: Dashboard page

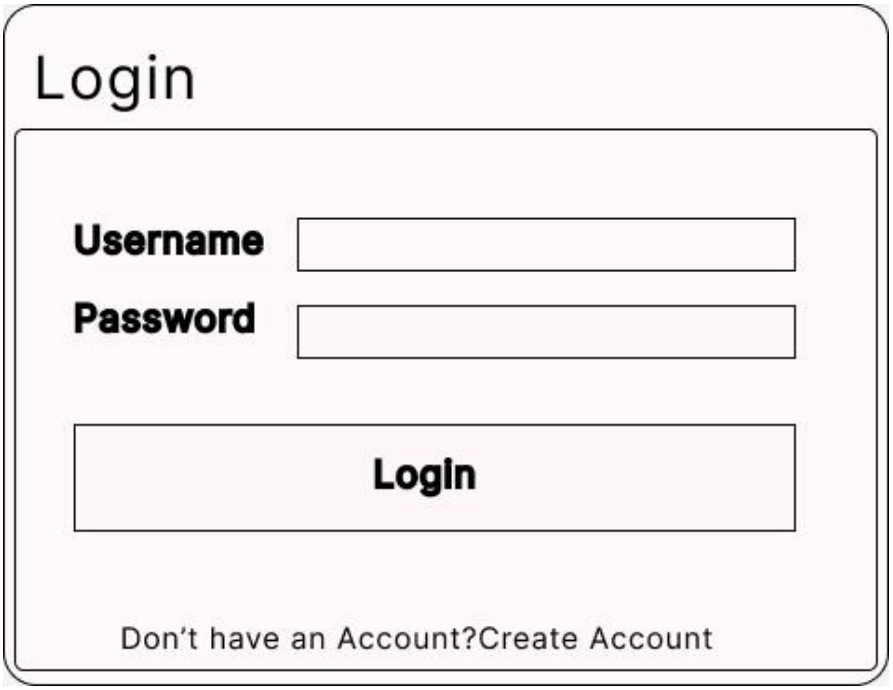


Figure 4.3: Login Page

Figure 4.4: Signup Page

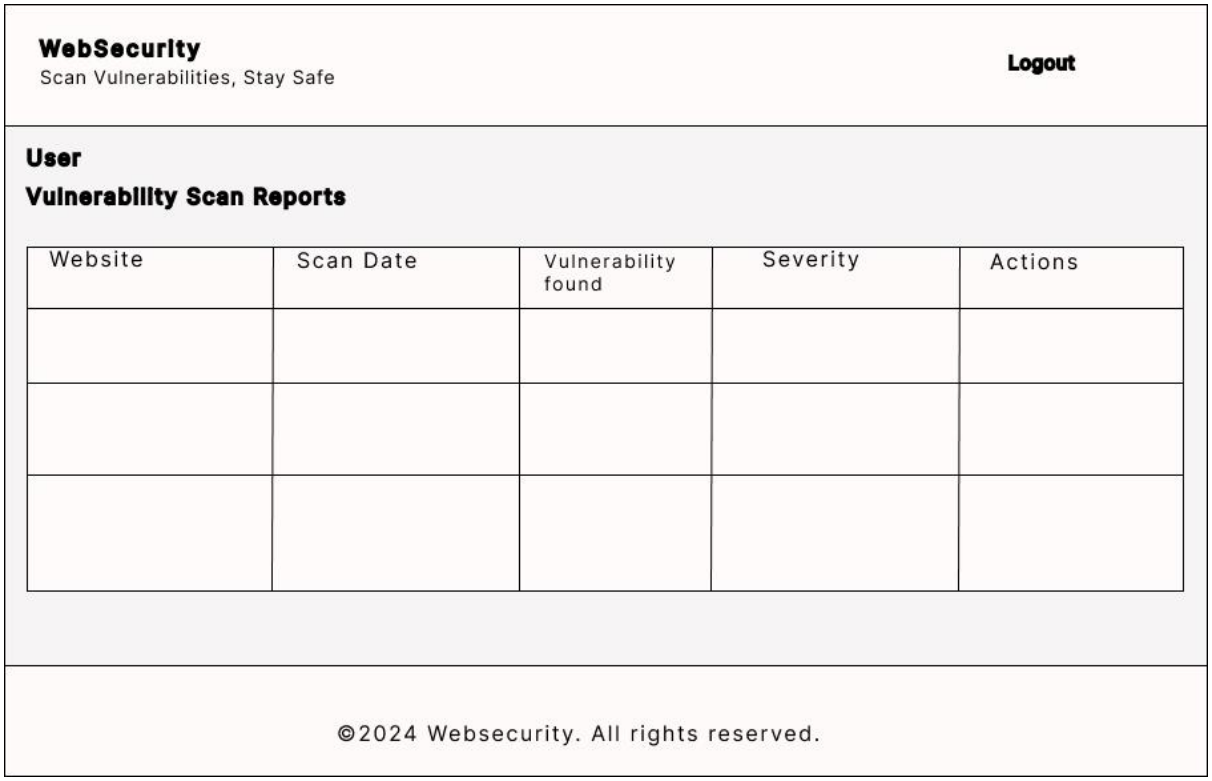


Figure 4.5: Vulnerability Scan Reports Design

4.2 Algorithm Details

4.2.1 Breadth-First Search (BFS) Algorithm for Crawling:

Breadth-First Search is a technique used to view the web pages in a structured manner. It starts its work by visiting the root or, in other words, the home page, followed by the exploration of all the direct links from the home page, then links from newly discovered pages, and so on. This happens level by level, ensuring that all accessible pages are discovered in a structured way. This is especially helpful for big websites, because, using the BFS algorithm, no page will ever be missed and it will never revisit any previously explored page. The BFS will keep track of the pages to visit next using a queue and will process each page in order. This method ensures very good coverage of the site and is very good at uncovering all reachable links within the structure of the site.

Steps towards BFS in Web Crawling:

- Initialization: A root URL is created from user input.
- Add Root URL to Queue: The root URL is inserted into a queue for processing.
- Visit and Process URL: The first URL in the queue is accessed and all of its hyperlinks are located.
- Add Unvisited URLs to Queue: The newly found URLs that are unvisited are added to the queue.
- Repeat: The process repeats, recursively visiting every URL and following its links until all reachable URLs have been crawled or the maximum depth is reached.
- Completion: It gives a list of all crawled URLs, ready for vulnerability analysis.

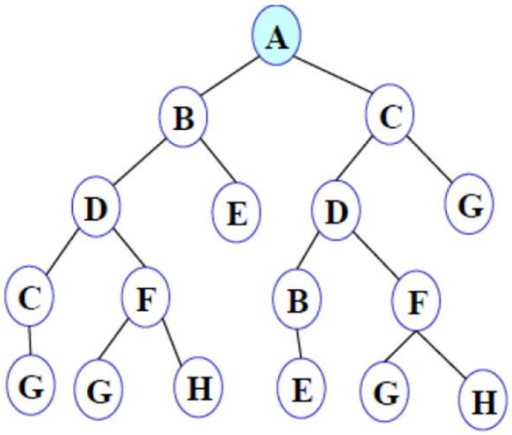


Figure 4.6: Breadth First Search (BFS)

4.2.2 Evaluation Matrix

When evaluating how well a classification model works, we rely on certain metrics to assess its consistency and correctness. Let's break them down in simple terms:

- True Positive (TP)**:**
  The model correctly identifies a vulnerability present on a website.
  Example**:** The crawler identifies an SQL injection vulnerability on a website and correctly reports it as a vulnerability.
- False Positive (FP)**:**
  The model incorrectly predicts that a vulnerability exists when it doesn't.
  Example**:** The model reports a non-existent vulnerability, such as a false alarm for cross-site scripting (XSS) on a secure page.
- True Negative (TN)**:**
  The model correctly identifies that there is no vulnerability.
  Example: The crawler correctly identifies that a web page is secure and does not contain any vulnerabilities.
- False Negative (FN)**:**
  The model fails to detect an actual vulnerability and incorrectly marks it as secure.
  Example**:** The model misses an existing SQL injection vulnerability on a page and reports it as secure.

4.2.2.1 Accuracy :Accuracy gives a general idea of how often the model is correct. It's the proportion of all correct predictions (both positives and negatives) to the total predictions.
Formula: Accuracy = (TP +TN)/(TP+TN+FP+FN)

4.2.2.2 Precision: Precision tells us how often the model is correct when it predicts something as positive. For example: If 20 websites are flagged as vulnerable, and 15 are correct, Precision= 75%.

Formula: Precision =TP/(TP+FP)

4.2.2.3 F1 Score : The F1 Score balances precision and recall. It's useful when there's an uneven distribution of positive and negative cases, as it considers both false positives and false negatives.

Formula: F1 Score = 2*((Precision*Recall)/(Precision + Recall))

4.2.2.4 Recall : Recall focuses on how many actual positives the model identifies correctly. It's like asking: If there are 25 vulnerabilities and 15 are detected, Recall = 60%.

Formula: Recall =TP/ (TP +FN)

4.2.2.5 Confusion Matrix

In the project **WebSecurity**, the confusion matrix is simply a table used in the determination of performance regarding the model for vulnerability detection; it's where it compares the real result or actual existence with what the model predicts.

		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Explanation:

- **True Positive (TP)**: Websites that are vulnerable and are correctly identified by the model.
- **False Positive (FP)**: Websites that are not vulnerable but are incorrectly flagged as vulnerable.
- **True Negative (TN)**: Websites that are secure and are correctly identified as such.
- **False Negative (FN)**: Websites that are vulnerable but are incorrectly flagged as secure.

Chapter 5: Implementation and Testing

5.1 Implementation

The Implementation section describes how the main features were developed, such as the login system, the signup sections, scanning of the URL, crawling process, and report generation. It also covers the technologies and tools used, how these features interact with each other, and the security measures implemented to make its usage safe. This section also includes any pitfalls encountered during the development, their solution, and testing and optimization of the system so that everything works smoothly.

5.1.1 Tools Used

- Python: Backend Programming Language.
- Django: Framework used for development of the project.
- Draw.io: For creating diagrams like Architecture and ER
- LucidChart: For creating DFD and Gantt Chart
- MySQL: MySQLdatabase service that simplifies deploying and managing your databases.
- Vs Code: IDE for writing and running code.
- Microsoft Word: For preparing documentation.

5.1.2 Implementation details of Modules

The detail about some of the modules implemented in our system are:

- User Management Module

  The User Management Module allows the administrator to manage user accounts and control access to the platform. An administrator can view, edit, or delete user accounts and even trace user activities, such as scanning history. This module ensures that only authorized users can submit scans, view, download and prefer solutions of the reports. It's essential in managing user interactions so that the platform runs smoothly and securely.
- URL Submission and Verification Module

  This module enables users to input a website's URL to be scanned and pre-processes it for correctness. It checks the structure of the URL, the right protocols like HTTP/HTTPS, and the accessibility of the URL before proceeding. The system

prevents invalid or malicious URLs from being processed by validating the URL, hence smoothing the scanning process.

- Crawling Module

  The Crawling Module will crawl through an entire website once the URL is provided. It follows internal and external links within the website, making sure all accessible pages are crawled. This is a very important feature because it might find vulnerabilities hidden in some part of the site.

- Vulnerability Scanning Module

  The Vulnerability Scanning Module, after crawling, scans the collected data for security vulnerabilities such as XSS, SQL Injection, CSRF, and many more. It follows a set of predefined security rules and patterns to identify vulnerabilities on the website. The module scans page by page for any flaws; it categorizes vulnerabilities by severity and provides recommendations for mitigation.

- Report Generation Module

  The Report Generation Module summarizes the results of the scan into a report. This is a detailed report that includes a description of each detected vulnerability, its severity, and suggested fixes. It can be viewed online by users or downloaded in PDF for offline use and further action on the identified security issues.

- Database Management Module

  The Database Management Module manages all the data of the platform. It will securely store user accounts, the URLs submitted for scanning, the scan results, and vulnerability details. It assures that the data is securely stored in the database and can be accessed efficiently when needed. It also provides the facility for tracking user activities, storing historical data of scans, and ensures integrity and security.

5.2. Testing

Testing is an essential part of the software development process, ensuring that the system is reliable, functional, and meets the desired quality standards. It involves a thorough examination of both individual components and the system as a whole to identify and address any potential issues.

5.2.1. Test Cases for Unit Testing

Unit testing focuses on checking each module or component independently to ensure it performs its specific task correctly.

Table 5. 1:Test Case for User login/Sign up

Test Case ID	Test Case Description	Steps to Perform	Expected Result	Actual Result	Status
TC_01	Register new users with valid inputs	Username: "Sandip Khadka", Email: "sandip@gmail.com", Password: "sandip@123"	User successfully registered and redirected to the login page.	As Expected	Pass
TC_02	Login new user with new data to check registration works or not	Username: “Kanchan Bhujel” , Password: "Kanchan@123"	Login Successfully	As Expected	Pass
TC_03	User login with valid input	Username: “Kanchan Bhujel” , Password: "Kanchan@123"	Login successfully	As Expected	Pass
TC_04	User login with invalid input	Username: “ABC” , Password: "Abc@123"	Login failed display error message.	As Expected	Pass

Table 5. 2:Test case for the Vulnerability Scanning Module

Test Case ID	Test Case Description	Steps to be performed	Expected Result	Actual Result	Status
TC_01	Valid URL Submission for Vulnerability Scan	URL: "http://owasp.com"	The scan should start and show a progress bar. Upon completion, the vulnerability report should display.	As Expected	Pass
TC_02	Invalid URL Submission for Vulnerability Scan	URL: "http://invalidsite"	The system will not proceed.	As Expected	Pass

Table 5. 3:Test cases for the Report Generation Module

Test Case ID	Test Case Description	Steps to be performed	Expected Result	Actual Result	Status
TC_01	Generate Vulnerability Report for a Valid URL	URL**:** "http://owasp.com"	The report should display vulnerabilities (e.g., SQL injection, XSS).	As Expected	Pass
TC_02	Verify Report Download Functionality	URL: "http://owasp.com"	The report should download successfully in the pdf file format.	As Expected	Pass

5.2.2. Test Cases for System Testing

System testing examines the entire system to confirm that all components work together harmoniously. This phase ensures that the integration of modules is seamless and that the system provides a smooth user experience.

Table 5. 4:Test case for Performance Testing

Test Case ID	Test Case Description	Steps to be performed	Expected Result	Actual Result	Status
TC_01	Measure the time taken to load the homepage, login page, and signup page.	localhost/websecurity/home	Pages should load within 2-5 seconds.	As Expected	Pass
TC_02	Test Response Time for URL Scanning	URL: http://owasp.com	The scan should complete within 50 seconds.	As Expected	Pass
TC_03	Test the download speed and ensure file integrity.	localhost/websecurity/home/download/report/1234	Files should download quickly, and the report content should be accurate.	As Expected	Pass

Table 5. 5:Test Cases for Usability Testing

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status
TC-01	Test page navigation	Navigate between various pages or modules of web app	Navigation to be smooth.	As expected	Pass
TC-02	Verify clarity of error messages	Enter invalid data in forms or simulate incorrect actions	Error messages are clear, helpful and easy to understand.	As expected	Pass

5.3 Result Analysis

The Vulnerability Scanning Project is designed to provide accurate identification of security weaknesses based on system configurations and known threat patterns. To evaluate its performance and user impact, various metrics and real-world scenarios were analyzed, offering valuable insights into its effectiveness and potential areas for enhancement.

5.3.1 Evaluating model performance:

- Methodology

  Under the result analysis of the system, we conducted a comparison of our system result with the existing system results i.e. Burp Suite Number of Header detected in each of the tools.

Table 5. 6:Headers Detected by the burpsuit

Header	Value
server	Apache
content_type	text/html; charset=UTF-8
connection	Keep-Alive
x_powered_by	Not present
cache_control	public, no-transform
vary	Accept-Encoding
strict_transport_security	max-age=31536000; includeSubDomains; preload
x_content_type_options	nosniff
x_frame_options	SAMEORIGIN
x_xss_protection	1; mode=block

Table 5. 7:Headers detected by our system

Header	Value
server	Apache
x_frame_options	SAMEORIGIN
vary	Accept-Encoding
x_xss_protection	1; mode=block
strict_transport_security	max-age=31536000; includeSubDomains; preload
x_content_type_options	nosniff
referrer_policy	origin
cache_control	public, no-transform
content_length	177873
connection	Keep-Alive
content_type	text/html; charset=UTF-8

After running the test for same domain in both system we find out the difference using the cosine similarity in the result we collected and then based on it we calculated the different parameters of the the analysis.

Cosine similarity measures the cosine of the angle between two non-zero vectors, essentially calculating how similar the two vectors are. In this case, we will treat the presence of headers as vectors (where headers are treated as "features") and then compute the cosine similarity between the two sets of headers from the scans.

First, we'll modify the approach so that the comparison works on vectorized data (e.g., headers presence as binary values), and then compute the cosine similarity between the vectors representing each scan.

- Calculation

```
# Load the JSON results
scan_a = json.loads(scan_results_scanner_1)
scan_b = json.loads(scan_results_scanner_2)

# Get a sorted list of all headers in both scans (combined)
all_headers = sorted(set(scan_a.keys()).union(set(scan_b.keys())))

# Create binary vectors for the presence of headers in each scan
vector_a = [1 if header in scan_a else 0 for header in all_headers]
vector_b = [1 if header in scan_b else 0 for header in all_headers]

# Cosine similarity (for overall comparison)
cosine_sim = cosine_similarity([vector_a], [vector_b])[0][0]

print(f"Cosine Similarity between the two scans: {cosine_sim:.2f}")

# Calculate confusion matrix components based on header comparison
true_positives = sum([1 for i in range(len(vector_a)) if vector_a[i] == 1 and vector_b[i] == 1])
false_positives = sum([1 for i in range(len(vector_a)) if vector_a[i] == 0 and vector_b[i] == 1])
false_negatives = sum([1 for i in range(len(vector_a)) if vector_a[i] == 1 and vector_b[i] == 0])
true_negatives = sum([1 for i in range(len(vector_a)) if vector_a[i] == 0 and vector_b[i] == 0])

# Calculate Precision, Recall, F1 Score, and Accuracy
precision = precision_score(vector_a, vector_b)
recall = recall_score(vector_a, vector_b)
f1 = f1_score(vector_a, vector_b)
accuracy = accuracy_score(vector_a, vector_b)

# Output results
print("\nMetrics based on header comparison:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

Figure 5. 1: Comparision of each headers

- Evaluation matrix

  Precision = TP / (TP + FP)

  Recall = TP / (TP + FN)

f1_score = 2 * (precision * recall) / (precision + recall)

Accuracy = (TP + TN) / (TP+ TN + FP + FN)

```
Cosine Similarity between the two scans: 0.91

Metrics based on header comparison:
Accuracy: 0.83
Precision: 0.91
Recall: 0.91
F1 Score: 0.91
```

Figure 5. 2:Overall Accuracy, Precision, Recall and F1-Score of the system

We also plot the confusion matrix for the easy visualization

```
# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Present in Scan 1", "Not Present in Scan 1"],
            yticklabels=["Present in Scan 2", "Not Present in Scan 2"])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix: Header Comparison')
plt.show()
```

```
Cosine Similarity between the two scans: 0.91

Metrics based on header comparison:
Accuracy: 0.83
Precision: 0.91
Recall: 0.91
F1 Score: 0.91
```

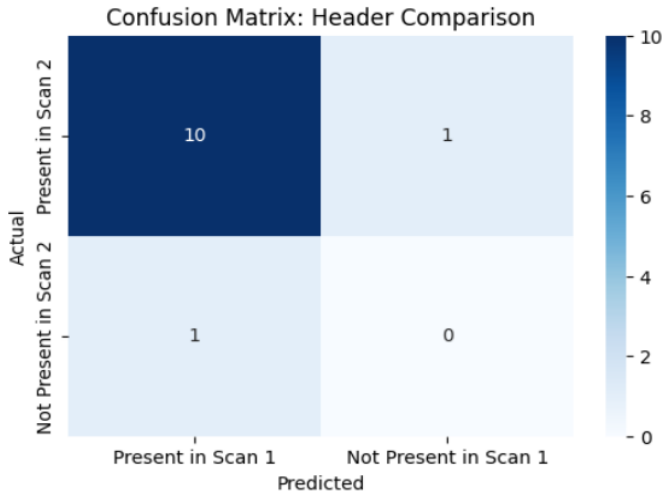


Figure 5.2: Confusion matrix generated by the model

- Conclusion:
  We compared our results with the existing system and concluded with an accuracy of 83%. However, this accuracy could be improved by incorporating crawler data, as most URLs would then be scanned. Additionally, the accuracy may vary depending on the tools and methods used.

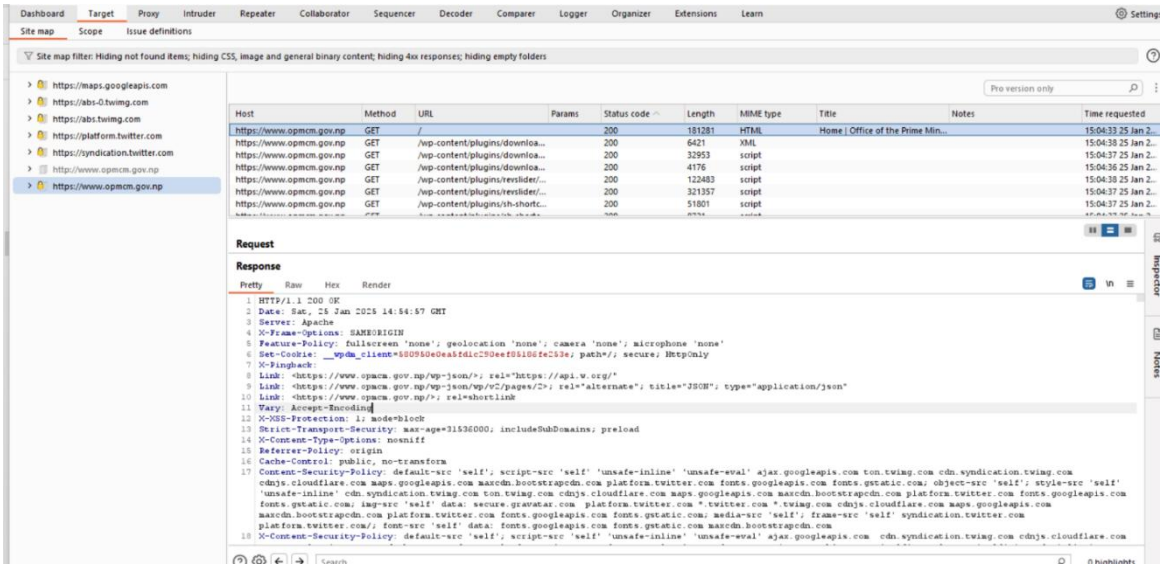


Figure 5.3: Screenshot for the test on burpsuit

Chapter 6: Conclusion and Future Recommendation

6.1 Conclusion

WebSecurity is a web-based set that enables users to scan their websites for vulnerabilities by combining Breadth-First Search for crawling the website. It allows users to enter the URLs in the search box for crawling, analyzing the website for some of its potential weaknesses in security. After scanning, a report is generated to show whether there is the presence of a vulnerability or not .The key performance indicators of the model include accuracy, precision, recall, and the F1 score. These metrics ensure that the model would effectively find vulnerabilities-true positives-while reducing the risks of false positives, or otherwise identifying a secure site as vulnerable, and false negatives, which would mean missing actual vulnerabilities. These metrics also provide a clear view of the balance between finding vulnerabilities and avoiding unnecessary alarms, which is very important for the reliability and trustworthiness of the WebSecurity platform.Overall, the WebSecurity website provides an easy-to-use, efficient, and reliable vulnerability scanning service. It can help website owners and developers to assess their site's security posture, identify potential risks, and take proactive steps to prevent cyber-attacks.

6.2 Future Recommendation

- Improved Detection Algorithms:
  Include more sophisticated machine learning models or hybrid algorithms, such as deep learning, to enhance the detection rate, particularly for complex vulnerabilities.

- Real-time Scanning:
  Provide real-time vulnerability scanning of websites so that users can monitor their website's security continuously instead of running manual scans from time to time.

- Broader Vulnerability Coverage:
  Widen the scope of vulnerabilities that can be detected to include new attack vectors such as zero-day exploits to keep the system updated with evolving threats.

- User Interface Improvement:
  Enhance the user's experience by introducing an even more intuitive and responsive UI that includes customized scan settings, more detailed reporting, and security recommendations.
- Integration with Other Security Tools:
  Enable the user to integrate **WebSecurity** with other security tools-for example, firewalls and anti-malware software-to provide a more comprehensive security assessment.
- Automated Remediation Suggestions:
  Provide automated suggestions or fixes for identified vulnerabilities, enabling users to directly act upon the issues in the platform itself.

References

[1] V. Dwivedi, "Web-securityand Analysis of Software Artifacts",2022.

[2] G. Yasmeen and S. Afaq, "The critical analysis of web application vulnerabilities.", pp. 22-37, 2023.

[3] Gibson, Dennis and Clive Harfield, "Amplifying victim vulnerability: Unanticipated harm and consequence in data breach notification policy.", *International Review of Victimology,* vol. 3, p. 29, 2023.

[4] M. G. Scott, Agile Estimating and Planning, 2024.

[5] M. C. S. K. and S. E., "Cyber security challenges and its emerging trends on latest technologies.", *IOP Conference Series: Materials Science and Engineering.,* vol. 981, p. 2, 2020.

[6] Y. Shamoo, "Comparative Analysis of LLMs vs. Traditional Methods in Vulnerability Detection.," *Application of Large Language Models (LLMs) for Software Vulnerability Detection,* p. 335, 2024.

[7] S. Nepal, "Cybersecurity in Foreign Policy: Nepal's Outlook and Considerations on," tucl.edu.np, 2022.

[8] P. Collumba and V. W. Nick, Critical Security Studies, London: Routledge, 2020, p. 294.

[9] J. Dybka, ""Qualys Inc"," 2020.

[10] HostedScan, "HostedScan",*HostedScan LLC,* [Online], 2019.

[11] S. I., "ImmuneWeb"," [Online], 2024.

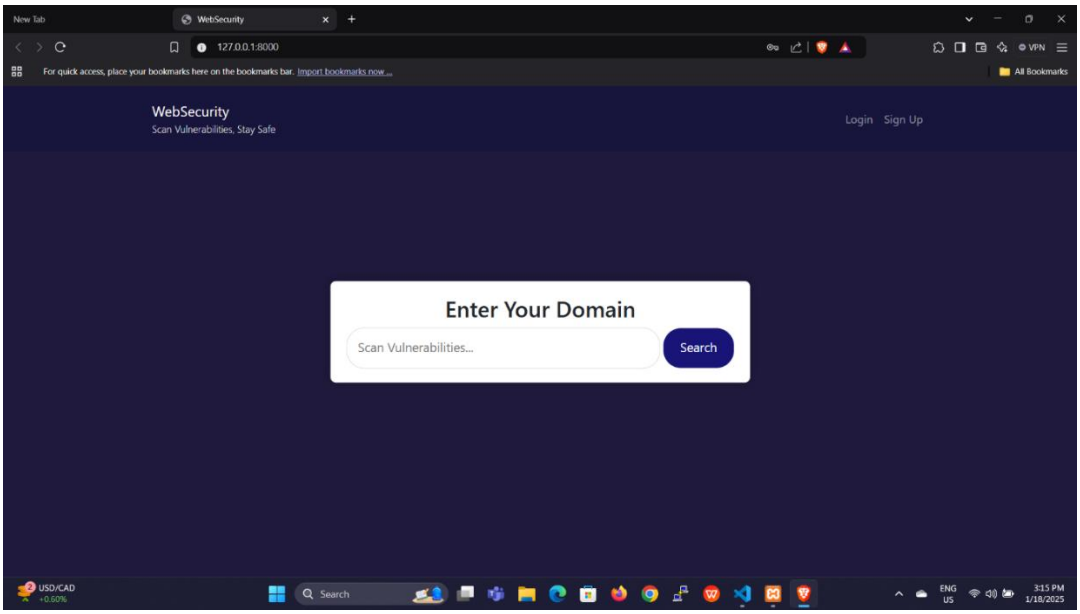
[12] A. Lavrenovas, & Melon and J. R. F, "HTTP security headers analysis of top one million websites.," *10th International Conference on Cyber Conflict (CyCon). ,* p. 26, 2018.

[13] SAMBATH and T. V. Dr, "A COMPLETE ANALYSIS OF SECURITY HEADERS AND SERVER CONFIGURATION," *Journal of Nonlinear Analysis and Optimization,* vol. 15, no. 2, p. 5, 2024.

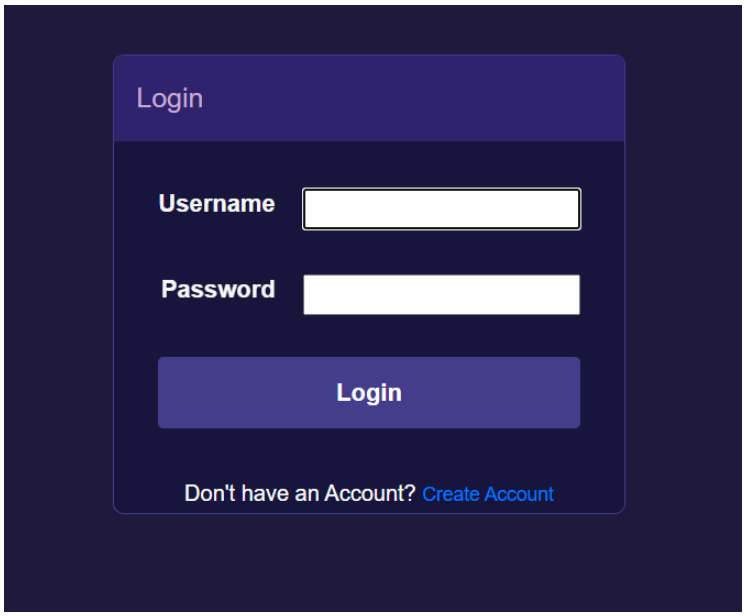
Appendices

Screenshots of the web application:

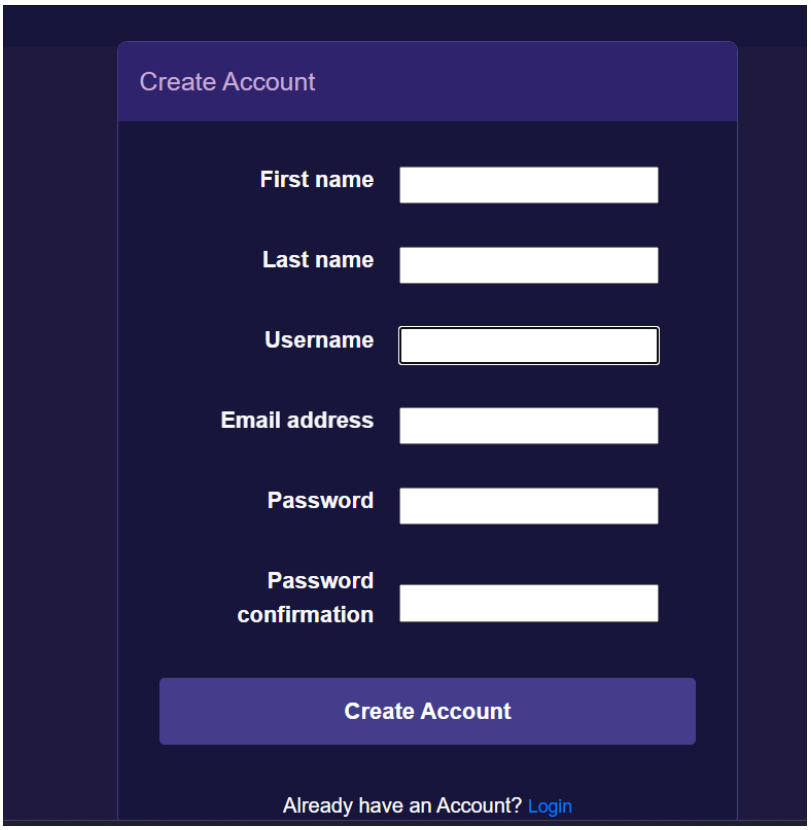
a) Home Page



b) Login Page:



c) Sign Up Page



d) Report Page