Kingdom of Saudi Arabia
Ministry of Education
Jouf University
Applied College

THESIS

In fulfillment of the requirements for the degree of
Diploma in Cybersecurity

VULNERASCAN

A FLASK-BASED WEB APPLICATION FOR NETWORK SECURITY SCANNING

PROJECT SUPERVISOR
Supervisor NAME
Associate Professor

GROUP MEMBERS

Aisha Saleh Al-Saqabi, 461208107
Sarah Adel Al-Fuwayhi, 462205364
Reem Mohammed Al-Anzi, 462204583
Rahaf Atef Al-Saleh, 461204698

Dr. Ibrahim Alrashdi

2025/2026
Semester 2

PROJECT IN BRIEF

PROJECT TITLE:	A Flask-Based Web app for Network Security Scanning (VulneraScan)
ORGANIZATION:	Jouf University - College of Computer and Information Science
OBJECTIVES:	1. Develop a user-friendly web interface for automated network discovery and service auditing. 2. Implement a risk-scoring algorithm to identify high-risk services and open ports. 3. Design a historical logging system to track network changes over time. 4. Visualize security data through interactive charts and downloadable audit summaries.
UNDERTAKEN BY:	• Aisha Saleh Al-Saqabi (461208107) • Sarah Adel Al-Fuwayhi (462205364) • Reem Mohammed Al-Anzi (462204583) • Rahaf Atef Al-Saleh (461204698)
SUPERVISED BY	
DATE STARTED:	19 / 02 / 2026
DATE COMPLETED:	01 / 06 / 2026
TECHNOLOGIES USED:	• **Languages & Frameworks:** Python 3.x, Flask, Jinja2 Templates. • **Libraries:** python-nmap, Flask-SQLAlchemy, Pandas. • **Frontend:** Bootstrap 5, Chart.js, Bootstrap Icons. • **Database:** SQLite. • **Scanning Engine:** Nmap Security Scanner.
OPERATING SYSTEM:	Windows 10/11 or Kali Linux.

SYSTEM USED:	• **Hardware:** Workstation (Laptop/PC) with Intel i5/i7, 8GB RAM, and 500MB storage. • **Virtualization:** VirtualBox or VMware for "Target Machines" simulation.

ACKNOWLEDGEMENT

First and foremost, we would like to express our deepest gratitude to **Allah** Almighty for giving us the strength, patience, and ability to complete this graduation project successfully.
We wish to extend our sincere thanks and appreciation to the **College of Computer and Information Science at Jouf University** for providing a supportive academic environment and the necessary facilities to conduct our research and development.
Our heartfelt gratitude goes to our supervisor, **[]** , for their invaluable guidance, continuous encouragement, and insightful feedback throughout the different stages of this project. Their expertise and dedication have been instrumental in the completion of this work.
We would also like to thank all the faculty members who have shared their knowledge with us over the years. Special thanks are also due to our families and friends for their unwavering support, prayers, and motivation during our academic journey.
Finally, we appreciate the efforts of our team members who worked collaboratively to bring "**VulneraScan**" to life.

DECLARATION

We hereby declare that this software, neither as a whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No portion of the work presented has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

CERTIFICATION

It is certified that the contents and form of the project entitled ……. submitted by ……..., ……., and …… has been found satisfactory for the requirements.

For the award of the degree of

Diploma in Applied computing and network technology

Supervisor: (NAME AND SIGN) .....

Committee Member 1: (NAME AND SIGN).....

Committee Member 2: (NAME AND SIGN).....

Dated: .....

ABSTRACT

In the modern digital landscape, ensuring network security and identifying vulnerabilities is a critical challenge for organizations of all sizes. While powerful tools like Nmap exist for network auditing, their reliance on complex command-line interfaces (CLI) often presents a barrier for junior IT staff and small organizations. This project introduces "**VulneraScan**", a centralized, web-based application designed to simplify and automate the process of network security scanning and risk assessment.

Developed using the **Flask (Python)** framework, VulneraScan bridges the gap between technical raw data and actionable security insights. The system integrates the **Nmap** engine via the python-nmap library to perform automated network discovery and service auditing. A core feature of the application is its intelligent logic layer, which categorizes detected ports and services into prioritized risk levels (High, Medium, and Low). To enhance usability, the platform features a real-time dashboard powered by **Chart.js**, providing visual representations of a network's security posture. Furthermore, the system includes a historical logging feature using an **SQLite** database, allowing administrators to track network changes and vulnerability trends over time.

The results of this project provide a user-friendly, visual, and efficient alternative to traditional CLI tools. By automating network auditing and offering structured reporting, **VulneraScan** empowers IT teams to identify security gaps more quickly and respond to potential threats more effectively, ultimately improving the overall security infrastructure of the organization.

Keywords: Network Security, Vulnerability Scanning, Flask, Nmap, Risk Assessment, Data Visualization, Cybersecurity Dashboard.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION

Chapter 1

PROJECT OVERVIEW

1.1 Introduction

Small and Medium Enterprises (SMEs) frequently encounter difficulties in effectively monitoring and securing their network environments due to limited resources and expertise. Traditional command-line scanning tools, while powerful, require a level of technical proficiency that may be beyond the reach of junior IT staff or organizations without specialized security teams. The absence of centralized dashboards, historical tracking, and visual risk analysis in these tools further complicates the process of long-term monitoring and auditing.

Consequently, SMEs are at an increased risk of undetected vulnerabilities, leading to potential security breaches and operational setbacks. The motivation behind this project is to develop a centralized, web-based platform, "**VulneraScan**", that simplifies network vulnerability assessment, enhances security visibility, and empowers SMEs to proactively manage their network security posture.

1.2 Problem Description

Many small to medium-sized organizations face significant challenges regarding network visibility. The primary issues identified include:

- **Complexity of Tools:** Professional scanning tools often lack a graphical user interface (GUI), making them difficult for junior IT staff or non-technical administrators to operate effectively.
- **Data Interpretation:** Raw scanning logs are often voluminous and difficult to interpret, leading to missed security gaps.
- **Lack of Historical Tracking:** Standard command-line tools do not inherently store scan history, making it impossible to track changes in the network over time or perform trend analysis.
- **Delayed Response:** Without a centralized platform and automated risk categorization, response times to potential threats are often delayed.

1.3 Proposed Solution

To address these challenges, we propose the development of **VulneraScan**. This solution is a centralized web platform that simplifies vulnerability detection through:

- **Automated Scanning:** Integrating the Nmap engine directly into a web interface via Python.
- **Intelligent Risk Scoring:** A logic layer that automatically categorizes detected open ports and services into risk levels (High, Medium, and Low).
- **Visual Dashboard:** Utilizing modern visualization libraries (Chart.js) to provide a real-time overview of the network's security posture.
- **Persistent Storage:** An integrated SQLite database to store scan history, enabling administrators to perform audits and track network changes over time.

1.4 Objective of the project

The primary objective of this project is to design and implement "VulneraScan," a Flask-based web application that integrates the Nmap scanning engine to automate network discovery and service auditing. The specific objectives include:

1. **Scanning Layer**: Integrate the Nmap engine using the python-nmap library to perform automated network scans on specified targets, retrieving detailed information about open ports, running services, and host status.
2. **Logic and Risk Analysis Layer**: Process and analyze scan results using a risk-scoring mechanism, categorizing detected ports and services into predefined risk levels (High, Medium, Low) based on service type, port sensitivity, and potential exposure.
3. **Data Management Layer**: Store all scan results in an SQLite database, enabling structured logging and historical tracking, which supports continuous monitoring and long-term security auditing.
4. **Visualization and Reporting Layer**: Provide an interactive dashboard built with Bootstrap and Chart.js to display security posture metrics in graphical formats, and generate structured reports to support documentation and compliance needs.

1.5 Methodology

The project follows a structured development approach divided into four functional layers:

1. **Scanning Layer:** Using the python-nmap library to interact with the Nmap engine for active network reconnaissance.
2. **Logic Layer (Processing):** Developing Python scripts to parse raw XML/JSON data from Nmap and apply a scoring system to evaluate the severity of open services.
3. **Visualization Layer:** Building a responsive frontend using Bootstrap 5 and Chart.js to present complex security data in an easy-to-read format.
4. **Reporting & Storage Layer:** Implementing an SQLite database with SQLAlchemy to manage scan records and generate structured reports for compliance and auditing.

1.6 Conclusion

VulneraScan aims to empower IT administrators by providing a powerful yet accessible tool for network security management. By moving away from complex manual commands and towards an automated, visual, and history-aware platform, organizations can significantly enhance their ability to detect vulnerabilities. This project not only simplifies the auditing process but also ensures that security data is transformed into actionable intelligence, ultimately strengthening the organization's overall cybersecurity defense.

Chapter 2

SYSTEM ANALYSIS

2.1 Introduction

In the contemporary digital landscape, small and medium-sized enterprises (SMEs) find themselves at an increasingly precarious intersection of technological advancement and cyber vulnerability. With the rapid digitization of business operations, SMEs have become attractive targets for cybercriminals who exploit their limited resources and often insufficient cybersecurity expertise. According to recent studies, more than 40% of cyberattacks are aimed at SMEs, highlighting a stark contrast between their critical role in the global economy and their inadequate preparedness against evolving cyber threats. This vulnerability is further compounded by a lack of comprehensive cybersecurity training, leading to a pervasive underestimation of potential risks within these organizations.

Traditional network vulnerability assessment tools, such as Nmap, serve as powerful instruments for identifying and analyzing security weaknesses in network architectures. Nmap, known for its robust scanning capabilities, can detect live hosts, open ports, and various services running on these hosts. While these tools are invaluable in the hands of cybersecurity professionals, they often necessitate a certain level of proficiency with command-line interfaces and in-depth technical knowledge to interpret the complex output they generate. For non-expert users within SMEs—many of whom are tasked with wearing multiple hats in their roles—this steep learning curve can prove to be a significant barrier to effective cybersecurity implementation.

To address the pressing need for accessible cybersecurity solutions, this chapter presents a comprehensive analysis of the VulneraScan project, which aims to democratize cyber defense by offering a user-friendly, Flask-based web application designed specifically for network security scanning. Flask, a micro web framework for Python, is

lauded for its simplicity and flexibility, making it an ideal choice for developing applications that require rapid deployment and intuitive user interfaces. The VulneraScan application stands out as a pivotal resource for SMEs, fostering a proactive approach to cybersecurity that aligns with the operational realities of smaller organizations.

VulneraScan operates on the principle of simplifying vulnerability assessments without sacrificing the depth of the technical analysis. By providing a centralized platform where users can easily initiate scans, review findings, and understand the implications of identified vulnerabilities, the application demystifies the previously arcane world of cybersecurity for SME employees. For example, the interface includes user-friendly dashboards that visualize scanning results in real-time, actionable insights tailored to the user's comprehension level, and step-by-step remediation guidelines that empower users to address vulnerabilities proactively.

2.2 Requirements

The requirements of the VulneraScan system are categorized into functional requirements, which define what the system does, and non-functional requirements, which define how the system performs its functions.

2.3 Specification Requirements

VulneraScan is specified to operate as a centralized web-based platform. The technical specifications include:

- **Backend:** Python 3.x using the Flask framework.
- **Scanning Engine:** Nmap (Network Mapper) integrated via the python-nmap library.
- **Database:** SQLite for local audit logging and historical tracking.
- **Frontend:** HTML5, CSS3, Bootstrap 5 for responsiveness, and Chart.js for data visualization.
- **Environment:** Compatibility with Windows 10/11 and Kali Linux for scanning local network targets.

2.4 Functional Requirements

VulneraScan aims to fulfill the following functional requirements:

- **Automated Network Scanning**: Perform comprehensive scans of specified network targets to identify open ports, running services, and host statuses.
- **Risk Assessment and Classification**: Analyze scan results to categorize detected vulnerabilities into predefined risk levels (High, Medium, Low) based on service type, port sensitivity, and potential exposure.
- **Data Storage and Historical Tracking**: Store scan results in a structured database to facilitate historical tracking, comparison between scans, and continuous monitoring.
- **Interactive Visualization and Reporting**: Provide a user-friendly dashboard with graphical representations of security metrics and generate structured reports for documentation and compliance purposes.

2.5 Non-Functional Requirements

The non-functional requirements for VulneraScan include:

- **Usability**: Design an intuitive web interface that simplifies complex security data, making it accessible to users without extensive technical backgrounds.
- **Scalability**: Ensure the system can handle increasing numbers of network targets and scan data as SMEs expand their operations.
- **Security**: Implement security best practices to protect the application from potential cyber threats, including secure coding practices and regular security audits.
- **Performance**: Optimize the system for efficient processing and quick response times to enhance user experience.

2.6 Use Case Diagram

The Use Case Diagram for **VulneraScan** illustrates the system's core functionalities and how the primary actor interacts with them.

- **Primary Actor: IT Administrator / SME User**, representing the individual responsible for monitoring the local network.
- **System Boundary:** The **VulneraScan System**, which encapsulates all the application's modules including scanning, logic, and data management.
- **Key Use Cases:**
    1. **Initiate Network Scan:** Allows the user to input a target IP or range and trigger the Nmap engine.
    2. **View Real-time Results:** Displays the active ports and services discovered during the scan.
    3. **View Visualization Dashboard:** Presents graphical analytics (charts/graphs) related to the network's risk posture.
    4. **Manage & Compare Scan History:** Enables the retrieval of previous records from the SQLite database to track changes over time.
    5. **Generate Security Reports:** Facilitates the export of scan summaries into structured formats for auditing purposes.

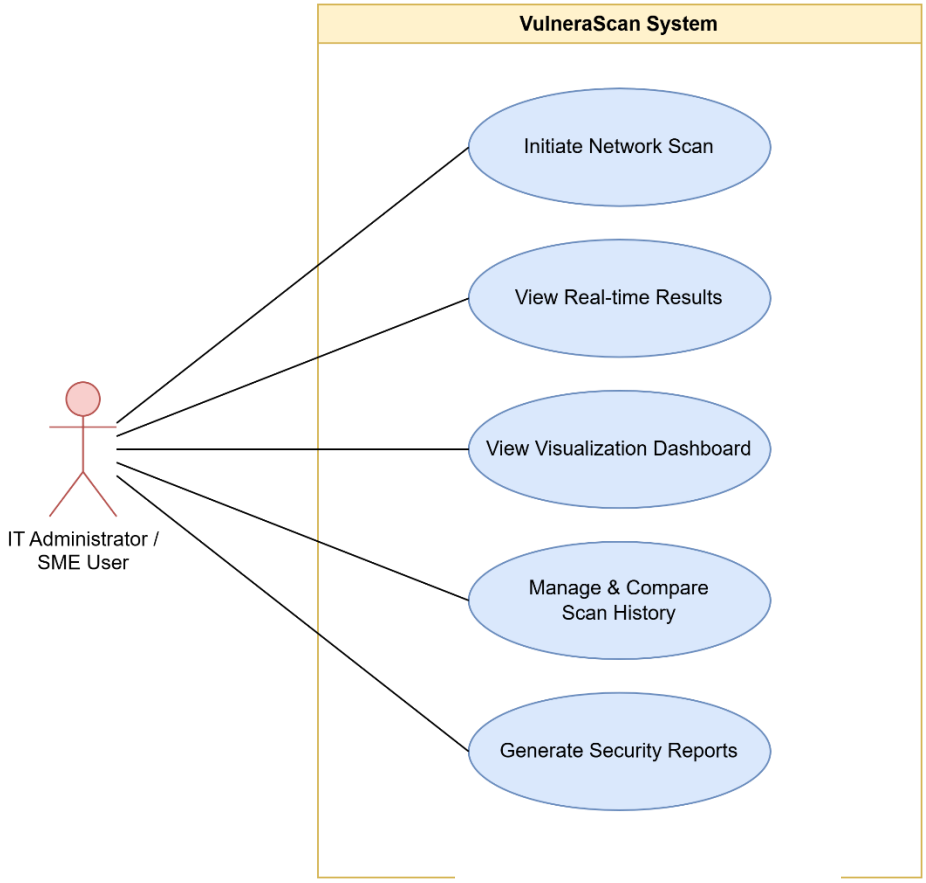


Figure 1: Use Case Diagram

2.7 General Constraints

The development and deployment of VulneraScan are subject to the following constraints:

- **Resource Limitations**: Limited computational resources may affect the scalability and performance of the system.
- **Security Considerations**: Ensuring the security of the application is paramount, requiring adherence to best practices and regular security assessments.
- **Compliance Requirements**: The system must comply with relevant industry standards and regulations to ensure its applicability and acceptance in various organizational contexts.

2.8 Product Requirements

- **Compatibility:** Support for modern browsers including Google Chrome, Mozilla Firefox, and Microsoft Edge.

- **Integration:** The system should be able to scan standard networking equipment (routers, switches) and servers without requiring client-side agents.
- **Documentation:** A comprehensive user manual detailing installation, scan initiation, and report interpretation must be provided.

2.9 Conclusion

VulneraScan is designed to address the challenges faced by SMEs in network vulnerability assessment by providing an accessible, scalable, and secure web-based platform. By integrating automated scanning, risk analysis, data management, and visualization into a unified system, VulneraScan aims to enhance the security posture of SMEs, enabling them to proactively identify and mitigate network vulnerabilities without the need for high-level technical expertise.

Chapter 3

SYSTEM DESIGN

3.1 Introduction

The design of **VulneraScan**, a Flask-based web application for network security scanning, is centered on creating a user-friendly platform that simplifies the process of network vulnerability assessment. By integrating the Nmap scanning engine within a web interface, VulneraScan aims to transform complex raw scan outputs into structured, interactive, and actionable security insights. This chapter outlines the system's architecture, including its layered design approach, sequence and activity diagrams, class and entity-relationship diagrams, database schema, and data dictionary.

3.2 Sequence Diagrams

Sequence diagrams illustrate the interactions between various components of the system over time, providing a clear understanding of the workflow and data flow within VulneraScan.

The interaction flows through four main phases:

1. **User Initiation:** The user logs into the interface, enters the target IP, and submits the form.
2. **Scan Execution:** The Flask backend triggers the Nmap engine via the python-nmap library.
3. **Data Processing:** The backend receives raw data, extracts open ports/services, and assigns risk scores.
4. **Visualization:** Processed results are stored in the database and rendered on the dashboard for the user.

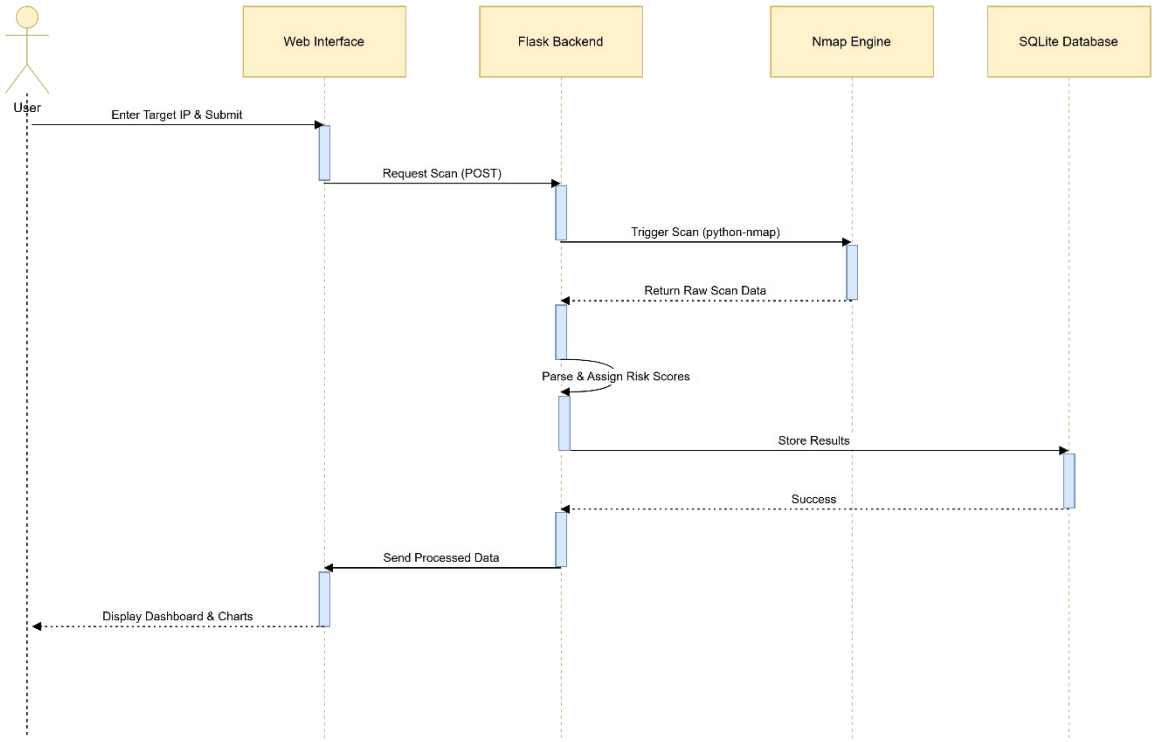


Figure 2: Sequence diagrams

3.3 Activity Diagrams

Activity diagrams depict the dynamic aspects of the system, highlighting the flow of control and data during various operations.

- **User Authentication:** Ensures only authorized users access the dashboard.
- **Scanning Workflow:** Includes input validation to ensure IP addresses are in the correct format before initiating Nmap.
- **Report Generation:** Shows the flow from retrieving database records to compiling them into a downloadable format (PDF/CSV).

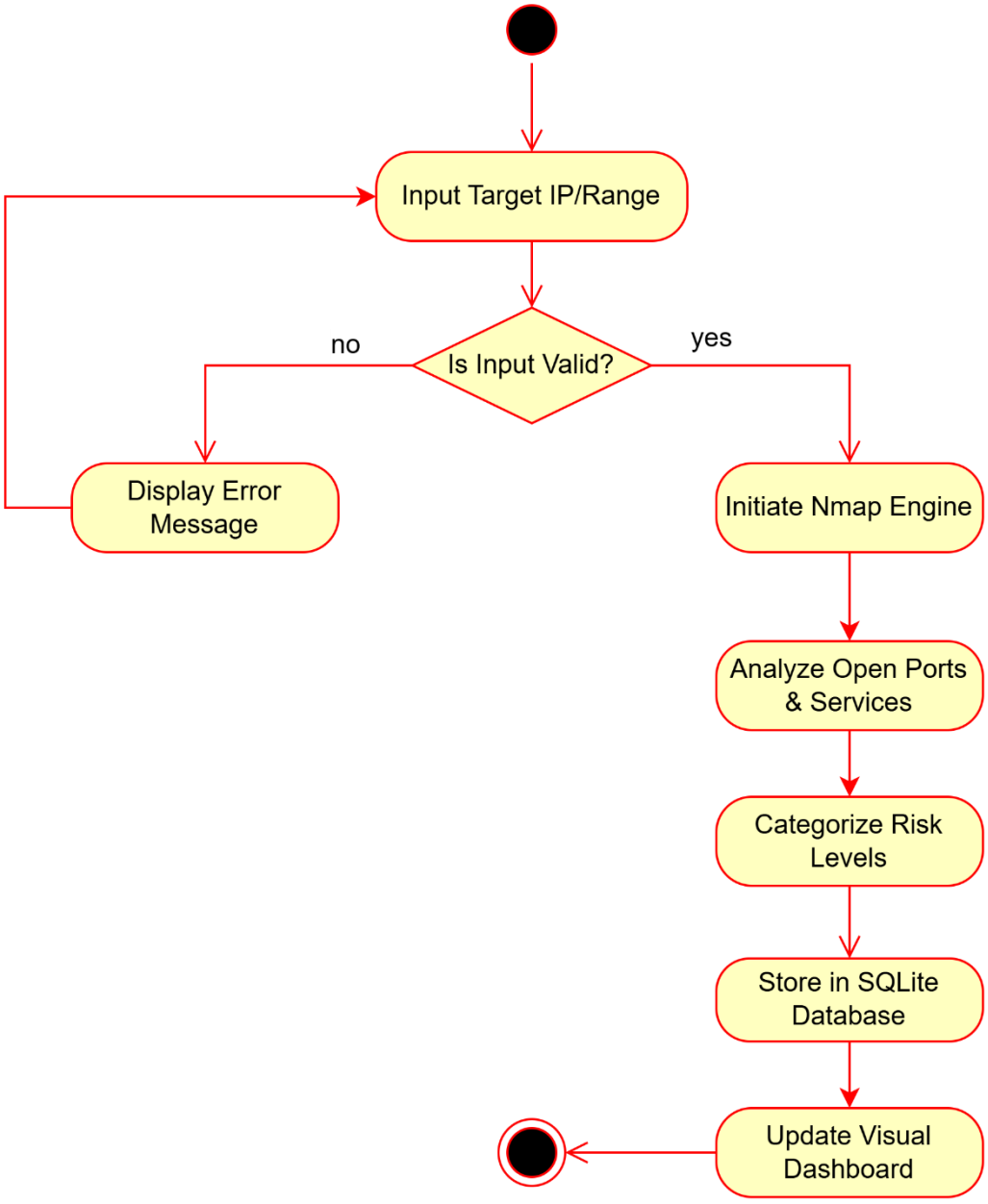


Figure 3: Activity diagrams

3.4 Class Diagram

The class diagram provides a static view of the system's structure, detailing the attributes and methods of each component.

- **User Class:** Manages authentication and profiles.
- **Scan Class:** Coordinates the target information and timing of scans.
- **Vulnerability Class:** The core logic class that parses port data and assigns severity.
- **Report Class:** Handles the formatting and exportation of data.

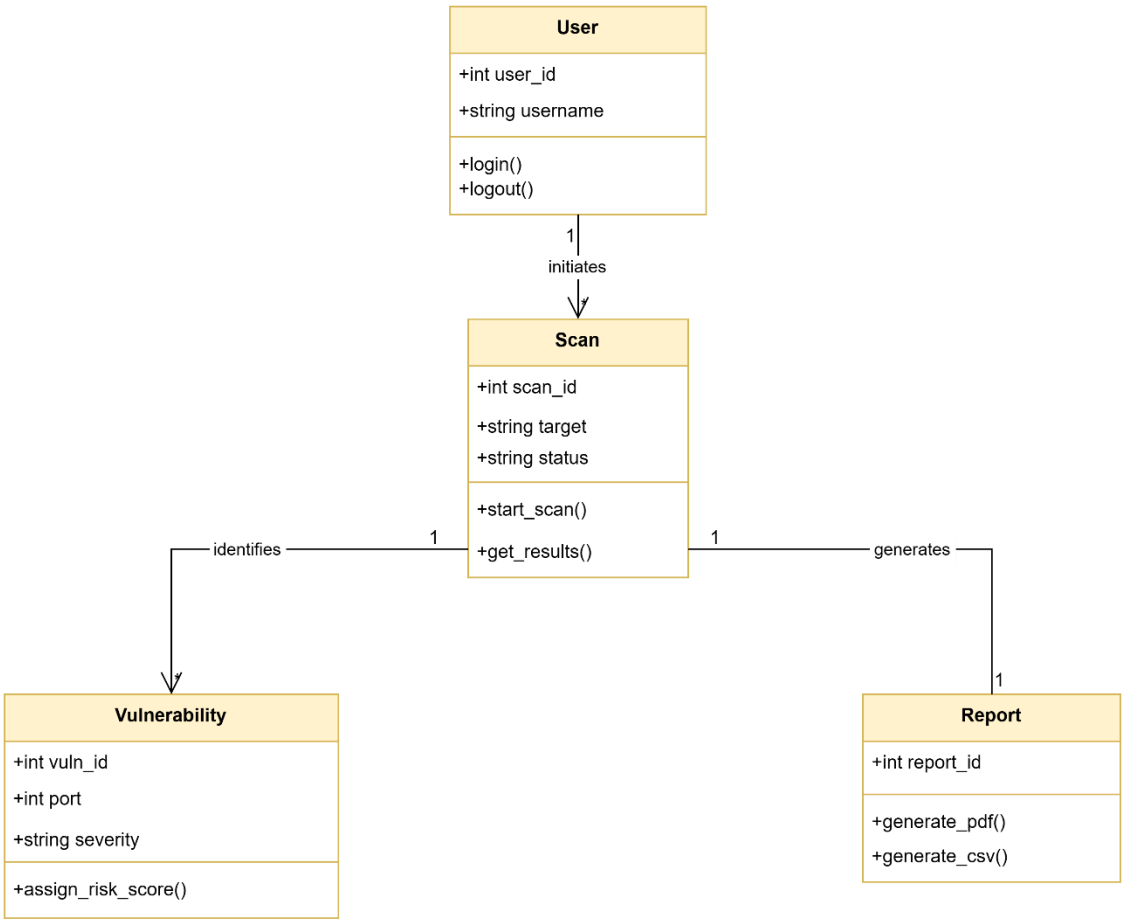


Figure 4: class diagram

3.5 Entity Relationship Diagram

The Entity Relationship Diagram (ERD) illustrates the logical structure of the database and the relationships between the core entities of the **VulneraScan** system. It defines how data is connected and ensures data integrity.

Entities:

- **User:** Represents the administrator who manages the system.
- **Scan:** Represents a single scanning session performed on a target.
- **Vulnerability:** Represents a specific security finding (open port/service) detected during a scan.
- **Report:** Represents the final document generated from scan results.

Relationships:

- **User to Scan:** One-to-Many (1:N) - One user can initiate multiple scan sessions.
- **Scan to Vulnerability:** One-to-Many (1:N) - One scan can identify multiple vulnerabilities.
- **Scan to Report:** One-to-One (1:1) - Each scan produces exactly one structured report.

3.6 Database Diagram

The database diagram represents the physical schema implemented in **SQLite**. It defines the tables, their primary keys (PK), foreign keys (FK), and data types.

1. **Users Table:** Stores credentials for authentication.
2. **Scans Table:** Logs every execution of the Nmap engine, including the target IP and timestamps.
3. **Vulnerabilities Table:** The most dynamic table, storing every port, service name, and its calculated risk level (High, Medium, Low).
4. **Reports Table:** Stores metadata about generated PDF/CSV files.

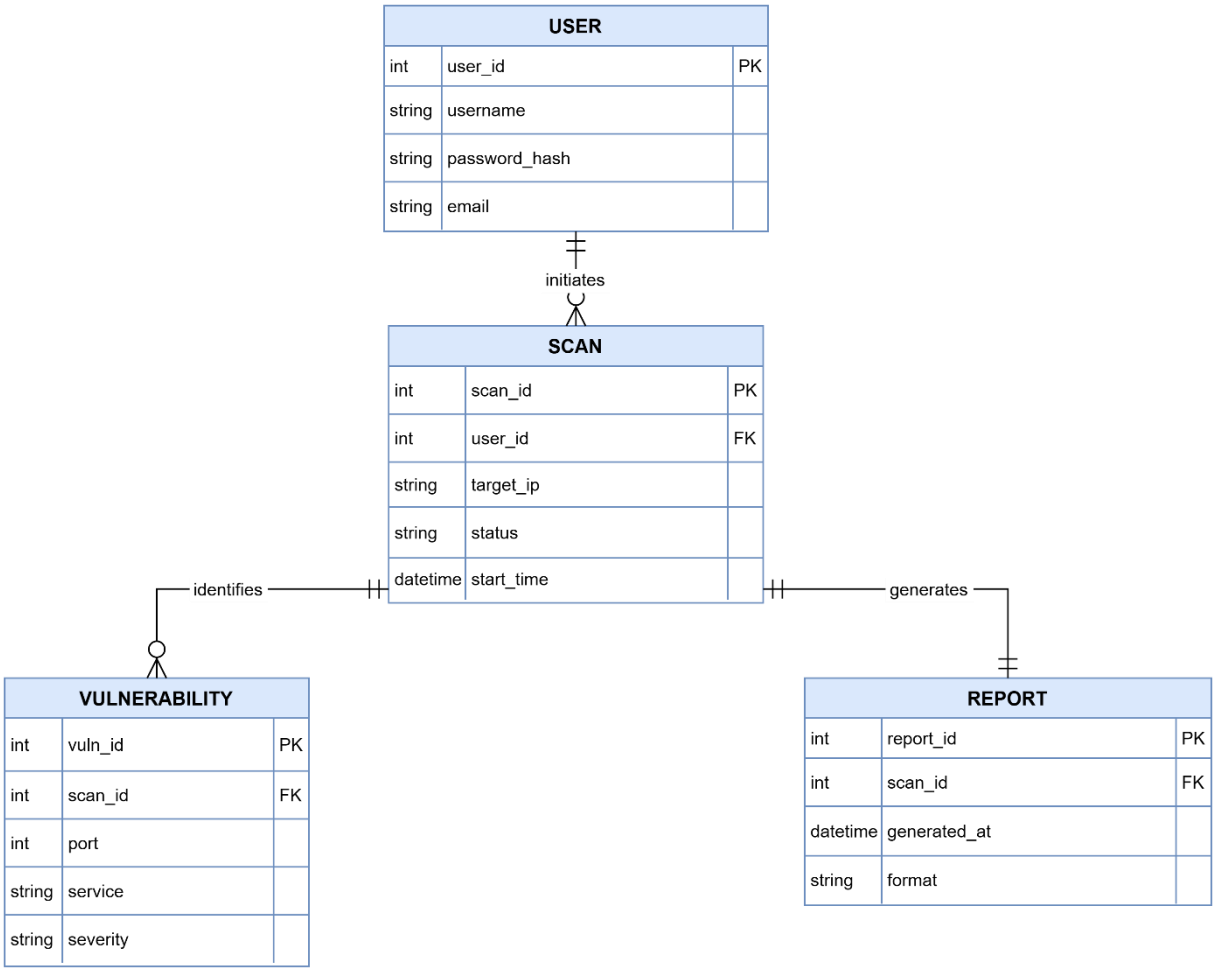


Figure 5: Entity Relationship Diagram (ERD)

3.7 Data Dictionary

The data dictionary provides a detailed breakdown of each table's attributes, ensuring a clear understanding of the data storage requirements.

Table 1: Users Table

Field Name	Data Type	Key	Nullable	Description
user_id	Integer	PK	No	Unique identifier for each user (Auto-increment).
username	Varchar(50)	-	No	Unique name used for login.
password_hash	Varchar(255)	-	No	Secured (hashed) password.
email	Varchar(100)	-	No	User's contact email.

Table 2: Scans Table

Field Name	Data Type	Key	Nullable	Description
scan_id	Integer	PK	No	Unique identifier for the scan session.
user_id	Integer	FK	No	Links the scan to the user who started it.
target	Varchar(100)	-	No	The target IP address or network range.
status	Varchar(20)	-	No	Current state (Completed, Failed, Scanning).
start_time	DateTime	-	No	The timestamp when the scan began.

Table 3: Vulnerabilities Table

Field Name	Data Type	Key	Nullable	Description
vuln_id	Integer	PK	No	Unique identifier for the vulnerability entry.
scan_id	Integer	FK	No	Links the vulnerability to a specific scan session.
port	Integer	-	No	The detected open port number (e.g., 80, 443).
service	Varchar(50)	-	Yes	The service name (e.g., HTTP, SSH, FTP).
severity	Varchar(10)	-	No	Risk level: High (Red), Medium (Orange), Low (Green).
description	Text	-	Yes	Detailed info about the vulnerability risk.

Table 4: Reports Table

Field Name	Data Type	Key	Nullable	Description
report_id	Integer	PK	No	Unique identifier for the report.
scan_id	Integer	FK	No	Links the report to the corresponding scan.
generated_on	DateTime	-	No	Date and time the report was created.
format	Varchar(10)	-	No	File format (e.g., 'PDF' or 'CSV').

3.8 Conclusion

We have presented in this chapter the complete system design for the **VulneraScan** application. By establishing a robust architectural framework, we have detailed the behavioral aspects through sequence and activity diagrams, as well as the structural foundations through class and ER diagrams. The database design and data dictionary ensure that all scan data—from raw Nmap logs to processed risk assessments—is stored efficiently and accurately. This design phase serves as the essential blueprint for the implementation stage, ensuring that the final product meets the security needs of SMEs with a scalable, modular, and user-centric approach.

Chapter 4

IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter describes the transition from the theoretical design phase to the practical realization of **VulneraScan**. It details the development environment, the technologies employed, and the implementation of core features such as the Nmap integration and risk-scoring logic. Furthermore, this chapter covers the testing strategies used to verify the system's functionality, accuracy, and security, ensuring that the final product meets the requirements of SMEs.

4.2 Technologies Used

To build a robust and scalable network scanner, the following technology stack was selected:

- **Programming Language: Python 3.x**, chosen for its extensive libraries in cybersecurity and web development.
- **Web Framework: Flask**, used for its lightweight nature and ease of routing.
- **Scanning Engine: Nmap (Network Mapper)**, the industry-standard tool for network discovery.
- **Database: SQLite**, a serverless relational database used for efficient local storage of scan history.
- **Frontend Frameworks: Bootstrap 5** for responsive UI design and **Chart.js** for dynamic data visualization.
- **Development Tools:** Visual Studio Code (IDE).

4.3 System Interfaces

The User Interface (UI) of VulneraScan is designed for simplicity and clarity to ensure accessibility for users with varying technical backgrounds. **Developed using Flask, HTML, CSS, and JavaScript, the UI allows users to input target IP addresses or domains, initiate scans, and view results.**

The interface consists of the following components:

1. **Scanning Dashboard:** A centralized form where users can enter target parameters and trigger the Nmap engine.

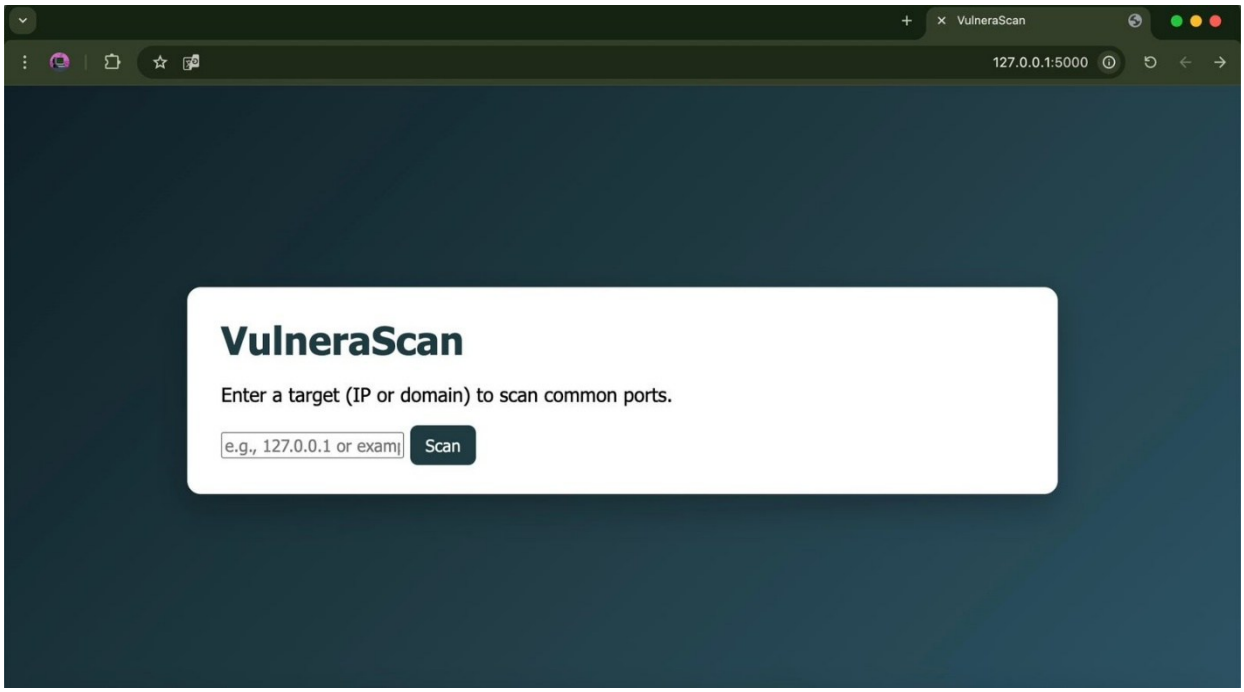


Figure 6: Dashboard Interface

2. **Result:** Dynamic tables and charts that display discovered ports, services, and their associated risk levels.

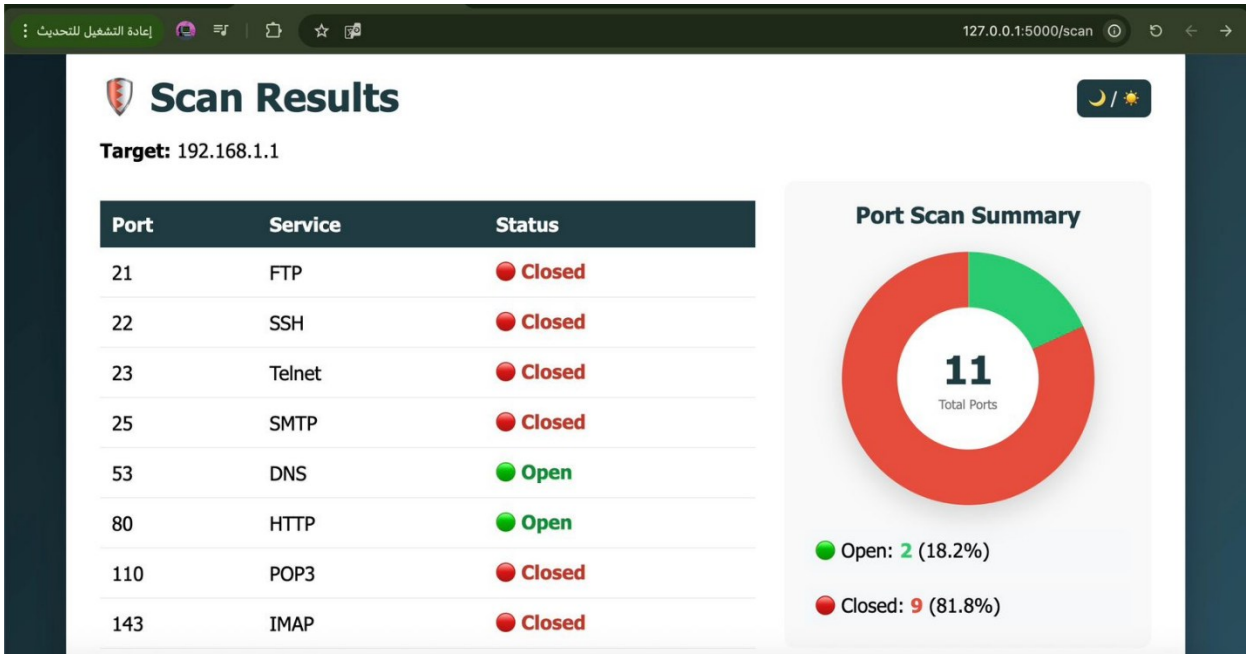


Figure 7: Result Visualization

3. **Historical Logs:** A management interface to review previous scanning sessions stored in the database.

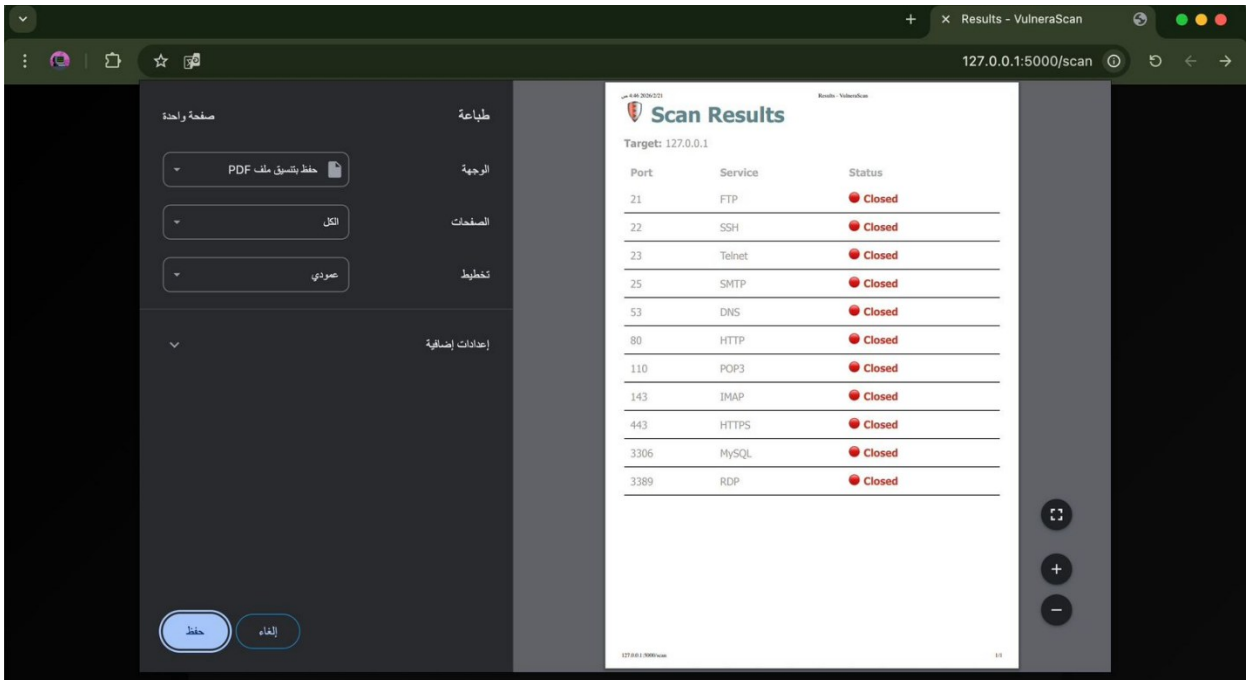


Figure 8: Historical Logs

4.4 System Coding

The implementation of VulneraScan is categorized into three functional layers:

- **Frontend Layer:** Built with **HTML, CSS, and JavaScript** to provide a dynamic experience. JavaScript is specifically used to handle asynchronous scan requests and update the UI without page reloads.
- **Backend Layer:** The **Flask** application serves as the bridge, receiving user inputs from the UI, executing the python-nmap scripts, and processing the raw output.
- **Logic Layer:** A Python-based risk assessment module that analyzes service versions and port numbers to categorize them into High, Medium, or Low risk levels.

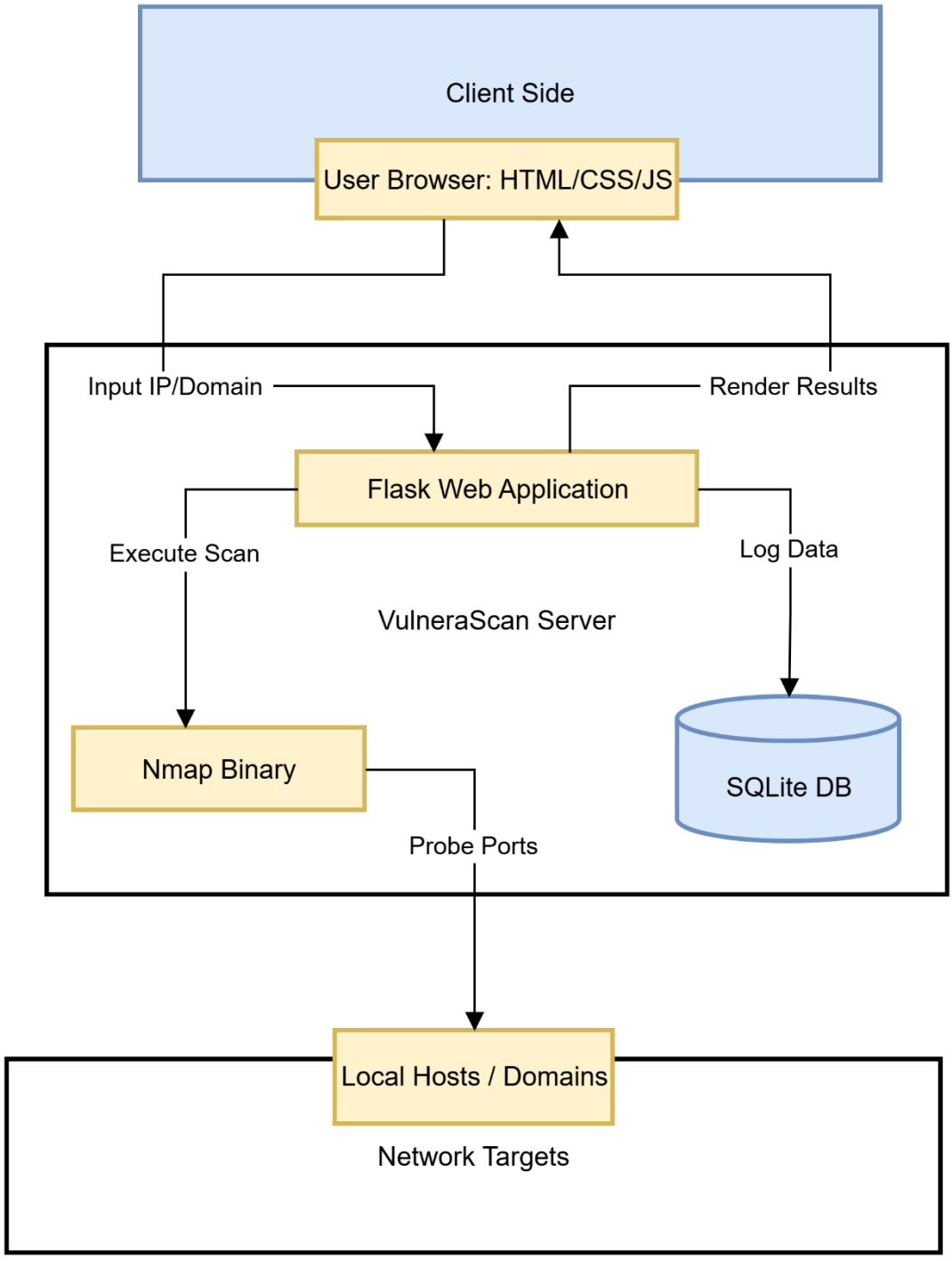
4.5 System Testing

Testing was conducted in a controlled environment to verify the reliability of the UI and the accuracy of the scanning engine:

1. **Unit Testing:** Testing the UI forms to ensure they correctly capture IP addresses and domain names.
2. **Functional Testing:** Confirming that the "Initiate Scan" button successfully triggers the Nmap engine and returns data to the UI.
3. **Compatibility Testing:** Ensuring the HTML/CSS layout is responsive across different web browsers (Chrome, Firefox, Edge).
4. **Security Testing:** Validating that the input fields are protected against common web attacks such as XSS (Cross-Site Scripting).

4.6 Deployment Diagram

The Deployment Diagram illustrates the physical hardware and software environment where VulneraScan is hosted and executed.



4.7 Conclusion

In this chapter, we detailed the implementation and testing of the **VulneraScan** system. By utilizing a stack of **Flask, HTML, CSS, and JavaScript**, we successfully developed a UI that simplifies complex network scanning tasks. The testing phase ensured that users can effectively input targets and receive accurate, visualized security insights. This implementation fulfills the project's goal of providing an accessible and powerful vulnerability assessment tool for SMEs.

Chapter

5

CONCLUSION & FUTURE WORK

5.1 Achievements

The primary goal of this project was to bridge the gap between complex command-line network security tools and the need for accessible, visual security auditing for Small and Medium Enterprises (SMEs). We have successfully achieved the following:

- **Successful Integration:** Developed a functional web application using the **Flask** framework that seamlessly interfaces with the **Nmap** scanning engine.
- **Automation of Risk Assessment:** Implemented a logic layer that automatically categorizes discovered ports and services into prioritized risk levels (High, Medium, and Low), removing the need for manual data interpretation.
- **Enhanced Visualization:** Created an interactive dashboard using **Chart.js**, **HTML**, and **CSS** that transforms raw technical data into intuitive graphical summaries.
- **Data Persistence:** Built a historical logging system using **SQLite** that allows administrators to track network changes over time and maintain an audit trail.
- **User-Centric Design:** Delivered a simplified user interface that empowers junior IT staff to perform professional-grade network scans with minimal training.

5.2 Limitations

While VulneraScan provides a robust foundation for network auditing, certain limitations were identified during the development and testing phases:

- **Scan Duration:** Extensive scans of large network ranges can be time-consuming due to the sequential nature of Nmap probes and hardware resource constraints.
- **Dependency on Nmap:** The application's detection capabilities are currently limited to what the Nmap engine can identify; it does not yet perform deep application-level exploitation.

- **Local Network Focus:** The current version is optimized for local network (LAN) environments and may require additional configuration for complex cloud-based or remote architectures.
- **Manual Triggering:** Scans must be initiated manually by the user, as the current system does not support automated scheduling.

5.3 Future Work

To further enhance the capabilities of **VulneraScan**, the following improvements are proposed for future development:

- **CVE Database Integration:** Integrating the system with the **National Vulnerability Database (NVD)** API to provide real-time information about specific vulnerabilities (CVEs) associated with detected service versions.
- **Automated Scheduling:** Adding a feature to schedule periodic scans (e.g., daily or weekly) and alert administrators via email or SMS if new high-risk ports are detected.
- **Multi-User Authentication:** Implementing advanced User Access Control (RBAC) to allow multiple administrators with different permission levels.
- **AI-Driven Risk Prediction:** Utilizing machine learning algorithms to analyze scan history and predict potential security trends or anomalies in the network.
- **Network Mapping:** Developing a dynamic network topology map that visually displays the connections between discovered hosts and their security status.

REFERENCES

1. **Aravindan, T.** (2025). *Complete Security Testing Guide: OWASP + Nmap with Practical Flask Example*. Medium. https://medium.com/@aravindanth/complete-security-testing-guide-owasp-nmap-with-practical-flask-example-921c1c1f1c1f
2. **Corgea.** (2024). *Flask Security Best Practices 2025*. Corgea Blog. https://corgea.com/blog/flask-security-best-practices
3. **Değerli, M.** (2025). *Web Based Application Programming — Web Security in Flask*. Middle East Technical University. https://users.metu.edu.tr/mdegerli/web-based-application-programming-web-security-in-flask/
4. **El-Yabroudi, N.** (2019). *Open Source Flan Scan Combines Nmap with Vulnerability Scanning*. Decipher. https://duo.com/decipher/open-source-flan-scan-combines-nmap-with-vulnerability-scanning
5. **Grinberg, M.** (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.
6. **Kumar, P., Antony, K., Banga, D., & Sohal, A.** (2024). *PhishNet: A Phishing Website Detection Tool Using XGBoost*. arXiv. https://arxiv.org/abs/2401.07134
7. **Lyon, G.** (2009). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.org. https://nmap.org/book/
8. **Mansukhani, D.** (2019). *Vulnerability Detection Using VulnDB Integration with Nmap*. Crest Data Systems. https://www.crestdata.ai/blog/vulnerability-detection-using-vulndb-integration-with-nmap
9. **NIST.** (2023). *Small Business Cybersecurity Corner*. National Institute of Standards and Technology. https://www.nist.gov/itl/smallbusinesscyber
10. **PortSwigger.** (n.d.). *Burp Suite Knowledge Base*. Wikipedia. https://en.wikipedia.org/wiki/Burp_Suite
11. **Sanchez, G.** (2024). *Python Flask Vulnerability Scanner*. Gilbert Sanchez Cyber Portfolio. https://gescyber.com/portfolio/python-flask-vulnerability-scanner/
12. **Siberoloji.** (2024). *Exploring Third-Party Nmap Tools and Add-Ons*. Siberoloji. https://www.siberoloji.com/exploring-third-party-nmap-tools-and-add-ons/
13. **Toxigon.** (2024). *Flask Security Best Practices 2025*. Toxigon Security. https://toxigon.com/flask-security-best-practices-2025
14. **Wikipedia contributors.** (2024, July 10). *ZMap (software)*. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/ZMap_(software)

APPENDIX

Index.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>VulneraScan</title>
  <link rel="stylesheet" href="/static/style.css" />
</head>
<body>
  <div class="card">
    <h1>VulneraScan</h1>
    <p>Enter a target (IP or domain) to scan common ports.</p>

    <form method="POST" action="/scan">
      <input name="target" placeholder="e.g., 127.0.0.1 or example.com"
required />
      <button type="submit">Scan</button>
    </form>
  </div>
</body>
</html>
```

results.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Results - VulneraScan</title>
  <link rel="stylesheet" href="/static/style.css" />
</head>
<body>
  <div class="card wide">
    <div class="top-bar">
      <h1>🛡️ Scan Results</h1>
      <button class="toggle" type="button" onclick="toggleMode()">🌙 /
☼</button>
    </div>
```

```
    <p class="target"><b>Target:</b> {{ target }}</p>
    {% if error %}
      <div class="error">{{ error }}</div>
    {% else %}
    <div class="results-layout">
      <!-- TABLE -->
      <div class="table-box">
        <table>
          <thead>
            <tr>
              <th>Port</th>
              <th>Service</th>
              <th>Status</th>
            </tr>
          </thead>
          <tbody>
            {% for r in results %}
              <tr>
                <td>{{ r.port }}</td>
                <td>{{ r.service }}</td>
                <td class="{{ 'open' if r.status == 'Open' else 'closed'
}}">
                  {{ '🟢 Open' if r.status == 'Open' else '🔴 Closed' }}
                </td>
              </tr>
            {% endfor %}
          </tbody>
        </table>
      </div>
      <!-- DONUT CHART (NO LIBRARIES) -->
      <div class="chart-box">
        <h3 class="chart-title">Port Scan Summary</h3>
        {% set total = open_count + closed_count %}
        {% set open_pct = (open_count / total * 100) if total > 0 else 0 %}
        {% set closed_pct = (closed_count / total * 100) if total > 0 else 0
%}
        <div class="donut-wrap">
          <div class="donut" style="--open: {{ '%.1f'|format(open_pct) }};">
            <div class="donut-center">
              <div class="donut-total">{{ total }}</div>
              <div class="donut-sub">Total Ports</div>
            </div>
          </div>
        </div>
        <div class="chart-stats">
          <div class="stat open-stat">🟢 Open: <b>{{ open_count }}</b> ({{
'%.1f'|format(open_pct) }}%)</div>
```

```
          <div class="stat closed-stat">🔘 Closed: <b>{{ closed_count
}}</b> ({{ '%.1f'|format(closed_pct) }}%)</div>
        </div>
      </div>
    </div>
    {% endif %}

    <div class="actions">
      <a class="back" href="/">← Back</a>
      <button type="button" onclick="window.print()">📄 Export PDF</button>
    </div>
  </div>
  <script>
    function toggleMode() {
      document.body.classList.toggle("dark");
    }
  </script>
</body>
</html>
```

app.py

```
from flask import Flask, render_template, request
import socket
app = Flask(__name__)
COMMON_PORTS = {
    21: "FTP",
    22: "SSH",
    23: "Telnet",
    25: "SMTP",
    53: "DNS",
    80: "HTTP",
    110: "POP3",
    143: "IMAP",
    443: "HTTPS",
    3306: "MySQL",
    3389: "RDP",
}
def scan_port(target: str, port: int, timeout: float = 0.5) -> bool:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            return s.connect_ex((target, port)) == 0
    except Exception:
        return False
@app.route("/", methods=["GET"])
def index():
    return render_template("index.html")
```

```
@app.route("/scan", methods=["POST"])
def scan():
    target = (request.form.get("target") or "").strip()
    if not target:
        return render_template("results.html", target="", results=[],
open_count=0, closed_count=0, error="Please enter a target")
    results = []
    open_count = 0
    closed_count = 0

    for port, service in COMMON_PORTS.items():
        is_open = scan_port(target, port)
        status = "Open" if is_open else "Closed"

        if is_open:
            open_count += 1
        else:
            closed_count += 1

        results.append({"port": port, "service": service, "status": status})

    return render_template(
        "results.html",
        target=target,
        results=results,
        open_count=open_count,
        closed_count=closed_count,
        error=None
    )

if __name__ == "__main__":
    app.run(debug=True)
```