Capstone Project-II

Report

On

"NetSentry-The Network Monitoring System"

Submitted By

Mustafa Jambughodawala	202202626010061
Divyang Mali	202202626010067
Ayush Nikose	202202626010072
MohammadRafe Shaikh	202202626010087

Under the Guidance of

Ms. Aakanksha Jain
Assistant Professor

Faculty of Engineering and Technology

GLS University

Academic Year

(2024-2025)

Acknowledgment

We would like to express our heartfelt gratitude to everyone who supported us throughout the journey of developing **NetSentry – The Network Monitoring System**.

Our deepest thanks to **Ms. Aakanksha Jain** for her invaluable guidance and constant encouragement during the development of this project. Her insights and support have played a crucial role in shaping the success of our work.

We would also like to thank our **Head of Department, Dr. Harshal Arolkar** for providing a supportive academic environment, which enabled us to carry out and complete this project successfully.

This acknowledgment stands as a token of our sincere appreciation to everyone who contributed to the successful completion of this project. We are proud of what we have achieved and truly grateful for the support that made it possible.

Sincerely,

Mustafa Jambughodawala
Divyang Mali
Ayush Nikose
MohammadRafe Shaikh

Abstract

The ***NetSentry*** is a comprehensive solution designed to monitor and manage devices connected to a local area network (LAN) efficiently. The system comprises a server and lightweight agents deployed across devices, enabling real-time detection, cataloging, and management of connected devices and peripherals. The project addresses critical challenges in network administration, including device classification, vulnerability detection, and peripheral monitoring. By leveraging NVD API usage, the system integrates system and network vulnerability detection. The solution ensures secure communication, and a privacy-centric approach by avoiding the storage of sensitive logs.

Key features include an intuitive web-based dashboard for administrators to view device details, approve or deny peripheral connections, and receive alerts about suspicious activity. The system is designed to operate efficiently across diverse network scales, offering high performance, reliability, and extensibility. This project contributes to modern network management by improving security, scalability, and usability, making it suitable for enterprises, educational institutions, and small to medium-sized networks. Future enhancements may include deeper AI/ML integration, support for predictive analytics, and expanded compatibility with emerging IoT technologies.

Content

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Project Detail

NetSentry – The Network Monitoring System is a cybersecurity-oriented solution designed to provide real-time visibility and monitoring of devices and their peripherals within a local network. The system helps administrators track device activity, identify unauthorized connections, and assess potential vulnerabilities. It aims to simplify network management by offering a centralized platform that displays detailed information about each connected device along with attached peripherals such as USB drives, printers, and external storage devices.

The system is divided into two core components: ***a lightweight agent*** and a ***centralized dashboard***. The agent, installed on individual devices, collects system details such as device name, IP address, operating system, installed software, system uptime, connected peripherals, and other required information. This includes real-time tracking of USB plug-ins and other hardware components, helping detect unauthorized hardware usage. The collected data is securely transmitted to the dashboard via socket communication for centralized monitoring and analysis.

The dashboard presents real-time information about all connected devices, including their status (online/offline), software inventory, vulnerability scores, and active peripherals. These vulnerability scores are calculated using the ***NVD (National Vulnerability Database) API***, allowing administrators to quickly identify outdated or risky software. With peripheral monitoring, the system adds another layer of security by alerting administrators to unusual or unauthorized hardware changes. The interface also includes features like search, sorting, and filtering for efficient management.

NetSentry is built with scalability in mind and can be extended with features such as automated alerts, role-based access control, and AI/ML-powered threat detection and anomaly analysis. With its ability to monitor both systems and their connected peripherals, it serves as an ideal solution for small businesses, educational institutions, and cybersecurity learners looking to improve network security and hardware oversight.

1.2 Purpose

The purpose of NetSentry is to develop an efficient and secure network monitoring system that not only tracks connected devices within a network but also monitors their attached peripherals. By providing real-time insights into system activity and hardware usage, the project aims to support proactive network management, enhance security, and reduce the risk of unauthorized device or peripheral access.

1.3 Scope

NetSentry is designed to cater to small and medium-sized networks, such as those found in educational institutions, offices, and labs. The system includes a lightweight agent for data collection on individual devices and a centralized dashboard for monitoring. It gathers detailed information about hardware, software, network status, and connected peripherals like USB devices and printers. The project also integrates vulnerability scoring using the NVD API to assess software security. Future scalability includes features like alerts, access control, and cloud compatibility.

1.4 Objective

- Detailed and Active Monitoring of device Activities
- Centralized management through a web dashboard
- Real Time Detection
- Peripheral Monitoring
- Vulnerability Detection

1.5 Literature Review

1.5.1 Existing Solutions

Several established tools like Nagios, Zabbix, and SolarWinds are widely recognized for their robust capabilities in network monitoring and management. These tools primarily operate using log-based monitoring techniques, capturing detailed logs of network activity and analyzing them for anomalies or performance issues.

- ***Nagios***: Known for its extensive plugin ecosystem, Nagios provides powerful monitoring capabilities for servers, switches, applications, and services. However, its reliance on manual configuration can lead to scalability challenges.

- ***Zabbix***: Offers sophisticated features for network visualization and real-time alerting but requires agents to be installed on monitored devices, potentially raising concerns about invasiveness.

- ***SolarWinds***: Focuses on enterprise-grade network management with advanced reporting and analytics but comes with high licensing costs and a steep learning curve.

1.5.2 Key Findings

While these tools are effective in their domains, they have certain limitations when it comes to addressing modern network security challenges.

- ***Efficiency***: These tools perform well in detecting known issues but may lack the ability to predict potential threats proactively.

- ***Invasiveness***: The reliance on extensive log storage and constant data collection can raise privacy concerns and lead to resource inefficiencies.

- ***Real-Time Responsiveness***: Although alerts are generated for detected issues, there is often a delay in identifying threats, which may compromise security in critical scenarios.

- ***AI/ML Integration***: Current solutions typically do not leverage advanced AI/ML techniques for anomaly detection or predictive threat analysis, making them less adaptive to evolving security risks.

1.5.3 Gaps in Existing Solutions

Despite their strengths, these tools leave significant gaps that need to be addressed:

- ***Privacy Concerns***: The reliance on large-scale log storage compromises the privacy of users and organizations.

- ***Proactive Security***: Limited ability to detect and mitigate emerging threats in real time.

- ***Real-Time Device Monitoring***: Insufficient focus on monitoring individual devices and peripherals dynamically within the network.

- ***User-Centric Design***: A lack of intuitive dashboards and user-friendly interfaces can hinder the usability of these tools for administrators.

1.5.4 Conclusion

There is a clear need for an innovative system that bridges the gaps in existing solutions by integrating real-time monitoring with strong privacy preservation. Such a system should offer immediate detection of device activity and network changes, enabling proactive responses to potential threats, while ensuring minimal invasiveness by avoiding the storage of user logs and maintaining robust monitoring capabilities.

By addressing these gaps, the proposed system will enhance network security, streamline management processes, and provide administrators with actionable insights without compromising user privacy.

Chapter 2: About The System

2.1 Functional and Non-Functional Requirements

Table 2.1: Functional Requirements

Requirement ID	Function	Description
FR-01	Real Time Detection & Notification	Detects and alerts administrators of new devices or anomalies in real-time.
FR-02	Detailed & Active monitoring of device activities	Continuously tracks and catalogs the status, behavior, and details of connected devices.
FR-03	Peripheral Monitoring	Monitors device-level peripheral connections (e.g., USB, printer) and generate alerts for activity changes.
FR-04	Vulnerability Detection	The system must compare software versions with NVD using APIs to assign vulnerability scores.
FR-05	Centralized management through a web dashboard	Provides a user-friendly web dashboard for network monitoring and administrative actions.

Table 2.2: Non-Functional Requirements

Requirement ID	Function	Description
NFR-01	Performance	Ensure real-time responses to device or network changes within 2 seconds.
NFR-02	Scalability	Supports small to large networks with minimal performance impact.
NFR-03	Security	Protects sensitive data and access using encryption and authentication mechanisms.
NFR-04	Availability	Ensures 99.9% uptime for uninterrupted monitoring and alerting.

NFR-05	Usability	Provides an intuitive web interface for efficient network management.
NFR-06	Interoperability	Ensures compatibility with various operating systems and devices.
NFR-07	Maintainability	Facilitates easy updates, debugging, and feature additions through modular design.
NFR-08	Data Privacy	Avoid storing sensitive data or logs to respect privacy requirements.
NFR-09	Adaptability	Supports future enhancements without significant architectural changes.

2.2 Project Planning

The project was planned and executed in multiple phases to ensure organized development and efficient progress:

Phase 1: Requirement Gathering and Analysis (July - August)

- Identified key functionalities required for device and peripheral monitoring.
- Researched existing tools and defined the scope for a lightweight, real-time system.

Phase 2: Design (September - October)

- Designed system architecture with agent-based data collection and socket communication.
- Designed UI mock-ups for the dashboard.

Phase 3: Implementation (November - February)

- Developed and tested the agent for collecting system and peripheral data.
- Built the dashboard for data visualization and implemented socket-based communication.
- Integrated the NVD API for software vulnerability scoring.

Phase 4: Testing and Debugging (March)

- Conducted functional and performance testing across multiple devices.
- Debugged issues related to data transmission and peripheral detection.

Phase 5: Final Review and Documentation (April)

- Finalized system features and prepared documentation.
- Outlined future enhancements like AI/ML-based anomaly detection and smart alerts.

Chapter 3: Analysis of the System

3.1 Use Case Diagram

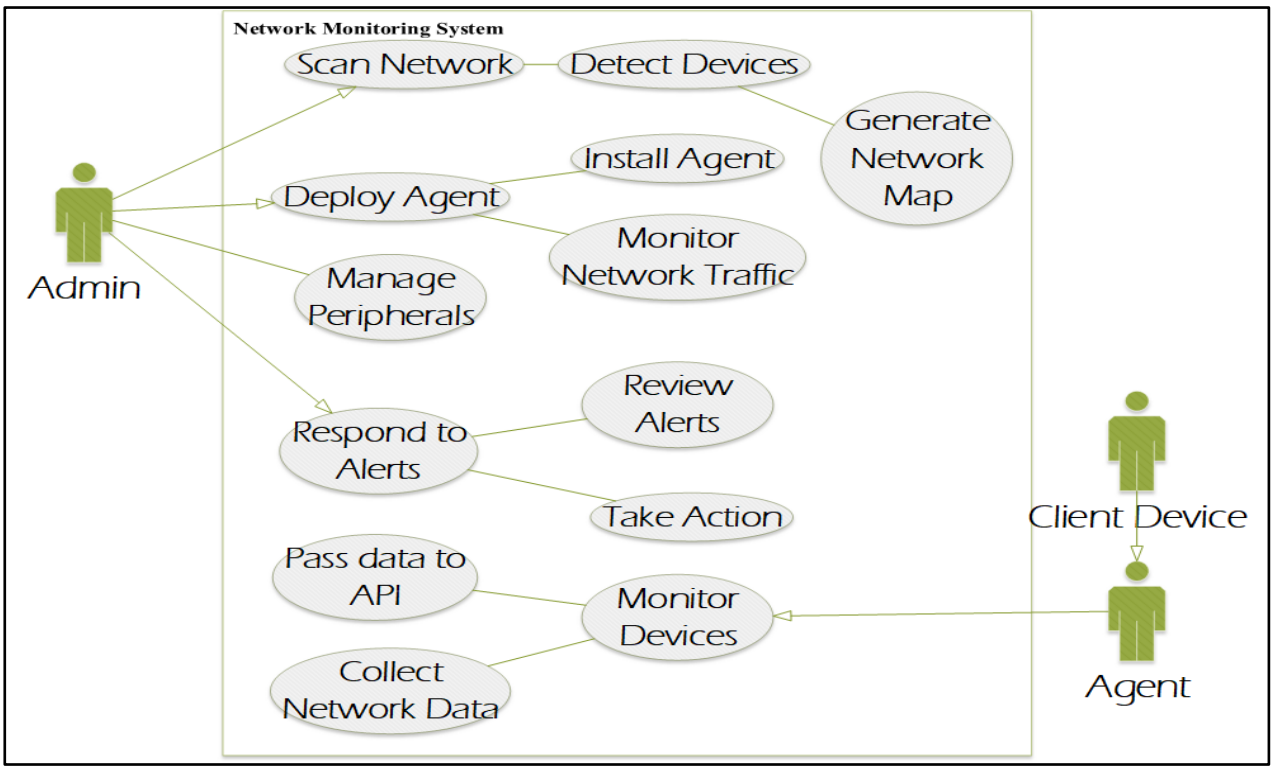

Fig. 3.1: Use Case Diagram

3.2 Sequence Diagram

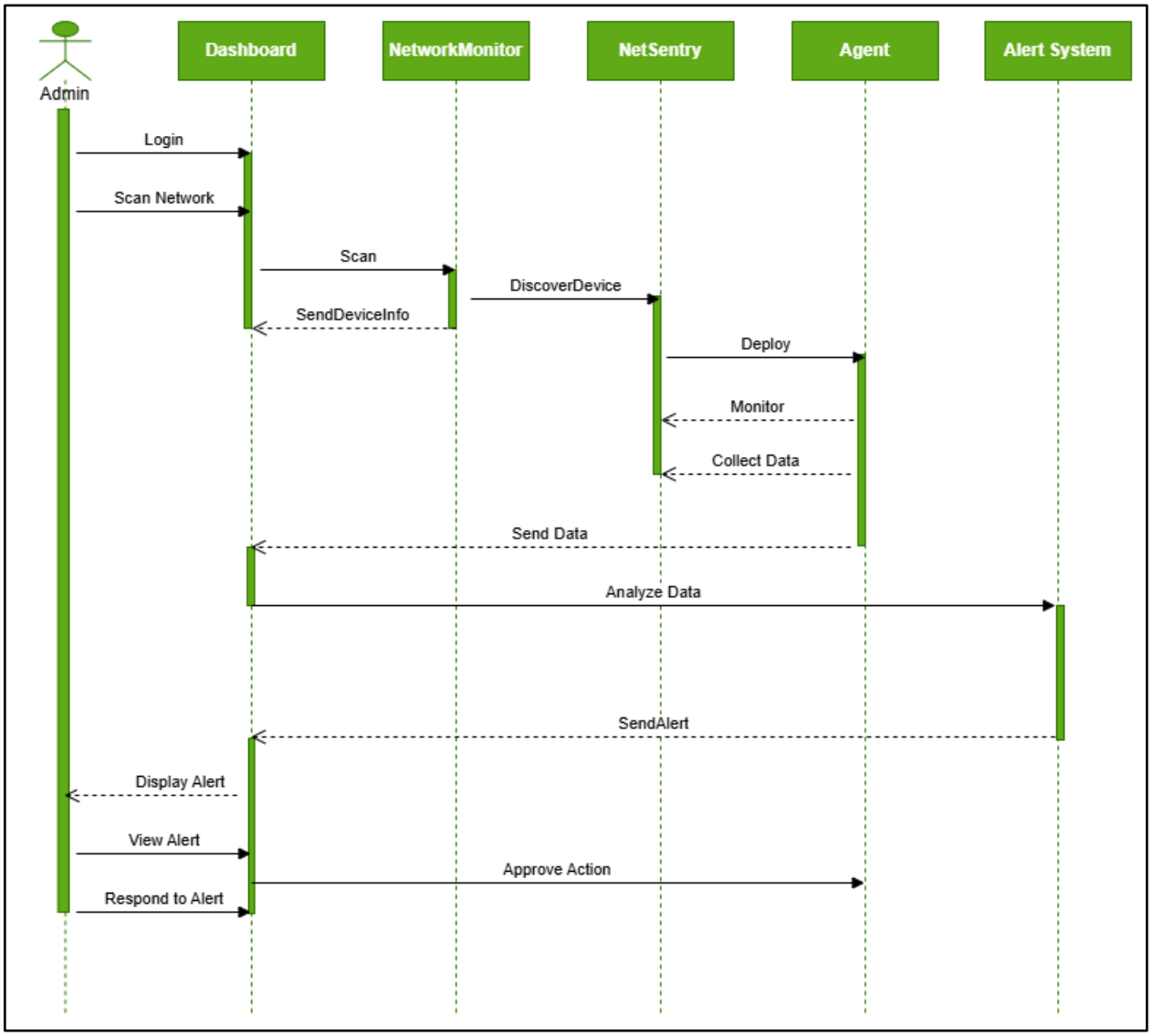

Fig. 3.2: Sequence Diagram

3.3 Activity Diagram

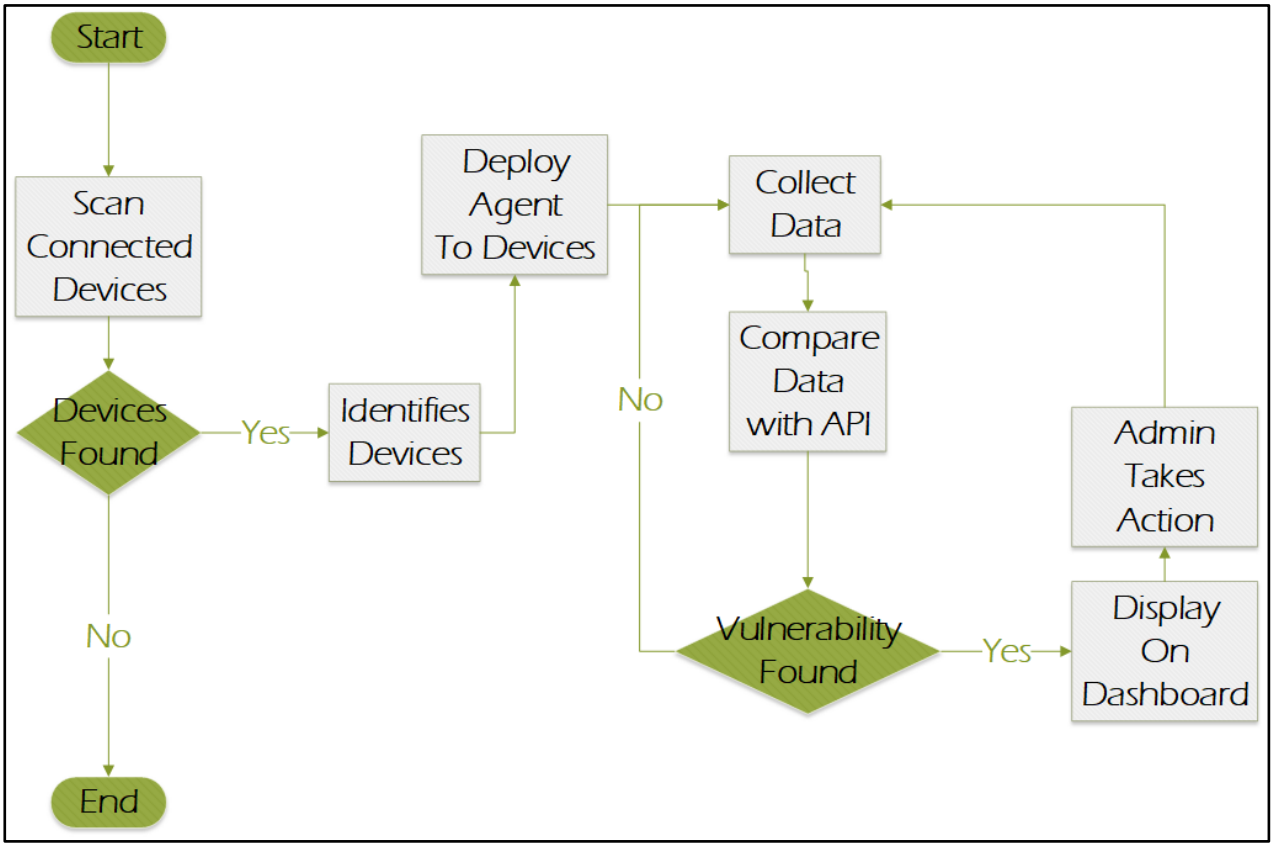

Fig. 3.3: Activity Diagram

3.4 Data Flow Diagram

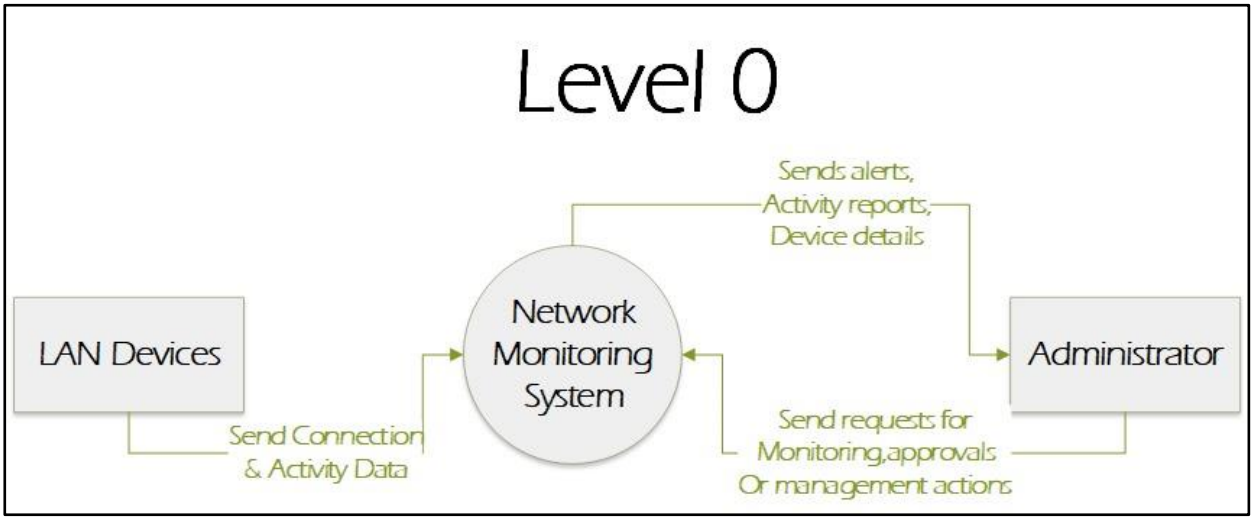

Fig. 3.4: Level 0 Data Flow Diagram

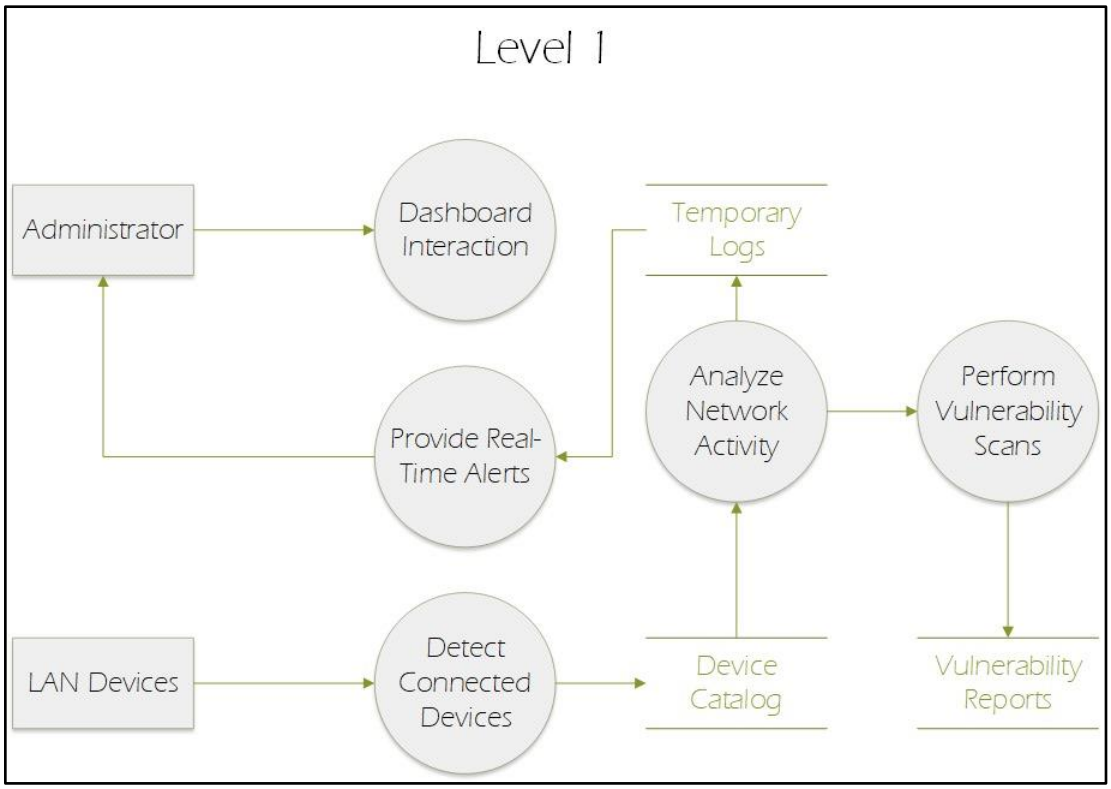

Fig. 3.5: Level 1 Data Flow Diagram

3.3 Class Diagram

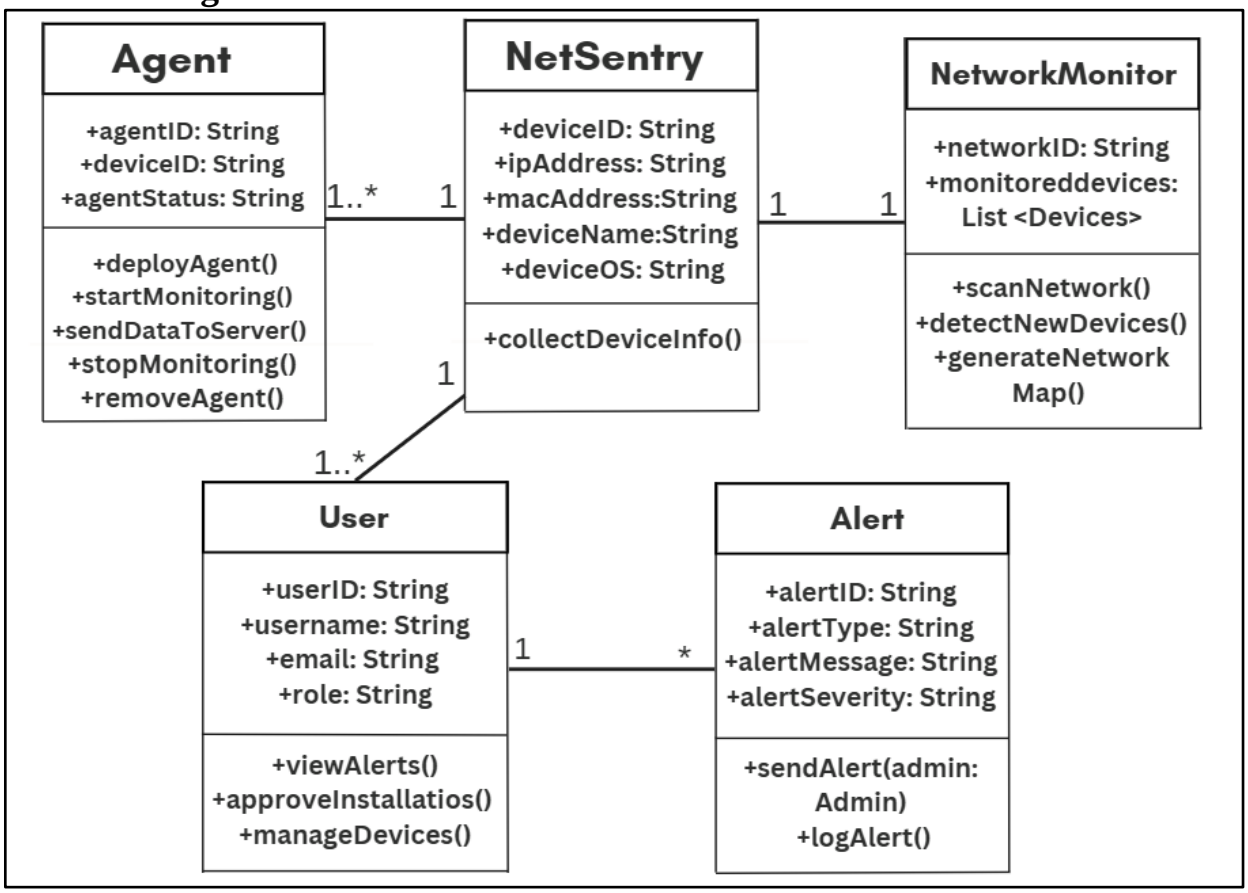

Fig. 3.6: Class Diagram

Chapter 4: Design

4.1 System Flow Diagram

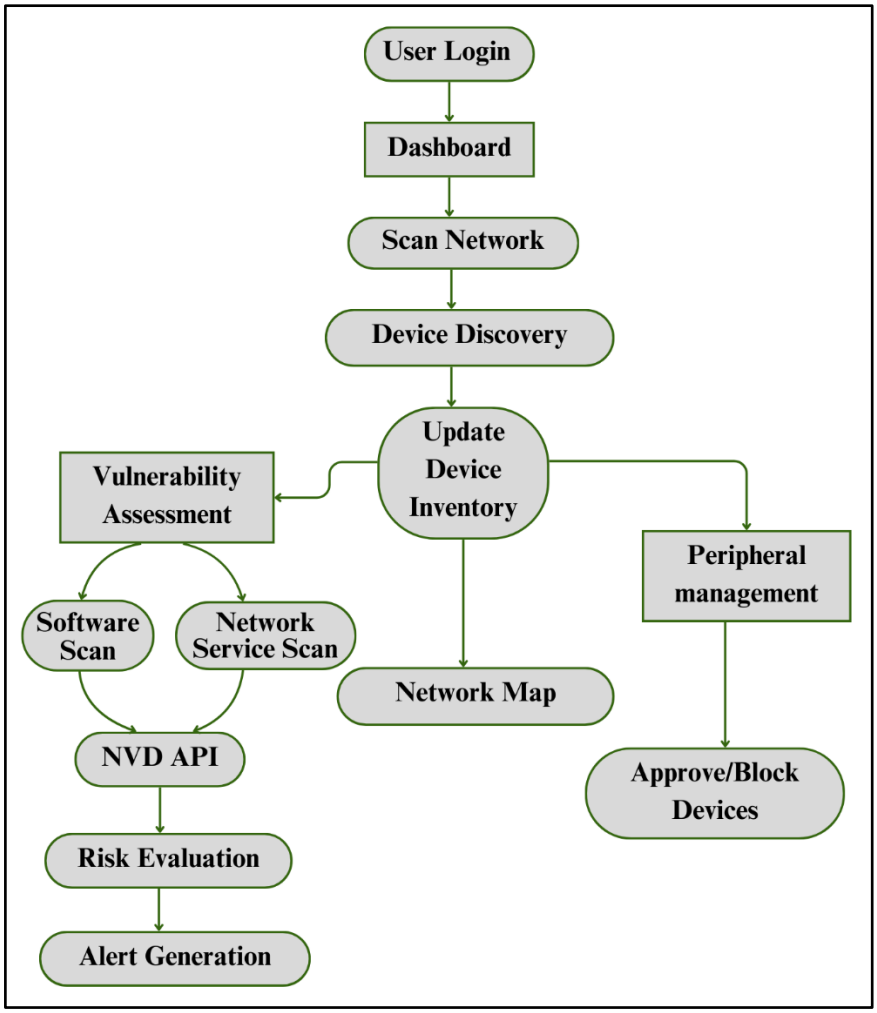

Fig. 4.1: System Flow Diagram

4.2 Data Dictionary

Table 4.1: System Monitoring Data Dictionary

Field	Data Type	Description	Source
timestamp	ISO 8601 datetime	Time of data collection	Agent
system.hostname	string	Device hostname	Agent
system.ip_address	string	Primary IP address	Agent

system.mac_address	string	MAC address	Agent
system.os	string	Operating system name	Agent
system.os_version	string	OS version	Agent
system.os_release	string	OS release	Agent
system.architecture	string	System architecture	Agent
system.processor	string	CPU information	Agent
system.boot_time	ISO 8601 datetime	Last system boot time	Agent
system.ram_total_gb	float	Total RAM in GB	Agent
system.ram_used_gb	float	Used RAM in GB	Agent
system.ram_available_gb	float	Available RAM in GB	Agent
system.disks[]	array	Disk information array	Agent
processes[]	array	Running processes	Agent
network_connections[]	array	Active network connections	Agent
users[]	array	System users	Agent
installed_software[]	array	Installed applications	Agent
security_events	string/list	Security log entries	Agent
antivirus_status	string/list	AV process status	Agent
sensitive_files[]	array	Critical system files	Agent
performance.cpu_percent	float	Current CPU usage %	Agent
performance.memory_percent	float	Current memory usage %	Agent
performance.disk_usage	float	Root disk usage %	Agent

Table 4.2: Peripheral Device Data Dictionary

Field	Data Type	Description	Source
id	string	Composite ID (vendor:product:serial)	Agent
vendor_id	string	USB vendor ID	Agent
product_id	string	USB product ID	Agent
manufacturer	string	Device manufacturer	Agent
product	string	Product name	Agent
serial	string	Serial number	Agent
sys_path	string	System device path	Agent
device_type	string	Device type	Agent
driver	string	Driver name	Agent
subsystem	string	Device subsystem	Agent
device_node	string	Device node path	Agent
detected_at	ISO 8601 datetime	Detection timestamp	Agent
request_id	UUID string	Unique request ID	Agent
status	string	Approval status (pending/approved/blocked)	Server
received_at	ISO 8601 datetime	Server receipt time	Server
decision_time	ISO 8601 datetime	Approval/block time	Server

Table 4.3: Vulnerability Data Dictionary

Field	Data Type	Description	Source
total_found	integer	Number of vulnerabilities found	Server
vulnerabilities[].cve_id	string	CVE identifier	Server
vulnerabilities[].description	string	Vulnerability description	Server
vulnerabilities[].cvss_version	string	CVSS version (2/3)	Server

vulnerabilities[].cvss_score	float	CVSS base score (0-10)	Server
vulnerabilities[].severity	string	Severity level (Critical/High/Medium/Low/None)	Server

Table 4.4: Network Scan Data Dictionary

Field	Data Type	Description	Source
os	string	Detected OS	Server
ports[].port	integer	Port number	Server
ports[].state	string	Port state (open/closed)	Server
ports[].name	string	Service name	Server
ports[].product	string	Service product	Server
ports[].version	string	Service version	Server

4.3 User Interface

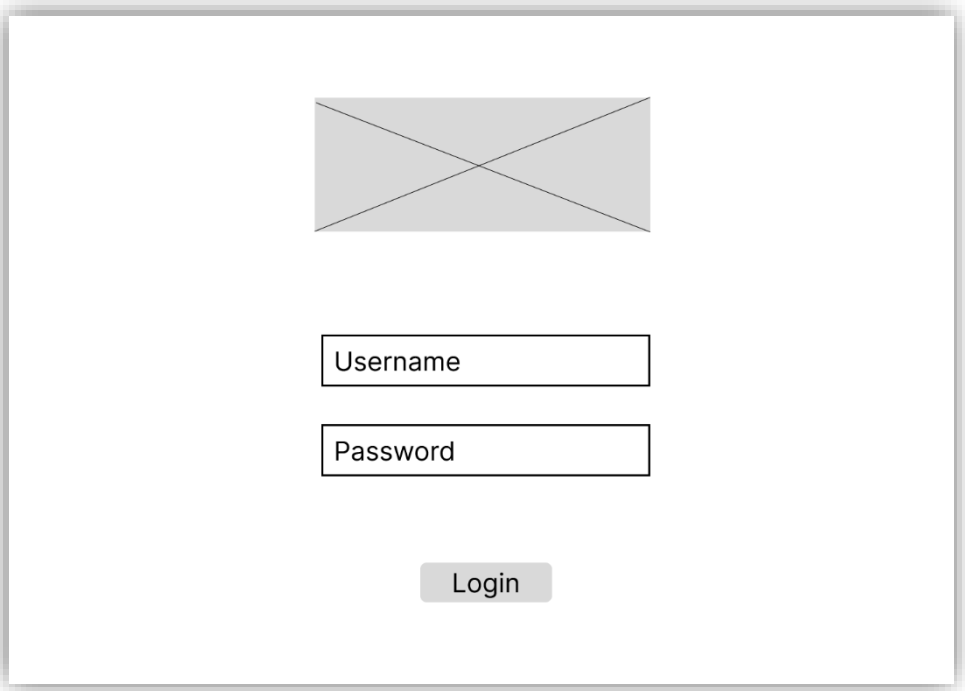

Fig. 4.2: Wireframe of Login Page

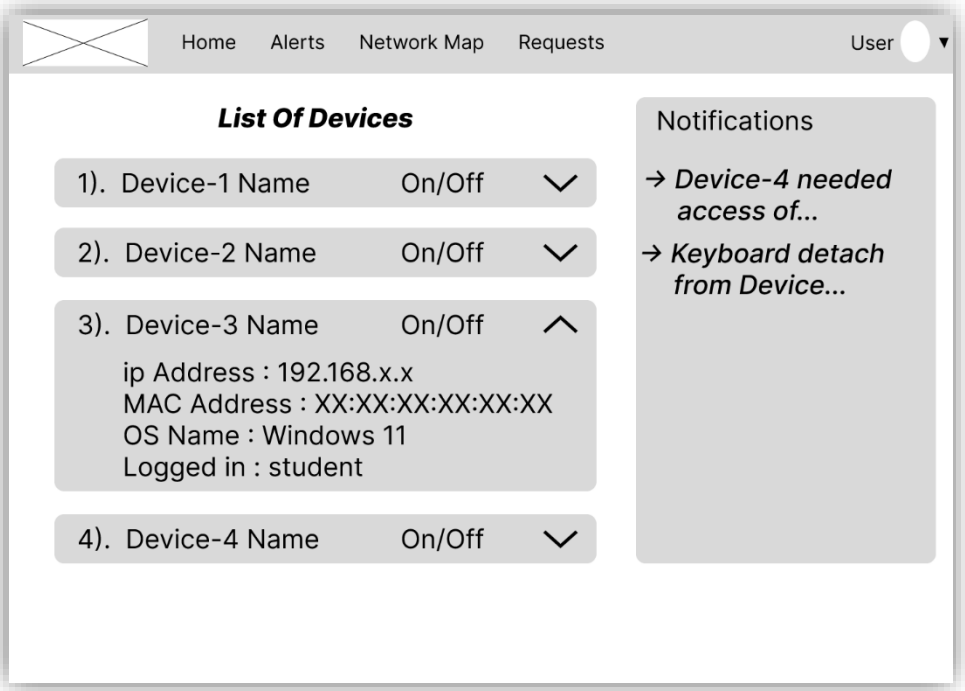

Fig 4.3: Wireframe of Home Page

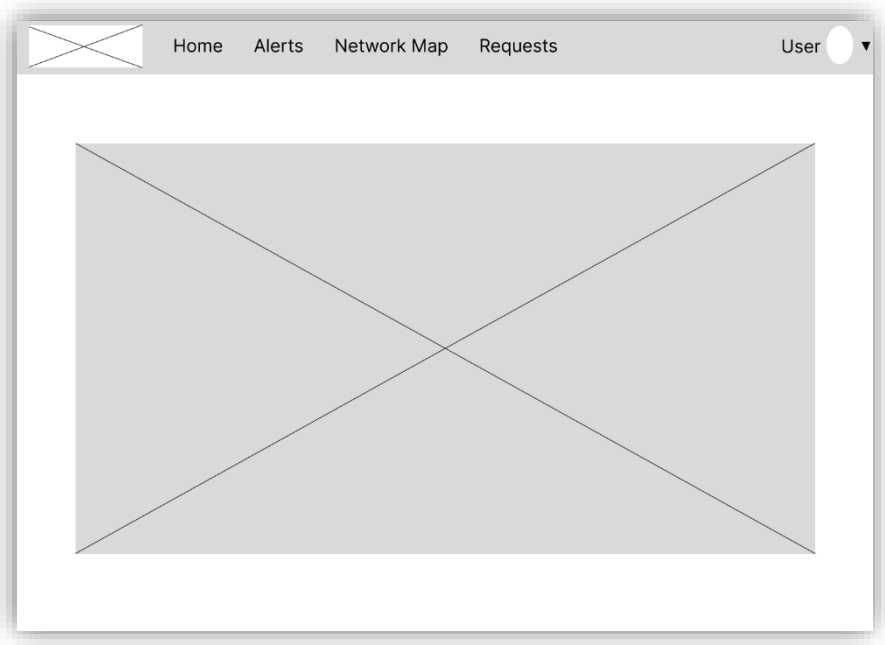

Fig. 4.4: Wireframe of Network Map Page

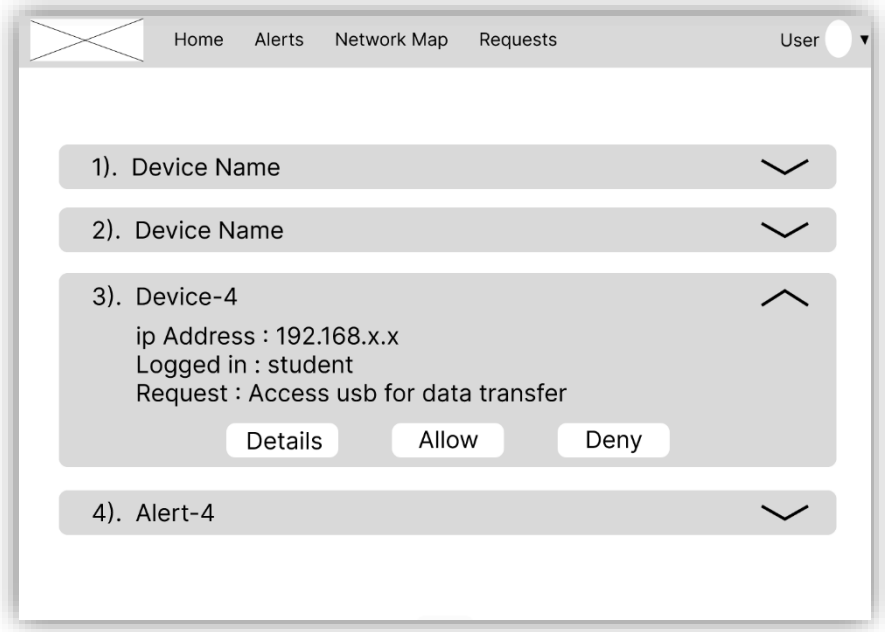

Fig. 4.5: Wireframe of Peripheral request Page

Fig. 4.6: Wireframe of User Dropdown Menu

Chapter 5: Implementation

5.1 Implementation Environment

1. Programming Languages

- ***Python:*** Used for developing the agent script, handling system data collection, socket communication, and integration with external APIs (e.g., NVD).
- ***JavaScript:*** For enhancing dashboard interactivity.

2. Web Technologies

- ***HTML/CSS:*** Used to build and style the user interface of the centralized dashboard.
- ***Flask:*** Python-based web framework used to serve the dashboard and handle backend logic.

3. Networking

- ***Socket Programming (Python's socket library):*** Facilitates secure communication between the agent and the dashboard in real-time.
- ***Nmap:*** For finding connected devices and port scanning.

4. Cybersecurity Integration

- ***NVD API:*** Integrated to fetch vulnerability data and assign CVE-based scores to software and service versions installed on network devices.

5. Peripheral Monitoring

- ***psutil / pyudev / platform libraries (Python):*** For accessing system-level information including hardware, software, and connected peripherals like USB devices.

5.2 Security Features

NetSentry integrates several key security features to protect data and enhance network safety:

- ***Secure Communication:*** Socket-based data transfer ensures controlled communication between agents and the dashboard. Future improvements may include encryption (e.g., SSL).
- ***Peripheral Monitoring:*** Detects and logs unauthorized hardware (e.g., USB devices), helping prevent data leakage or hardware-based attacks.
- ***Software Vulnerability Detection:*** Uses the NVD API to identify known CVEs in installed software, allowing proactive threat management.

- ***Access Control (Planned):*** Role-based access can be added to restrict dashboard access based on user permissions.
- ***Minimal Attack Surface:*** The agent operates with limited system privileges to reduce potential exploitation.

5.3 Coding Standard

To maintain code quality, readability, and consistency, the following coding standards were followed:

- ***PEP 8 (Python Enhancement Proposal 8):*** Adhered to for Python code styling including naming conventions, indentation, and structure.
- ***Modular Coding:*** Code was organized into reusable and maintainable functions and modules for scalability and debugging.
- ***Meaningful Naming:*** Variables, functions, and classes were named clearly to reflect their purpose.
- ***Error Handling:*** Try-except blocks were used to handle exceptions gracefully and ensure system stability.
- ***Code Comments:*** Inline comments were added throughout the codebase for clarity and future maintenance.

Chapter 6: Project Screenshot

6.1 Prototype with Results/Screenshots

```
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5053
 * Running on http://192.168.31.222:5053
2025-04-07 00:32:40,481 - INFO - Press CTRL+C to quit
2025-04-07 00:32:40,483 - INFO -  * Restarting with stat

==============================================
NetSentry Server starting...
Make sure your firewall allows connections to port 5053
Server will be accessible at:
• http://127.0.1.1:5053 (Primary IP)
• http://localhost:5053 (Local access only)
==============================================
```

Fig. 6.1: Server Startup

Fig. 6.2: Login Page

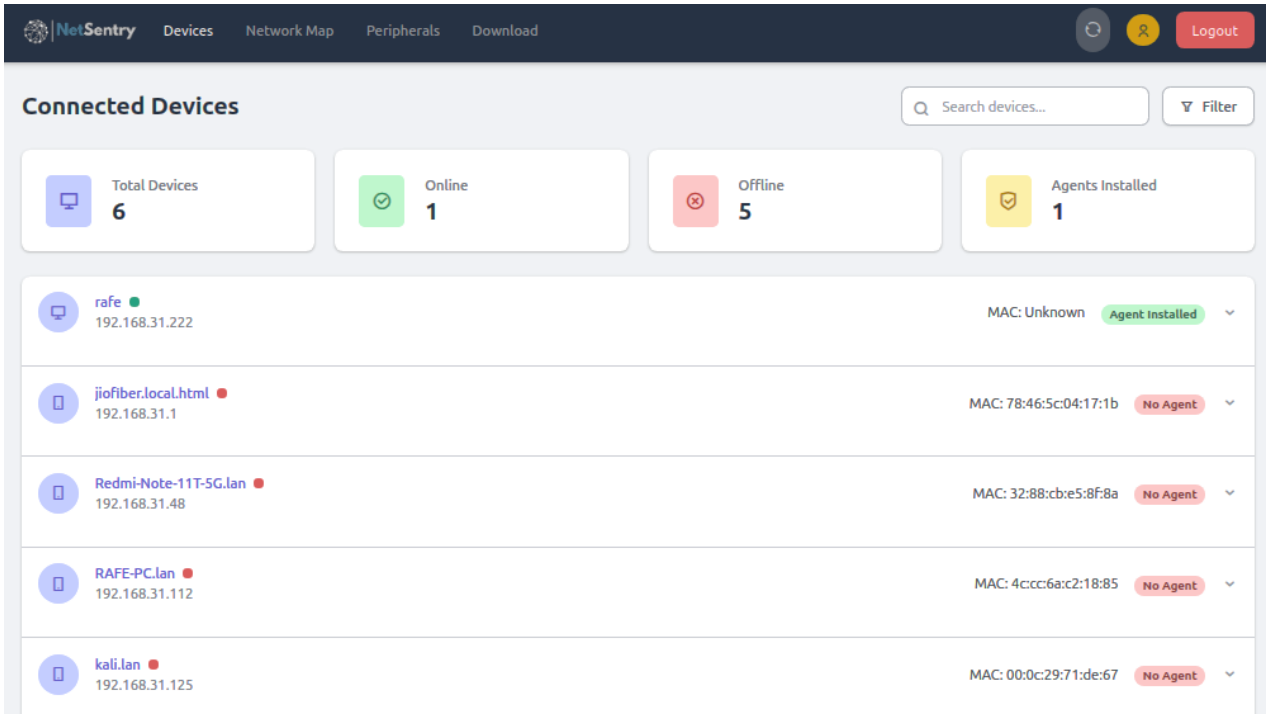

Fig. 6.3: Home Page

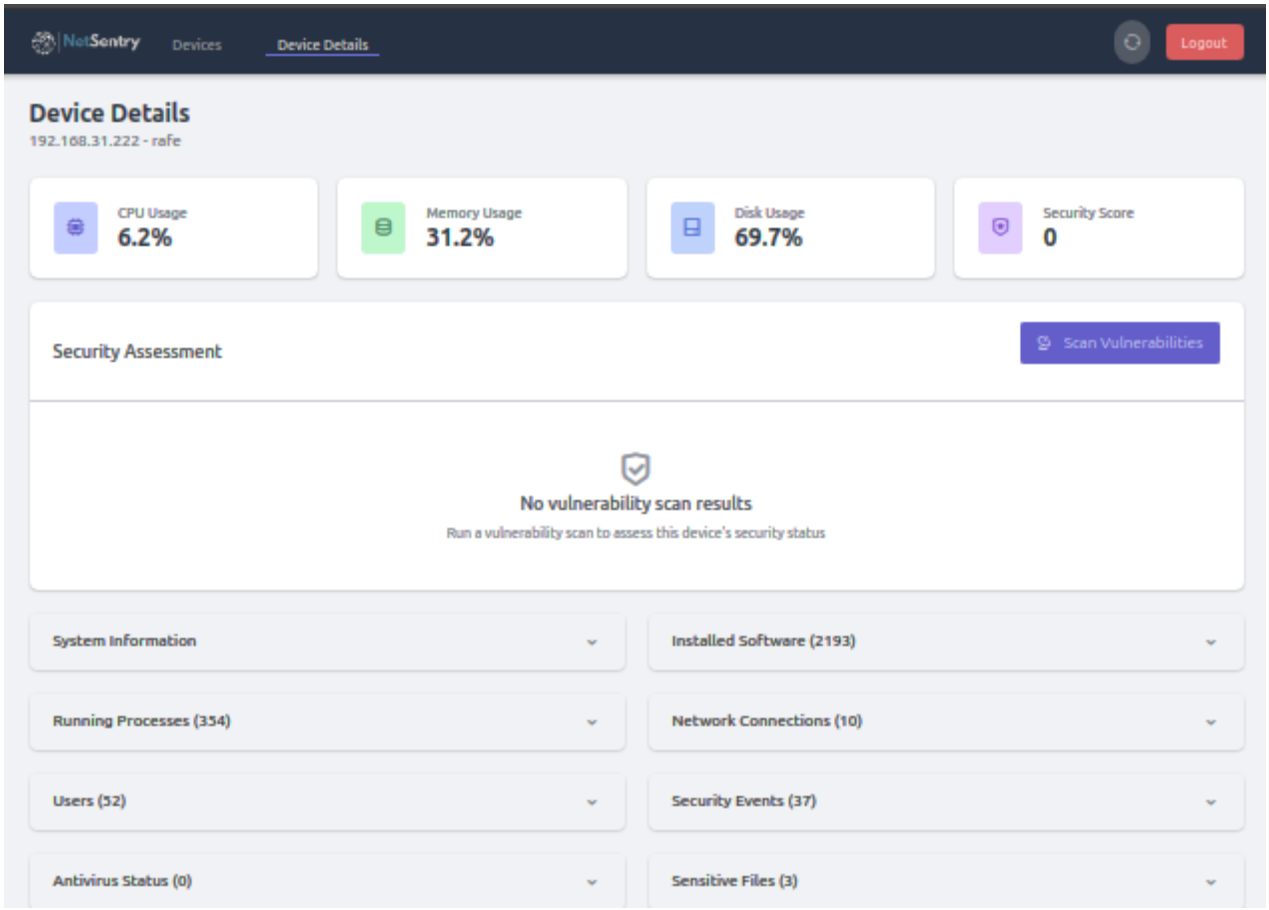

Fig. 6.4: Device Details Page

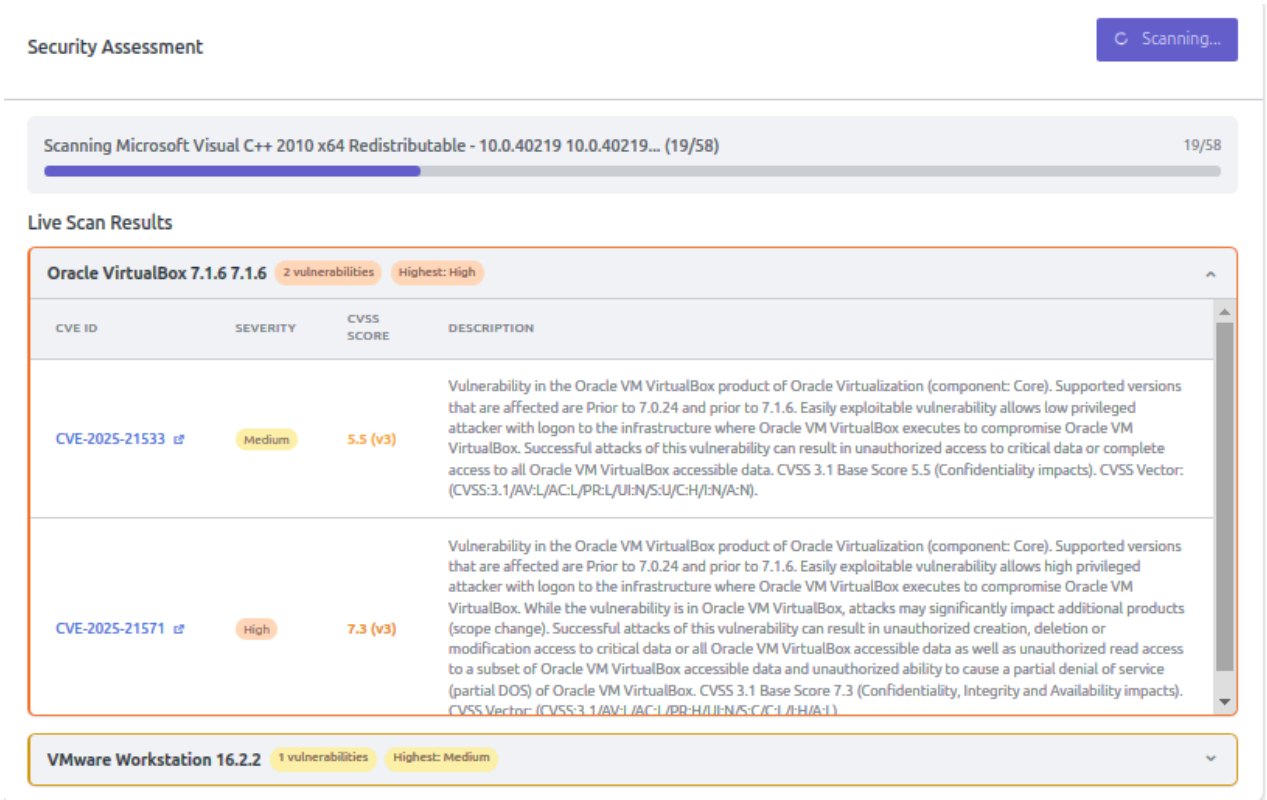

Fig. 6.5: Live Security Assessment

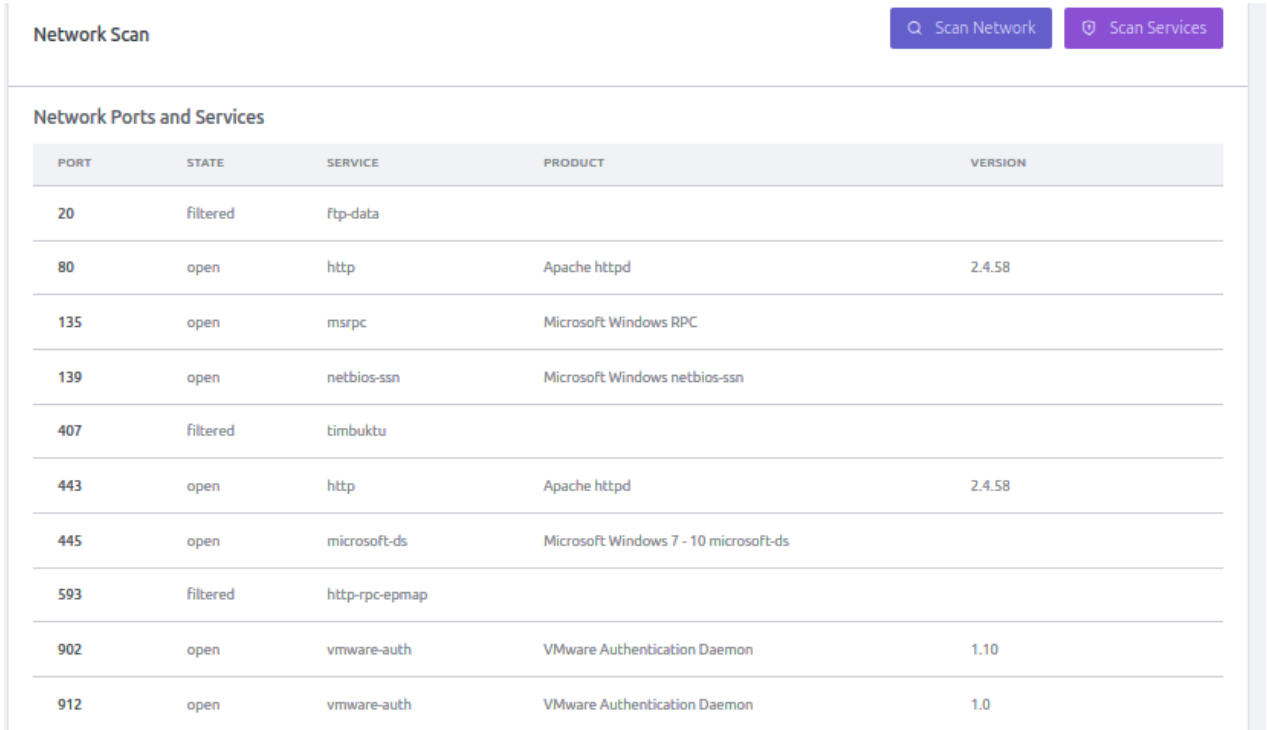

Fig. 6.6: Network Port Scan Result

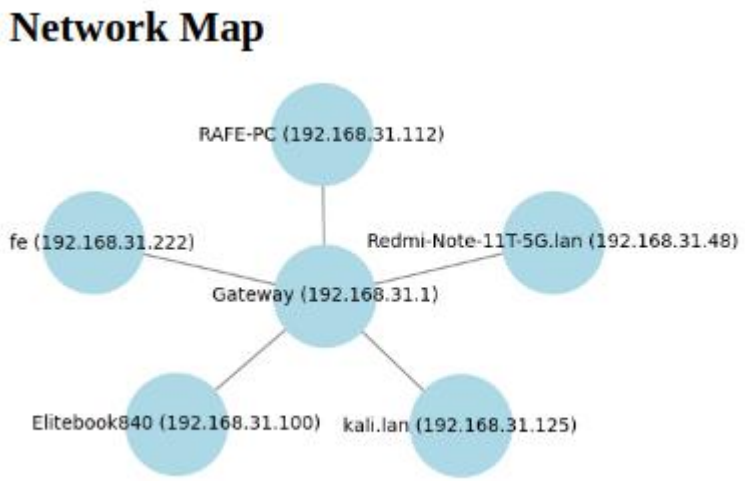

Fig. 6.7: Network Map

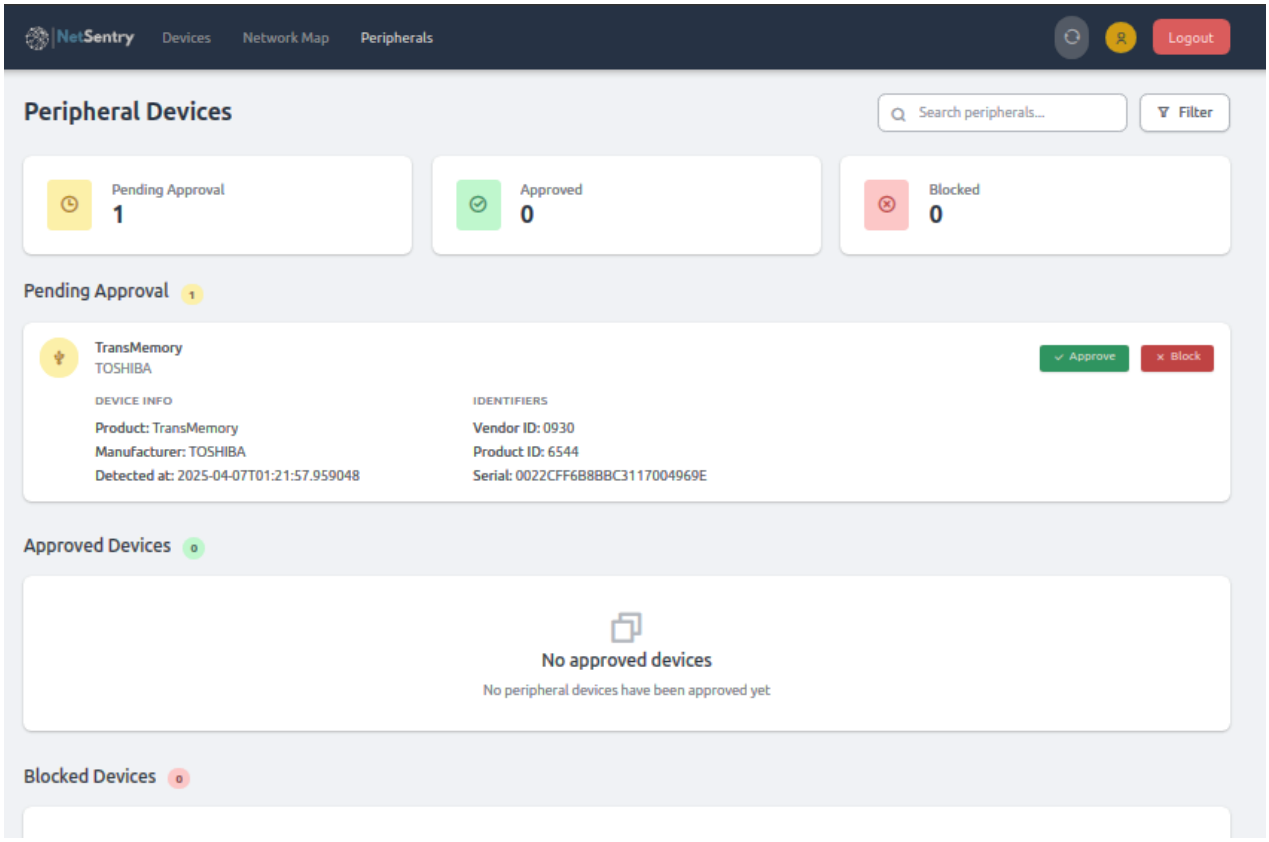

Fig. 6.8: Peripherals Page

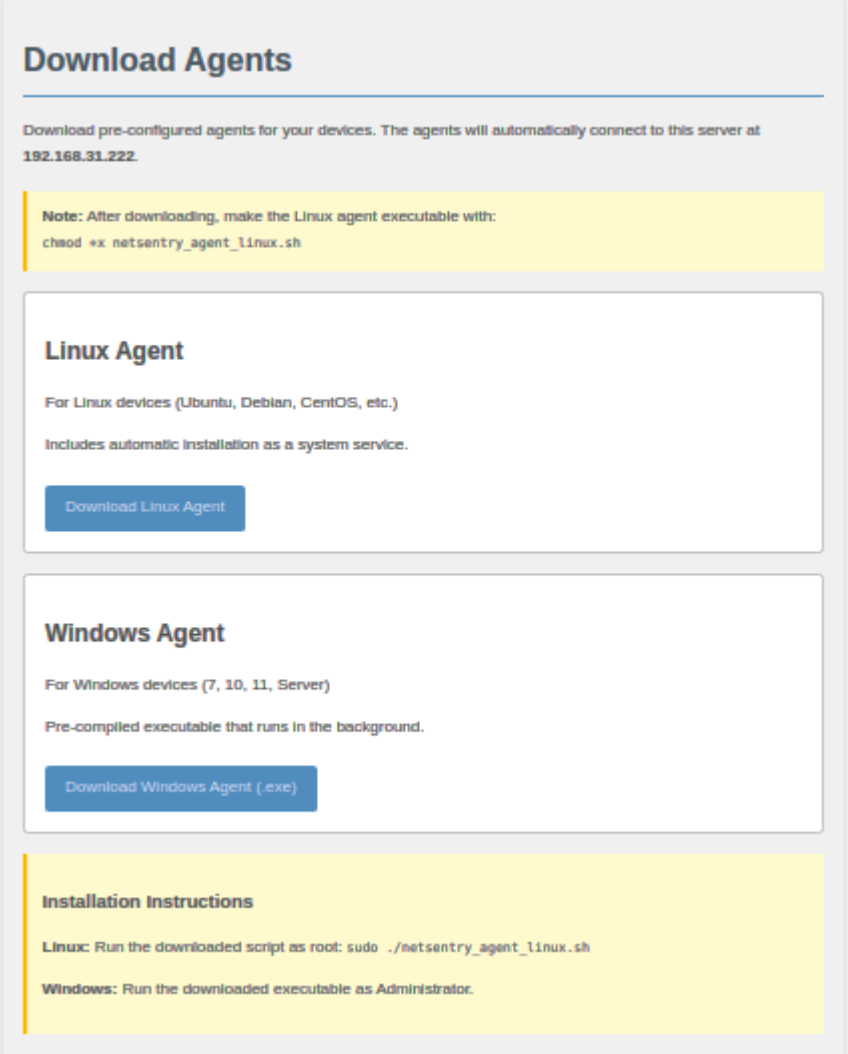

Fig. 6.9: Agent Download Page

Chapter 7: Conclusion & Future Work

7.1 Conclusion

NetSentry – The Network Monitoring System has successfully achieved its goal of providing real-time monitoring of devices and their connected peripherals within a local network. The system offers detailed visibility into system configurations, installed software, hardware activity, and potential vulnerabilities, all through a centralized and user-friendly dashboard. The integration of the NVD API for vulnerability scoring enhances the project's relevance in cybersecurity contexts, enabling administrators to identify and mitigate risks proactively. With its agent-based architecture and socket communication, NetSentry is both lightweight and efficient, making it suitable for use in small to medium-sized networks such as educational institutions and small businesses.

7.2 Future Work

While the current implementation meets the core objectives, there is significant scope for enhancement and expansion. Planned future developments include:

- ***AI/ML Integration:*** Implementing machine learning models to detect anomalies, unusual behavior, or potential threats based on device activity patterns.
- ***Automated Alerts:*** Adding real-time notifications via email or dashboard alerts for critical events like vulnerability detection or unauthorized peripheral use.
- ***Role-Based Access Control (RBAC):*** Introducing user roles and permissions to restrict and manage access to sensitive data on the dashboard.
- ***Data Encryption:*** Securing socket-based communication using SSL/TLS protocols to ensure data privacy and integrity.
- ***Extended Device Support:*** Enhancing compatibility across more operating systems and IoT devices.

These additions will further enhance the robustness, intelligence, and security of NetSentry, transforming it into a comprehensive and scalable solution for modern network monitoring challenges.

References

- National Institute of Standards and Technology. (n.d.). *National Vulnerability Database (NVD)*. Retrieved from https://nvd.nist.gov/

- Zabbix Team. Zabbix Network Monitoring Overview. Retrieved from https://www.zabbix.com/documentation

- Flask Development Team. Flask Documentation, https://flask.palletsprojects.com/en/2.1.x/

- Nagios. Network Monitoring Documentation. Retrieved from https://www.nagios.org/documentation/

- OpenVAS. Vulnerability Scanning Guide. Retrieved from https://openvas.org/documentation

- OWASP Foundation. Top 10 Vulnerabilities and Threats. OWASP, 2024. Retrieved from https://owasp.org/www-project-top-ten/