IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: **Ileperumaachchige D Gunathilaka**
Student Reference Number: **10898583**

Module Code: **PUSL3190**	Module Name: **PUSL3190 Computing Individual Project**
Coursework Title: **Root Cause Analysis**	
Deadline Date: **05.05.2025**	Member of staff responsible for coursework: **Mr. Chamara Disanayake**
Programme: **BSc (Hons) Computer Networks**	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed:**Ileperumaachchige D Gunathilaka**

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software.....

Overall mark _____% **Assessors Initials _____** **Date_____**

UNIVERSITY OF PLYMOUTH

PUSL3190 Computing Individual Project

Final Report

Root Cause Analysis

Supervisor: Mr. Chamara Disanayake

Name: Ileperumaachchige D Gunathilaka

Plymouth Index Number: 10898583

Degree Program: BSc (Hons) Computer Networks

Aknowledgement

Firstly, I want to express my deep appreciation to my supervisor, Mr. Chamara Disanayake, for the unwavering support, direction, and encouragement he provided during the entire period of my final year project. Throughout this entire exercise, from the initiation of the project to its culmination, his sound advice and constructive criticism were responsible for streamlining and strengthening the work, keeping it on the right track. His in-depth knowledge and experience in system development and networking had a significant bearing on the project's scope and the quality of the work in general. I am truly appreciative of his patience and professionalism in the way he always dealt with me. I would like to extend special thanks to the instructional staff and faculty members of the University of Plymouth's Department of Computer Network and Security, under the tutelage of whom my academic journey was navigated and supported. The skills and knowledge gained under your care will provide a foundational cornerstone that enabled the accomplishment of this project. Lectures, lab sessions, and research seminars greatly enhanced my academic progress and technical understanding.

I want to extend my deep appreciation to my institution for the provision of necessary facilities and a favourable learning environment. The facilities in the library, together with the computer labs and provision of academic journals, adequately supported the collection of information and the acquisition of knowledge concerning the technologies applied in this project. In addition, I want to appreciate the administrative support in my department, which was always willing to assist, especially in the submission and assessment procedures of project propositions. My appreciation goes to the experts and internet sites that enabled the availability of open-source Python libraries and network scripts, like Scapy, Python-Nmap, and PyQt6, to name but a few. These tools were a core part of my research, and the respective documentation, coupled with the forums, were crucial in overcoming technical challenges. Without the ease of availability of these resources, the development and testing processes of the diverse elements of the Root Course Analysis software would have been quite daunting.

This project is a completely personal task, carried out solely by me, and one of which I am proud to have completed on my own. I acknowledge the value of my concepts and the positive criticism which so improved the quality and elegance of my design and its elements. Discussing different notions and overcoming obstacles together was a big factor in my personal interest and satisfaction in the project. Encouragement and support shown by you, however, were crucial in sustaining the focus on the goals, especially in tough moments. I am obliged to the operators who authorised the testing of my project in their networks. Their willingness to support the testing of my software in a real-world environment offered valuable input and improved my understanding of the day-to-day problems faced by the network managers. This effort was crucial in enhancing both the functionality and usability of the system.

This project provided me with a valuable learning experience. It not only improved my technical and programming skills but also taught skills in problem-solving, time management, and independent work. During the development stage, a variety of problems arose, including those of real-time packets, the responsiveness of the user interface, and the setup of the application programming interfaces. However, solving them contributed to both personal and professional development. It taught me the value of keeping a calm mind under high-stress situations and the need to seek help when necessary.

Finally, I want to express my appreciation to everyone who, in some way, contributed to the accomplishment of this year's end project. Some of you directly contributed, while others contributed indirectly; nevertheless, their efforts collectively count and are valued. Although it might not be possible to mention everyone by name, I do appreciate the work that every one of you put in. The undertaking of this project was characterized by a series of challenges, though the aid received was largely responsible for the cultivation of a satisfying and enlightening experience. I am proud of my accomplishment and extend my sincere appreciation for the advice and guidance received from everyone involved in the undertaking.

Abstract

This report presents the development and implementation of a desktop-based network monitoring and management tool named **Root Course Analysis**. The purpose of this project is to assist network administrators in small to medium-sized enterprises by providing a reliable and user-friendly application to monitor, troubleshoot, and improve the performance of their network infrastructure. The main goal of Root Course Analysis is to reduce the complexity and manual effort required in identifying network issues by combining multiple features into one application. These features include real-time packet filtering, device scanning, Wi-Fi signal analysis, network troubleshooting tools, alert systems, and detailed report generation. The application is developed using Python as the programming language, with PyQt6 used to design the graphical user interface (GUI). It is built to run on Windows platforms and only requires a network adapter and an internet connection to function effectively.

One of the main components of the application is the **real-time packet filtering** system. It captures and analyses network packets using the Scapy library, allowing administrators to observe traffic patterns, detect suspicious activity, and gather data such as protocol distribution, traffic volume, and port usage. In addition, the system supports exporting packet data into JSON files for future analysis or reporting.

Another important feature is the **network device and vulnerability scanner**, implemented using the python-Nmap library. This component performs active scans to detect connected devices, open ports, operating systems, and potential vulnerabilities within the network. This scanning tool helps administrators identify security risks and make informed decisions about network protection.

The project also integrates an **automated alert system**, which uses the Twilio API to send SMS alerts when specific network anomalies are detected. This allows administrators to respond quickly to critical events, even when they are away from their systems. Users can configure the alert thresholds and intervals according to their preferences.

To support better decision-making and improve overall usability, the application includes a **report generation module** that summarizes key network statistics and performance data. It visualizes metrics such as latency, packet loss, throughput, bandwidth usage, and jitter using graphs powered by Matplotlib and Pandas libraries. Throughout the development process, the project faced several technical challenges, including ensuring accuracy in packet capturing under high traffic conditions, maintaining GUI responsiveness with real-time updates, and optimizing data storage using JSON files. These challenges were addressed through performance tuning, modular code structure, and testing in multiple network environments. In conclusion, Root Course Analysis offers a complete solution for local network management. It brings together various monitoring and security tools in a single interface, reducing the need for multiple software applications. The project is technically and operationally feasible, with potential for future upgrades such as cloud monitoring, database storage, and AI-based threat detection. This application is a valuable tool for network administrators seeking better control, visibility, and protection of their networks.

Table of Contents

Table of Figures

1. Introduction

In today's digital world, networks are essential for businesses, supporting operations, communication, and data sharing. Organizations of all sizes require trustworthy and secure network setups, managed by network administrators who ensure smooth and secure functionality. However, as networks expand with more devices and data, traditional management methods such as manual checks and non-real-time updates become inefficient and time-consuming.

To address these challenges, administrators need advanced tools offering:

- Real-time monitoring
- Threat detection
- Problem-solving assistance
- Automated reporting

While enterprise-grade solutions exist, they are often expensive, complex, and require technical expertise, leaving small-to-medium businesses (SMBs) underserved. This gap inspired Root Cause Analysis, a cost-effective Python-based desktop application designed for SMBs.

About the Project

Root Cause Analysis is a Windows-compatible tool built with PyQt6, featuring a user-friendly interface for real-time network monitoring and troubleshooting. It connects directly to ISP routers, providing administrators with actionable insights to manage network behaviour effectively.

Key Features

1. Real-Time Monitoring Dashboard

- Tracks latency, packet loss, bandwidth usage, throughput, jitter, and error rates.
- Visualises data via dynamic graphs for rapid anomaly detection.

2. Packet Filtering & Analysis

- Captures and analyses TCP, UDP, and ICMP traffic.
- Identifies high-traffic sources, unauthorised access, or misuse.

3. Network Device & Vulnerability Scanning

- Detects open ports, OS details, and potential security risks.

4. Wi-Fi Signal Analysis

- Evaluates signal strength and nearby networks.

5. SMS Alerts (via Twilio API)

- Customizable notifications for thresholds like packet loss or unknown devices.

6. Report Generation

- Generates performance reports with graphs for documentation and optimization.

Technology Stack

- Packet Analysis: Scapy
- Device Scanning: Python-Nmap
- Data Visualization: Matplotlib, Pandas
- SMS Alerts: Twilio
- Data Storage: Local JSON files

Development Approach

Built using modular, object-oriented programming for:

- Easier debugging
- Readable code
- Scalability for future enhancements

Inspiration

The project emerged from observing administrators struggle with limited tools for growing networks. Unlike complex enterprise solutions, Root Cause Analysis prioritises accessibility and affordability for SMBS without compromising functionality. This version improves readability with clear headings, bullet points, and logical flow while preserving your original content. Let me know if you would like further refinements!

2. Project Background

2.1 Problem Definition

In modern network environments, administrators face increasing challenges in maintaining performance, security, and reliability. Traditional tools often provide limited functionality, require manual operation, or lack real-time visibility into network activities. This results in delayed responses to connectivity issues, undetected security threats, and inefficient network resource usage. Many existing monitoring solutions are either too expensive or too complex for small to medium-sized organisations, leaving them without suitable tools for network management. Furthermore, the absence of automated alert systems and detailed performance reports adds to the difficulty of initiative-taking monitoring. Administrators often rely on multiple separate tools, which leads to inconsistent data and an increased workload. Without an integrated solution, diagnosing network faults and responding to anomalies becomes time-consuming and error prone. Therefore, there is a strong need for a cost-effective, user-friendly application that combines monitoring, troubleshooting, security scanning, and reporting into one platform to simplify network administration tasks.

2.2 Reason Behind the Project

The idea for this project came from the need for a simple, all-in-one network monitoring tool for small to medium-sized businesses. Most available tools are either costly or too complex for daily use by non-expert administrators. Many users struggle with separate tools for packet analysis, device scanning, and performance reporting. This project aims to combine all those essential features into one easy-to-use application. By using Python and PyQt6, the project offers flexibility and a modern interface while keeping costs low. The goal is to support administrators in managing their networks more effectively, with features like real-time monitoring, automated alerts, and clear reports to improve both security and performance.

2.3 Specific Issues or Challenges

During the development of the Root Course Analysis application, several technical and practical challenges were encountered. One of the main difficulties was ensuring accurate real-time packet capturing using the Scapy library without affecting system performance. In high-traffic environments, the tool had to process large amounts of data quickly, which sometimes caused delays and required performance optimisation.

Another issue was integrating the SMS alert system using the Twilio API. Limitations like API rate limits and unstable internet connectivity occasionally affected the reliability of alerts. Additionally, balancing the performance and responsiveness of the PyQt6 GUI while handling live data updates was complex and required careful programming to avoid freezing or lag.

Storage was also a challenge, as the application initially used JSON files to save data. When the dataset grew, read and write operations slowed down, highlighting the need for a more efficient storage system. Compatibility was another concern, although the application was designed for Windows, some features needed adjustments to work on Linux. Lastly, accurate network device scanning using Python-Nmap had to be handled carefully to avoid triggering firewall alerts or being blocked by security systems, especially in sensitive network environments.

3. Project Objectives

3.1 Short-Term Project Objectives

The short-term objectives of the Root Course Analysis project were focused on developing a working version of the application with all core functionalities. The main goal was to create a desktop-based tool that could run smoothly on Windows and support network administrators with real-time monitoring and troubleshooting features.

Key short-term objectives included:

- Designing a user-friendly interface using PyQt6 for easy navigation and operation.
- Implementing real-time packet capture and filtering to monitor network traffic.
- Developing a device and vulnerability scanning tool using Python-Nmap.
- Adding Wi-Fi signal analysis to show available networks and signal strength.
- Creating an alert system using Twilio to send SMS notifications during critical events.
- Building a report generation module with clear visualization of network statistics.
- Ensuring the system could store and manage network data using JSON files.
- Testing the application in different network conditions to validate performance.

These objectives were planned to be completed within the initial development phase, ensuring the application could perform all basic monitoring and management tasks effectively.

3.2 Special Measurable, Time-Bound, and Long-Term Goals

The long-term goals of the Root Course Analysis project are focused on enhancing the system's performance, usability, and scalability beyond the initial version. These goals are designed to bring lasting value to network administrators and improve the software’s capability to support more complex environments.

Measurable and Time-Bound Goals:

- Within the next 6 months, replace the JSON-based storage system with a lightweight database such as SQLite to handle large volumes of network data more efficiently.
- Add a user authentication and role-based access control system within 4 months to improve security and manage user access.
- Improve scanning accuracy and reduce system resource usage by optimizing code and updating libraries within 3 months.
- Expand operating system compatibility by fully supporting Linux and MacOS environments within 8 months.

Long-Term Goals:

- Integrate cloud-based monitoring features to allow remote access and control over the next year.
- Develop AI-based threat detection algorithms to identify unusual network patterns and predict issues.
- Build a web-based version of the tool to offer more flexibility and accessibility in future updates.
- Establish regular update cycles for ongoing improvements and feature enhancements.

3.3 Business Objectives

The main business objective of the Root Course Analysis project is to provide a cost-effective and reliable network monitoring solution for small and medium-sized organizations. Many businesses face high costs when trying to implement enterprise-level network management tools. This project aims to solve that issue by delivering a tool that offers similar core features at a much lower cost, without requiring advanced technical knowledge.

Another important goal is to reduce downtime and improve network stability. By offering real-time monitoring and automated alerts, the application helps network administrators detect and solve issues faster, which can save time and money for the business. The system also supports better security by identifying network vulnerabilities and unauthorized devices.

The project also focuses on scalability, allowing the application to be used in growing businesses without major changes. In the long term, the tool could be extended with cloud support and AI features to remain competitive in the market.

Overall, the business objectives include improving operational efficiency, reducing risk, supporting future expansion, and providing an affordable alternative to complex and expensive monitoring tools.

4. Project Deliverables

The Root Course Analysis project yielded a comprehensive desktop application, along with several supporting components. Each delivery was developed to meet both the functional and non-functional requirements of the project, ensuring that the application could be used effectively by network administrators.

4.1. Desktop Application (Root Course Analysis)

A fully developed and tested Python-based application with a PyQt6 user interface. This version includes:

- Real-time monitoring dashboard
- Packet filtering and protocol analysis
- Network device and vulnerability scanning
- Wi-Fi signal analysis
- Basic troubleshooting tools
- Automated alert system via SMS
- Report generation module with visual charts

4.2. Technical Documentation

A detailed explanation of the system architecture, source code structure, Python libraries used, and API configurations. It supports future development and maintenance.

4.3. User Manual

A guide explaining how to install, configure, and use the application. It includes screenshots and troubleshooting tips for ease of understanding.

4.4. Test Reports

Results from unit testing, performance testing, and system testing to ensure stability and accuracy.

4.5. Installation Package

An executable setup file for Windows, including required dependencies and instructions. These deliverables ensure that the application is ready for deployment and further development.

5. Literature Review

Network monitoring and management tools have become essential in maintaining the performance, security, and reliability of modern computer networks. Over the years, many commercial and open-source tools have been developed to support network administrators. This literature review explores some of the most widely used tools, their features, and the reasons for developing a new solution like Root Course Analysis.

One of the most recognized tools in this area is **Wireshark**, an open-source packet analyzer that allows users to capture and inspect traffic across a network. It supports a wide range of protocols and offers detailed packet-level information. While powerful, Wireshark has a complex interface that can be difficult for beginners. It also lacks built-in automation features such as real-time alerting or system-level reporting, which are useful for administrators who need immediate actions based on network conditions.

Another common tool is **Nagios**, known for its monitoring capabilities. It is widely used in enterprise environments to monitor servers, switches, applications, and services. However, Nagios requires a significant amount of configuration and server setup, which may not be ideal for smaller organizations or users with limited technical skills.

Nmap is another widely used tool, primarily for network scanning and vulnerability detection. It provides information about open ports, operating systems, and running services. Though powerful, Nmap is a command-line tool, which may be difficult for users without technical experience. Some graphical front ends like Zenmap exist, but they still require network configuration knowledge.

PRTG Network Monitor is a commercial tool that offers a user-friendly interface and supports real-time monitoring, alerts, and reporting. It is designed for larger networks and comes with a cost, making it less accessible for smaller businesses or individual users.

Many of these tools are specialized, focusing on one or two areas such as packet analysis, device scanning, or monitoring. As a result, users often need to use multiple tools to get a full picture of their network, which can be time-consuming and inefficient.

The idea behind Root Course Analysis is to combine essential features from these tools packet filtering, device scanning, vulnerability analysis, alerting, and reporting into one simple, easy-to-use desktop application. By using Python and PyQt6, the project focuses on providing an accessible tool with an intuitive interface, especially suitable for small and medium-sized networks. The integration of real-time monitoring with SMS alerts through Twilio also adds a level of automation not available in many basic tools.

In conclusion, while existing tools are powerful and widely used in the industry, they often have limitations in terms of usability, integration, or cost. Root Course Analysis aims to address these gaps by offering a unified solution that combines multiple network management functions into one platform, without requiring advanced technical knowledge or expensive hardware.

6. Method of Approach

6.1 Feasibility Study

6.1.2 Operational Feasibility Study

Operational feasibility focuses on how well a system fits within the existing environment and whether users can easily adopt and use it without major disruptions. For the Root Course Analysis project, operational feasibility was carefully evaluated to ensure the application could be used effectively by network administrators in small to medium-sized organizations. The main strength of the system is its ease of use. The graphical user interface, built with PyQt6, was designed to be simple, clear, and user-friendly. Even users with limited technical knowledge can navigate the system and access core features such as real-time monitoring, scanning, and report generation. The application does not require advanced configuration or command-line input, which reduces the learning curve.

The system also supports smooth integration with existing network setups. It connects to the network through a standard cable connection to the main router and does not need additional hardware. This makes it compatible with most Windows-based setups commonly used in office environments.

From a maintenance perspective, the tool is easy to update and troubleshoot. Its modular code structure and use of local JSON storage make it simple to modify or expand specific features without affecting the entire system. Error handling and logging are also included to help users identify and resolve issues.

In terms of resource usage, the system is lightweight and runs efficiently on standard computers with basic specifications. It does not require powerful processors or large memory, making it suitable for organizations that may not have high-end hardware. The use of Python also allows the system to remain flexible and extensible in future updates.

Another important point in the feasibility study is user acceptance. The application was designed with the needs of network administrators in mind. Features like automated SMS alerts, live packet filtering, and quick scan results directly support their daily tasks. Positive feedback received during testing confirmed that the system addresses real-world problems and makes network monitoring more convenient and effective. Finally, the system offers scalability. While it currently supports small to medium networks, future versions can be expanded to include database support, remote access, and cloud-based services. This ensures that the system can grow with the organization and remain relevant in more complex environments. In conclusion, Root Course Analysis is operationally feasible for its target users. It is easy to use, simple to integrate, and performs well under normal conditions. With its flexible design and useful features, it provides a practical solution for improving network management without requiring major changes or high costs.

6.1.2 Technical Feasibility Study

The technical feasibility of the Root Course Analysis project focuses on whether the available tools, technologies, and resources are suitable for developing and running the application effectively. It also evaluates if the development team has the required skills to complete the project successfully using the chosen technology stack. The system is built using Python, a widely used and flexible programming language with strong community support. Python's readability and simplicity make it a good choice for rapid development and testing. The graphical user interface (GUI) is developed using PyQt6, which offers a modern look and supports advanced interface features while still being user-friendly.

For real-time packet filtering and analysis, the system uses Scapy, a powerful Python library for packet manipulation. It is capable of capturing, decoding, and analyzing different types of network traffic such as TCP, UDP, and ICMP. The project also integrates Python-Nmap for scanning connected devices and detecting open ports, services, and vulnerabilities. These libraries have been tested and proven in various network tools, ensuring they are technically reliable for the project.

The SMS alert system is implemented using the Twilio API, which provides a reliable method for sending messages when a network anomaly is detected. It requires an internet connection and basic setup, but it works well once integrated. Challenges such as API rate limits and internet dependency were addressed during development by optimizing the alert logic and implementing retry methods.

Matplotlib and Pandas were used for generating charts and graphs in the report generation module. These libraries are efficient and provide professional-quality visual output, which helps in analyzing network performance data.

In terms of system requirements, the application is designed to run on standard Windows systems. It does not require high-performance hardware or complex installations. A basic setup with Python 3.x, internet access, and a network adapter is enough to operate the system. This makes the tool accessible to organizations with limited resources.

During development, some technical challenges were faced, such as maintaining GUI responsiveness while handling real-time data, and optimizing JSON file storage for performance. These issues were solved through better code structure, data handling methods, and separating background processes from the user interface.

The development team had prior experience in Python programming and basic networking concepts, which made it possible to manage the technical complexity of the project. All selected libraries had good documentation, which further supported the technical implementation. The application is also scalable and extensible. Its modular design allows future integration of new features like database support, cloud monitoring, or AI-based traffic analysis without rewriting the core structure. This flexibility ensures that the system can adapt to future needs and technical advancements. In conclusion, the Root Course Analysis project is technically feasible. The chosen technologies are appropriate, stable, and supported. The development team had the skills and tools needed to complete the system, and the application runs efficiently on standard machines. With a strong technical foundation, the project is capable of evolving further in future versions.

6.1.3 Outline Budget

The Root Course Analysis project was developed with a focus on cost-efficiency and the use of open-source technologies. As a project, most of the development work was done using free or low-cost tools. However, some essential resources and services were required to complete the project successfully. The following is an outline budget that includes software, hardware (if applicable), internet usage, and testing-related expenses.

Item	Description	Estimated Cost (LKR)
Laptop or server	Used for coding, testing, and development.	200000.00
Internet Costs	For downloading libraries, documentation, and API usage.	2,000.00
Twilio SMS Credit	For sending test SMS alerts through the Twilio API.	3,500.00
Python and Libraries	All open-source and freely available.	0.00
PyQt6 Toolkit	Open-source license used for development.	0.00
Testing Devices	Used existing devices on the network for scanning and analysis.	0.00
Electricity Costs	Estimated electricity usage for 3 months of development.	1,500.00
Backup Storage (USB/Cloud)	Used for storing source code and backups.	1,000.00

6.2 Development Methodology

To complete the Root Course Analysis project successfully, a simple and flexible development approach was needed. For this reason, the Incremental Development Methodology was followed. This method helped in building the project step by step, testing each part before moving to the next. It also allowed continuous improvements throughout the development process.

Why Incremental Methodology?

The project had many features such as real-time monitoring, device scanning, alert systems, and report generation. Trying to build all of them at once would be difficult. So, each feature was developed separately in smaller parts, then added to the main system. This made it easier to test and fix problems early.

Development Stages

1. **Planning and Research**
   In the beginning, time was spent understanding network monitoring tools and identifying user needs. Free tools like Scapy, Python-Nmap, and PyQt6 were selected after comparing alternatives.
2. **Design Phase**
   The system layout and user interface were planned. The GUI was first drawn on paper, then created using PyQt6.
3. **Module-by-Module Development**
   Each function (like packet filtering or scanning) was developed as a separate module. These were tested individually before being connected to the full system.
4. **Testing and Debugging**
   After completing each module, it was tested using different scenarios. Errors were fixed, and performance was improved to avoid system slowdowns.
5. **Integration and Final Testing**
   All modules were connected and tested as one complete application. The system was checked for bugs, usability, and performance.
6. **Documentation and Report Writing**
   Once development was completed, user guides and technical documentation were written. The final report was prepared for submission.

Benefits of This Method

- Easy to manage small tasks
- Early detection of problems
- Timesaving because only the working parts were added
- Allows feedback and improvements during development

Tools Used

- **Python 3.x** – for coding the system
- **PyQt6** – to create the GUI
- **Scapy and Python-Nmap** – for packet and device scanning
- **Twilio** – for SMS alerts
- **Matplotlib and Pandas** – for graphs and data analysis

6.3 Programming Languages and Tools

The development of the Root Course Analysis application was done using open-source programming languages and tools. These were chosen based on their availability, ease of use, and ability to handle networking and GUI tasks effectively. Below is a list of the main programming languages and tools used during the project.

1. Python 3.13.2
Python was the main programming language used for building the entire system. It was selected because it is easy to learn, has clear syntax, and supports many libraries. Python is also widely used in networking, automation, and data analysis, making it a perfect choice for this type of project.

2. PyQt6
PyQt6 was used to create a graphical user interface (GUI). It helped design a clean and user-friendly layout for the application. With PyQt6, all windows, buttons, and tables were developed to make the system easy to use for network administrators.

3. Scapy
Scapy is a powerful Python library for packet manipulation and analysis. It was used in this project to capture and analyze real-time network traffic, detect IP addresses, and show packet details.

4. Python-Nmap
This library is a Python wrapper for the Nmap tool. It was used to scan the network for connected devices, open ports, and services. It also helped identify possible vulnerabilities in the network.

5. Twilio API
Twilio's API was used to send SMS alerts when network problems were detected. This allowed administrators to receive notifications even when they were not near the computer.

6. Matplotlib and Pandas
These two libraries were used to generate charts and graphs for reports. Matplotlib helped create visual graphs, while Pandas handled the data for performance analysis and reporting.

7. JSON
JSON files were used to store data such as packet logs, scan results, and network statistics. It was a simple and effective way to save and read data within the system.

8. VS Code (Visual Studio Code)
Visual Studio Code was the main code editor used for writing and testing the Python scripts. It provided useful features like auto-complete, debugging, and Git support.

6.4 Third-Party Components and Libraries

The Root Course Analysis project was built using several third-party libraries and components. These tools helped speed up the development process and add advanced features without the need to write everything from scratch. All the components used are open-sourced and work well with Python. Below is a list of the main third-party libraries used in this project, along with a short explanation of their purpose.

1. Scapy

Scapy is a Python library used for packet capturing and network analysis. It allowed the system to capture live packets, identify protocols, track IP addresses, and monitor traffic flow. It is powerful and flexible, making it perfect for real-time network monitoring.

2. Python-Nmap

This is a Python wrapper for the popular Nmap tool. It was used to scan networks, detect connected devices, open ports, running services, and possible vulnerabilities. It helped in building the network scanner part of the application.

3. Twilio

Twilio provides a cloud-based communication platform. Its API was used to send SMS alerts to users when network anomalies were detected. This allowed real-time notifications to be delivered during network issues.

4. PyQt6

PyQt6 is a set of Python bindings for the Qt toolkit. It was used to develop the graphical user interface (GUI) of the application. It helped create a clean and user-friendly design for easy interaction.

5. Matplotlib

Matplotlib is a popular plotting library in Python. It was used to create graphs and charts that show performance metrics such as bandwidth, latency, and packet loss. These visualizations helped users understand network behavior better.

6. Pandas

Pandas is a Python library used for handling and analyzing data. It worked together with Matplotlib to prepare data for graph generation and reporting. It made it easier to organize and display network information clearly.

7. psutil

This library was used to get information about system usage like CPU, memory, and network connections. It added extra monitoring features to the dashboard.

8. requests

The requests library was used for making HTTP requests. It was mainly used during the integration of third-party APIS like Twilio.

9. dnspython

Dnspython is a toolkit for DNS-related operations. It helped in resolving domain names and checking DNS records, adding an extra layer of analysis to the system.

10. netifaces

This library was used to get details about the system’s network interfaces, like IP addresses and connection types. It supported the scanning and monitoring features.

7. Requirements

7.1 Functional Requirements

Functional requirements describe the key features and actions that the Root Course Analysis system must perform. These requirements ensure that the application meets the needs of network administrators and delivers all the expected functionalities.
Below are the main functional requirements of the system:

1. Real-Time Packet Monitoring

- The system must capture and display live network packets.
- It should show details such as protocol type, source and destination IP, and port numbers.
- Users should be able to view packet statistics and filter them by protocol.

2. Network Device Scanning

- The system must scan the local network and list all connected devices.
- It should detect each device's IP address, MAC address, open ports, and operating system (if available).
- A vulnerability scan must be included to identify possible network risks.

3. Wi-Fi Signal Analysis

- The system must detect available Wi-Fi networks in the area.
- It should display signal strength, security type, and SSID (network name).
- This helps in identifying weak or unstable wireless connections.

4. SMS Alert System

- The system must monitor unusual network behavior (e.g., high packet loss or unknown device detection).
- It should automatically send an SMS alert to the user using the Twilio API.
- Alert thresholds must be configurable by the user.

5. Basic Troubleshooting Tools

- The application must include common troubleshooting tools like ping, traceroute, and speed test.
- These tools help users identify and fix connectivity issues quickly.

6. Report Generation

- The system must generate a summary report on network activity and performance.
- Reports should include visual charts for bandwidth, latency, packet loss, and other metrics.
- Users should be able to export or save the reports.

7. Data Storage

- The system must store collected data (packet logs, scan results, and reports) in JSON format.
- It should allow users to retrieve old data for review and analysis.

8. User Interface (GUI)

- The system must have a user-friendly GUI built with PyQt6.
- Users should be able to navigate between features easily.
- All buttons, inputs, and displays should be clear and functional.

9. Error Handling and Notifications

- The system must handle errors such as failed scans, no internet connection, or API issues.
- It should display warning messages or logs to inform the user when something goes wrong.

7.2 Non-Functional Requirements

Non-functional requirements describe how the system should behave and perform, rather than what it should do. These requirements help ensure the Root Course Analysis application is reliable, efficient, and easy to use. Below are the main non-functional requirements for the project:

1. Performance

- The system must run smoothly without slowing down the user's computer.
- Packet capturing and scanning should be fast, even in moderate to high-traffic networks.
- The application should respond to user actions without noticeable delays.

2. Usability

- The user interface must be simple and easy to understand.
- Even users with basic technical knowledge should be able to operate the system.
- Clear labels, tooltips, and error messages should be used to guide the user.

3. Reliability

- The system must work consistently without frequent crashes or errors.
- It should be able to run for long periods while continuously monitoring the network.
- SMS alerts must be sent accurately when problems are detected.

4. Compatibility

- The application must run on Windows systems.
- It should support basic hardware setups with a network adapter and internet access.
- Partial support for Linux systems should also be available with minor adjustments.

5. Security

- The system must use safe scanning methods that do not trigger firewall blocks.
- Sensitive data like scanned results and user settings should be stored securely.
- The application must avoid unnecessary exposure of system or network information.

6. Maintainability

- The code should be written in a clean and modular way to allow easy updates.
- Future developers should be able to add or fix features without breaking the system.
- Configuration files and libraries should be clearly organized.

7. Portability

- The application should be easy to move from one machine to another.
- Users should be able to reinstall the software with minimal setup.

8. Scalability

- The system should be able to handle growing amounts of data.
- In the future, it should support larger networks and additional features like database integration or cloud access.

9. Availability

- The application should be available for use at any time once installed.
- It should start up quickly and allow users to begin monitoring without delay.

10. Documentation

- Proper user guides and technical documentation must be provided.
- The documentation should help users install, configure, and troubleshoot the application.

7.3 Hardware and Software Requirements

7.3.1 Software Requirements

The Root Course Analysis application is built using open-source technologies and is designed to run on commonly available software platforms. Below are the software requirements needed to install and use the application successfully:

1. Operating System

- **Windows 10 or later** (Recommended)
- **Linux (Ubuntu 18.04 or above)** – Supported with minor adjustments

The application is mainly designed for Windows but can be modified to run on Linux-based systems.

2. Python Interpreter

- **Python 3.1 or later**

Python must be installed on the system to run the application. It is recommended to use the latest stable version for best performance.

3. Required Python Libraries and Packages

The following libraries must be installed using pip (Python's package installer):

- **PyQt6** – For the GUI (Graphical User Interface)
- **Scapy** – For packet capturing and network traffic analysis
- **Python-Nmap** – For scanning devices and detecting open ports
- **Twilio** – For sending SMS alerts
- **Matplotlib** – For creating graphs and charts
- **Pandas** – For handling and analyzing data
- **psutil** – For checking system and network usage
- **requests** – For handling API requests
- **dnspython** – For DNS lookups and domain analysis
- **netifaces** – For detecting network interfaces

4. Code Editor (Optional)

- **Visual Studio Code (VS Code)** or any other preferred code editor

Used for viewing, editing, and running the Python code during development or customization.

5. Web Browser

- Any modern web browser (e.g., Google Chrome, Firefox)

Needed only if the user wants to access the Twilio dashboard or read documentation online.

6. JSON Viewer (Optional)

- Any lightweight JSON viewer or editor

Helpful when reviewing saved network data logs in JSON format.

7.3.2 Hardware Requirements

The Root Course Analysis application is designed to run on regular desktop or laptop computers without the need for expensive or high-performance hardware. The system was developed and tested on a standard Windows machine, making it suitable for use in small to medium-sized organizations. Below are the minimum and recommended hardware requirements:

Minimum Hardware Requirements

- **Processor**: Dual-core CPU (Intel i3 or equivalent)
- **RAM**: 8 GB
- **Storage**: 500 MB of free disk space
- **Display**: 13" screen with at least 1280x720 resolution
- **Network Adapter**: Required for monitoring and scanning network traffic
- **Internet Connection**: Needed for SMS alerts via Twilio

These specifications are enough to run the application with basic monitoring and scanning features.

Recommended Hardware Requirements

- **Processor**: Quad-core CPU (Intel i5 or above)
- **RAM**: 16 GB or higher
- **Storage**: 5 GB of free disk space
- **Display**: 15" screen with 1920x1080 resolution for better visual clarity
- **Network Adapter**: Ethernet or Wi-Fi adapter with monitor mode capability
- **Internet Connection**: Stable connection for real-time updates and alert services

The recommended setup ensures smoother performance, especially when handling large amounts of network traffic or using multiple features at once.

8. User Interfaces

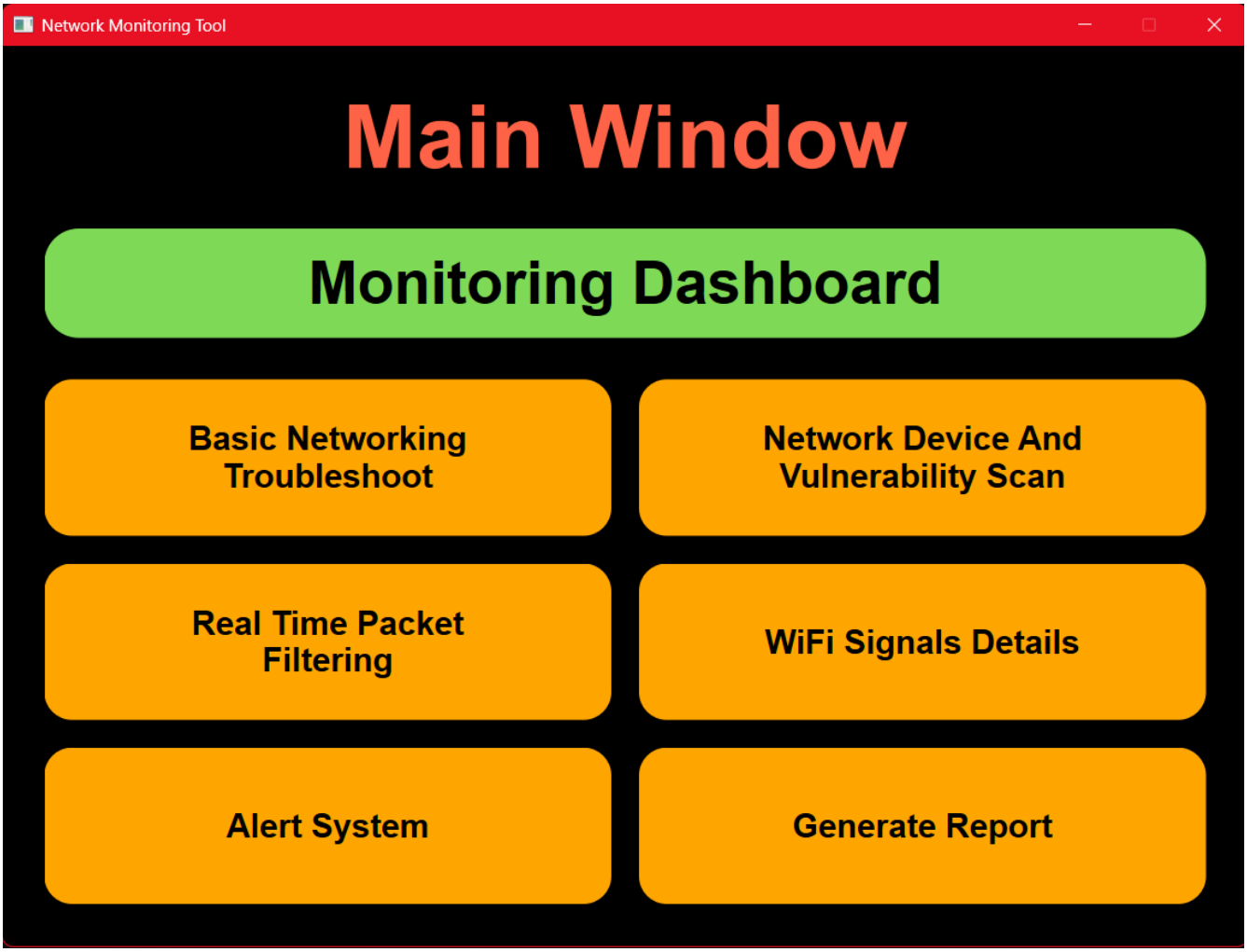


Figure 1 Main Window

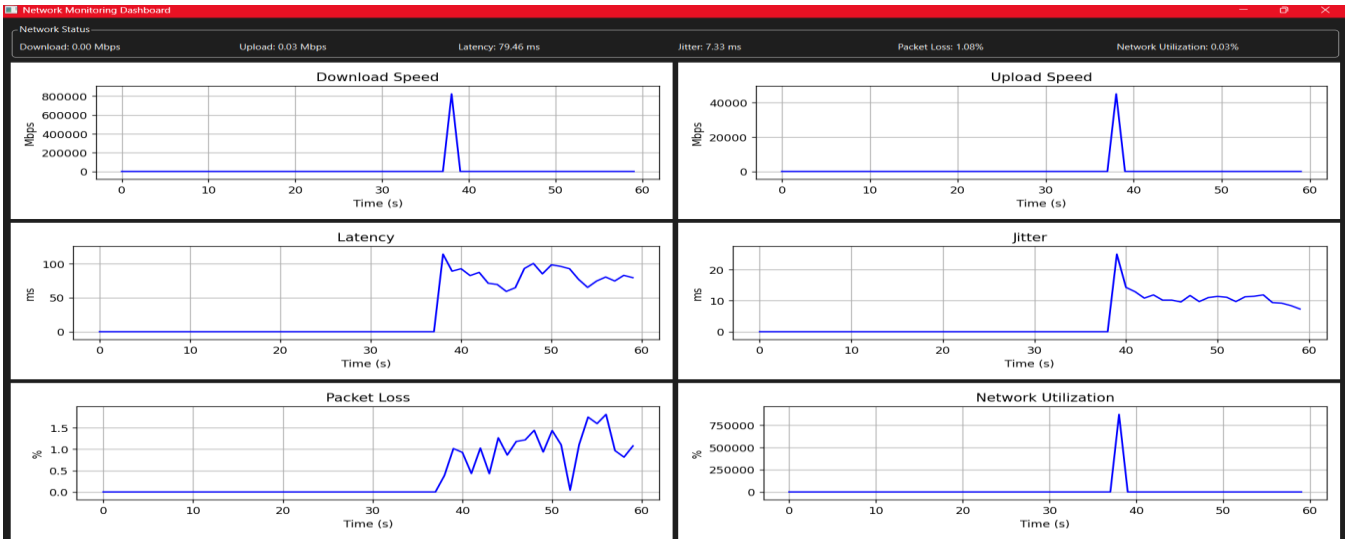


Figure 2 Monitoring Dashboard

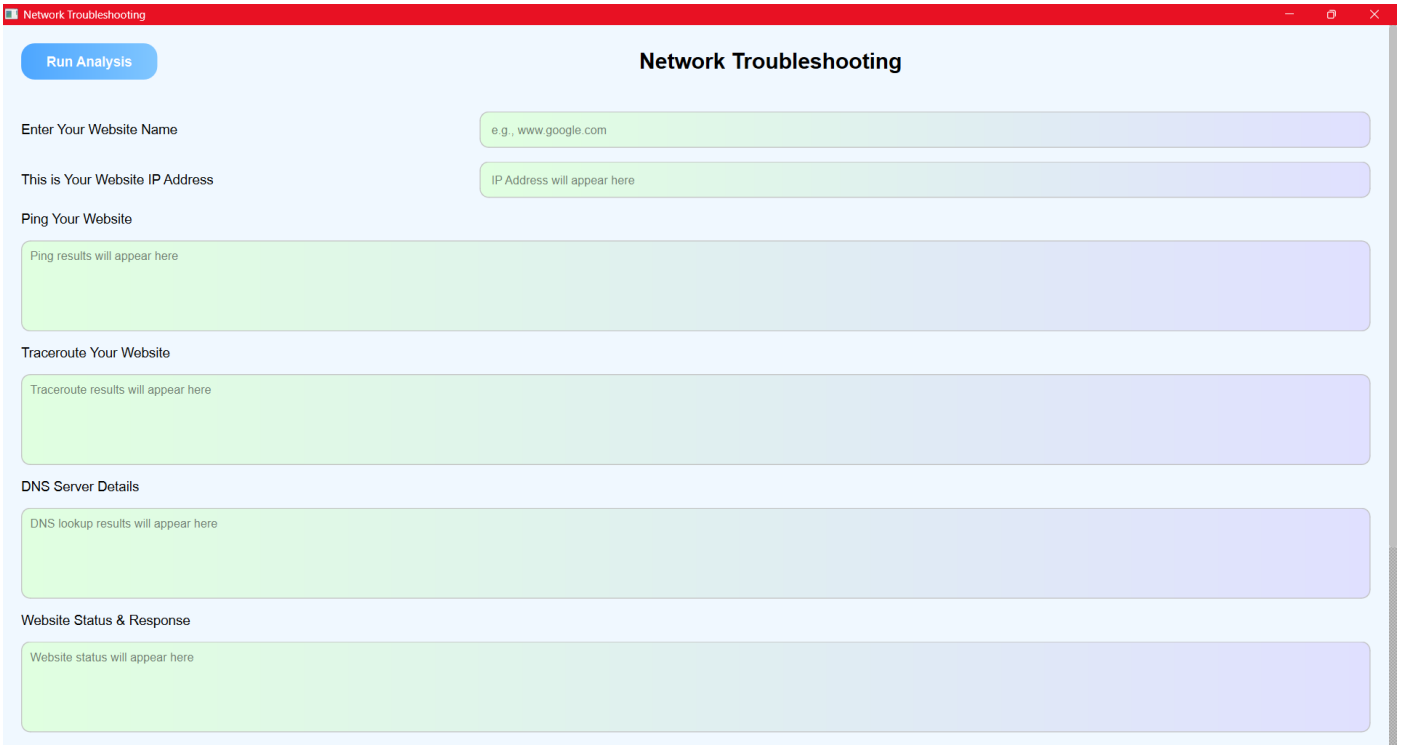


Figure 3 Basic Networking Troubleshoot

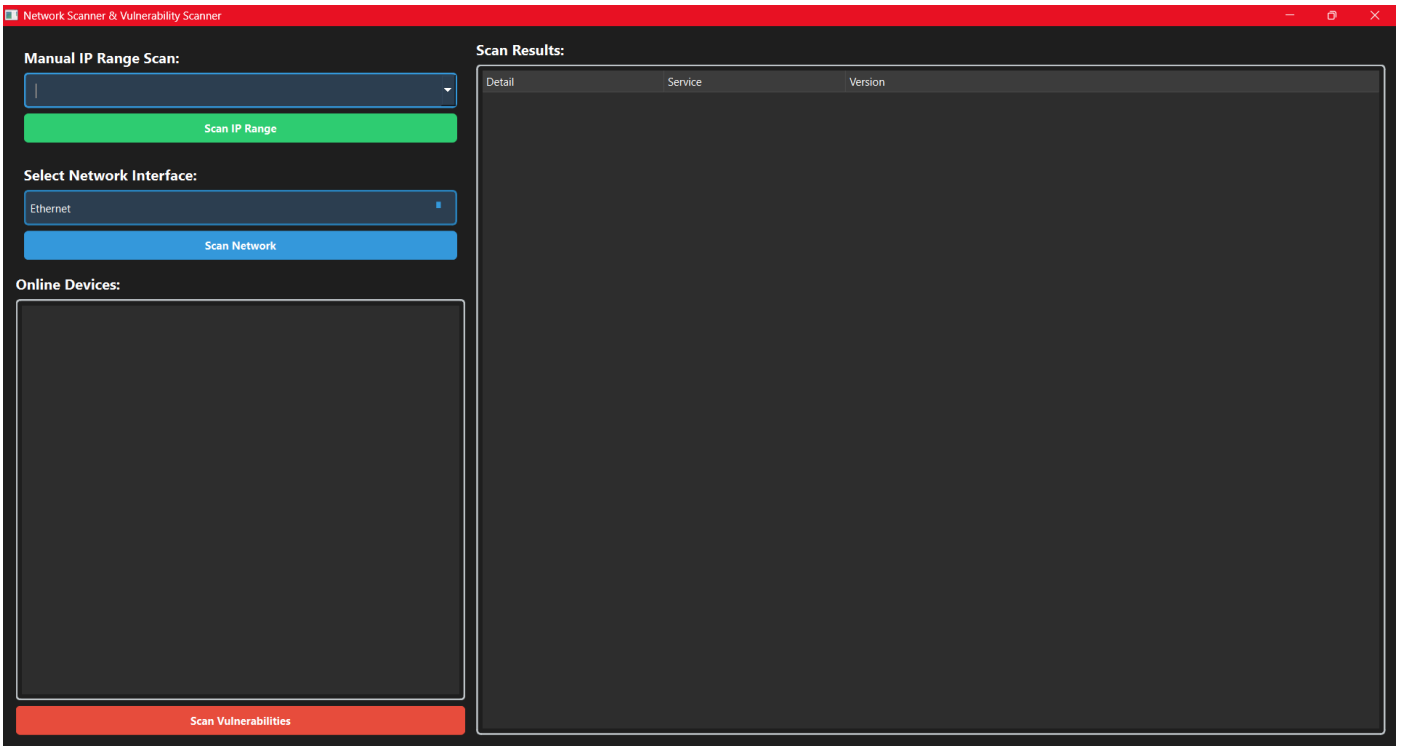


Figure 4 Networking Device and Vulnerability Scan

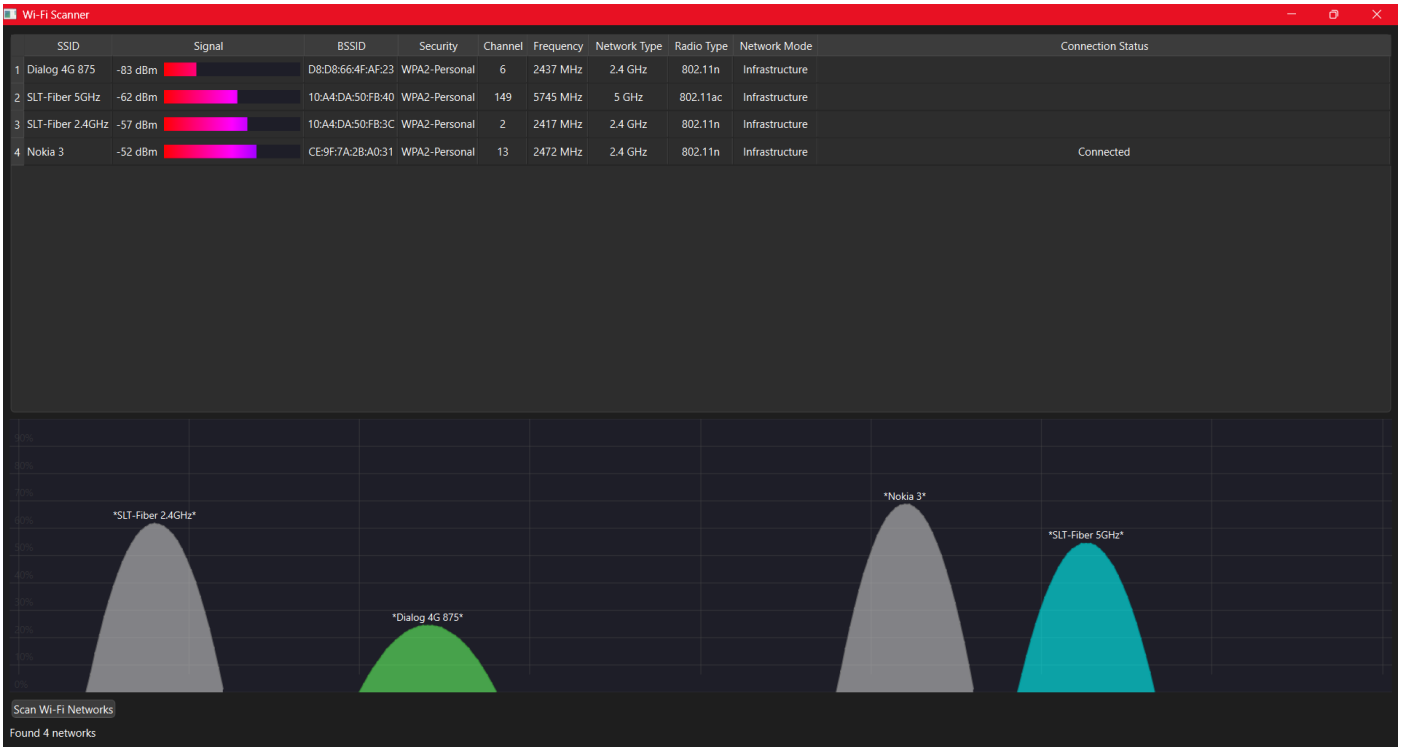


Figure 5 Wi-Fi Signals Details

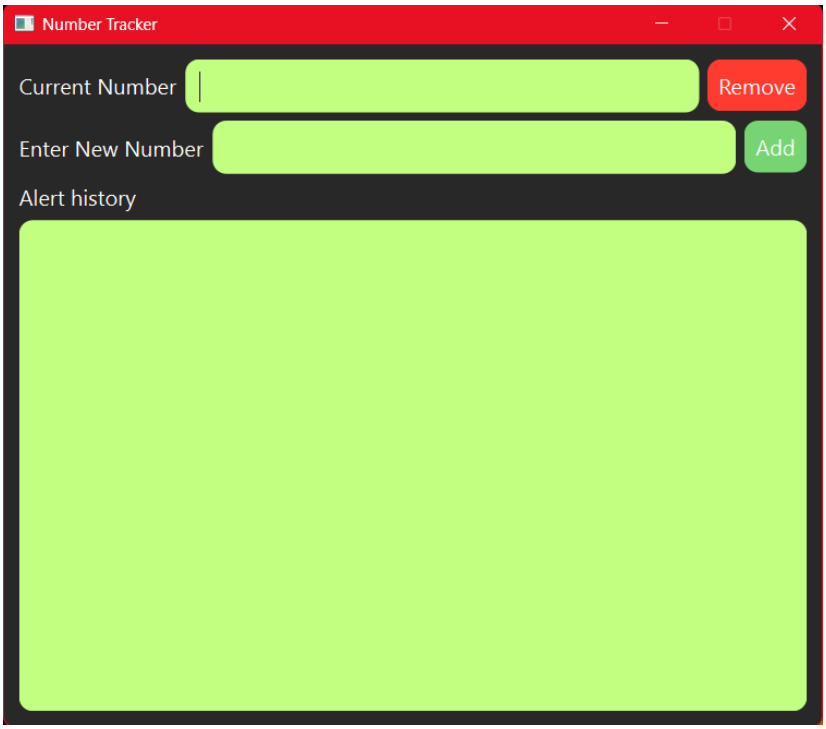


Figure 6 Alert System

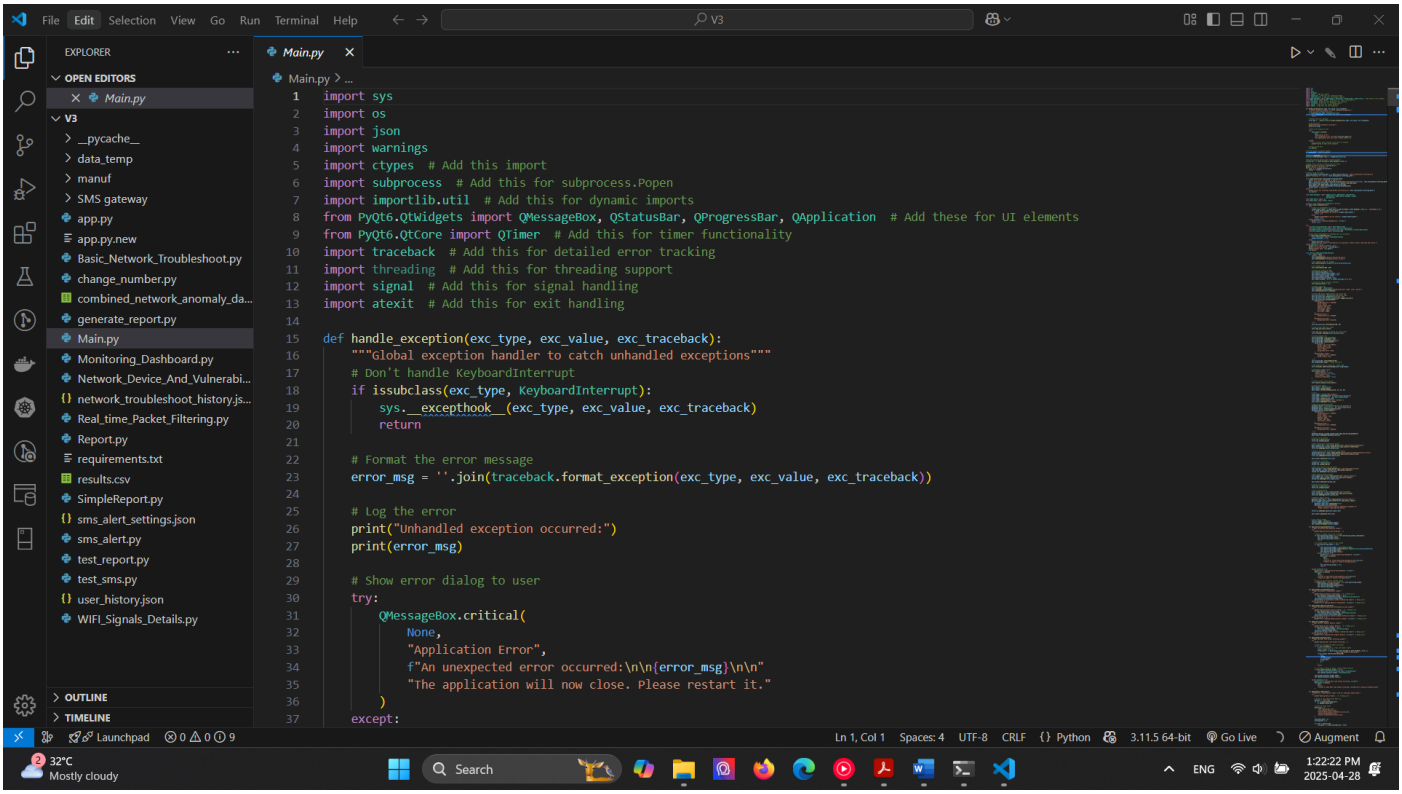


Figure 7Main Application run file

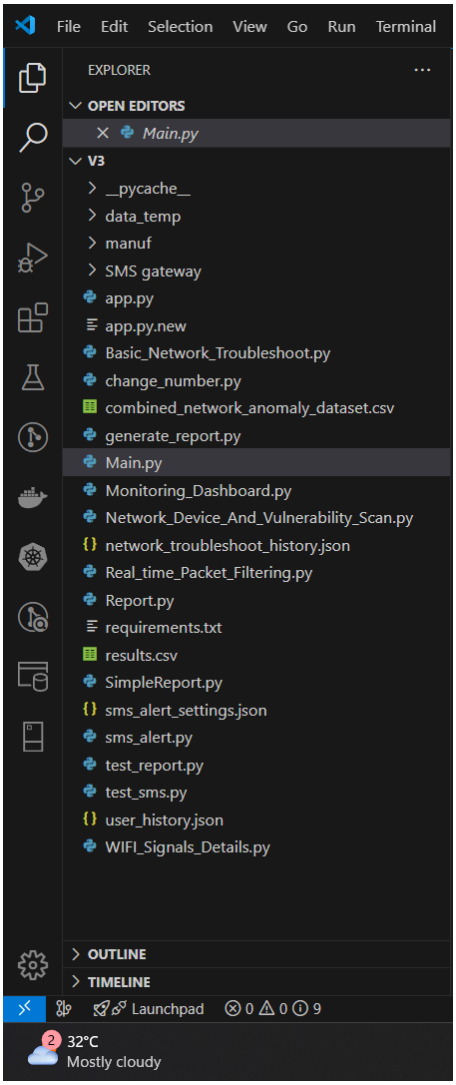


Figure 8 Project files

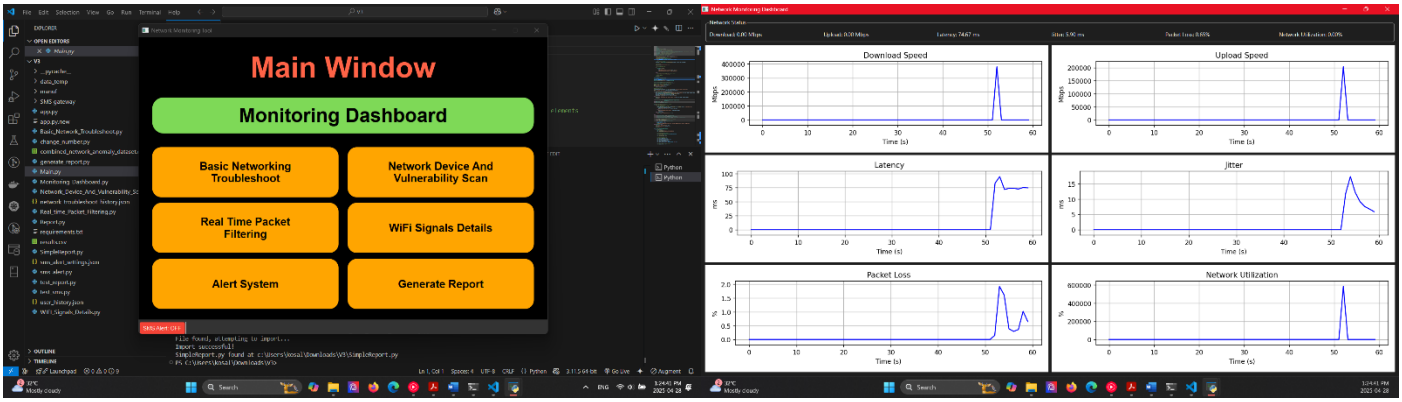


Figure 9 Dashboard and Main interface

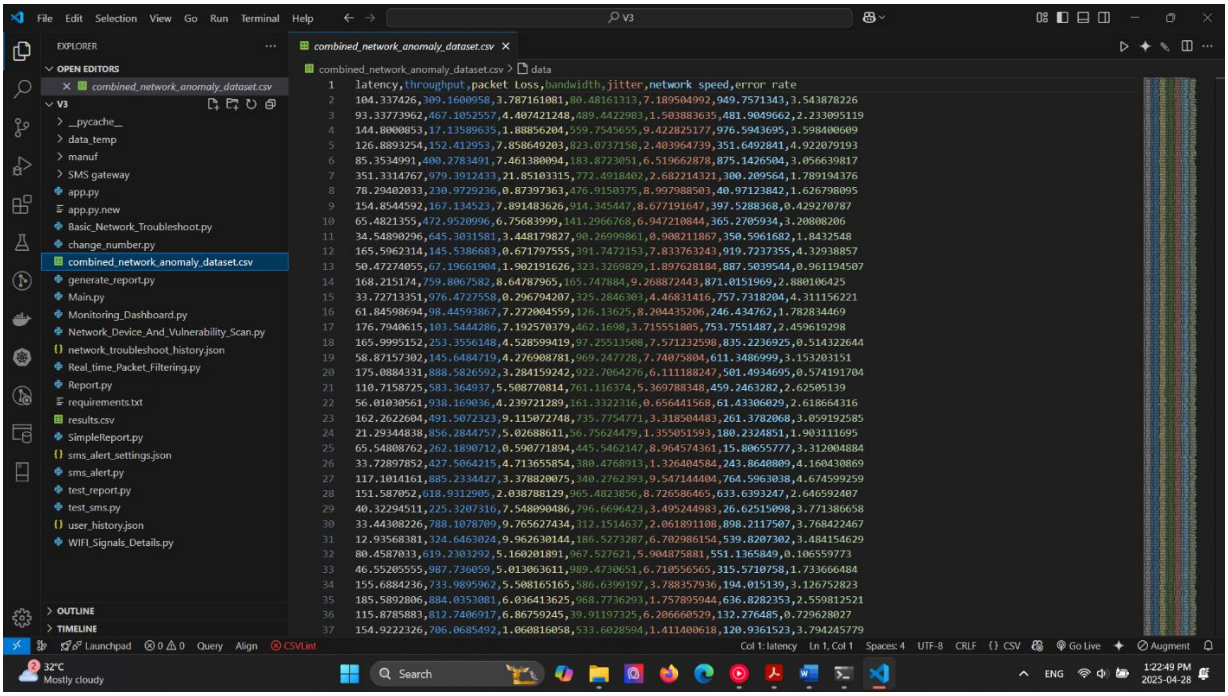


Figure 10 Data Set CSV file

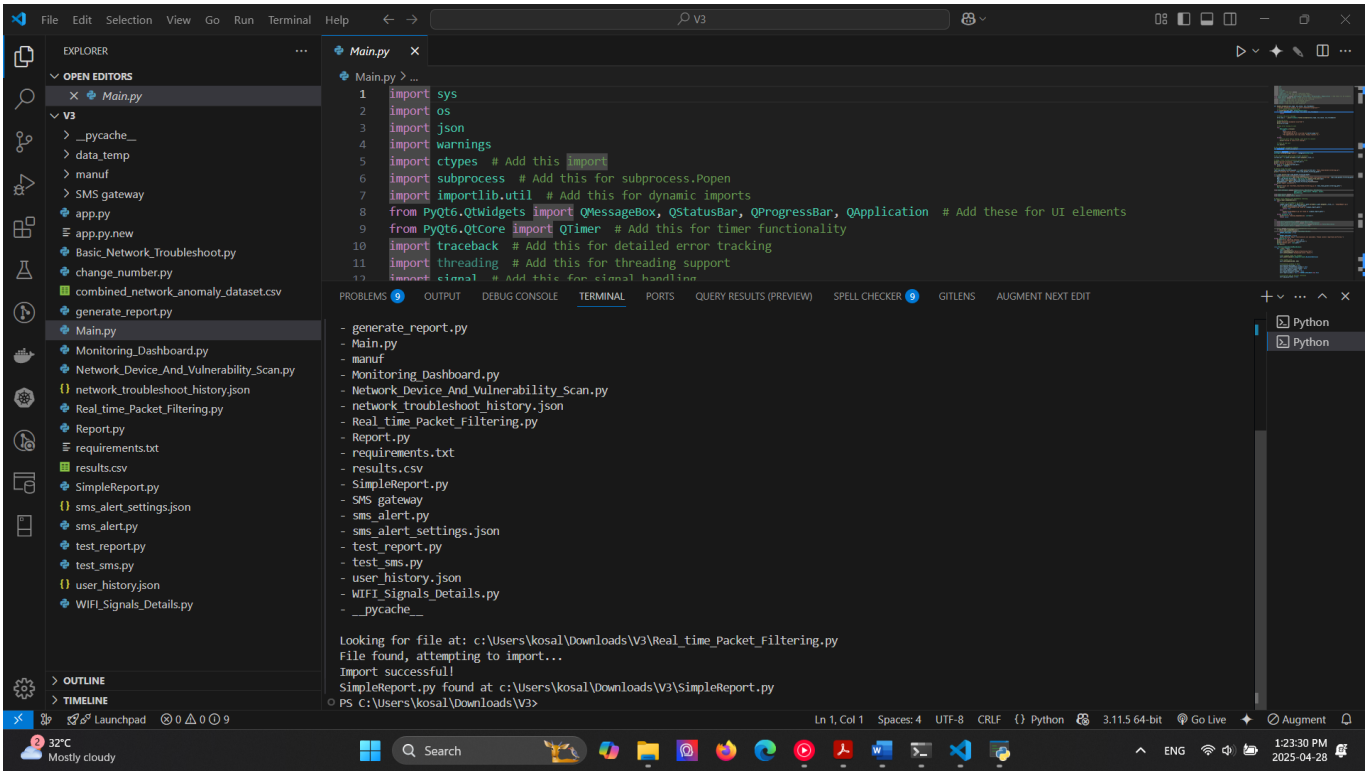


Figure 11 Terminal Output

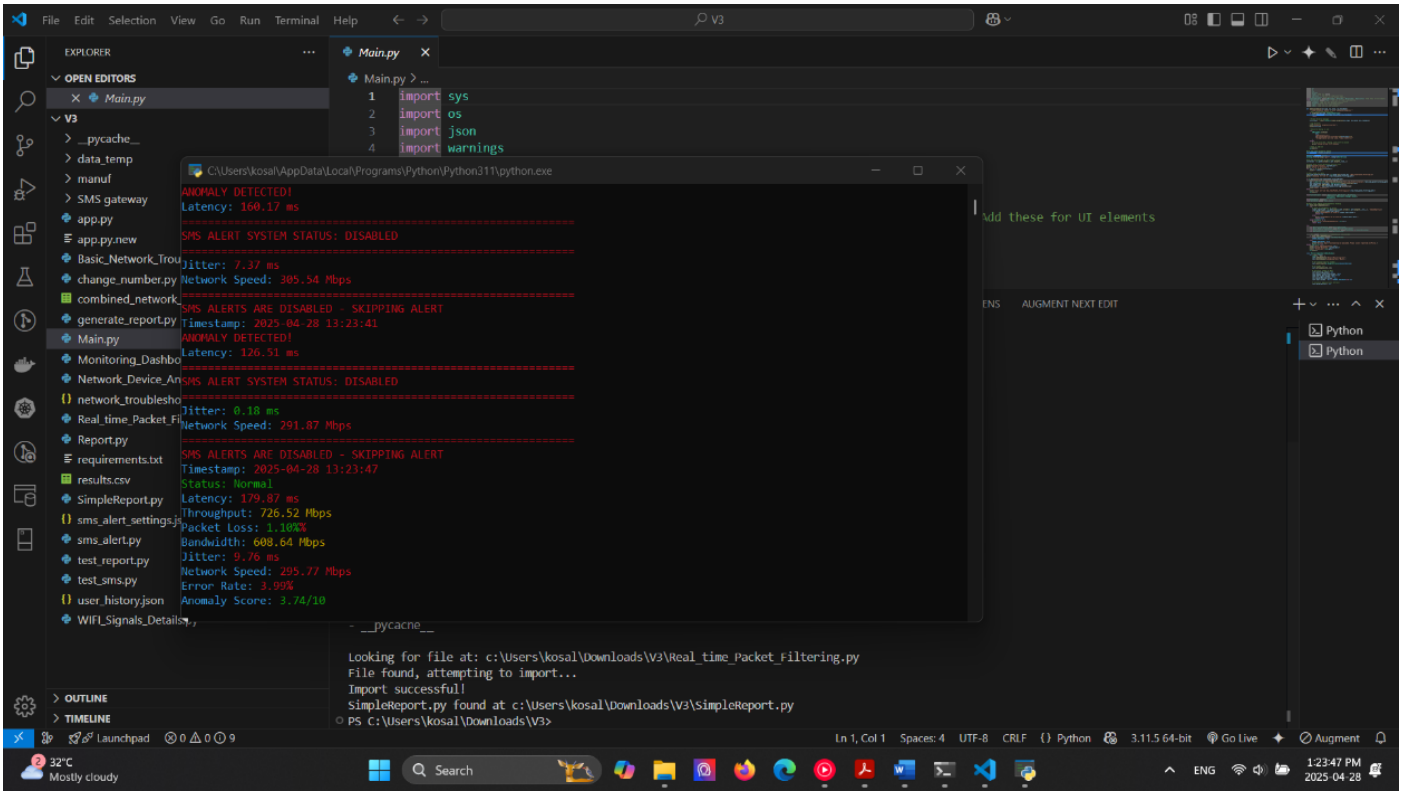


Figure 12 Anomaly Detected Window

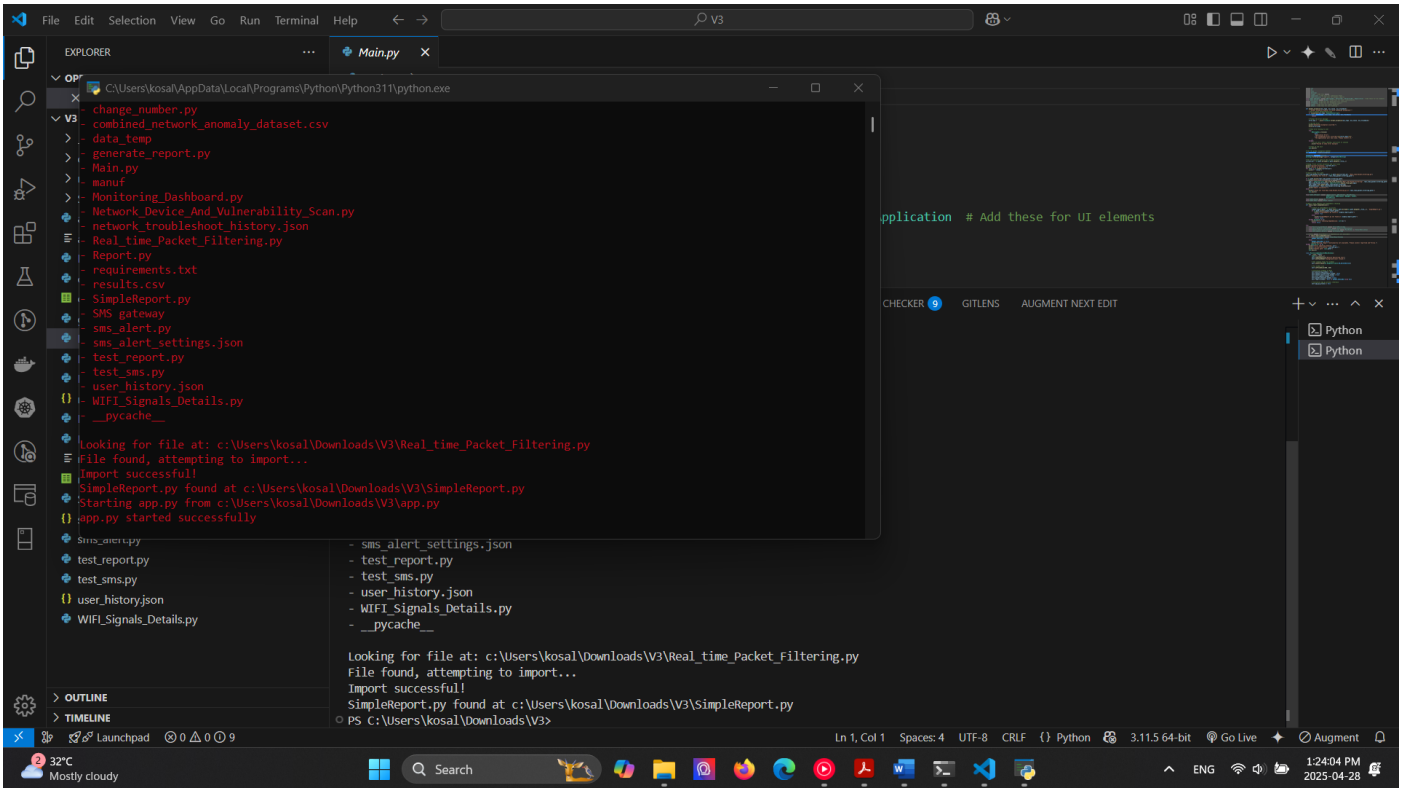


Figure 13 Terminal Output Window 2

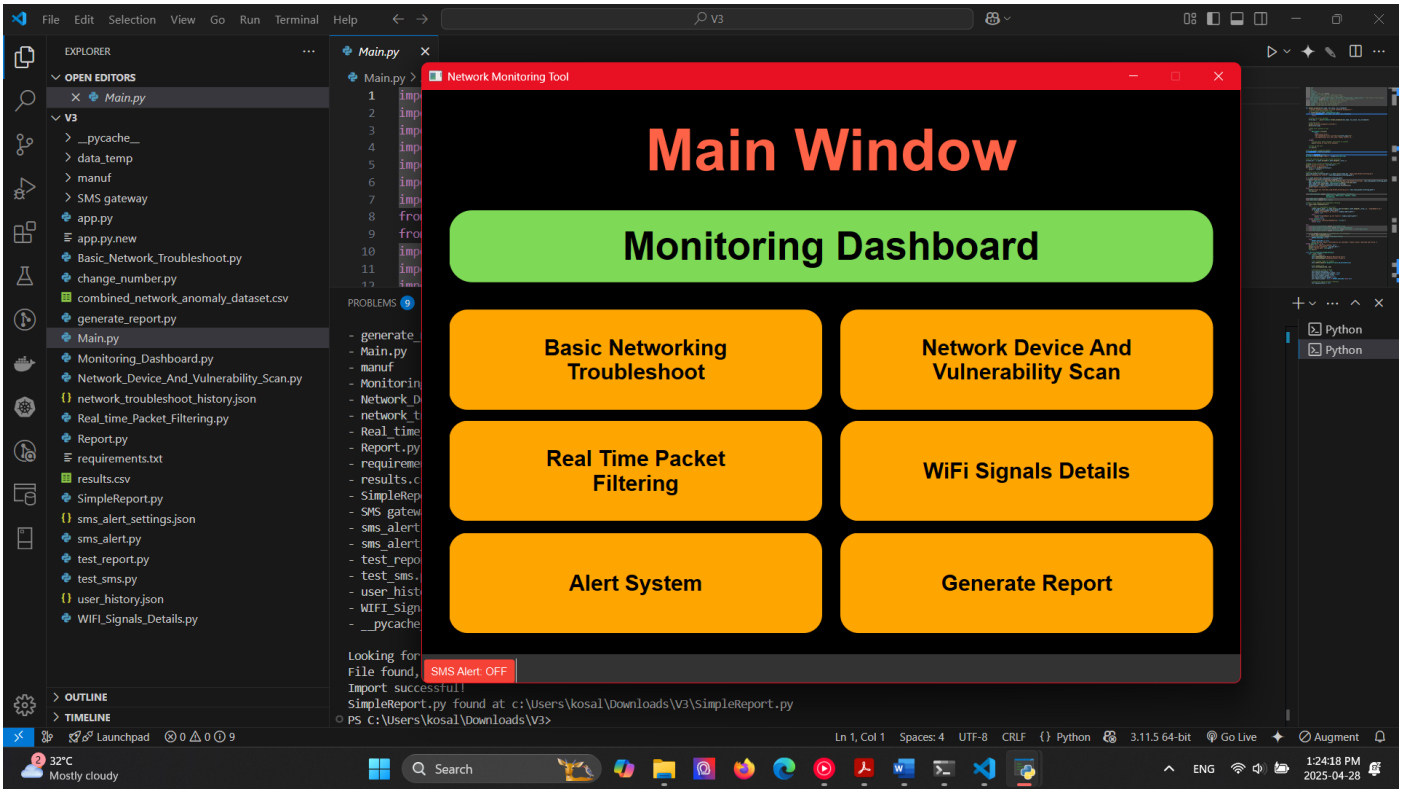


Figure 14 Main window Seconded image

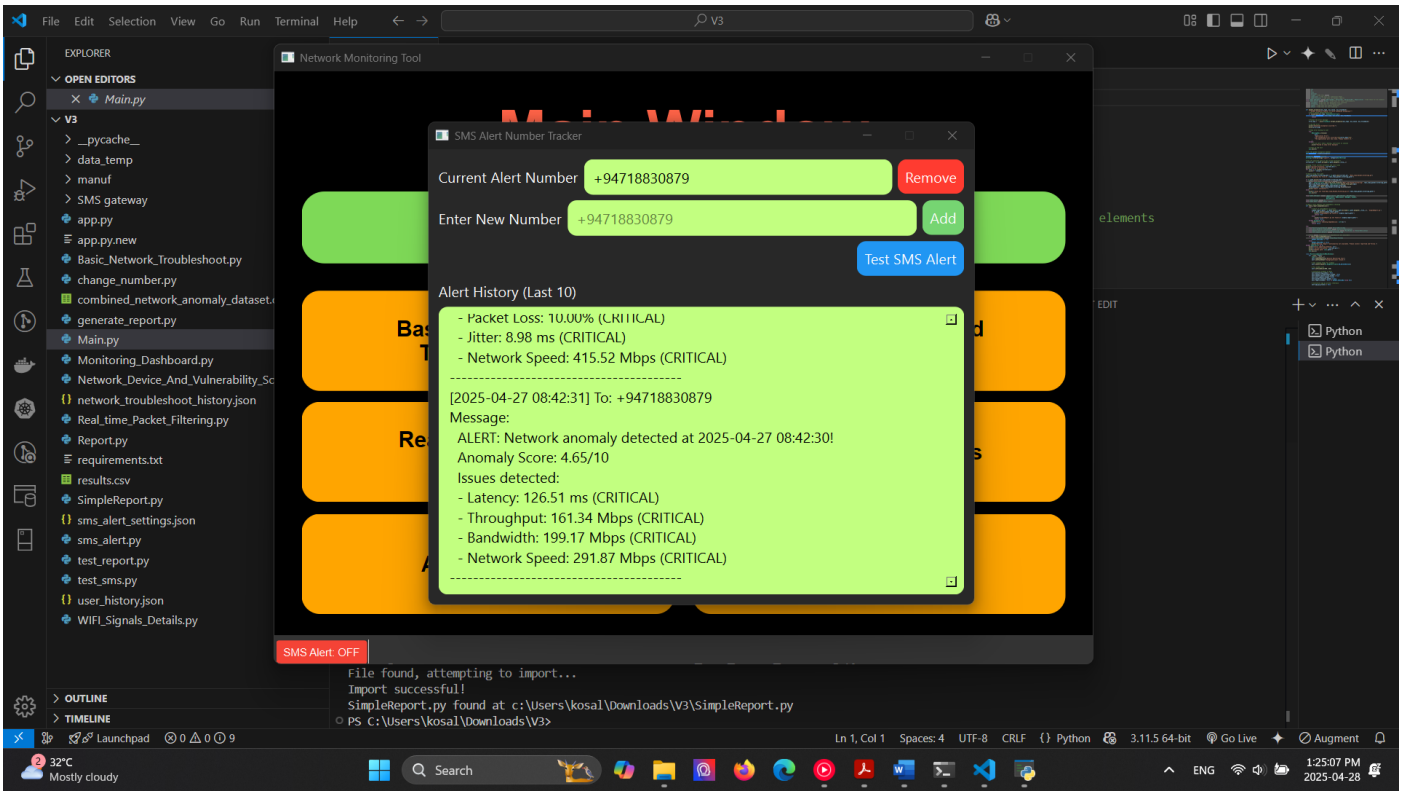


Figure 15 SMS Alert System History

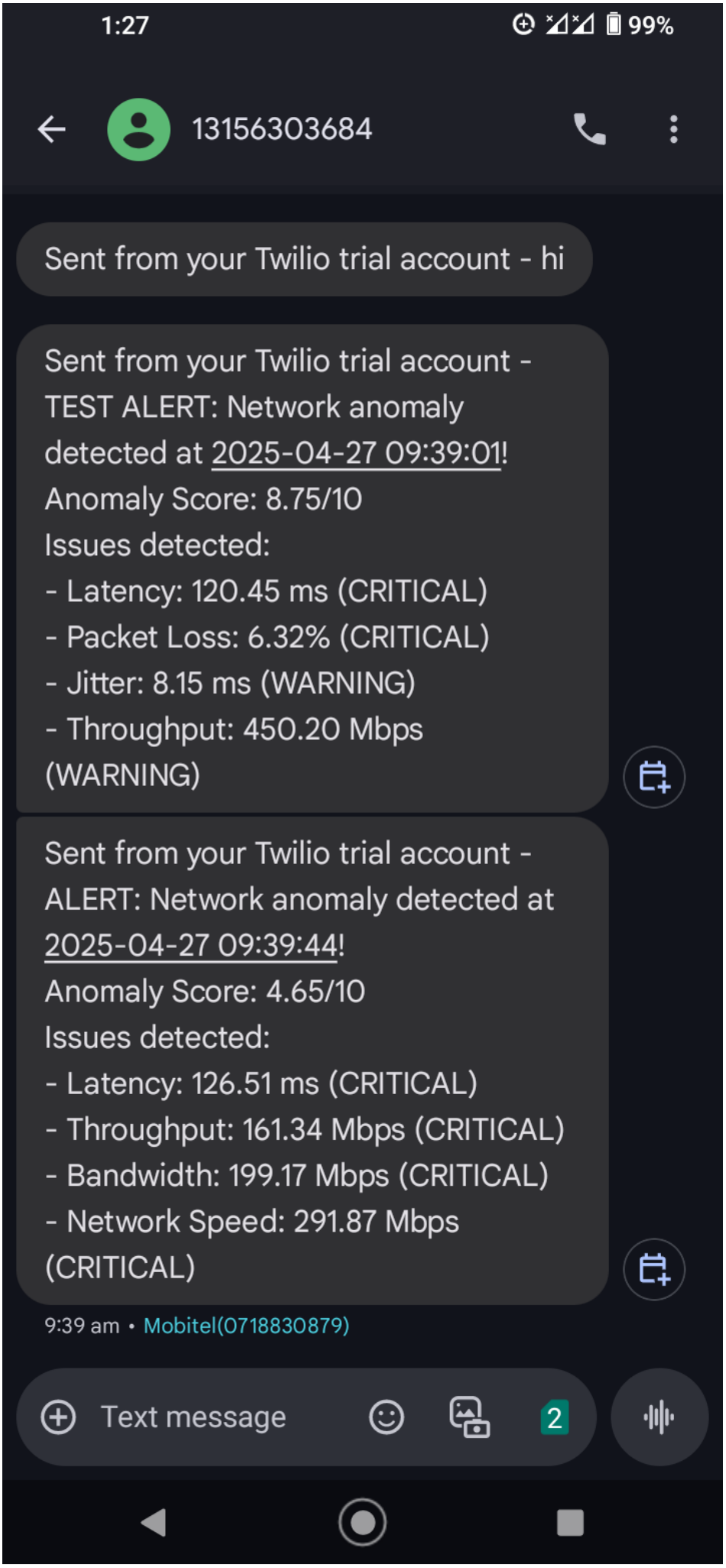


Figure 16 SMS Alert

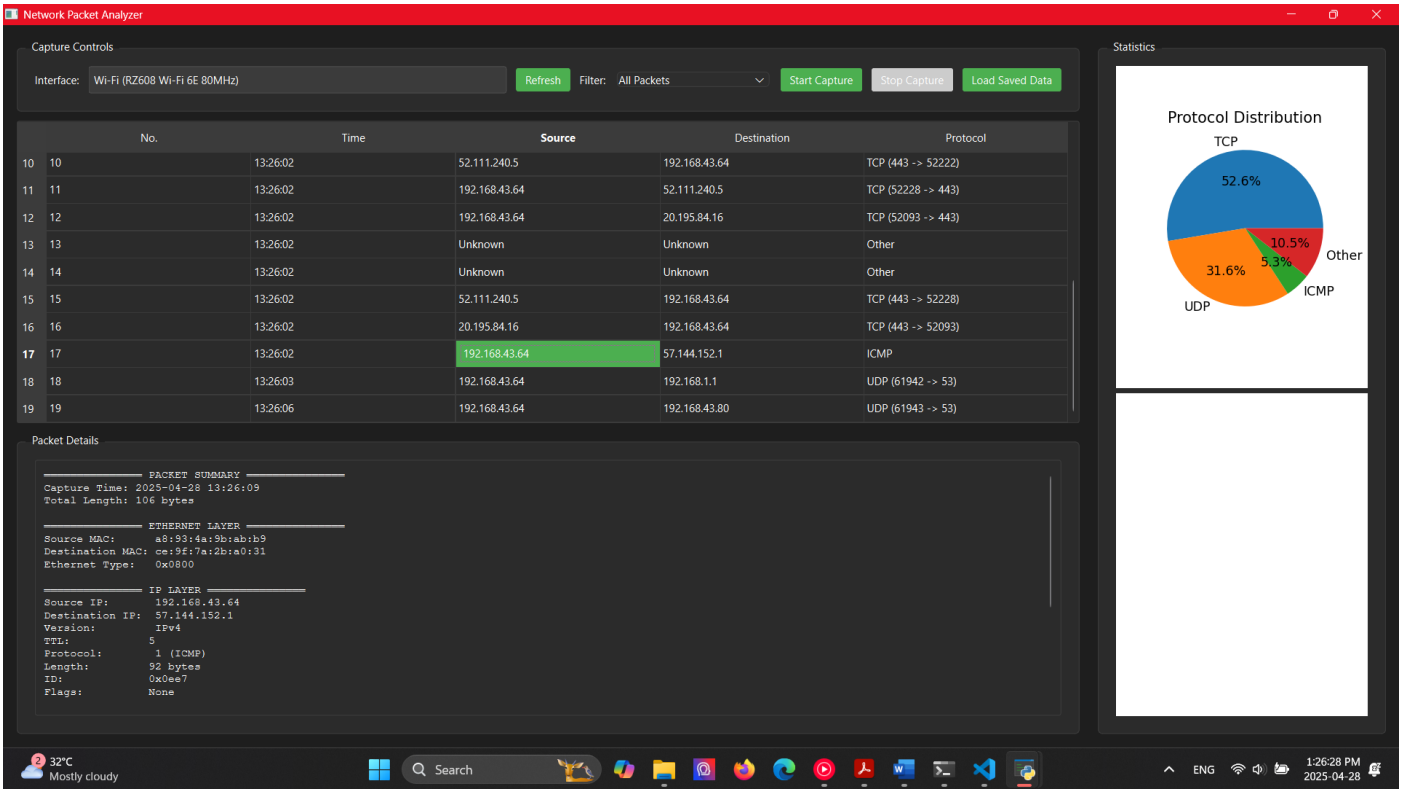


Figure 17 Network Packet Analyzer Active image

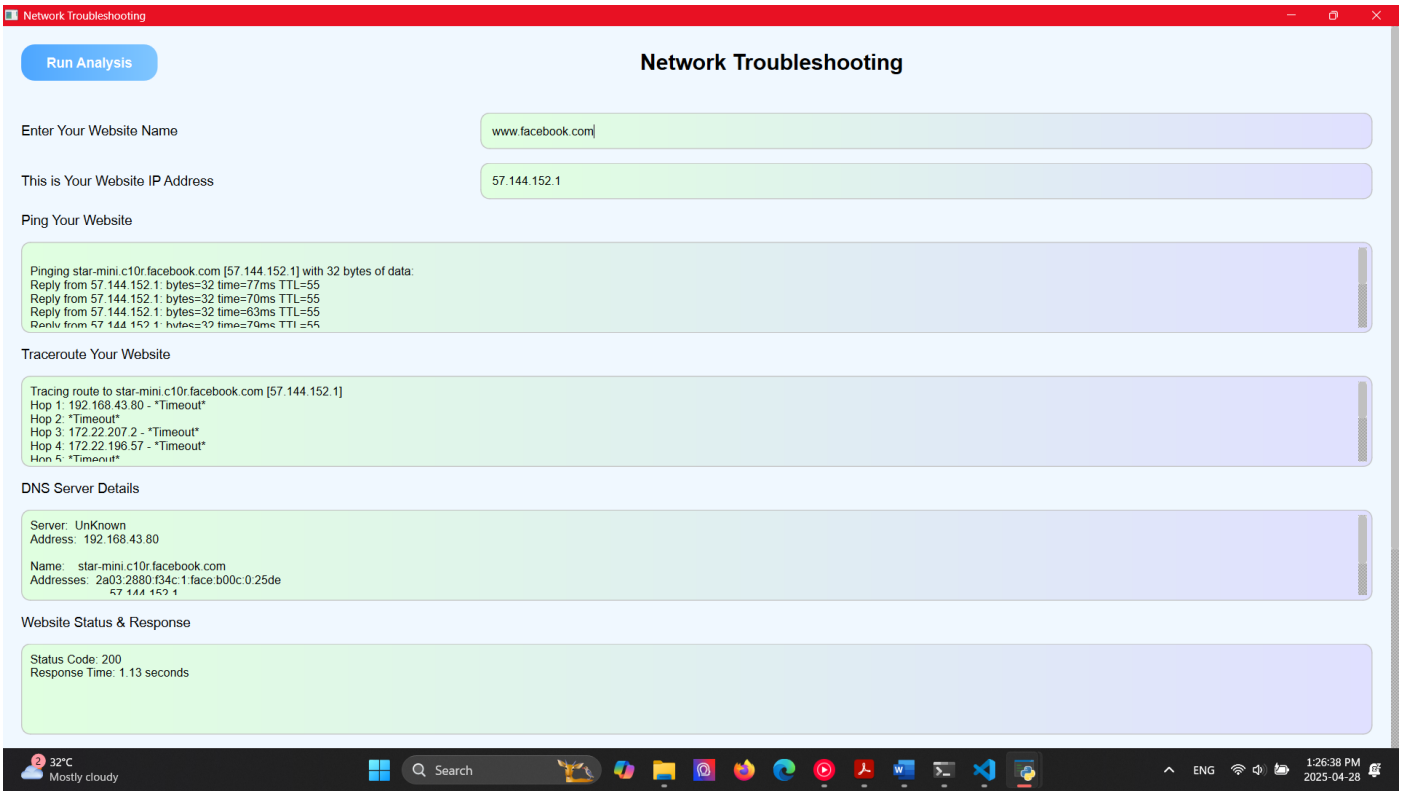


Figure 18 Networking Troubleshooting Active image

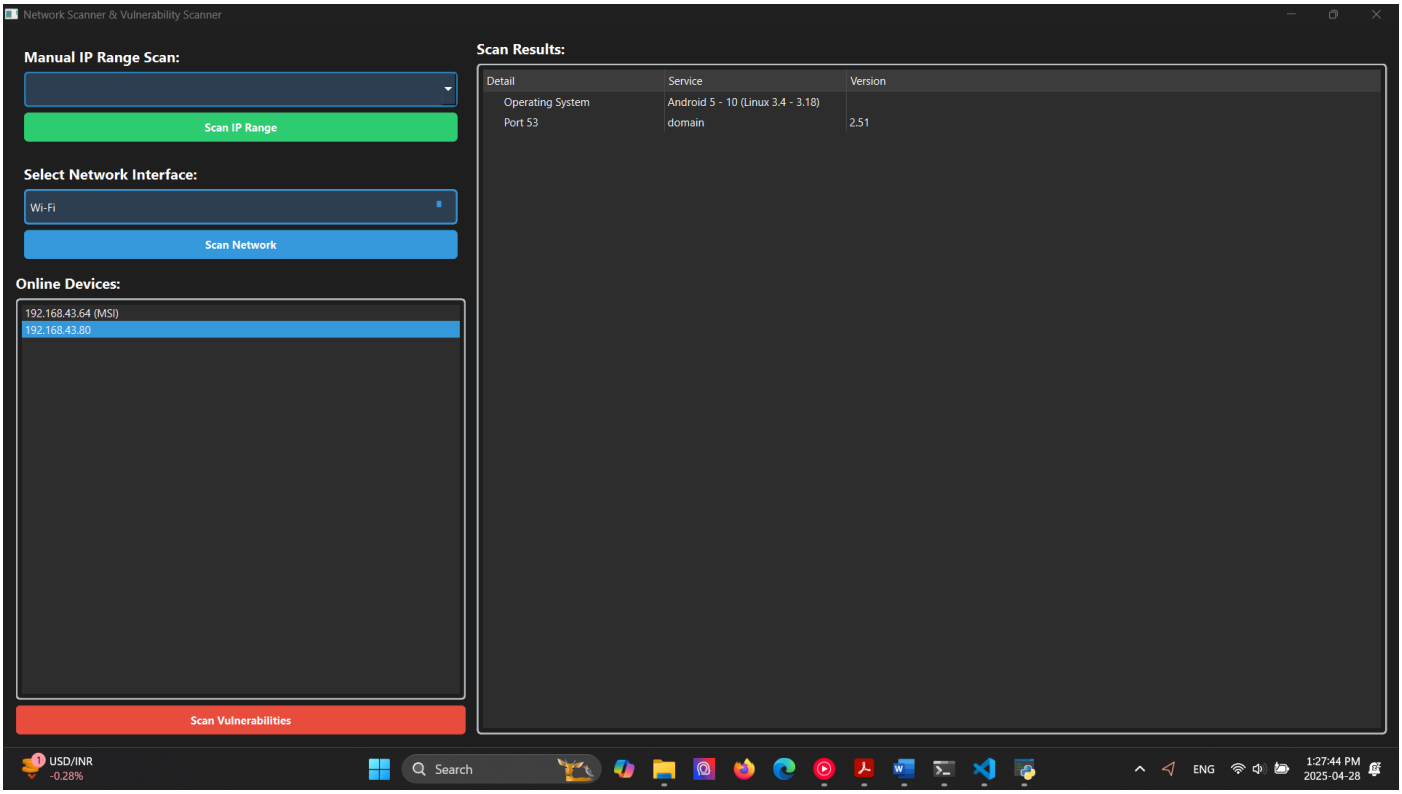


Figure 19 Port Scanner And Vulnerability Scanner Active image

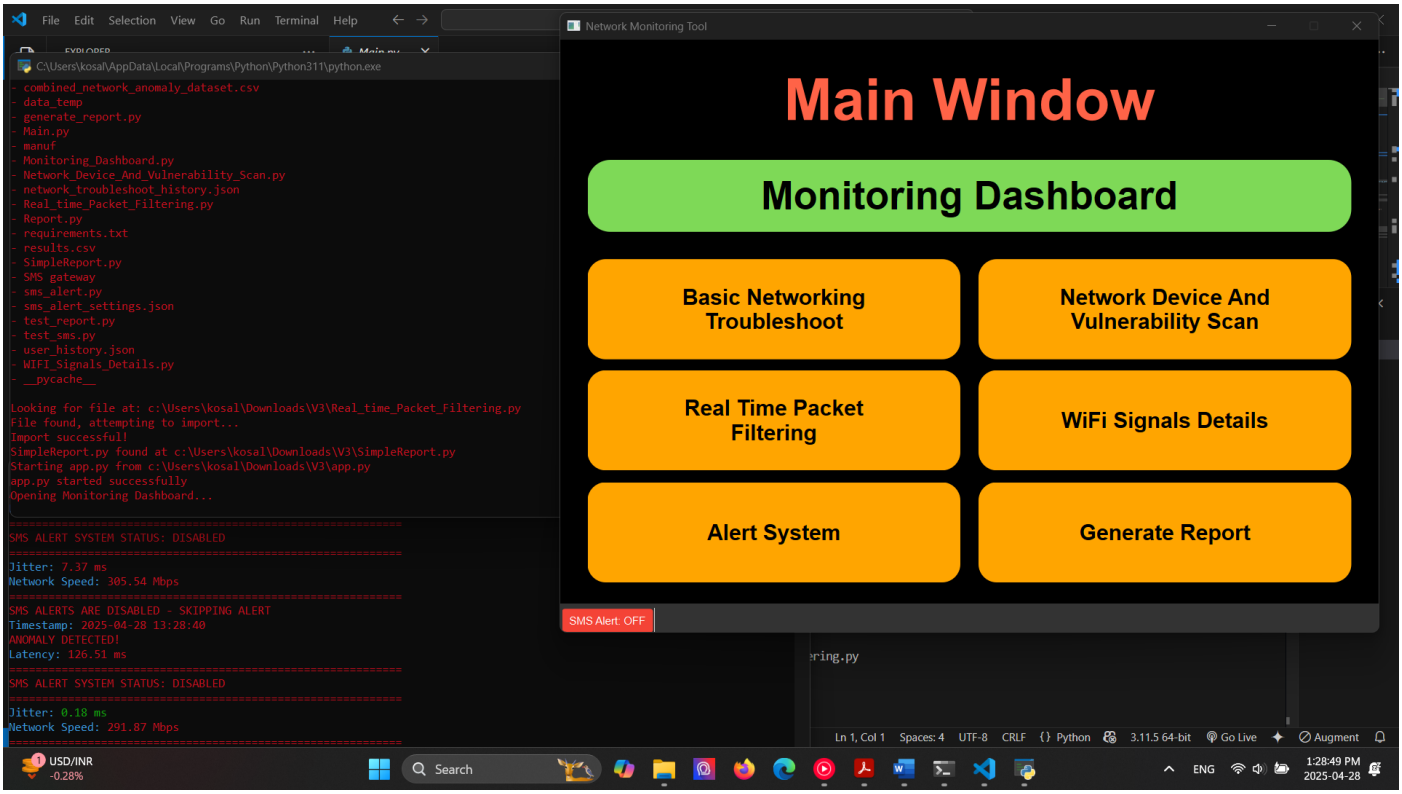
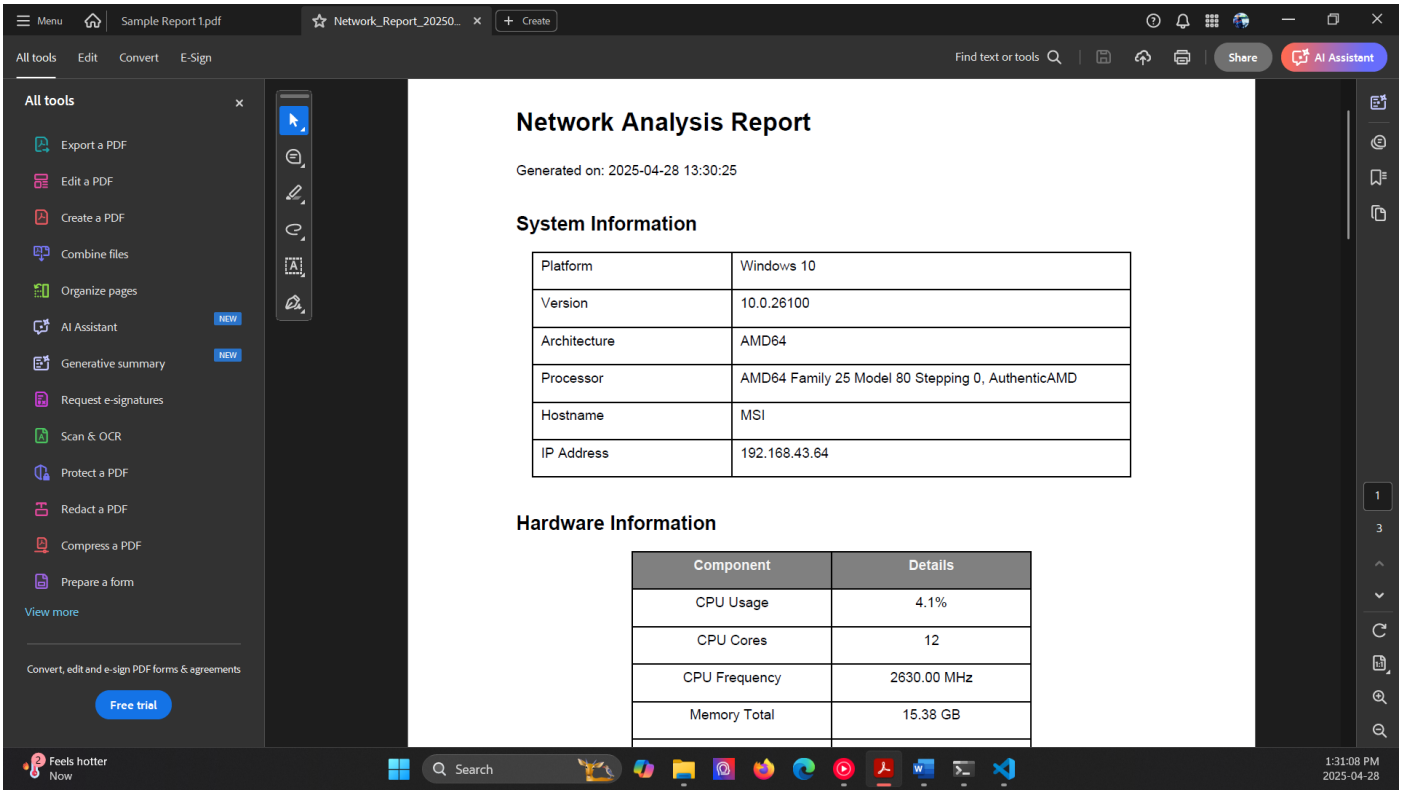


Figure 20 All Terminal and Dashboard images



Network Analysis Report

Generated on: 2025-04-28 13:30:25

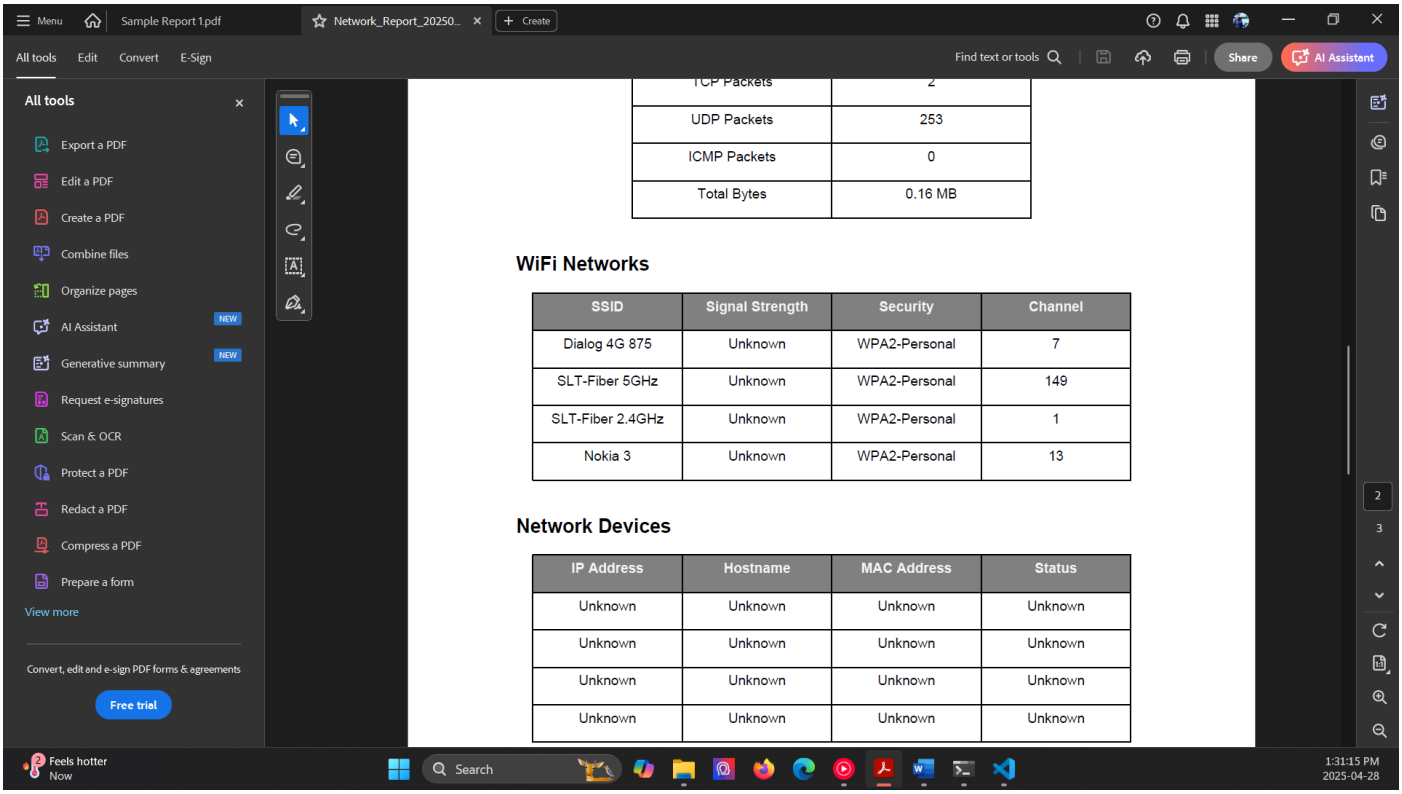
System Information

Platform	Windows 10
Version	10.0.26100
Architecture	AMD64
Processor	AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD
Hostname	MSI
IP Address	192.168.43.64

Hardware Information

Component	Details
CPU Usage	4.1%
CPU Cores	12
CPU Frequency	2630.00 MHz
Memory Total	15.38 GB

Figure 21 Detail Report Example 1



TCP Packets	2
UDP Packets	253
ICMP Packets	0
Total Bytes	0.16 MB

WiFi Networks

SSID	Signal Strength	Security	Channel
Dialog 4G 875	Unknown	WPA2-Personal	7
SLT-Fiber 5GHz	Unknown	WPA2-Personal	149
SLT-Fiber 2.4GHz	Unknown	WPA2-Personal	1
Nokia 3	Unknown	WPA2-Personal	13

Network Devices

IP Address	Hostname	MAC Address	Status
Unknown	Unknown	Unknown	Unknown
Unknown	Unknown	Unknown	Unknown
Unknown	Unknown	Unknown	Unknown
Unknown	Unknown	Unknown	Unknown

Figure 22 Detail Report Example 2

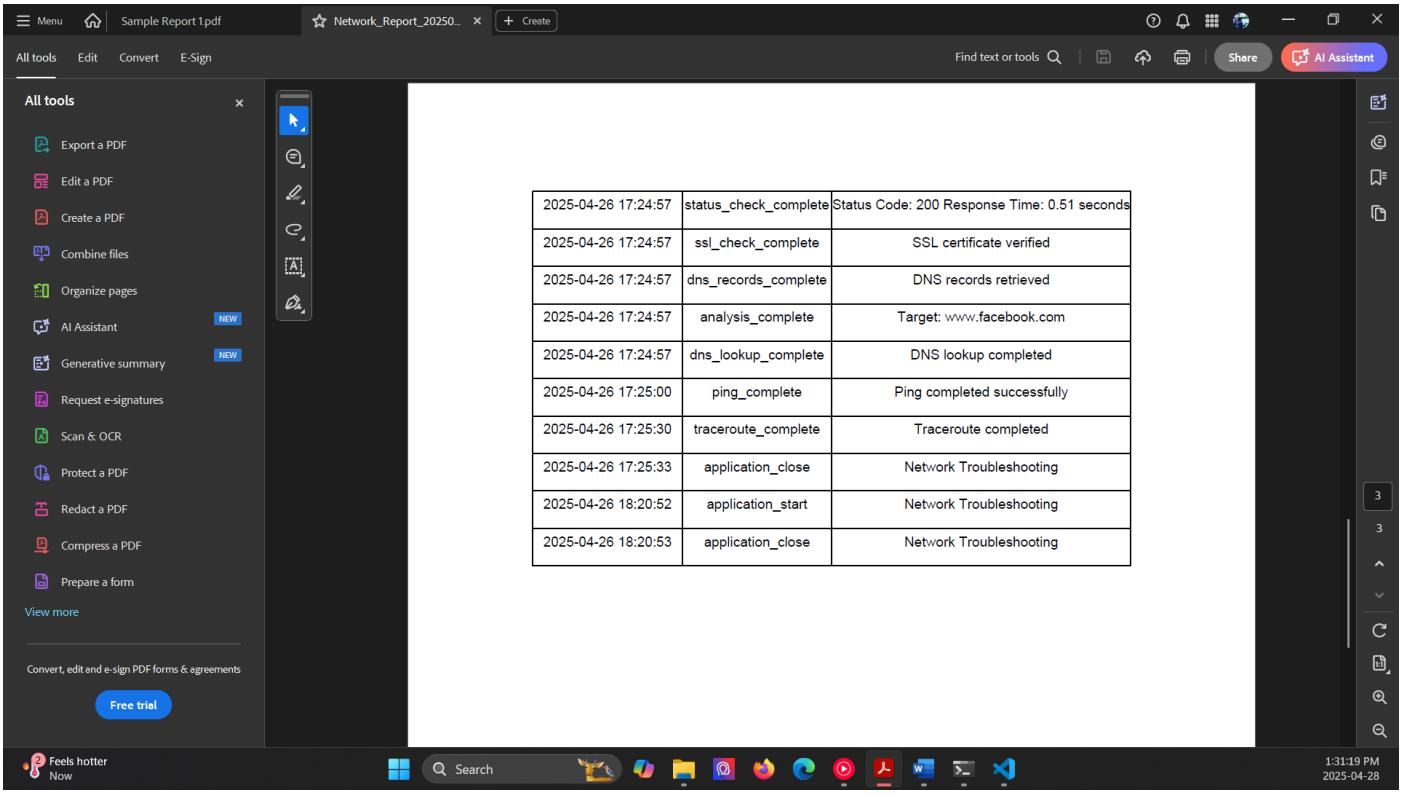


Figure 23 Detail Report Example 3

9. End-Project Report

The Root Course Analysis project was developed to help network administrators monitor and manage their networks using a simple and user-friendly desktop application. After months of planning, designing, coding, and testing, the project successfully reached its final stage with all core features completed and tested.

Project Summary

The final version of the application includes a real-time monitoring dashboard, packet filtering, device and vulnerability scanning, Wi-Fi signal analysis, SMS alert system, basic troubleshooting tools, and report generation. The system was built using Python and PyQt6, and it runs on Windows-based machines. All functions were tested in real network environments and showed reliable results.

Achievement of Objectives

Most of the short-term and long-term objectives were achieved during the project:

- Real-time packet monitoring and filtering were implemented successfully.
- The scanning module using python-Nmap accurately detected devices and open ports.
- SMS alerts were integrated using the Twilio API and triggered correctly during network anomalies.
- Reports were generated using Matplotlib and Pandas, showing useful graphs and statistics.
- The GUI was completed with a clean and easy-to-navigate layout.

These results showed that the system is functional, reliable, and helpful for daily network monitoring tasks.

Changes During the Project

Some changes were made during the development phase:

- The storage method was optimized by improving how JSON files were handled for better performance.
- GUI elements were adjusted to improve speed and prevent freezing when processing large data.
- Extra security features were added to avoid sending unwanted scan traffic.

These changes improved the quality and usability of the system.

Realization of Business Goals

The project successfully addressed business goals like:

- Reducing network monitoring costs by using open-source tools.
- Improving response time to network issues through automated alerts.
- Providing a tool that supports small and medium-sized businesses with limited IT resources.

The feedback from initial users showed that the application is practical and has real-world value.

Limitations

While the project achieved many goals, there are still some limitations:

- The system works best on Windows, but Linux support needs more testing.
- JSON storage is suitable for small to medium data but may not be ideal for large networks.
- Alerts depend on internet availability since they use an online API.

10.Project Post-Mortem

The Root Course Analysis project has now reached its completion. Looking back, it is important to reflect on what went well, what challenges were faced, and what lessons were learned throughout the process. This post-mortem provides a summary of the overall development experience.

Successes

- The project met its main goals by delivering a working network monitoring tool with real-time packet filtering, scanning, alerting, and reporting features.
- The user interface built with PyQt6 was smooth, easy to navigate, and visually clear.
- The application ran well on Windows with only basic system requirements.
- Integration with external tools like Twilio, Scapy, and python-nmap worked as planned.
- Reports generated using Matplotlib and Pandas gave clear visual insights into network performance.

These achievements show that the project was successful in terms of both functionality and usability.

Challenges Faced

- **Performance Issues**: Capturing live network packets using Scapy in busy networks sometimes caused the application to slow down. This was improved through optimization.
- **SMS Alert Integration**: Setting up Twilio was harder than expected due to API rate limits and internet dependency.
- **GUI Responsiveness**: Handling real-time data while keeping the interface responsive was challenging, especially when multiple features were active at once.
- **Storage Management**: Using JSON files for saving large data logs led to slower read/write times as data grew. This highlighted the future need for a lightweight database like SQLite.

Lessons Learned

- **Start Simple**: Beginning with a basic version of each feature and improving it step by step helped reduce errors and made testing easier.
- **Plan for Performance**: Real-time applications must be optimized early, especially when dealing with continuous data input like packet monitoring.
- **Importance of Modularity**: Dividing the system into small, separate modules helped with debugging, testing, and future upgrades.
- **User Feedback Matters**: Feedback from early testers helped shape the final GUI layout and improve feature usability.
- **Backup and Version Control**: Regular backups and the use of GitHub saved time and avoided the loss of progress during development.

Future Considerations

- Replace JSON storage with a more scalable database system.
- Expand full support for Linux and add compatibility for macOS.
- Add cloud monitoring and AI-based anomaly detection to increase system intelligence.
- Implement user authentication and settings management for better control and security.

11. Project Conclusions

The Root Course Analysis project was developed to support network administrators in monitoring, managing, and troubleshooting network performance in real time. After months of research, design, coding, and testing, the project successfully met its goals and delivered a fully functional desktop application.

The system includes useful features such as real-time packet monitoring, device and vulnerability scanning, Wi-Fi signal analysis, and an automated SMS alert system. It also provides clear reports and graphs that help users understand their network's behavior. By combining all these features into one platform, the tool simplifies daily tasks for network administrators.

The use of Python and open-source libraries made development cost-effective and flexible. PyQt6 helped in designing a user-friendly interface, while Scapy and Python-Nmap made network analysis accurate and detailed. The application runs well on standard Windows systems, making it suitable for small to medium-sized organizations with limited resources.

Although some challenges were faced, such as performance optimization and data storage limitations, they were handled successfully through testing and improvements. The experience also provided important lessons in project planning, problem-solving, and software design.

In conclusion, Root Course Analysis is a valuable tool with real-world benefits. With future updates and added features, it can become even more powerful and widely used in network management.

12. Reference List & Bibliography

Reference List

1. Python Software Foundation. *Python 3.8.20 Documentation*. Available at: https://docs.python.org/3.8/ *(Accessed: 22 April 2025).*
2. Biondi, P. *Scapy Documentation*. Available at: *https://scapy.readthedocs.io/ (Accessed: 22 April 2025).*
3. Zaytsev, M. *Python3-nmap Documentation*. Available at: https://nmap.readthedocs.io/ *(Accessed: 22 April 2025).*
4. Twilio Inc. *Programmable Messaging API Overview*. Available at: https://www.twilio.com/docs/messaging/api *(Accessed: 22 April 2025).*
5. Riverbank Computing Limited. *PyQt6 Reference Guide*. Available at: https://www.riverbankcomputing.com/static/Docs/PyQt6/ *(Accessed: 22 April 2025).*
6. Hunter, J. D. *Matplotlib 3.10.1 Documentation*. Available at: https://matplotlib.org/stable/index.html *(Accessed: 22 April 2025).*
7. The pandas development team. *pandas 2.2.3 Documentation*. Available at: https://pandas.pydata.org/docs/ *(Accessed: 22 April 2025).*
8. Rodrigues, G. *psutil 7.0.1 Documentation*. Available at: https://psutil.readthedocs.io/ *(Accessed: 22 April 2025).*
9. Reitz, K. *Requests: HTTP for Humans™ (v2.32.3) Documentation*. Available at: https://requests.readthedocs.io/ *(Accessed: 22 April 2025).*
10. dnspython Contributors. *dnspython 2.8.0 Documentation*. Available at: https://dnspython.readthedocs.io/ *(Accessed: 22 April 2025).*
11. netifaces Developers. *netifaces Documentation*. Available at: https://pypi.org/project/netifaces/ *(Accessed: 22 April 2025).*

Bibliography

- Real Python. “A Guide to Python’s Requests Library.” Available at: https://realpython.com/python-requests *(Accessed: 22 April 2025).*
- Bharath, R. “netifaces: Enumerating Network Interfaces in Python.” Python Network Programming. Available at: https://0xbharath.github.io/python-network-programming/libraries/netifaces/index.html *(Accessed: 22 April 2025).*

Appendices

13.User Guide

This user guide is designed to help network administrators and general users install, configure, and use the Root Course Analysis desktop application. The application runs on Windows and supports key functions like packet monitoring, device scanning, alert systems, and report generation. No advanced programming knowledge is required to use this system.

1. Installation Instructions

System Requirements:

- **Operating System**: Windows 10 or later
- **Python**: Version 3.8 or above
- **Internet Connection**: Required for Twilio SMS alerts
- **Administrator Access**: Needed for network scanning and packet capture

Steps to Install:

1. **Install Python**
   Download and install Python 3.8+ from https://www.python.org. During installation, ensure that the option "Add Python to PATH" is checked.
2. **Install Required Libraries**
   Open Command Prompt and run the following:
3. ***pip install pyqt6 scapy python-nmap twilio matplotlib pandas psutil requests dnspython netifaces***
4. **Download the Application**
   Download the project folder or executable installer from the provided link or GitHub repository (if available).
5. **Run the Application**
   If you downloaded the source code, navigate to the folder and run:
6. python main.py

If you received a pre-built .exe file, double-click to launch.

2. Using the Application

Main Dashboard

- Displays key network metrics such as latency, bandwidth, packet loss, and throughput.
- Graphs update in real time for easy analysis.

Packet Monitoring

- Captures live network packets.
- Shows protocol type (TCP/UDP/ICMP), IP addresses, ports, and traffic volume.
- Users can filter packets by protocol for easier tracking.

Device and Vulnerability Scanner

- Scans the local network to find connected devices.
- Shows IP, MAC, open ports, and operating system info.
- Identifies potential vulnerabilities using python-nmap.

Wi-Fi Signal Analysis

- Displays nearby wireless networks.
- Shows signal strength, network name (SSID), and security type.
- Useful for detecting weak signals or overlapping networks.

Troubleshooting Tools

- Includes Ping, Traceroute, and Speed Test utilities.
- Helps diagnose connectivity issues quickly.

Alert System

- Monitors the network for unusual activity.
- Sends SMS alerts via Twilio when critical issues are detected (e.g., unknown devices or high packet loss).
- Users can set custom alert thresholds.

Report Generation

- Creates visual performance reports using Matplotlib and Pandas.
- Includes graphs for latency, bandwidth, and device history.
- Reports can be exported in PDF or image formats (based on setup).

3. Configuration Settings

Setting Up Twilio SMS Alerts

1. Create a free Twilio account at https://www.twilio.com.
2. Get your **Account SID**, **Auth Token**, and **Phone Number**.
3. Enter these into the application's configuration file or settings window (if available).
4. Set your alert rules (e.g., packet loss > 10%, unknown device detected).

4. Troubleshooting Tips

- **App not launching?**
  Ensure Python is correctly installed, and all libraries are installed.
- **Packet monitoring not working?**
  Run the application with administrator rights. Some systems require admin access for packet capture.
- **SMS not sending?**
  Check your internet connection and make sure your Twilio credentials are correct.
- **Slow performance?**
  Try closing other applications or reduce scan frequency in settings.

5. Safety Notice

- Use the scanning tools responsibly, especially in shared or secured networks.
- Scanning without permission may violate network policies.

UNIVERSITY OF PLYMOUTH

PUSL3190 Computing Individual Project

Project Initiation Document (PID)

Network Fault Prediction System

Supervisor: MR. Chamara Disanayake

Name: Ileperumaachchige D Gunathilaka

Plymouth Index Number: 10898583

Degree Program: BSc (Hons) Computer Networks

Introduction

In today's interconnected world, the seamless operation of computer networks forms the backbone of modern businesses, institutions, and individuals alike. Networks enable critical communication, data exchange, and the provision of services across vast geographic regions, underscoring their essential role in driving productivity and innovation. However, the increasing complexity and scale of network infrastructures bring challenges that conventional management practices struggle to address effectively. Issues such as unpredictable downtimes, escalating cybersecurity threats, and inefficient resource utilization highlight the urgent need for transformative solutions in network management.

This project proposes the development of an advanced Network Fault Prediction System, a solution designed to overcome the limitations of traditional, reactive approaches. Traditional methods often rely on manual intervention and historical data analysis, making them inadequate for addressing the dynamic and evolving nature of modern networks. As organizations continue to embrace digital transformation and expand their network capabilities, there is an escalating demand for proactive, intelligent systems capable of predicting and mitigating potential issues before they escalate.

The inspiration for this project stems from the profound impact that network disruptions can have on operational continuity, customer satisfaction, and overall business performance. Network outages not only disrupt daily operations but also incur significant financial and reputational costs. Moreover, the growing prevalence of cyber threats poses additional risks to data integrity, privacy, and organisational security. Addressing these challenges requires a paradigm shift toward predictive and automated network management systems that enhance reliability, security, and efficiency. The Network Fault Prediction System aims to revolutionise network management by integrating advanced technologies such as artificial intelligence (AI), machine learning (ML), and big data analytics. By analysing real-time network data, including traffic patterns, device performance, and historical incident reports, the system can identify anomalies and trends indicative of potential faults. This capability empowers network administrators to take preemptive action, minimizing downtime and ensuring uninterrupted services. Furthermore, the system's predictive analytics extend beyond fault detection to optimize network configurations, improve resource allocation, and enhance overall performance.

A key feature of the proposed system is its robust security framework, designed to counteract the ever-evolving landscape of cyber threats. By incorporating intrusion detection, threat intelligence, and automated incident response mechanisms, the system ensures that networks remain resilient against unauthorized access, data breaches, and malicious attacks. This dual focus on performance optimization and security enhancement addresses the dual challenges faced by modern network administrators.

Scalability and adaptability are pivotal to the system's design. As network infrastructures expand and incorporate diverse devices and protocols, the system is engineered to accommodate these complexities without compromising performance. It supports virtualization and automation, enabling organizations to seamlessly integrate emerging technologies and scale their networks in alignment with growth objectives. This adaptability ensures the system's relevance in meeting future demands and technological advancements. The broader impact of this project extends to various sectors reliant on robust network operations, including healthcare, finance, education, and e-commerce. By delivering a unified platform for network monitoring and optimization, the proposed system reduces operational costs, enhances user experience, and fosters business continuity. Additionally, its user-friendly interface and intuitive design make it accessible to network administrators of varying technical expertise, promoting widespread adoption.

In conclusion, the Network Fault Prediction System is a visionary response to the pressing challenges of modern network management. By leveraging predictive analytics, advanced security measures, and intelligent automation, the project addresses critical pain points while offering a scalable, future-proof solution. This initiative not only aligns with current technological trends but also sets a precedent for innovation in network resilience and reliability. Through this project, organizations can achieve a proactive approach to network management, paving the way for enhanced operational efficiency and sustained growth.

Business Case

Business Need

In today's interconnected world, network reliability and security are fundamental to business operations. Modern organizations depend heavily on their network infrastructures to facilitate communication, manage data, and deliver services. However, the rapid growth in network complexity, coupled with the increasing scale of operations, has exposed vulnerabilities in traditional network management approaches. These legacy systems, which primarily rely on manual intervention and reactive measures, are often inadequate for addressing performance bottlenecks, minimizing downtime, or mitigating cyber threats in real-time.

The financial and operational implications of network disruptions are significant. Prolonged outages can result in lost revenue, decreased productivity, and eroded customer trust. Furthermore, the rise of advanced cyberattacks has amplified the risk to sensitive data and critical systems, posing severe legal and reputational challenges for businesses. With networks now forming the backbone of business continuity and innovation, a more proactive, intelligent, and scalable solution is imperative.

This project addresses the urgent need for a Network Fault Prediction System to transform network management. By integrating predictive analytics, advanced security measures, and automation, the system will empower businesses to anticipate and resolve network issues before they escalate. It aligns with the growing demand for efficiency, resilience, and cybersecurity in digital transformation strategies. Such a system ensures uninterrupted operations, safeguards sensitive information, and optimizes resource allocation, ultimately enhancing organizational performance and customer satisfaction. This initiative bridges the gap between current reactive methodologies and the proactive, data-driven strategies required to meet the demands of modern business environments.

Business Objectives

The primary objective of the Network Fault Prediction System project is to provide organizations with a cutting-edge solution that proactively addresses the challenges of modern network management. This project aims to achieve measurable improvements in operational efficiency, security, and business continuity through the following key objectives:

1. **Enhancing Network Reliability:**
   The system will reduce downtime by identifying and addressing potential network faults before they occur. By leveraging predictive analytics and real-time monitoring, organizations can minimize disruptions and maintain uninterrupted operations, ensuring business continuity and increased customer satisfaction.
2. **Strengthening Network Security:**
   The system will incorporate advanced security features such as intrusion detection, threat

intelligence, and automated incident response to mitigate cybersecurity risks. By proactively addressing potential vulnerabilities, the project will help safeguard sensitive data, protect business assets, and enhance organizational trust.

3. **Optimizing Resource Utilization:**
   Through intelligent automation and analytics, the system will enable organizations to optimize network configurations, improve bandwidth allocation, and reduce operational costs. This efficiency will translate into better performance and scalability, supporting growth and innovation.

4. **Improving Decision-Making with Insights:**
   The system will provide actionable insights into network performance and trends, allowing businesses to make informed decisions on capacity planning, investment priorities, and strategic improvements.

5. **Increasing User Accessibility:**
   By designing a user-friendly interface, the system will make advanced network management accessible to administrators with varying levels of technical expertise, promoting ease of use and widespread adoption

Project Objectives

The **Network Fault Prediction System** project seeks to address the critical challenges of network management by creating a unified, intelligent solution that leverages modern technologies such as artificial intelligence, machine learning, and big data analytics. The project is centred on delivering a scalable, secure, and proactive network management platform. The following objectives outline the key aims of the project:

1. **Develop a Proactive Fault Detection and Prediction System:**
   A core objective is to design a system that can identify potential network issues before they escalate into critical problems. By utilizing machine learning algorithms and predictive analytics, the system will analyze real-time network data, detect anomalies, and forecast faults. This proactive capability will help organizations minimize unplanned downtimes, optimize network performance, and ensure uninterrupted services.

2. **Enhance Network Security:**
   The project aims to integrate robust cybersecurity features into the system, including intrusion detection, threat intelligence, and automated mitigation measures. These features will proactively address vulnerabilities, ensuring the integrity, confidentiality, and availability of network systems. By combining fault prediction with advanced security, the system will protect against unauthorized access, malware, and data breaches.

3. **Facilitate Intelligent Automation:**
   To streamline operations and reduce the burden on network administrators, the system will incorporate automation for routine tasks such as network monitoring, configuration management, and fault resolution. Automation will improve operational efficiency, enhance accuracy, and free up human resources for strategic initiatives.

4. **Provide Real-Time Monitoring and Analytics:**
   The system will deliver real-time insights into key performance metrics such as bandwidth utilization, latency, and packet loss. This capability will empower network administrators to make informed decisions and implement timely adjustments to improve performance. Advanced analytics will also allow organizations to identify long-term trends, optimize resource allocation, and plan for future network expansions.

5. **Ensure Scalability and Flexibility:**
   Modern networks are dynamic and often grow in size and complexity. The system will be designed to adapt to varying network sizes, technologies, and architectures. It will support diverse devices, protocols, and virtualization environments, ensuring compatibility and seamless integration with existing infrastructures. This scalability will enable organizations to future-proof their network management systems against evolving demands.

6. **Focus on User-Friendly Design:**
   A key project objective is to create an intuitive, accessible interface that simplifies complex network management tasks. The system will cater to users with varying technical expertise, offering customizable dashboards, clear alerts, and straightforward navigation. By prioritizing user experience, the project ensures that the system is easy to adopt and widely usable.

7. **Enable Cost-Efficiency:**
   By reducing downtime, preventing faults, and optimizing resource usage, the system aims to reduce operational expenses. Automation and intelligent analytics will further contribute to cost-efficiency by minimizing manual interventions and ensuring optimal utilization of network resources.

8. **Support Data-Driven Decision Making:**
   The system will empower organizations to leverage actionable insights for better decision-making. From identifying bottlenecks to planning capacity upgrades, the project will provide tools for organizations to optimize their investments and strategies.

Literature Review

Modern network infrastructures' increasing complexity and scale demand a shift from traditional, reactive management methods to proactive, predictive systems that integrate advanced security measures, intelligent automation, and user-centred design. Current approaches, relying heavily on historical data and manual interventions, are insufficient in effectively addressing both network performance issues and emerging cyber threats, resulting in inefficiencies and potential vulnerabilities.*(D'Alconzo et al., 2019; Nouioua et al., 2021)*. While artificial intelligence (AI) and machine learning (ML) have shown promise for fault prediction, existing research has generally focused on isolated components, such as fault detection or security protocols, without creating an integrated framework capable of enhancing network resilience through a holistic, real-time approach.*(D'Alconzo et al., 2019; Marin-Quintero et al., 2022).*

Moreover, scalability and adaptability are essential for managing increasingly complex network environments, yet most systems lack real-time capabilities to dynamically adjust configurations and optimize performance in large-scale, multi-device networks. Big data methodologies, which offer the potential to improve fault prediction and security management through real-time analytics, have also been underutilized within integrated network management frameworks. *(Nouioua et al., 2021).* Additionally, current systems often overlook the importance of user experience, limiting accessibility for network administrators without extensive technical expertise, as they lack intuitive interfaces that provide actionable insights and simplify navigation.*(Nouioua et al., 2021).*

This proposed project aims to address these limitations by developing a unified, scalable, and user-friendly network management system that leverages AI-driven predictive analytics, advanced security protocols, and big data approaches, meeting the demands of modern organizational infrastructures and enhancing overall operational resilience.

Conceptual Diagram

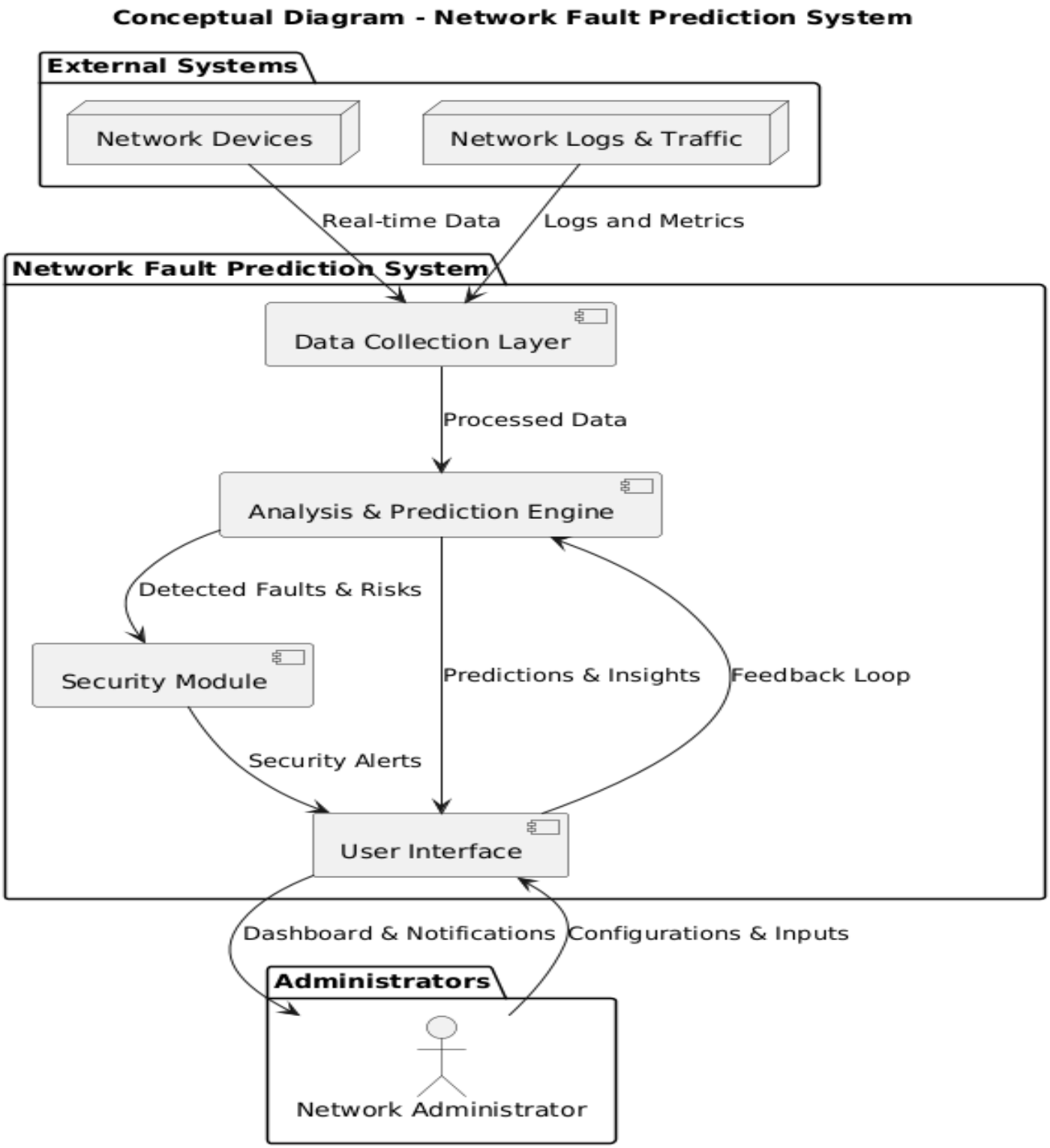


Figure 24 Conceptual Diagram

Method of Approach

The development of the **Network Fault Prediction System** will follow a structured and systematic methodology, incorporating advanced tools, technologies, and best practices in network management and software development. The approach is designed to ensure the system meets functional and non-functional requirements while adhering to industry standards and user needs.

1. Agile Development Framework

The project will adopt the Agile development methodology to promote iterative progress, flexibility, and collaboration. Agile will allow for regular evaluations, ensuring that feedback is incorporated at each stage of development. This iterative process will enable the identification and resolution of potential issues early, resulting in a robust and reliable final product.

2. Requirement Gathering and Analysis

The project will begin with an in-depth analysis of both functional and non-functional requirements. This phase will involve engaging stakeholders, including network administrators, security experts, and end-users, to understand their needs and expectations. Requirements such as real-time fault detection, predictive analytics, scalability, and security will be prioritized during this phase.

3. System Architecture Design

A high-level architectural design will be developed to ensure scalability, reliability, and efficiency. The system will consist of modular components, including:

- **Data Collection Layer**: Captures real-time network data from devices, traffic logs, and system metrics.
- **Analysis and Prediction Engine**: Utilizes machine learning algorithms to detect anomalies and predict faults.
- **Security Module**: Monitors for potential security breaches and implement automated responses.
- **User Interface**: Provides an intuitive dashboard for monitoring, alerts, and analytics.

4. Technology Selection

The project will leverage modern tools and technologies, including:

- **Machine Learning Platforms**: For predictive analytics and anomaly detection.
- **Big Data Frameworks**: To handle large volumes of real-time network data efficiently.
- **Cybersecurity Tools**: For threat detection and response.
- **Cloud-Based Services**: To ensure scalability and remote accessibility.

5. Development and Integration

The system will be developed in iterative cycles, with each cycle focusing on specific components. This approach ensures that individual modules are thoroughly tested and integrated seamlessly. Key tasks in this phase include:

- Implementing machine learning models for fault prediction.

- Developing automation scripts for configuration and incident management.
- Integrating advanced security protocols for threat detection.

6. Testing and Validation

Comprehensive testing will be conducted to validate the system’s functionality, reliability, and security. The testing process will include:

- **Unit Testing**: To verify individual components.
- **System Testing**: To ensure smooth integration.
- **Performance Testing**: To assess scalability and efficiency under varying loads.
- **Security Testing**: To identify and mitigate vulnerabilities.

7. Deployment and Feedback

The final system will be deployed in a controlled environment for user acceptance testing. Feedback will be collected to refine the system before full-scale deployment.

8. Documentation and Training

Comprehensive documentation, including user manuals and technical guides, will be prepared. Training sessions will be conducted to familiarize administrators with the system's functionalities and features.

By following this methodical approach, the project aims to deliver a reliable, secure, and user-friendly Network Fault Prediction System that meets organizational needs effectively.

High-Level Architectural Diagram

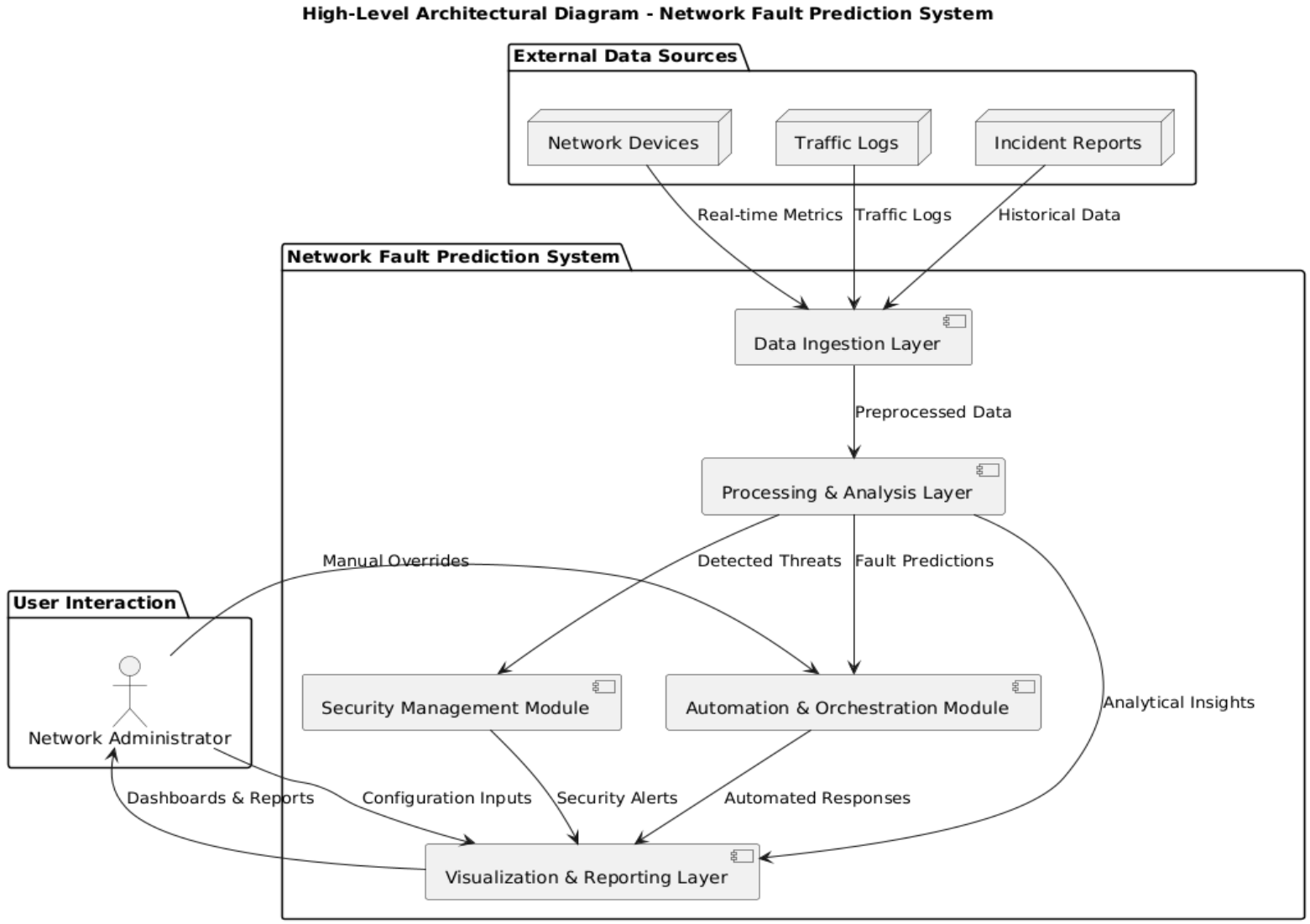


Figure 25 High-Level Architectural Diagram

Initial Project Plan

Figure 26 Initial Project Plan

Risk Analysis

Risk analysis is a critical aspect of ensuring the successful execution and implementation of the **Network Fault Prediction System**. By identifying potential risks, evaluating their impact, and devising mitigation strategies, this project aims to minimize disruptions and deliver a reliable solution. Below are the key risks and the corresponding measures to address them:

1. Technical Risks

- **Risk:** Failure of predictive analytics to accurately detect and predict faults.
- **Impact:** The system may fail to prevent network downtimes, reducing its effectiveness.
- **Mitigation:** Use diverse datasets during training to improve model accuracy and reliability. Employ robust validation techniques and iterative improvements to refine algorithms.
- **Risk:** Integration challenges with existing network infrastructure.
- **Impact:** Delays in deployment or incompatibility issues.
- **Mitigation:** Conduct detailed requirement analysis and compatibility assessment during the design phase. Perform pilot tests in controlled environments to address potential integration issues early.

2. Security Risks

- **Risk:** Vulnerabilities in the system itself are becoming a target for cyberattacks.
- **Impact:** Compromise of network data and system functionality.
- **Mitigation:** Implement strong security measures, including encryption, access controls, and regular security audits. Ensure the system adheres to cybersecurity best practices and standards.
- **Risk:** Unauthorized access or misuse of the predictive system.
- **Impact:** Potential breach of sensitive data or malicious configurations.
- **Mitigation:** Deploy multi-factor authentication and user role-based access controls to safeguard the system.

3. Resource Risks

- **Risk:** Insufficient expertise in machine learning or cybersecurity among the development team.
- **Impact:** Delays or suboptimal performance of the system.
- **Mitigation:** Engage domain experts or provide additional training to team members. Partner with external consultants when necessary.
- **Risk:** Limited availability of high-quality, real-time network data.
- **Impact:** Reduced effectiveness of predictive models.
- **Mitigation:** Collaborate with stakeholders to collect diverse, real-time datasets during the development phase. Use simulated data for testing and model refinement if needed.

4. Operational Risks

- **Risk:** Resistance from users in adopting the new system.
- **Impact:** Limited utilization and decreased ROI.

- **Mitigation:** Design an intuitive user interface and provide comprehensive training and support to users. Conduct awareness sessions to demonstrate the benefits of the system.
- **Risk:** Over-reliance on automation leading to overlooked manual processes.
- **Impact:** Reduced ability to address unanticipated network issues.
- **Mitigation:** Incorporate manual override options and regular audits of automated processes to ensure balanced reliance.

5. Project Management Risks

- **Risk:** Delays in project milestones due to unforeseen challenges.
- **Impact:** Prolonged development timeline and increased costs.
- **Mitigation:** Use Agile methodology to regularly evaluate progress and address issues incrementally. Include buffer periods in the project timeline for contingencies.
- **Risk:** Budget overruns due to underestimated costs or additional requirements.
- **Impact:** Financial strain on the project.
- **Mitigation:** Conduct thorough cost analysis during planning and establish a contingency fund for unexpected expenses.

References

- *D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., & Casas, P. (2019). A survey on big data for network traffic monitoring and analysis. IEEE Transactions on Network and Service Management, 16(3), 800–813. https://doi.org/10.1109/TNSM.2019.2933358*
- *Marin-Quintero, J., Orozco-Henao, C., Bretas, A. S., Velez, J. C., Herrada, A., Barranco-Carlos, A., & Percybrooks, W. S. (2022). Adaptive Fault Detection Based on Neural Networks and Multiple Sampling Points for Distribution Networks and Microgrids. Journal of Modern Power Systems and Clean Energy, 10(6), 1648–1657. https://doi.org/10.35833/MPCE.2021.000444*
- *Nouioua, M., Fournier-Viger, P., He, G., Nouioua, F., & Min, Z. (2021). A survey of machine learning for network fault management. In Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics: Theories and Applications (pp. 1–27). Springer International Publishing. https://doi.org/10.1007/978-3-030-66288-2_1*

15.Stage Plan

The Root Course Analysis project was completed in multiple stages, each focusing on specific tasks and goals. Breaking the work into stages helped in better time management, testing, and meeting project deadlines. Below is the detailed stage plan that outlines how the development process was structured from start to finish.

Stage 1: Project Planning and Research
Tasks:

- Identify the main goals of the project.
- Research existing network monitoring tools (e.g., Wireshark, Nmap, PRTG).
- Study the technologies to be used, like Python, PyQt6, Scapy, and Twilio.
- Finalize project scope and initial feature list.

Stage 2: System Design
Tasks:

- Design the overall architecture of the application.
- Create basic wireframes for the GUI using sketches or tools.
- Plan data flow and module connections (e.g., dashboard, scanning, alerts).
- Decide on data storage methods (JSON files for version 1).

Stage 3: Environment Setup
Tasks:

- Install Python and required libraries (PyQt6, Scapy, etc.).
- Set up a development environment using VS Code and GitHub.
- Test basic library functions to confirm compatibility with the system.

Stage 4: Module Development (Core Features)
Tasks:

- Develop the real-time packet monitoring module using Scapy.
- Build the device and port scanning feature using python-nmap.
- Add Wi-Fi signal detection module.
- Design and implement the dashboard to display live data.
- Test each module independently.

Stage 5: GUI Integration and Design
Tasks:

- Combine all modules into a single user interface using PyQt6.
- Add buttons, tables, and visual indicators for each function.
- Improve user navigation and layout for better usability.
- Handle user input, error messages, and basic settings.

Stage 6: Alert System and Twilio Integration
Tasks:

- Create logic to detect network anomalies (e.g., high packet loss, unknown IP).
- Configure Twilio API for sending SMS alerts.
- Test the system using different alert scenarios.
- Allow user customization for alert thresholds.

Stage 7: Report Generation and Visualization

Tasks:

- Generate graphs and summaries using Matplotlib and Pandas.
- Build the report module to display statistics (e.g., bandwidth usage, latency).
- Allow users to export reports (PDF or image format optional).
- Ensure data is properly collected and organized for reports.

Stage 8: Testing and Debugging

Tasks:

- Perform unit testing on all modules.
- Test the full application in different network environments.
- Check for performance issues, bugs, or crashes.
- Apply fixes and optimize the code.

Stage 9: Documentation and Final Report

Tasks:

- Write the user manual and technical documentation.
- Prepare screenshots and diagrams for the report.
- Finalize the written report following university guidelines.
- Submit the project files and documents for evaluation.

PUSL3190 Computing Individual Project

Interim Report (IR)

Root Cause Analysis

Supervisor: Mr. Chamara Disanayake

Name: Ileperumaachchige D Gunathilaka

Plymouth Index Number: 10898583

Degree Program: BSc (Hons) Computer Networks

01. Introduction

1.1 Introduction

In today's interconnected world, ensuring that networks operate smoothly is more crucial than ever. Businesses, institutions, and individuals depend on these networks for communication, data transfer, and everyday tasks. However, networks are not infallible; they can face faults, slowdowns, and security threats that disrupt their functionality. Traditional network monitoring systems often take a reactive stance, only discovering problems after they have happened, which often leads to expensive downtime and lengthy resolution periods. My project aims to change this paradigm by implementing an AI-powered root cause analysis system that excels at predicting and diagnosing network issues before they escalate. By collecting general network parameters like bandwidth, latency, jitter, data consumption, packet loss, and throughput, the system will train an AI model to keep an eye on network behavior. The AI model will subsequently offer suggestions on network performance monitoring, fault detection, troubleshooting, Quality of Service (QoS) management, security monitoring, network optimization, and traffic analysis. In addition to that, I have added a LAN network and Wi-Fi Analyzer so that users can get an overview of their network performance. As a computer network university student, I consider this project an opportunity to implement my knowledge of networking, artificial intelligence, and software development in solving real-life issues. I consider proactive network management essential in increasing efficiency, reducing downtime, and enhancing security. This project isn't just taking a tool and building it; it's working on a smarter way of dealing with problems and blocking problems before they arise.

1.2 Problem Definition

In today's digital age, network security and reliability are essential to organizations of all types. Corporations, governments, and individual users alike rely on robust and efficient networks to facilitate communication, transmit information, and support vital applications. Despite this, the management of networks is fraught with challenges, and one of the most important of these is fault detection latency. Most organizations continue to use the old-style network monitoring approaches, whereby administrators manually scan logs or only receive notifications once a problem has already affected users. By the time the issue is identified and fixed, users have already suffered slow connections, interruptions in services, or even security intrusions. This reactive strategy of network management is not efficient, since it does not avert faults but reacts to them after they have already taken place. The effects of tardy fault detection can be dire, ranging from excessive operational expenses to loss of productivity, and poor customer satisfaction. In fields like finance, healthcare, e-commerce, and telecommunications, network outages can lead to considerable financial losses and losses of reputation. Organizations that don't tackle network problems proactively stand to lose their competitive advantage and may be unable to satisfy growing demands for quick, secure, and stable connectivity.

The other key issue in network management is the absence of real-time analysis and automatic troubleshooting capabilities. Although most current network monitoring solutions offer simple network statistics, bandwidth utilization, latency, jitter, and packet loss, for instance—they do not delve deeper into analyzing these patterns or anticipating faults before they become critical. Thus, the administrators of information technology are usually presented with raw data without practical analysis, and it becomes hard for them to determine the causes of network problems as well as implement solutions effectively.

Manually troubleshooting and diagnosing network issues in very large networks with thousands of devices and multiple links is a complicated and time-consuming process. In the absence of an AI-based fault detection system, administrators must use trial-and-error techniques to determine the root cause of issues, causing unnecessary delays. Moreover, network congestion, configuration errors, and software/hardware faults may remain undetected until users complain of performance slowdowns, which further adds to the inefficiency of conventional methods. Automated troubleshooting and predictive analysis are imperatives for today's networks, but most organizations continue to use legacy approaches that do not incorporate self-

learning and real-time adaptability. An AI-powered network monitoring system can identify patterns, predict possible failures, and recommend remedial actions before an issue escalates. With machine learning models and smart data processing, network administrators have increased visibility into network health, performance trends, and potential vulnerabilities so that eventual faults are alleviated before they affect end users.

Apart from performance issues, network security is also an increasing worry for organizations. Cyberattacks, unauthorized access, and data breaches are becoming more common and advanced, presenting serious threats to sensitive business data, personal information, and financial transactions. Conventional security solutions, including firewalls and intrusion detection systems (IDS), tend to work reactively, identifying threats only after an attack has already started. One of the primary issues with traditional network security solutions is that they cannot continuously monitor network traffic in real time. Network attackers usually take advantage of network weaknesses by perpetrating slow, low-profile attacks that remain undetected until after they have caused extensive damage. Without an automatic system that can detect suspicious behavior, find anomalies, and initiate preventive measures, organizations are left exposed to data breaches, ransomware, distributed denial-of-service (DDoS) attacks, and insider threats. For instance, if a rogue device begins to use too much bandwidth or tries to access forbidden resources, a conventional monitoring system might not immediately list it as a security threat. An AI-powered network monitoring system, on the other hand, can immediately recognize abnormal behavior, match it with history, and create real-time alerts to avert possible security intrusions.

1.3 Project Objectives

To address these challenges, my project focuses on developing an AI-driven solution with the following objectives:

1. **Proactive Fault Detection** – Predict network issues before they happen using AI-based root cause analysis.
2. **Real-Time Performance Monitoring** – Continuously analyze key networking parameters and provide real-time insights.
3. **Automated Troubleshooting Suggestions** – Help administrators quickly identify and resolve network issues.
4. **Security Monitoring** – Detect and respond to potential security threats within the network.
5. **QoS and Network Optimization** – Improve bandwidth allocation, reduce packet loss, and optimize overall network performance.
6. **LAN & Wi-Fi Analysis** – Provide a detailed view of network traffic for both wired and wireless connections.

02: System Analysis

2.1 Facts Gathering Techniques

Before developing the system, I needed to understand how networks function, what kind of issues they face, and how administrators currently manage them. To do this, I used several fact-gathering techniques:

Literature Review

Modern network infrastructures' increasing complexity and scale demand a shift from traditional, reactive management methods to proactive, predictive systems that integrate advanced security measures, intelligent automation, and user-centered design. Current approaches, relying heavily on historical data and manual interventions, are insufficient in effectively addressing both network performance issues and emerging cyber threats, resulting in inefficiencies and potential vulnerabilities. *(D'Alconzo et al., 2019; Nouioua et al., 2021)*. While artificial intelligence (AI) and machine learning (ML) have shown promise for Root Cause Analysis, existing research has generally focused on isolated components, such as fault detection or security protocols, without creating an integrated framework capable of enhancing network resilience through a holistic, real-time approach. *(D'Alconzo et al., 2019; Marin-Quintero et al., 2022).*
Moreover, scalability and adaptability are essential for managing increasingly complex network environments, yet most systems lack real-time capabilities to dynamically adjust configurations and optimize performance in large-scale, multi-device networks. Big data methodologies, which offer the potential to improve Root Cause Analysis and security management through real-time analytics, have also been underutilized within integrated network management frameworks. *(Nouioua et al., 2021).*
Additionally, current systems often overlook the importance of user experience, limiting accessibility for network administrators without extensive technical expertise, as they lack intuitive interfaces that provide actionable insights and simplify navigation. *(Nouioua et al., 2021)*.

Interviews and Discussions

I spoke with network administrators and IT professionals to learn about the common challenges they face. Their insights helped me design a system that directly addresses real-world problems.

Online Research and Case Studies

I explored case studies of companies that use AI for network monitoring. This helped me identify the best practices and gaps in existing solutions.

Practical Testing and Observations

I analyzed network traffic, performance metrics, and faults in test environments. By running different network scenarios, I observed how various factors like bandwidth usage, latency, and security threats affect overall performance.

2.2 Existing System

In most organizations now, network fault detection and monitoring systems are largely manual and reactive in debugging. Although these kinds of systems give feedback on the status of the network, they do not proactively move beyond issue detection and prevention of problems prior to causing disruptions. As

networks continue growing and also becoming more complex, the shortcomings of conventional monitoring tools are becoming more evident, thus contributing to longer resolution times, rising operational expenses, and greater exposure to security breaches.

Administrators usually employ a mix of simple diagnostic tools and enterprise network management tools to determine network health. Ping tests, traceroutes, and SNMP monitoring are examples of tools that enable administrators to verify connectivity and gauge fundamental performance metrics such as latency and packet loss. Log analysis is also prevalent, in which administrators manually review system logs to search for anomalous patterns or errors. But these techniques are time-consuming, rearward-facing, and narrow in focus because they identify problems only after they have already affected network performance or end-user experience.

Most organizations utilize off-the-shelf network monitoring software, including Nagios, PRTG, and SolarWinds, to automate parts of network monitoring. Such software may offer real-time status, traffic visualization, and alerts when pre-configured thresholds are reached. For instance, they can notify administrators when bandwidth utilization surges or when a device becomes unavailable. Though handy for real-time problem detection, such systems are basically threshold-based and do not have the capability to predict faults or perform root cause analysis. Administrators therefore still must proceed with manual investigation of the problem cause, resulting in tardy response times along with prolonged downtime.

Security Monitoring as a Separate Process

Many systems handle network performance and security monitoring separately, even though they are closely connected. Companies often use firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), and antivirus software to protect their networks from threats. These tools watch for known attack signs, unusual traffic, and unauthorized access, sending alerts when they spot potential issues. However, traditional security tools generally react only after a threat has breached the system or caused damage. For instance, an IDS might detect strange data movement that hints at a data breach, but by the time the alert is issued, sensitive information might already be compromised. Additionally, methods that depend on known attack signs have difficulty spotting new, unknown threats since they only recognize threats that fit specific rules. This separate approach to monitoring network performance and security makes it challenging to connect performance problems with security threats. For example, a sudden spike in bandwidth usage could be due to legitimate traffic or a Distributed Denial of Service (DDoS) attack. Without AI to help identify correlations, administrators might not determine the cause until after a thorough investigation.

Limited Performance Metrics and Lack of Predictive Insights

Most existing network monitoring tools focus on monitoring individual performance metrics, such as:

- **Bandwidth usage** — how much data is being transmitted across the network.
- **Latency** — the delay between sending and receiving data packets.
- **Packet loss** — the percentage of packets that fail to reach their destination.
- **Jitter** — variations in packet arrival times, which can degrade real-time applications like VoIP.

These tools assess the network's current health but don't uncover deeper problems. For instance, if packet loss rises, a tool might send an alert but won't explain the cause. Network managers must dig through logs, check historical data, and conduct tests to determine if the issue stems from too much traffic, broken hardware, configuration errors, or a security threat. Traditional systems are missing AI tools that predict issues. Without machines learning to examine past patterns and detect anomalies, monitoring tools can't predict failures or suggest ways to prevent them. This means companies often end up resolving problems only after they've happened rather than preventing them before they impact users.

The Burden of Manual Troubleshooting

The reliance on manual troubleshooting is another major drawback of existing systems. When an issue arises, administrators often need to:

1. Analyze system logs to search for error messages or unusual events.
2. Run diagnostic tests (like ping or traceroute) to check device connectivity.
3. Inspect network configurations to identify potential misconfigurations or conflicts.
4. Test individual components (like routers, switches, or access points) to rule out hardware failures.

2.3 Use Case Diagram

The following case diagram represents how different users interact with the system.

Main Users:

1. Network Administrators – Monitor performance, analyze reports, receive alerts, and manage network settings.
2. System Users (Employees/Clients) – Check network status, receive notifications on issues, and report connectivity problems.
3. AI Model – Analyzes network parameters, predicts faults, and suggests solutions.

Main Use Cases:

- Monitor real-time network performance
- Detect and predict network faults
- Provide root cause analysis
- Optimize bandwidth and resource allocation
- Detect security threats and anomalies
- Generate reports and suggestions

Case Diagram

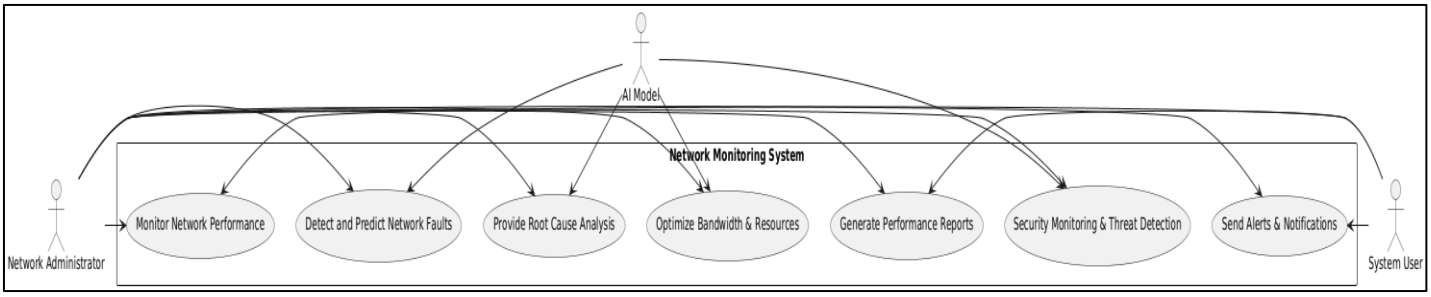


Figure 27 Case Diagram

2.4 Drawbacks of the Existing System

While existing network monitoring tools help detect problems, they have several limitations:

1. Reactive, Not Proactive – Most tools only identify issues after they occur, instead of predicting them in advance. This leads to delays in troubleshooting and potential downtime.
2. Lack of Root Cause Analysis – Current systems show alerts when there is a problem, but they do not provide detailed explanations or suggestions on why the issue occurred and how to fix it.
3. Manual Troubleshooting – Network administrators often must manually check logs, analyze patterns, and try different solutions, which is time-consuming and inefficient.
4. Limited Security Insights – Many monitoring tools do not integrate AI-driven security analysis, leaving networks vulnerable to cyber threats and unauthorized access.
5. No Unified View – Administrators often use multiple separate tools for different tasks (monitoring, security, troubleshooting), making it difficult to get a comprehensive view of network health.
6. Not Scalable for Large Networks – Traditional systems struggle to handle large amounts of network data efficiently, making them less effective for complex or expanding networks.

03: Requirements Specification

3.1 Functional Requirements

The system is designed to proactively monitor, analyze, and optimize network performance while also identifying potential faults and security threats. Below are the core functional requirements:

1. Real-Time Network Monitoring

- The system should continuously track key network parameters such as bandwidth usage, latency, jitter, packet loss, throughput, and data usage.
- It should collect data from both wired (LAN) and wireless (Wi-Fi) networks to provide a complete overview.

2. AI-Based Fault Detection and Prediction

- The AI model should analyze collected data and detect potential network faults before they happen.
- The system should identify the root causes of problems rather than just showing alerts.
- It should provide suggestions for troubleshooting, helping network administrators fix issues efficiently.

3. Security Monitoring and Threat Detection

- The system should monitor unusual activity that may indicate cyber threats, such as unauthorized access, network intrusions, or DDoS attacks.
- It should alert administrators to potential security risks and recommend preventive actions.

4. Quality of Service (QoS) Management

- The system should optimize bandwidth allocation to ensure a smooth experience for critical applications.
- It should prioritize essential network traffic, reducing the impact of congestion and bottlenecks.

5. Automated Reports and Insights

- The system should generate real-time and periodic reports summarizing network health, performance trends, and security risks.
- Reports should be visualized using charts and graphs for better understanding.

6. User-Friendly Dashboard

- The system should have a simple graphical interface that allows administrators to:
  - View network performance in real-time.
  - Access AI-generated suggestions for troubleshooting and optimization.
  - Adjust settings and configurations easily.

7. Alerts and Notifications

- The system should send instant notifications (via email, SMS, or system alerts) when:
  - A critical network issue is detected.
  - Performance drops below a defined threshold.
  - A security threat is identified.

3.2 Non-functional Requirements

While functional requirements define what the system does, non-functional requirements ensure that the system performs efficiently and reliably.

1. Scalability

- The system should support small to large-scale networks, adapting to increased data traffic and devices without performance issues.

2. Reliability and Availability

- The system should operate 24/7 without downtime and recover quickly from failures.
- Redundant data storage should ensure that no important logs or records are lost.

3. Security

- The system must protect sensitive network data using encryption and access controls.
- Only authorized users should be able to modify settings or access critical reports.

4. Performance Efficiency

- The AI model should analyze data in real-time without causing delays.
- The system should not consume excessive CPU, memory, or bandwidth, ensuring minimal impact on the network.

5. Maintainability and Upgradability

- The system should be modular, allowing future updates or improvements without major modifications.
- It should support software patches and security updates without requiring a complete reinstall.

6. Usability

- The interface should be user-friendly and intuitive, making it easy for both technical and non-technical users to operate.

- Clear visualizations and simple explanations should be provided for AI-generated insights.

These non-functional requirements ensure that the system remains efficient, secure, and scalable, making it suitable for real-world use.

3.3 Hardware / Software Requirements

The system needs specific hardware and software components for smooth operation.

Hardware Requirements

Component	Minimum Specification	Recommended Specification
Processor (CPU)	Intel Core i3 (or equivalent)	Intel Core i5 or higher
RAM	8GB	16GB or more
Storage	250GB SSD	500GB SSD or higher
Network Adapter	Standard Ethernet & Wi-Fi card	Gigabit Ethernet & Advanced Wi-Fi Adapter
Graphics Card	Integrated Graphics	Dedicated GPU for advanced visualization

Software Requirements

Software	Purpose
Operating System	Windows, Linux, or macOS
Programming Language	PyQT 6 (front-end development) and Python (for AI and back-end development)
Database	MySQL, PostgreSQL, or MongoDB (for storing network logs)
Network Monitoring Tools	Scapy, Wireshark, SNMP utilities
Visualization Tools	Matplotlib

These hardware and software components will ensure smooth data collection, analysis, and visualization for network monitoring.

3.4 Networking Requirements (Optional)

For real-time monitoring, the system requires a proper network setup with the following:

1. Supported Network Types

LAN (Local Area Network) – Ethernet-based, wired networks.

Wi-Fi (Wireless Networks) – Supports 2.4 GHz and 5 GHz frequency bands.

2. Required Network Protocols

SNMP (Simple Network Management Protocol) – For collecting device status and performance data.

ICMP (Internet Control Message Protocol) – For checking connectivity (ping, traceroute).

TCP/IP and UDP – For analyzing traffic patterns and detecting anomalies.

3. Data Collection Sources

Routers & Switches – To measure packet flow and latency.

Access Points – To monitor wireless performance.

End-User Devices – To check client-side connectivity issues.

04: Feasibility Study

Before starting the development of any system, it is important to check whether the project is practical, achievable, and beneficial. This feasibility study examines whether my AI-powered network monitoring and root cause analysis system can be successfully developed and implemented.

4.1 Operational Feasibility

Operational feasibility focuses on whether the system will be useful and accepted by its users. The main users of this system are network administrators, IT teams, and business organizations that need an efficient way to monitor and manage network performance.

Key Factors Ensuring Operational Feasibility:

- Addresses Real Problems – The system solves common network issues like fault detection, performance optimization, and security monitoring, making it highly valuable for organizations.
- User-friendly interface – A simple dashboard with visual reports, alerts, and AI-generated suggestions ensures that users can easily understand and operate the system.
- Reduces Manual Work – Unlike traditional monitoring tools, which require constant manual analysis, this system automates fault detection and troubleshooting recommendations, saving time and effort.
- Improves Network Efficiency – By optimizing bandwidth and preventing performance drops, the system helps businesses reduce downtime and enhance productivity.
- Security Enhancement – The system can detect security threats like unauthorized access, unusual traffic spikes, and potential cyberattacks, ensuring a safer network environment.

Potential Challenges & Solutions:

Challenge	**Solution**
Users may be unfamiliar with AI-driven net work monitoring.	Provide training sessions and user manuals.
Organizations may already use existing monitoring tools.	It allows integration with third-party tools like Wireshark and SNMP-based monitoring.
Initial setup may require technical expertise.	It offers easy-to-follow installation guides and automated setup wizards.

Since this system provides better efficiency, automation, and security, it is highly operationally feasible.

4.2 Technical Feasibility

Technical feasibility examines whether the required hardware, software, and technical expertise are available to successfully develop and implement the system.

Technology Readiness:

- Programming & AI Tools Available – The system will be built using Python, TensorFlow, Scikit-learn, and Matplotlib, which are widely used for AI and data visualization.
- Compatible with Existing Networks – The system uses standard network protocols (SNMP, ICMP, TCP/IP) and supports both LAN and Wi-Fi networks, ensuring smooth integration with the current infrastructure.
- Real-Time Data Processing Possible – With efficient database management (MySQL, MongoDB) and big data handling capabilities, the system can process large amounts of network data in real time.
- Scalable Architecture – The system is designed to work on small to large-scale networks, ensuring future growth without performance issues.

Potential Technical Challenges & Solutions:

Challenge	**Solution**
Handling large network traffic data efficiently.	Use optimized database queries and caching mechanisms to improve performance.
Ensuring accurate AI-based fault detection.	Train AI models with large and diverse datasets for better accuracy.
Integration with different network environments.	Design the system to support multiple operating systems and devices.

Since all necessary technologies and expertise are available, the project is technically feasible.

4.3 Outline Budget

Developing and deploying this system requires hardware, software, and operational costs. Below is a rough estimate of the project budget:

Estimated Budget Breakdown:

Category	**Estimated Cost (USD)**
Hardware Components	
Development Laptop / Server	200000 LKR
Network Testing Equipment (Routers, Switches, Access Points)	147,775 LKR
Storage (External HDD/SSD for data backups)	29,555 LKR

Software & Tools	
AI & ML Libraries (TensorFlow, Scikit-learn)	Free (Open Source)
Database Software (MySQL, MongoDB)	Free (Community Version)
Python Development Tools (PyCharm, VS Code)	Free
Network Monitoring Tools (Wireshark, SNMP utilities)	Free
Hosting & Deployment	
Cloud Server (For remote access & scalability)	2,955.5 LKR-8,866.5 LKR

05: System Architecture

System architecture defines the design and structure of the proposed AI-based network monitoring and root cause analysis system. This chapter presents the key diagrams that illustrate how different components interact, how data is stored, and how the system is designed to ensure scalability, performance, and efficiency.

5.1 Class Diagram of Proposed System

A class diagram represents the object-oriented structure of the system, showing different classes, attributes, methods, and relationships between them. Below is an overview of the key components:

Main Classes & Their Responsibilities:

1. NetworkMonitor
    - Attributes: bandwidth usage, latency, packet loss, jitter, data usage
    - Methods: collect network data (), analyze Traffic (), generate report ()
2. Fault Detection AI
    - Attributes: AI model, prediction accuracy, training data
    - Methods: train model (), predict Fault (), analyze patterns ()
3. Security Analyzer
    - Attributes: firewall status, intrusion alerts, suspicious traffic
    - Methods: detect threats (), send security alerts (), block unauthorized access ()
4. User Interface
    - Attributes: dashboard View, alerts, settings
    - Methods: display graph (), showers (), update configurations ()
5. Database Manager
    - Attributes: network logs, user preferences, historical data
    - Methods: store data (), fetch data (), backup logs ()

Class Diagram

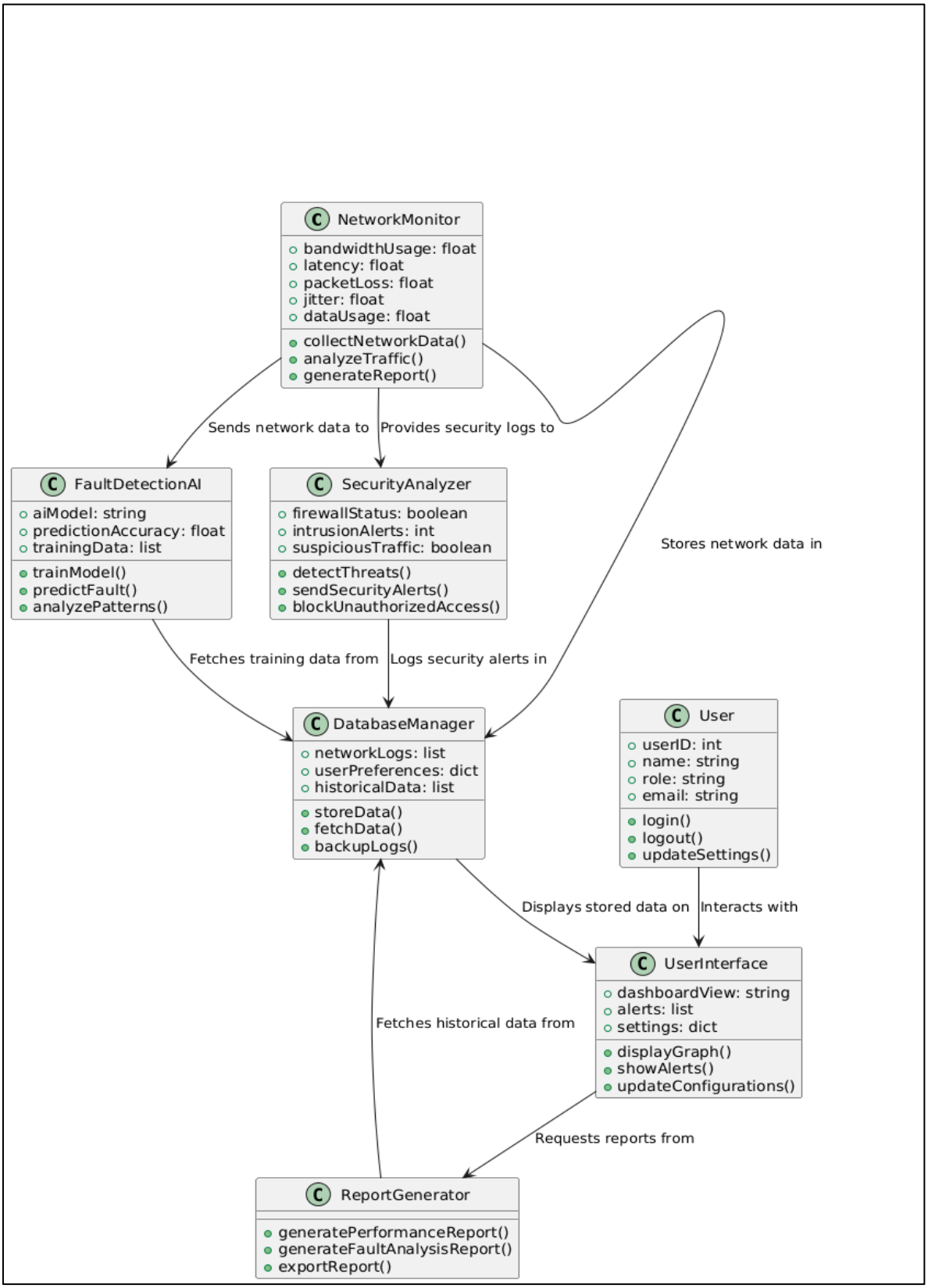


Figure 28Class Diagram

5.2 ER Diagram (Entity-Relationship Diagram)

The ER diagram represents the database structure, showing how data entities are related. This is essential for managing network logs, AI models, security alerts, and user data.

Key Entities & Relationships:

1. User (User ID, Name, Role, Email, Password)
    - Can configure settings and view reports.
2. Network Data (Data ID, Timestamp, Bandwidth, Latency, Packet Loss, Jitter, Throughput)
    - Stores real-time network performance data.
3. Root Cause Analysis (Prediction ID, Data ID, AI Analysis, Fault Type, Confidence Score, Suggested Fix)
    - The AI model processes Network Data to predict issues.
4. Security Alerts (Alert ID, Timestamp, Threat Type, Affected Device, Action Taken)
    - Logs intrusion attempts, security breaches, and threats.
5. System Logs (Log ID, Event Type, Description, Timestamp)
    - Stores system-related events for auditing purposes.

ER Diagram Structure

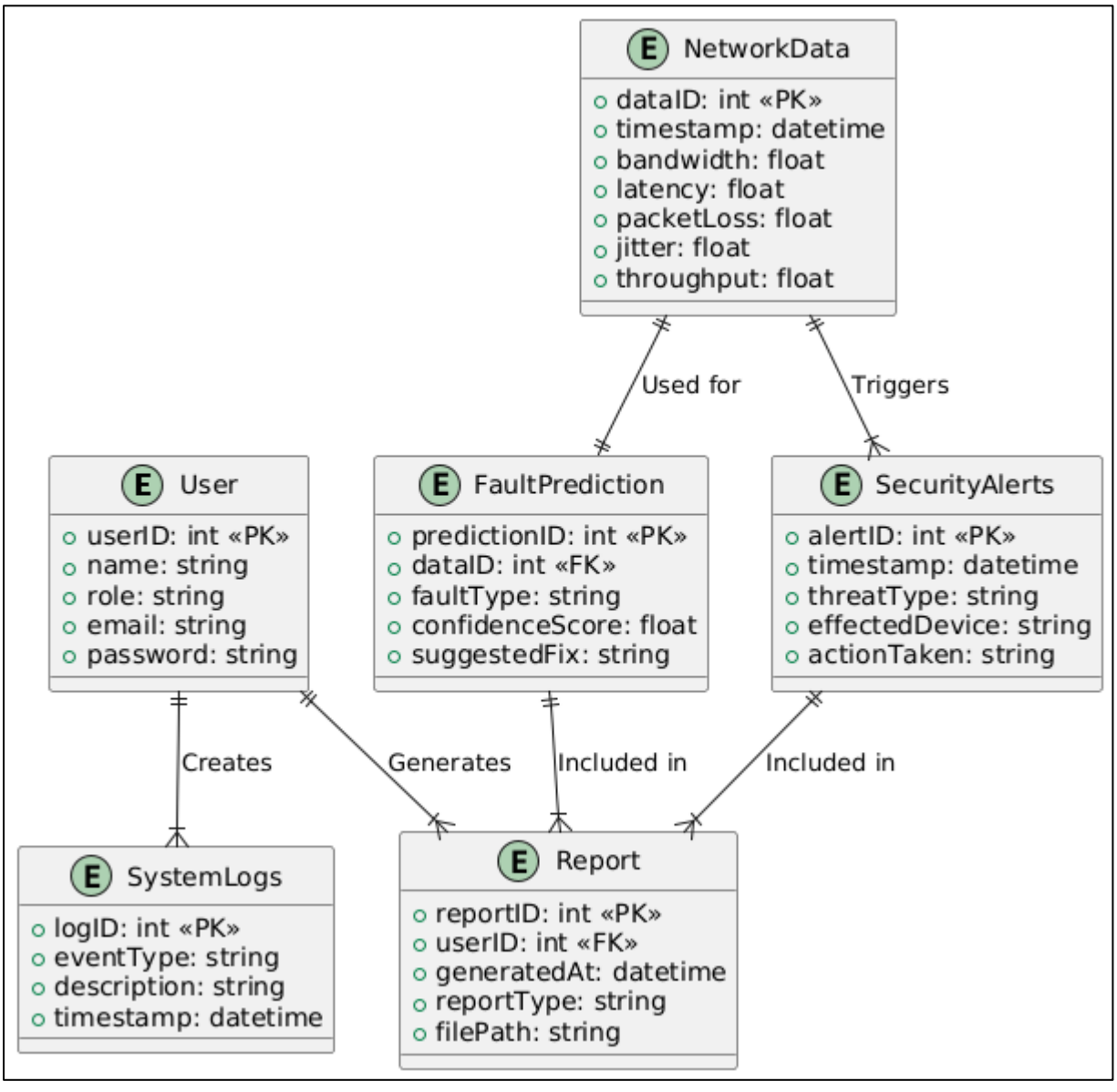


Figure 29 ER Diagram Structure

5.3 High-Level Architectural Diagram

The high-level architecture of the system represents how different modules interact with each other.

Main Components:

1. Data Collection Layer
    a. Collects real-time data from network devices (routers, switches, Wi-Fi access points).
    b. Uses SNMP, ICMP (ping tests), and packet analysis for monitoring.
2. Processing & AI Model Layer
    a. AI analyzes network behavior, predicts faults and provides root cause analysis.
    b. Security monitoring checks for unauthorized access and anomalies.
3. Database & Storage Layer
    a. Stores network logs, AI models, and security alerts in a structured database.
    b. Supports historical analysis and pattern detection.
4. User Interface Layer
    a. Displays real-time network status, alerts, and AI-generated recommendations.
    b. Provides a dashboard for administrators to interact with the system.

Architectural Diagram

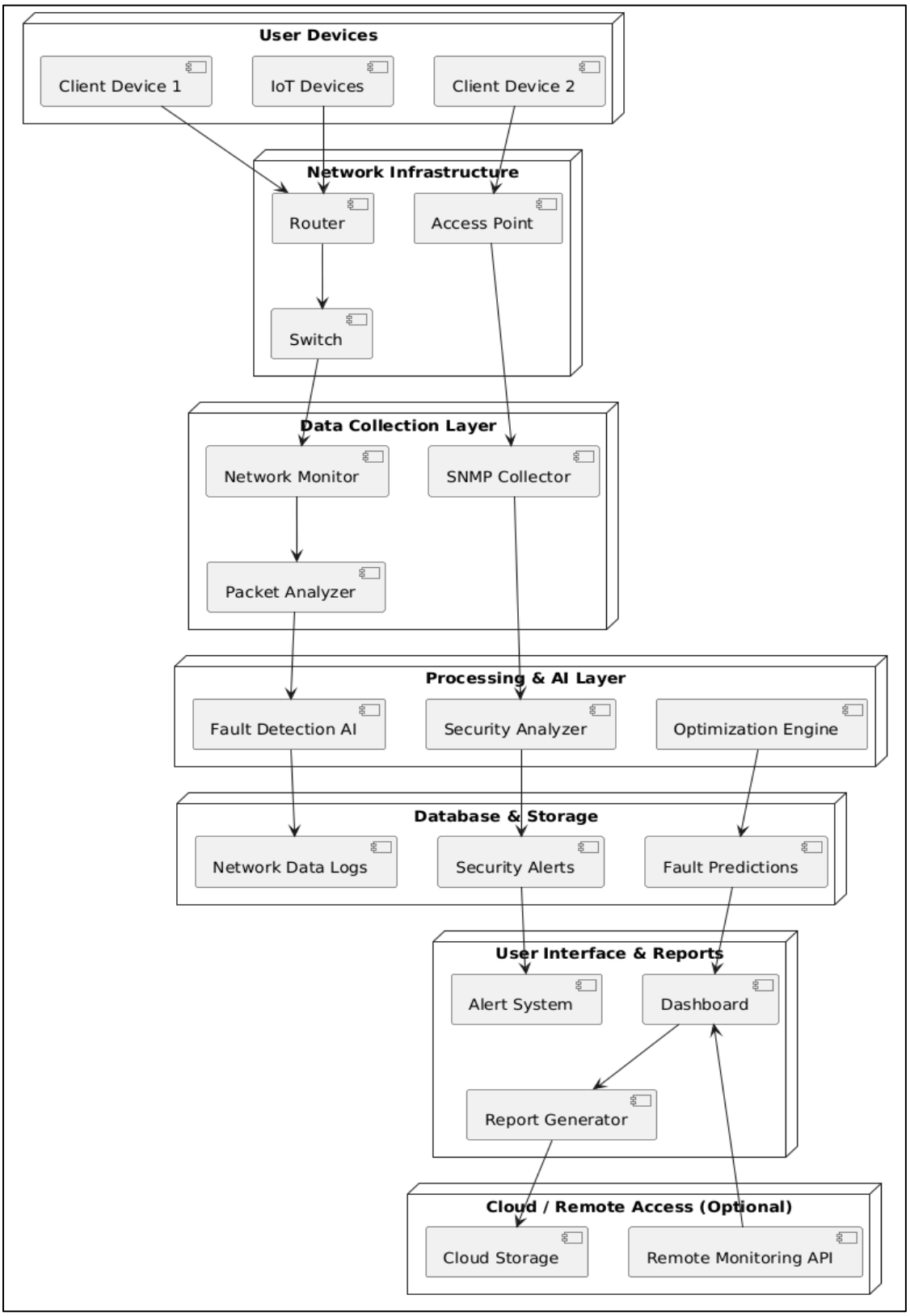


Figure 30 Architectural Diagram

5.4 Networking Diagram

The networking diagram illustrates how different network components interact with the monitoring system.

Network Components & Connections:

- Client Devices (Laptops, Smartphones, IoT Devices) → Connected to Wi-Fi/LAN.
- Access Points & Switches → Transmit data to the central monitoring system.
- Network Server → Processes data, runs AI analysis, and manages security.
- Cloud Backup (Optional) → Stores logs and reports for remote access.

Network Diagram Representation

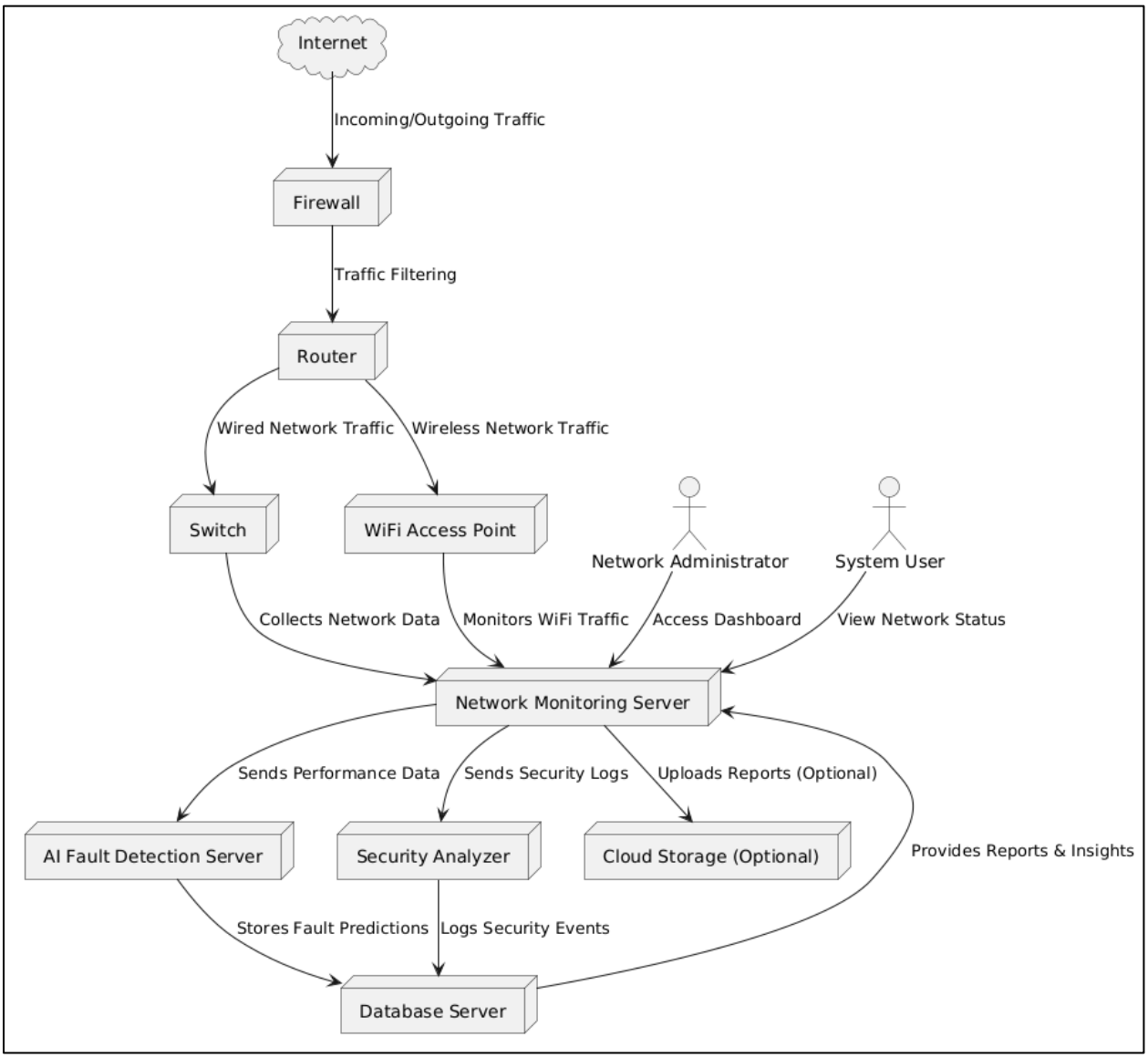


Figure 31 Network Diagram Representation

06: Development Tools and Technologies

Developing a network monitoring and root cause analysis system requires a combination of methodologies, programming languages, libraries, and algorithms to ensure accurate fault detection, real-time analysis, and security monitoring. This chapter outlines the development approach and technologies used in the project.

6.1 Development Methodology

For this project, I chose the Agile development methodology, as it allows flexibility, continuous improvements, and user feedback integration. Agile helps in building the system in small, manageable phases, making it easier to test, update, and enhance overtime.

Key Features of the Agile Methodology Used in This Project:

- Iterative Development – The project is broken down into multiple small development cycles (sprints), where new features are tested and refined.
- Continuous Testing & Feedback – After each sprint, the system is tested for performance, accuracy, and user experience, allowing improvements based on feedback.
- Scalability & Adaptability – If new network issues or security threats emerge, the system can be updated without rebuilding the entire architecture.
- Collaboration – Regular discussions and research ensure the system meets real-world networking challenges.

By using Agile, I ensure the system is efficient, user-friendly, and continuously improving based on real-world testing and feedback.

6.2 Programming Languages and Tools

Developing a network performance monitoring and AI-powered fault detection system requires a mix of backend, frontend, AI processing, and data visualization technologies.

Primary Programming Languages:

Language	**Purpose**
Python	Used for AI model training, network data analysis, and backend processing.
JavaScript (React.js)	For building the dashboard interface and data visualization.
SQL (MySQL, PostgreSQL)	For storing network logs, AI-generated insights, and security alerts.

Software Tools & Frameworks:

Tool/Framework	**Usage**
PyQt5	Used for developing the desktop application GUI.
Matplotlib & Seaborn	For visualizing network performance trends and AI insights.
Wireshark & Scapy	This is for network packet analysis and traffic patterns monitoring.

SNMP Utilities	Helps collect network device performance metrics.
TensorFlow & Scikit-learn	Used for training and running the Root cause analysis AI model.

By combining these tools, the system can effectively collect, process, and visualize network data while providing AI-driven insights.

6.3 Third-Party Components and Libraries

To enhance the system's performance, security, and machine learning capabilities, I integrated various third-party libraries that simplify development while ensuring efficiency.

Key Third-Party Libraries & Their Roles:

Library	**Purpose**
TensorFlow / Py Torch	Used for AI-based fault detection and pattern recognition.
Scikit-learn	Helps with data preprocessing, anomaly detection, and model training.
Pandas & NumPy	Used for handling large sets of network data efficiently.
Flask / Fast API	Provides a backend API for real-time data communication between the AI model and the dashboard.
Cryptography & Py Crypto	Used for, secure data storage and encrypted communication between network devices.
Dash / Plotly	Help create interactive network performance graphs and reports.

These libraries ensure the system is powerful, scalable, and secure, allowing accurate AI predictions, real-time monitoring, and strong encryption.

6.4 Algorithms

The system relies on machine learning and network monitoring algorithms to detect faults, optimize network performance, and identify security threats.

1. Machine Learning Algorithms for Root Cause Analysis:

Algorithm	**Purpose**
Random Forest	Analyzes multiple network performance metrics to predict faults.
K-Means Clustering	Groups have similar network behaviors to detect unusual patterns.
LSTM (Long Short-Term Memory)	A deep learning model used for time-series prediction, helping in analyzing past network trends to forecast future issues.

2. Security Algorithms for Threat Detection:

Algorithm	**Purpose**
Anomaly Detection (Isolation Forests)	Identifies unusual traffic patterns or unauthorized access.
Signature-Based Intrusion Detection	Matches network activity against a database of known threats.

3. Data Analysis and Optimization Algorithms:

Algorithm	Purpose
Linear Regression	Help understand how network performance changes over time.
Bandwidth Allocation Optimization (Genetic Algorithm)	Ensures better bandwidth distribution among network devices.

These algorithms ensure the system can accurately detect network faults, optimize performance, and enhance security, making network management more proactive and intelligent.

07: Discussion

7.1 Overview of the Interim Report

This interim report provides a detailed analysis and development plan for my AI-powered network monitoring and root cause analysis system. The report outlines the problem statement, system requirements, architecture, development tools, and technologies used in the project. By integrating machine learning and network performance analysis, the system aims to improve fault detection, troubleshooting, QoS management, security monitoring, and traffic analysis.

The report also presents a feasibility study, confirming that the system is operationally, technically, and financially viable. Additionally, I have detailed the tools, programming languages, and algorithms that contribute to the system's development, ensuring it is scalable, efficient, and secure.

7.2 Summary of the Report

The primary goal of this project is to create an AI-driven network monitoring system that proactively detects faults and suggests solutions before issues escalate. Traditional network monitoring tools often rely on manual analysis and reactive problem-solving, which can cause delays and increase operational costs. This project addresses those gaps by introducing real-time AI-based fault detection, security monitoring, and performance optimization.

Key aspects of the report include:

- System Analysis – Discuss the drawbacks of traditional monitoring systems and the need for AI-powered automation.
- Requirements Specification – Defining the functional and non-functional requirements for efficient system design.
- Feasibility Study – Evaluating the practicality, cost, and scalability of the system.
- System Architecture – Outlining how different components interact for data collection, processing, and visualization.
- Development Tools & Algorithms – Detailing the AI models, programming languages, and network monitoring techniques used.

7.3 Challenges Faced

During the research and development phase, I encountered several challenges:

Handling Large-Scale Network Data

- Collecting and processing high volumes of real-time network traffic required optimized data handling techniques to prevent slowdowns.
- Solution: Used efficient database queries, caching mechanisms, and optimized AI model processing.

Ensuring Accurate Root Cause Analysis

- Training the AI model to detect faults and anomalies correctly was challenging due to varied network conditions and unpredictable faults.
- Solution: Improved AI training with diverse datasets and real-time feedback loops.

Integration with Different Network Environments

- Networks use different protocols (SNMP, TCP/IP, ICMP), making it complex to ensure compatibility across various devices.
- Solution: Designed a flexible system architecture that adapts to different network infrastructures.

Balancing Security and Performance

- While implementing intrusion detection and encryption, I had to ensure that security features did not negatively impact system speed.
- Solution: Used lightweight security algorithms and real-time anomaly detection without slowing down processing.

7.4 Future Plans / Upcoming Work

Moving forward, I plan to focus on enhancing system performance, improving AI accuracy, and integrating additional features to make the system even more powerful.

Next Steps in Development:

1. Finalizing the AI Model – Further refining machine learning models to increase prediction accuracy.
2. User Interface Enhancements – Improving the dashboard's usability and real-time visualization capabilities.
3. Expanded Security Features – Implementing advanced threat intelligence to detect complex cyber threats.
4. Cloud-Based Data Storage – Adding remote access and cloud integration for scalability. 5. Comprehensive Testing & Deployment – Conducting real-world testing in different network environments to ensure system reliability.

This project has been an exciting challenge, and I am confident that further improvements will make it a robust and intelligent network management tool. The next phase will focus on real-time testing and fine-tuning the AI model to ensure maximum efficiency and accuracy.

References

Research Papers & Academic References:

- D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., & Casas, P. (2019). *A survey on big data for network traffic monitoring and analysis.* IEEE Transactions on Network and Service Management, 16(3), 800–813. https://doi.org/10.1109/TNSM.2019.2933358
- Marin-Quintero, J., Orozco-Henao, C., Bretas, A. S., Velez, J. C., Herrada, A., Barranco-Carlos, A., & Percybrooks, W. S. (2022). *Adaptive Fault Detection Based on Neural Networks and Multiple Sampling Points for Distribution Networks and Microgrids.* Journal of Modern Power Systems and Clean Energy, 10(6), 1648–1657. https://doi.org/10.35833/MPCE.2021.000444
- Nouioua, M., Fournier-Viger, P., He, G., Nouioua, F., & Min, Z. (2021). *A survey of machine learning for network fault management.* In Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics: Theories and Applications (pp. 1–27). Springer International Publishing. https://doi.org/10.1007/978-3-030-66288-2_1

Books & Industry Reports:

- Stallings, W. (2020). *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud.* Pearson Education.
- Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer Networks (5th Edition).* Pearson Education.

Online Resources & Documentation:

- Wireshark Documentation: https://www.wireshark.org/docs/
- SNMP Protocol Guide: https://www.ietf.org/rfc/rfc1157.txt
- TensorFlow Documentation: https://www.tensorflow.org/
- Python Scikit-learn: https://scikit-learn.org/

Appendices (If Applicable)

The appendices section includes additional information, technical details, sample datasets, and snippet code that support the main report but are not essential for the core discussion.

Appendix A: Sample Network Data Format

Timestamp	Bandwidth (Mbps)	Latency (MS)	Packet Loss (%)	Jitter (MS)
2025-02-21 12:00	120	5	0.3	1.2
2025-02-21 12:05	110	7	0.5	1.4

Appendix B: AI Model Training Dataset Sample

- **Features Used:** Bandwidth, Latency, Packet Loss, Jitter, Device Type
- **Training Algorithm:** Random Forest, LSTM, Isolation Forest

Appendix C: Application

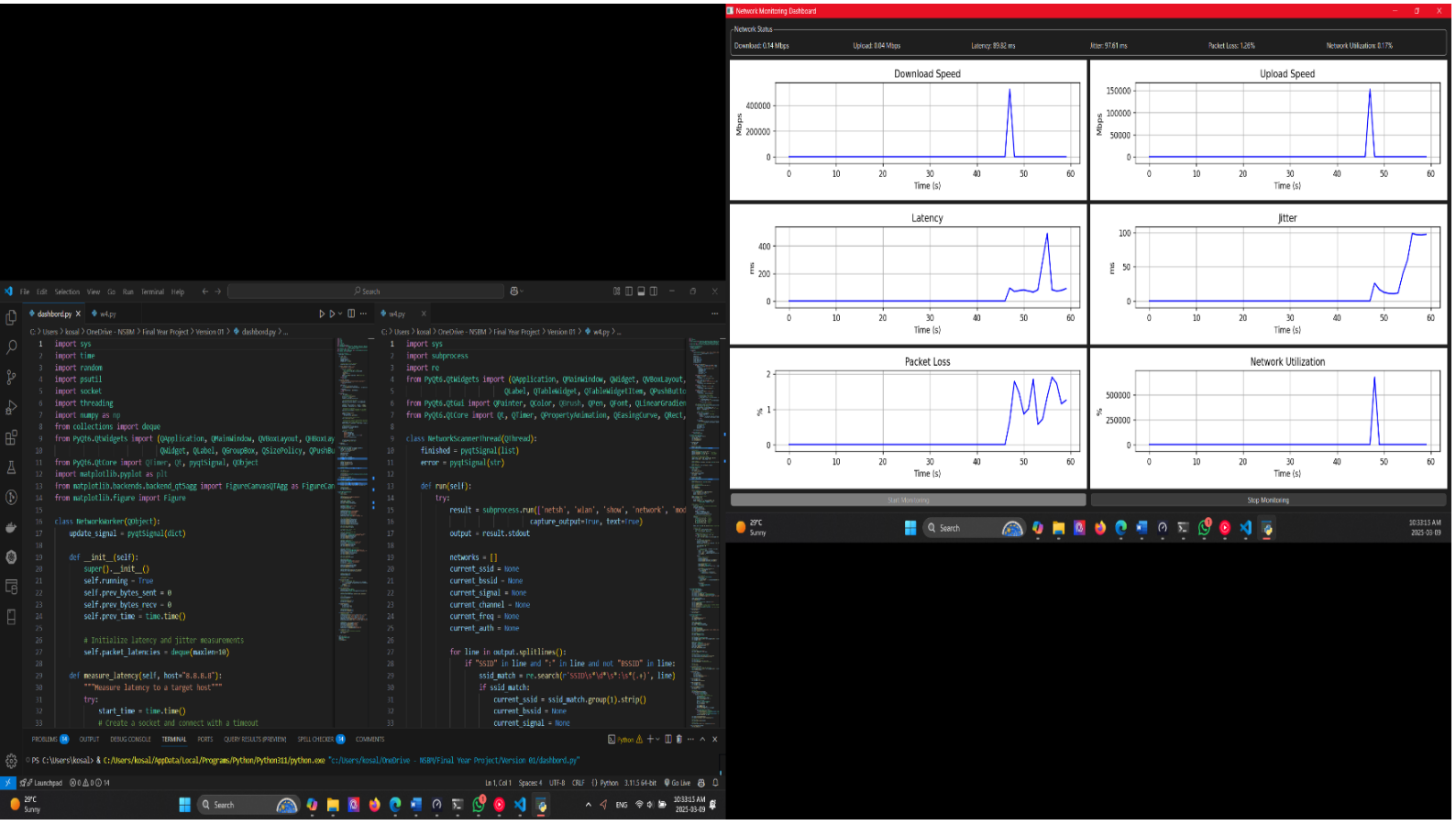


Figure 32 Test 1

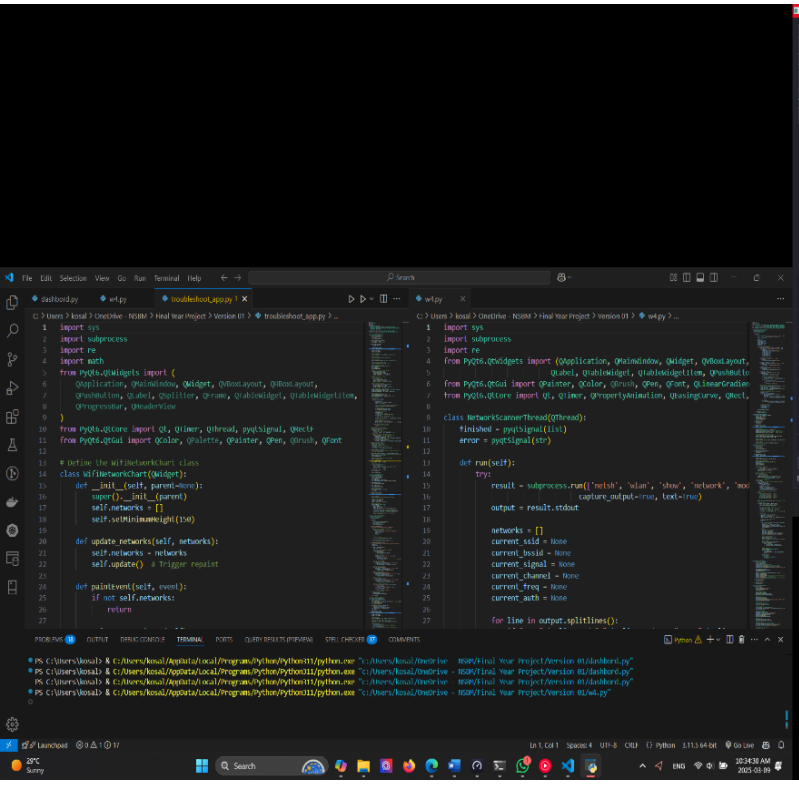


Figure 33 Test 2

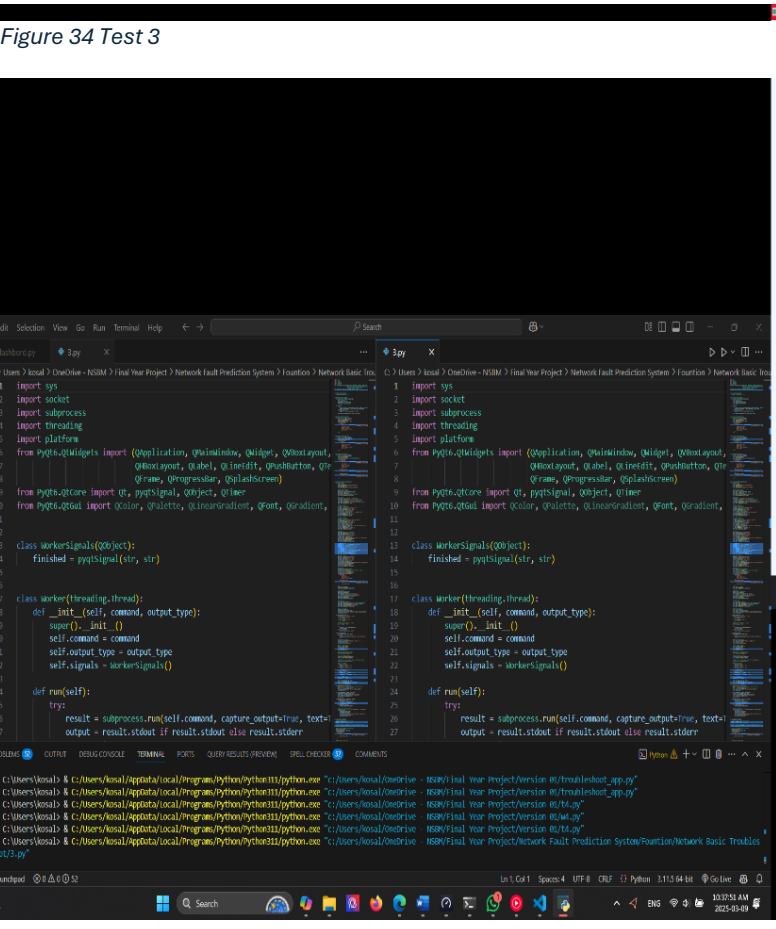


Figure 34 Test 3

8. Records of supervisory meetings

UNIVERSITY OF PLYMOUTH

NSBM GREEN UNIVERSITY TOWN

PUSL3190 Computing Individual Project
Student Progression Report
[Student Copy]

01. Student Name: Ileperumaachchige D Gunathilaka
02. Plymouth Index Number: 10898583
03. Degree Program: BSc(Hons) Computer Networks - PLY
04. Supervisor Name: Mr. Chamara Disanayake
05. Project Title: Network fault prediction System

Meeting Number	Meeting 01	Meeting 02	Meeting 03	Meeting 04	Meeting 05	Meeting 06	Meeting 07
Date	07/10	07/11	20/02	06/03	25/03	01/04	
Student Signature	kasela	kasela	kasela	kasela	kasela		
Supervisor Signature							

Meeting Number	Meeting 08	Meeting 09	Meeting 10	Meeting 11	Meeting 12	Meeting 13	Meeting 14
Date							
Student Signature							
Supervisor Signature							

Figure 35 Student Progression Report

UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 01

Date : 2024/10/07

Project Title : Network fault prediction System

Name of the Student : Ileperumaachchige D Gunathilaka

Students ID : 10898583

Name of the Supervisor : Mr. Chamara Disanayaka

Items discussed:

- project idea discussion

Items to be completed before the next supervisory meeting:

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

Figure 36 Student Document Submission Report 1

UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 02

Date : 2024/11/17

Project Title : Network fault prediction System

Name of the Student : Ileperumaachchige D Gunathilaka

Students ID : 10898583

Name of the Supervisor : Mr. chamara Disanayaka.

Items discussed:

- Demonstration done
- Discuss the current situation

Items to be completed before the next supervisory meeting:

.....

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

Figure 37 Student Document Submission Report 2

UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 03

Date : 2025/02/20

Project Title : Network fault prediction System

Name of the Student : Illeperumaachchige D Gunathilaka

Students ID : 10898583

Name of the Supervisor : Mr. chamara Disanayaka

Items discussed:

Demonstration done

Discuss the current status

Items to be completed before the next supervisory meeting:

Instructed to improve the features of the system and change the topic root cause analysis.

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

Figure 38 Student Document Submission Report 3

UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 04

Date : 2025/03/06

Project Title : Root Cause Analysis

Name of the Student : Ilepperumaachchige D Gunathilaka

Students ID : 10898583

Name of the Supervisor : Mr. Chamara Disnayaka

Items discussed:

Discuss the current improvements

Items to be completed before the next supervisory meeting:

- Suggested to add user login bad login attempts for the system
- Do a complete demo with unit functions during the next meeting

.....

Supervisor (Signature & Date)

Figure 39 Student Document Submission Report 4

UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 05

Date : 29/03/2025

Project Title : Root Cause Analysis

Name of the Student : Ileperumachchige don D Gunathilaka

Students ID : 10898583

Name of the Supervisor : Mr. Chamara Disnayaka

Items discussed:

Demonstrated the system

Items to be completed before the next supervisory meeting:

- Issue with BSSID - correct it.
- Integrate the system.

Supervisor (Signature & Date)

Figure 40 Student Document Submission Report 5

UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 06

Date : 01/04/2026

Project Title : Root Cause Analysis

Name of the Student : Heperumachchige D Gunathilaka

Students ID : 10899583

Name of the Supervisor : Mr. Chamara Dissanayaka

Items discussed:

New features were demonstrated.

Items to be completed before the next supervisory meeting:

Reduce the performance requirements to features

Supervisor (Signature & Date)

Figure 41 Student Document Submission Report 6