Machine Learning Based Network Monitoring System

Date: November 20, 2025

Supervisor:
Dr. Muhammad Zain Siddiqi

Co-Supervisor:
Dr. Khurram Jadoon
Madam Beenish, Lecturer

Group Members:
Muhammad Younas – 2022456
Muhammad Umar Maqsood – 2022447
Shamina Durrani – 2022453

ML – Based Network Monitoring System

Revision History:

Revision History	*Date*	*Comments*
1.00	November 19, 2025	Initial Design Document Draft.
2.00		

Document Approval:
The following document has been accepted and approved by the following:

Signature	*Date*	*Name*
	November 20, 2025	Dr. Muhammad Zain Siddiqi
	November 20, 2025	Madam Beenish

List of Contents

List of Figures

1. INTRODUCTION

The Machine Learning-based Network Monitoring System is a FYP being developed at the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI). Aims to provide real-time anomaly detection within a LAN to enhance network security and assist network administrators in identifying and responding to cyber threats.

The platform will serve as a prototype for advanced network intrusion detection, demonstrating the application of machine learning techniques to network traffic analysis for security purposes.

1.1. PURPOSE

The purpose of this Software Requirements Specification (SRS) is to define the functional and non-functional requirements for the **Machine Learning-Based Network Monitoring System**. This system is a Final Year Project (FYP) designed to enhance network security by detecting anomalies in network traffic behavior. Unlike traditional signature-based systems, this project leverages machine learning to identify novel and polymorphic threats in real-time within a Local Area Network (LAN). This document serves as the roadmap for development, testing, and validation.

1.2. PRODUCT SCOPE

The Machine Learning-based Network Monitoring System is a FYP being developed at the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI). The system aims to provide real-time anomaly detection within a LAN to enhance network security and assist network administrators in identifying and responding to cyber threats.

The platform will serve as a prototype for advanced network intrusion detection, demonstrating the application of machine learning techniques to network traffic analysis for security purposes.

Table 1: Terms used in this document and their description

Name	Description
SRS	(Software Requirements Specification). A document that describes the nature of a project, software or application.
ML-NMS	Machine Learning-Based Network Monitoring System.
ML	Machine Learning: algorithms that enable systems to learn from data.
SPAN	Switched Port Analyzer: a method to copy network traffic to a monitoring device.
CLI	Command-Line Interface
RBAC	Role-Based Access Control
SRS	Software Requirement Specification
METADATA	High level data (e.g., packet size, flow duration e.tc) used for analysis, protecting privacy by ignoring payloads.
IDS	Intrusion Detection System
FYP	(Final Year Project). A substantial project undertaken by university students in their final year of study.
AI	(Artificial Intelligence). The simulation of human intelligence processes by machines, especially computer systems.
IPS	(Intrusion Prevention System). A network security device that monitors network and/or system activities for malicious or unwanted behavior and can react to block or prevent those activities.
PCAP	(Packet Capture Format). A file format for recording network traffic.
TLS	(Transport Layer Security). A cryptographic protocol designed to provide communications security over a computer network
FPR	(False Positive Rate). The proportion of actual negatives that are incorrectly identified as positives.
FNR	(False Negative Rate). The proportion of actual positives that are incorrectly identified as negatives.
TPR	(True Positive Rate). The proportion of actual positives that are correctly identified as positives.
TNR	(True Negative Rate). The proportion of actual negatives that are correctly identified as negatives.
IT	(Information Technology). The use of computers, storage, networking, and other physical devices, infrastructure, and processes to create, process, store, secure, and exchange all forms of electronic data.
SIEM	(Security Information and Event Management). A software solution that aggregates and analyzes security alerts and log data from various sources across an IT infrastructure.

LAN	(Local Area Network). A computer network that interconnects computers within a limited area such as a home, school, computer laboratory, or office building.
Client	In a network context, typically refers to a device (e.g., workstation, server, IoT device) that requests resources or services from another device (the server).
Server	A computer program or device that provides functionality or services for other programs or devices (clients) over a network.
SPAN	(Switched Port Analyzer). A feature on network switches that allows traffic from one or more source ports to be copied and sent to a designated destination port for monitoring.
Mirror Port	A common alternative term for a SPAN port, used to describe the capability of duplicating network traffic to another port for analysis.
Data Collectors	Components (software agents or dedicated hardware) deployed within a network responsible for gathering various types of telemetry data (e.g., metrics, logs, network flows) for monitoring and analysis.
Telemetry Data	Data collected from various sources (e.g., network devices, servers, applications) that provides information about the performance, health, and activity of a system.
MFA	(Multi-Factor Authentication). An authentication method that requires users to provide two or more verification factors to gain access to a resource.
OAuth	(Open Authorization). An open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites without giving them their passwords.
METADATA	High-level descriptive information about network traffic (e.g., packet size, timestamps, source/destination IPs, ports)
FLOW	A unidirectional sequence of packets sharing common attributes such as IP addresses, ports, and protocol
FEATURE VECTOR	A structured representation of extracted attributes used as input for ML models
SUPERVISED LEARNING	ML approach using labeled data to train classification algorithms
UNSUPERVISED LEARNING	ML approach used to identify patterns or anomalies in unlabeled data
ANOMALY DETECTION	ML-based identification of unusual patterns or deviations from normal traffic behavior
ALERT	A system generated notification indicating detected anomalous or suspicious activity
JSON	(JavaScript Object Notation). A lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate.

REAL-TIME PROCESSING	Data analysis and detection performed with minimal latency
IT Teams	Information Technology teams (which could be internal to an SMB or a managed service provider supporting SMBs)
DATASET	A structured collection of network samples used for training and testing ML models
MODEL DRIFT	Degradation of ML performance when input data characteristics change over time
CLASS IMBALANCE	Unequal representation of classes within the dataset
PREPROCESSING	Transformation steps applied to raw data before model training

1.3. OVERVIEW

The Machine Learning-Based Network Monitoring System is a standalone security solution designed to address the limitations of traditional signature-based IDSs. As we outlined in our project scope (FYP-Presentation -1), traditional systems often struggle with **unknown attack patterns** and high-volume traffic. This system employs behavioral analysis and machine learning algorithms to identify irregular patterns in real-time.

The system captures network packets, extracts flow-based features, and compares them against a dynamic baseline. If the "anomaly score" exceeds a threshold, the system alerts the network administrator. The primary goal is to provide a scalable, adaptive system that enhances security in high-traffic environments while minimizing false positives.

2. THE OVERALL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The system operates as an independent monitoring node within a LAN. It interfaces with:

- **Network Infrastructure:** Captures traffic via a Switched Port Analyzer (SPAN/Mirror) port or a direct network interface.
- **System Administrator:** Interacts with the system via a Web Dashboard and Command Line Interface (CLI) to view alerts and configure settings.
- **Local Storage:** Stores trained ML models, traffic feature datasets, and anomaly logs.

3. WORK BREAKDOWN STRUCTURE

The project is divided into **8 Major Epics**, subdivided into specific actionable tasks.

1. **Epic: Network Traffic Capture**
   - **FR-01:** Design packet capture architecture and flow.
   - **Task:** Implement traffic capture script (using Scapy/Socket) on mirror port.
   - **Task:** specific parsing logic for TCP/UDP/ICMP headers.
   - **Task:** Conduct unit testing for packet ingestion rate and loss.
2. **Epic: Feature Extraction**
   - **FR-04:** Define specific features required for the ML model (e.g., Flow Duration, Packet Size).
   - **Task:** specific feature extraction algorithms in Python.
   - **Task:** Convert raw packets into structured CSV/Vector format.
   - **Task:** Validate extraction accuracy against a known dataset (e.g., CIC-IDS).
3. **Epic: ML Anomaly Detection**
   - **FR-05:** Research and select unsupervised learning models (e.g., Isolation Forest, Autoencoders).
   - **Task:** Integrate the trained model with the real-time streaming pipeline.
   - **Task:** specific hyperparameters to balance False Positives vs. Detection Rate.
   - **Task:** Evaluate model performance using baseline normal traffic.
4. **Epic: Data Storage Management**
   - **FR-09:** Design the Database Schema (ERD) for traffic logs, features, and incidents.
   - **Task:** specific database connection handler (SQLite/PostgreSQL).
   - **Task:** Implement efficient write operations for high-volume logs.
   - **Task:** Test CRUD operations for incident retrieval.
5. **Epic: Alert Mechanism**
   - **FR-11:** Develop logic to trigger alerts based on ML Anomaly Scores.
   - **Task:** specific SMTP (Email) notification service.
   - **Task:** Configure threshold settings for Low, Medium, and High severity.
   - **Task:** Simulate attacks to verify alert delivery latency.
6. **Epic: Web Dashboard Visualization**
   - **FR-13:** Design UI Wireframes for the "Network Health" and "Live Traffic" pages.
   - **Task:** specific frontend components (Line Charts, Incident Tables).
   - **Task:** specific backend API endpoints to serve real-time data to the frontend.
   - **Task:** Integrate the dashboard with the Authentication module.
7. **Epic: User Management (RBAC)**
   - **FR-15:** Implement secure Login and Registration screens.
   - **Task:** specific Role-Based Access Control (Admin vs. Observer).
   - **Task:** Hash user passwords before database storage.
   - **Task:** Test permission restrictions (ensure Observers cannot change configs).
8. **Epic: Documentation & Testing**
   - **Task:** Compile the User Manual and Installation Guide.
   - **Task:** specific Deployment Documentation (Dependencies, Environment Setup).
   - **Task:** Perform Integration Testing (End-to-End system flow).

- **Task:** Finalize the Final Year Project Report.

4.5. PROJECT TIMELINE

The following schedule outlines the estimated duration for each module, starting from **November 25, 2025**.

Table 1: ML – NMS Project Timeline

Module / Epic	Start Date	End Date	Duration
Planning & Setup	Nov 25, 2025	Nov 30, 2025	5 Days
1. Dataset Cleansing	Nov 30, 2025	Dec 05, 2025	5 Days
2. Feature Extraction	Dec 06, 2025	Dec 15, 2025	10 Days
3. ML Anomaly Detection	Dec 16, 2025	Dec 27, 2025	12 Days
4. Data Storage	Dec 20, 2025	Jan 03, 2026	15 Days
5. Alerting Mechanism	Dec 29, 2025	Jan 10, 2026	13 Days
6. Web Dashboard	Jan 04, 2026	Jan 18, 2026	15 Days
7. RBAC / User Mgmt	Jan 10, 2026	Jan 22, 2026	13 Days
8. Docs & Testing	Jan 15, 2026	Jan 31, 2026	17 Days
Final Review & Buffer	Feb 01, 2026	Feb 07, 2026	7 Days

1. **Epic: Network Traffic Capture**
   - **FR-01:** Design packet capture architecture and flow.
   - **Task:** Implement traffic capture script (using Scapy/Socket) on mirror port.
   - **Task:** specific parsing logic for TCP/UDP/ICMP headers.
   - **Task:** Conduct unit testing for packet ingestion rate and loss.
2. **Epic: Feature Extraction**
   - **FR-04:** Define specific features required for the ML model (e.g., Flow Duration, Packet Size).
   - **Task:** specific feature extraction algorithms in Python.
   - **Task:** Convert raw packets into structured CSV/Vector format.
   - **Task:** Validate extraction accuracy against a known dataset (e.g., CIC-IDS).
3. **Epic: ML Anomaly Detection**
   - **FR-05:** Research and select unsupervised learning models (e.g., Isolation Forest, Autoencoders).
   - **Task:** Integrate the trained model with the real-time streaming pipeline.
   - **Task:** specific hyperparameters to balance False Positives vs. Detection Rate.
   - **Task:** Evaluate model performance using baseline normal traffic.
4. **Epic: Data Storage Management**
   - **FR-09:** Design the Database Schema (ERD) for traffic logs, features, and incidents.
   - **Task:** specific database connection handler (SQLite/PostgreSQL).
   - **Task:** Implement efficient write operations for high-volume logs.
   - **Task:** Test CRUD operations for incident retrieval.

5. **Epic: Alert Mechanism**
   - **FR-11:** Develop logic to trigger alerts based on ML Anomaly Scores.
   - **Task:** specific SMTP (Email) notification service.
   - **Task:** Configure threshold settings for Low, Medium, and High severity.
   - **Task:** Simulate attacks to verify alert delivery latency.
6. **Epic: Web Dashboard Visualization**
   - **FR-13:** Design UI Wireframes for the "Network Health" and "Live Traffic" pages.
   - **Task:** specific frontend components (Line Charts, Incident Tables).
   - **Task:** specific backend API endpoints to serve real-time data to the frontend.
   - **Task:** Integrate the dashboard with the Authentication module.
7. **Epic: User Management (RBAC)**
   - **FR-15:** Implement secure Login and Registration screens.
   - **Task:** specific Role-Based Access Control (Admin vs. Observer).
   - **Task:** Hash user passwords before database storage.
   - **Task:** Test permission restrictions (ensure Observers cannot change configs).
8. **Epic: Documentation & Testing**
   - **Task:** Compile the User Manual and Installation Guide.
   - **Task:** specific Deployment Documentation (Dependencies, Environment Setup).
   - **Task:** Perform Integration Testing (End-to-End system flow).
   - **Task:** Finalize the Final Year Project Report.

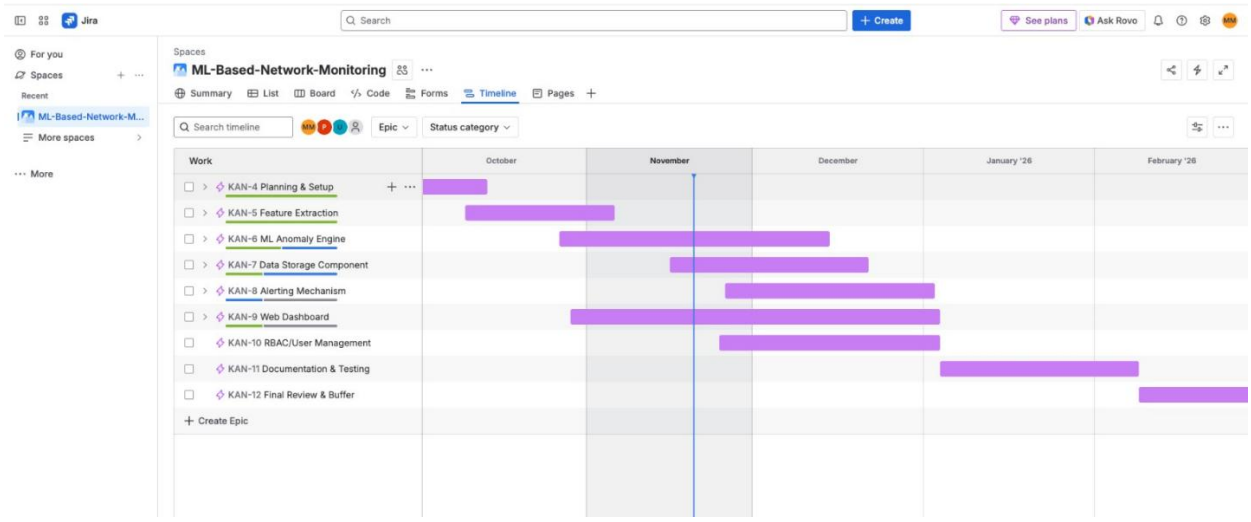


Figure 1 Work Breakdown Structure

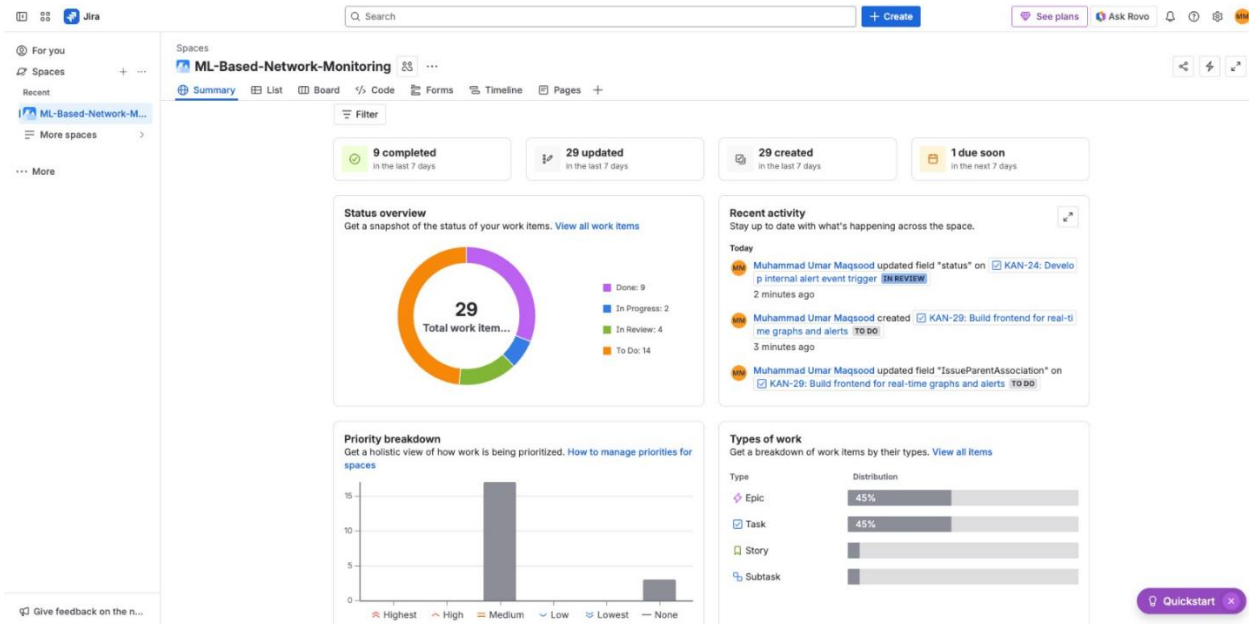


Figure 2: ML NMS Work Breakdown

4. Design

4.1 ARCHITECTURAL DESIGN

For the ML-Based Network Monitoring System, we have adopted **Layered Architecture**. This pattern organizes the system into horizontal layers, where each layer performs a specific role and communicates only with the layers immediately above or below it. This structure allows us to separate the machine learning processing logic from the user interface and data storage.

4.2. Why we choose Layered Architecture Design?

We selected the Layered Architecture based on the following engineering principles:

4.1.1 Separation of Concerns (High Cohesion):

The system has distinct functionalities: visual monitoring (Dashboard), complex data processing (ML/Sniffing), and storage (Database). Layering ensures that the UI code does not get mixed with the raw packet capture logic. This makes the codebase cleaner and easier to debug.

4.1.2 Modularity and Maintainability:

If we need to change the Machine Learning algorithm (e.g., switching from Isolation Forest to Autoencoders) in the *Business Logic Layer*, we can do so without rewriting the *Presentation Layer* (Dashboard). This reduces the risk of breaking the system during updates.

4.1.3 Abstraction:

The Network Administrator (User) interacts only with the Presentation Layer. They do not need to know the complexity of how packets are sniffed or how the database is structured. The architecture hides this complexity.

4.1.4 Scalability:

In the future, the *Data Layer* (Database) could be moved to a separate physical server to handle more traffic without affecting the *Application Layer*.

4.3. MODULE IDENTIFICATION

Based on the architecture, the system is divided into the following functional modules:

Layer 1: Presentation Module (User Interface)

- **Web Dashboard:** The front-end interface developed using HTML/CSS/JavaScript (or React). It is responsible for:
  - Visualizing traffic trends (Graphs/Charts).
  - Displaying alert tables.
  - Accepting user commands (Start/Stop Monitoring).

Layer 2: Application / Business Logic Module

This is the core of the system, containing the Python-based backend logic.

- **Packet Capture Module (The "Sniffer"):**
  - Uses the Network Interface Card (NIC) in promiscuous mode.
  - Captures TCP/IP packets in real-time using libraries like Scapy.
- **Feature Extraction Module:**
  - Parses raw binary packets.
  - Extracts key attributes (Source IP, Destination Port, Protocol, Flow Duration) to create a Feature Vector.
- **ML Anomaly Detection Module:**
  - Loads the trained Machine Learning model.
  - Takes the Feature Vector as input and outputs an Anomaly Score (0.0 to 1.0).
- **Alert Manager Module:**
  - Compares the anomaly score against the defined threshold.
  - Triggers email notifications via SMTP if a threat is detected.

Layer 3: Data Persistence Module

- **Database Handler:**
  - Manage connections to the SQLite/PostgreSQL database.
  - Stores historical traffic data, incident logs, and user configuration profiles.

4.4 ARCHITECTURAL DESIGN

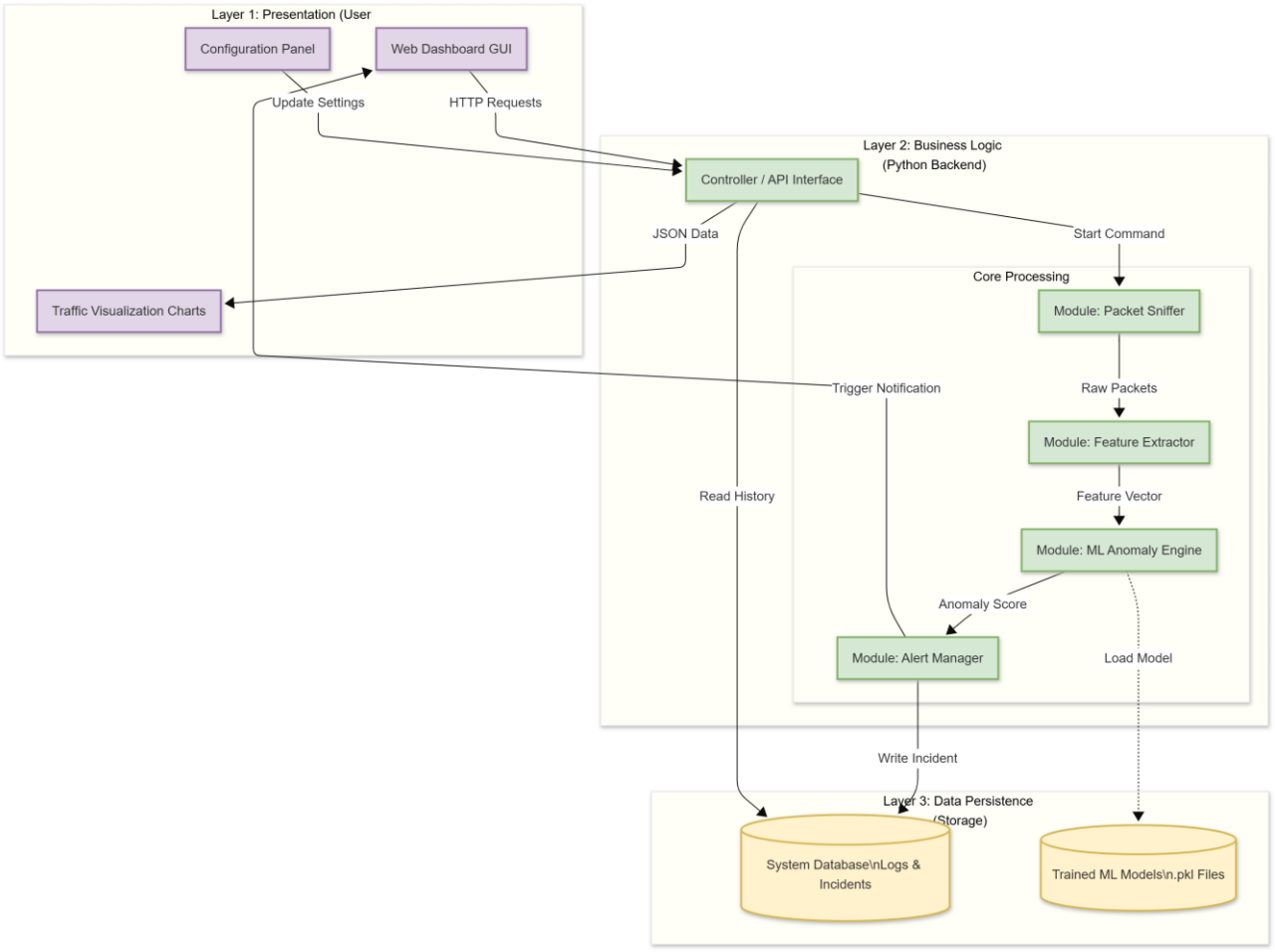
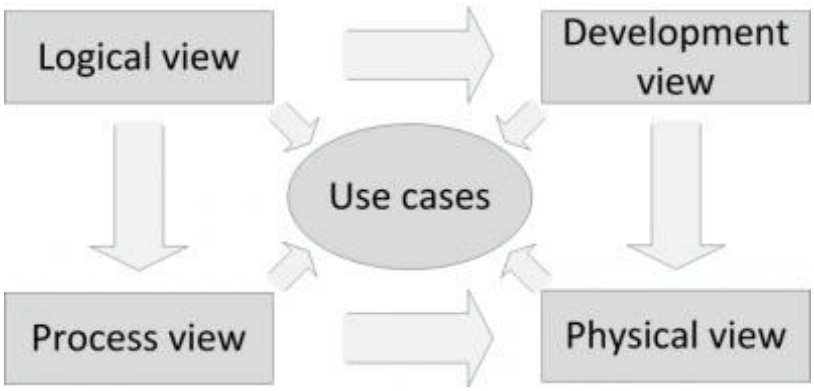


Figure 3: Layered Architecture Diagram

5. 4+1 ARCHITECTURE VIEW MODEL

The architecture of the ML-Based Network Monitoring System is described using the **4+1 View Model** (Kruchten, 1995). This ensures all stakeholder concerns from end-user functionality to physical deployment, are addressed.



3 4+1 Architecture View

5.1. Use Case View

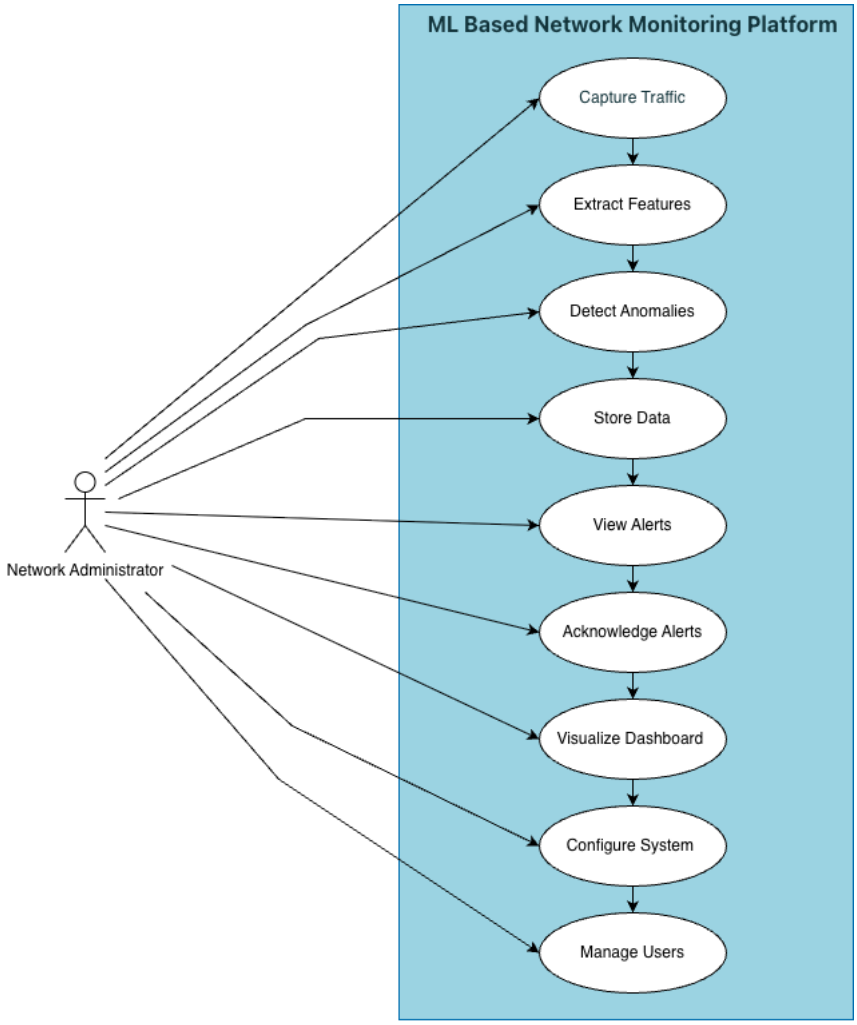


Figure 4: Use Case View Of Network Administrator (Primary User)

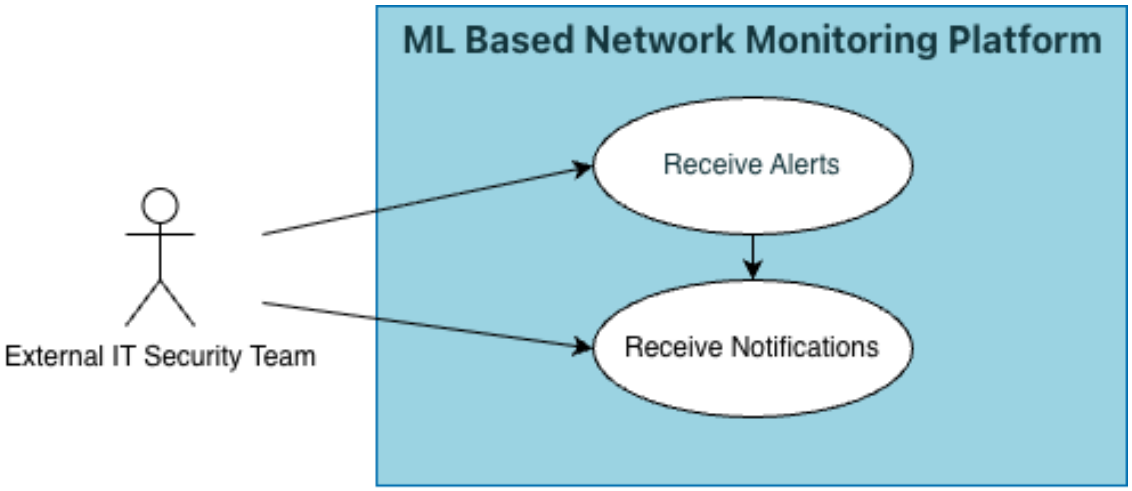


Figure 5: External IT Security Team Use Case Diagram (Secondary User)

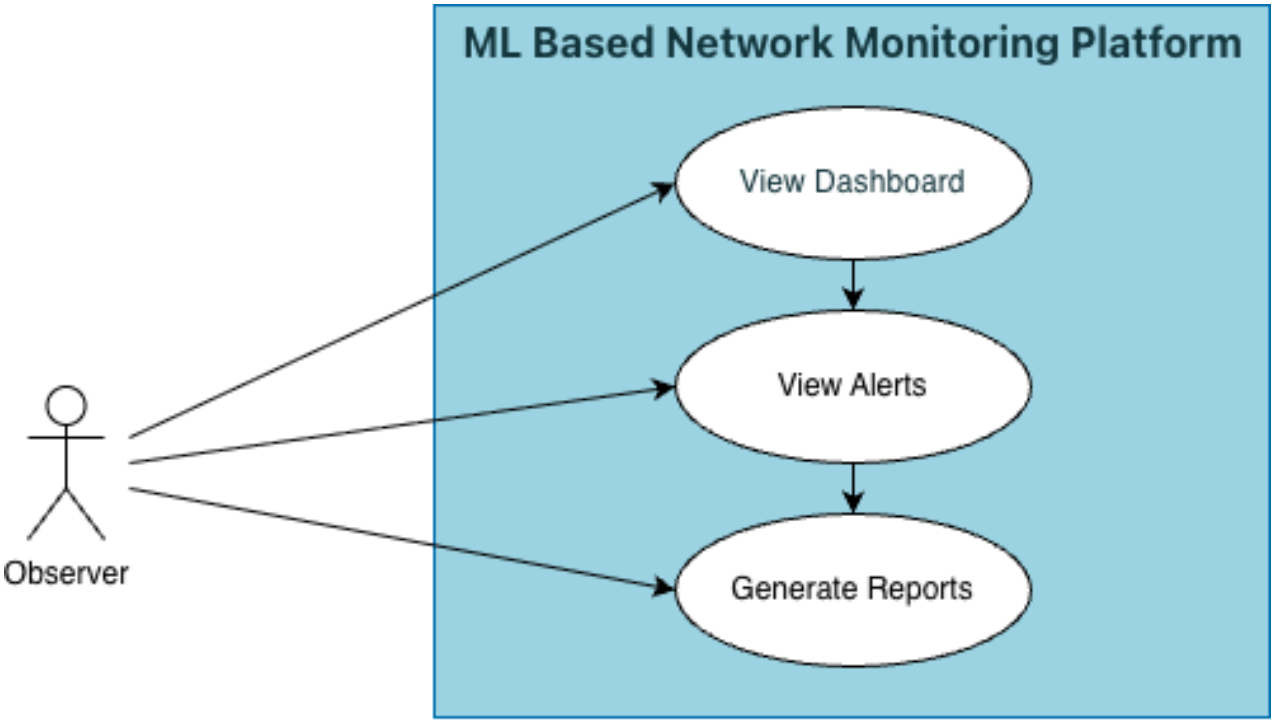


Figure 6: Observer Use Case Diagram (Secondary User)

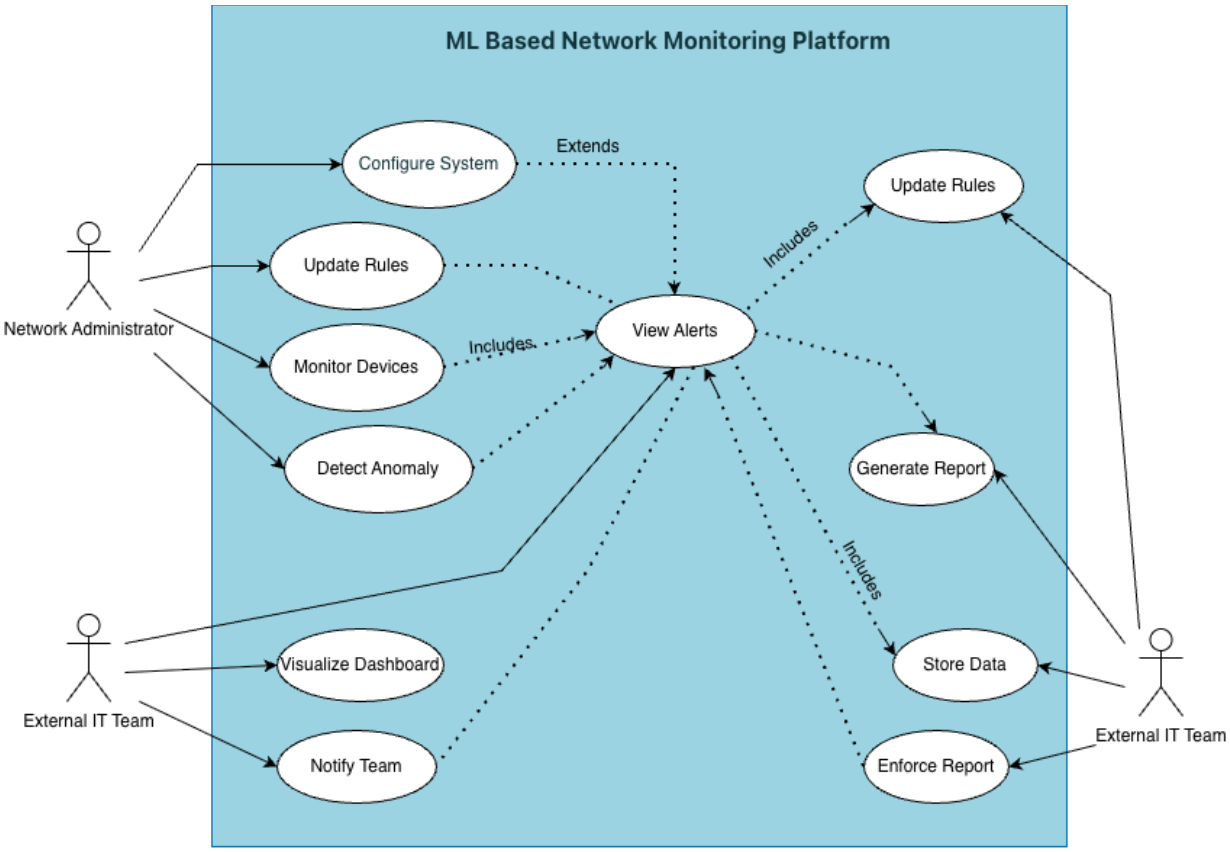


Figure 7: Overall System Use Case Diagram

5.2. Logical View:

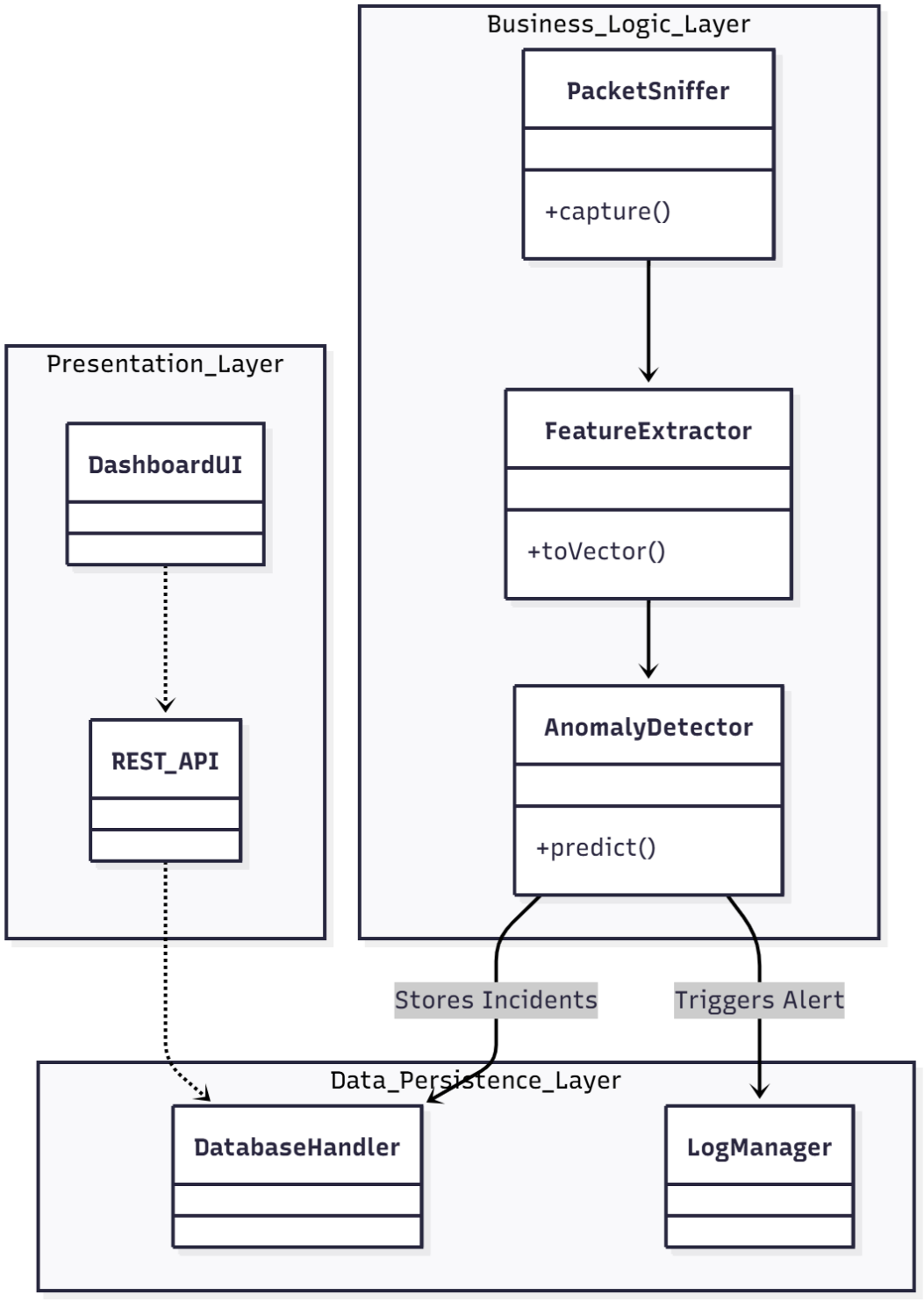


Figure 8: ML – NMS Logical View Diagram

5.3. Development View

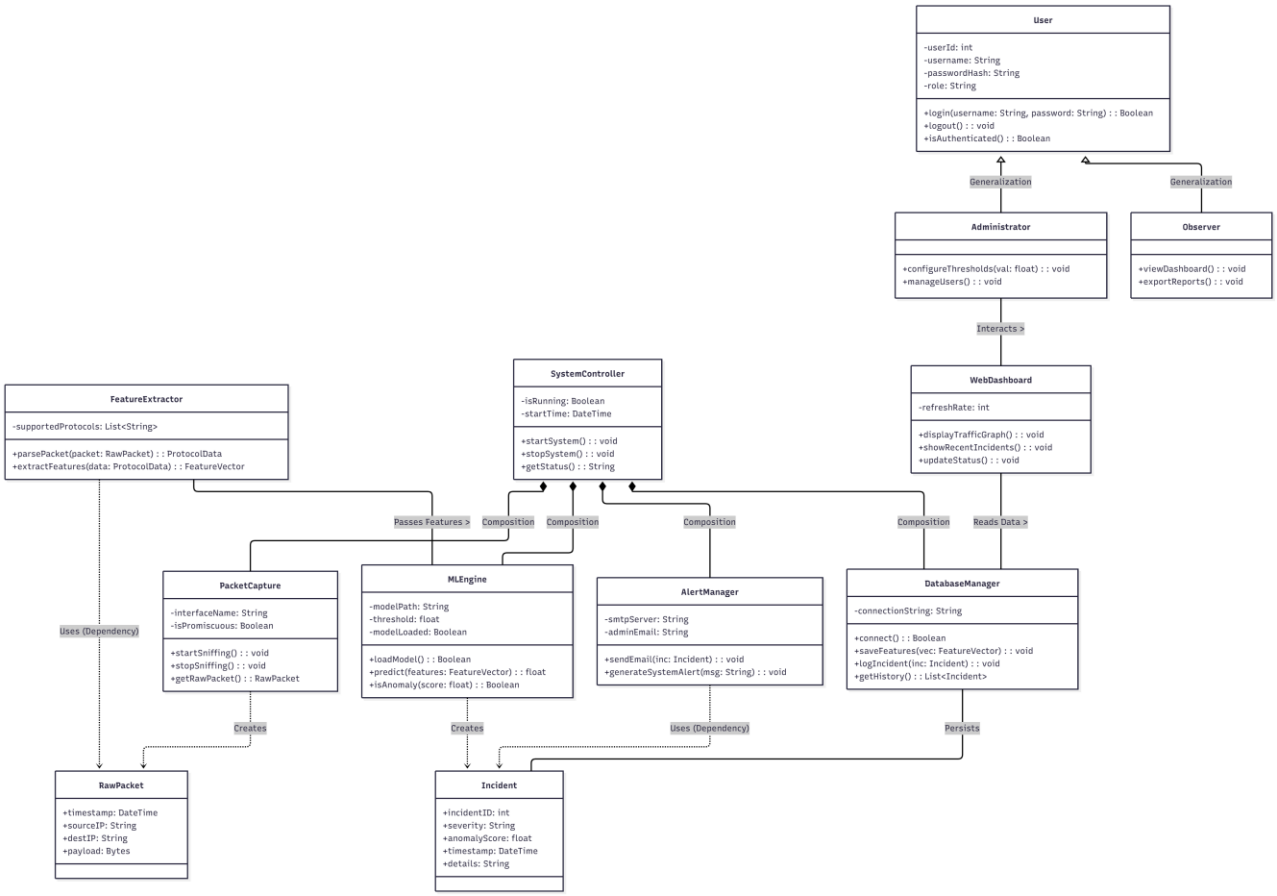


Figure 9: ML – NMS Development View Diagram (Low Level)

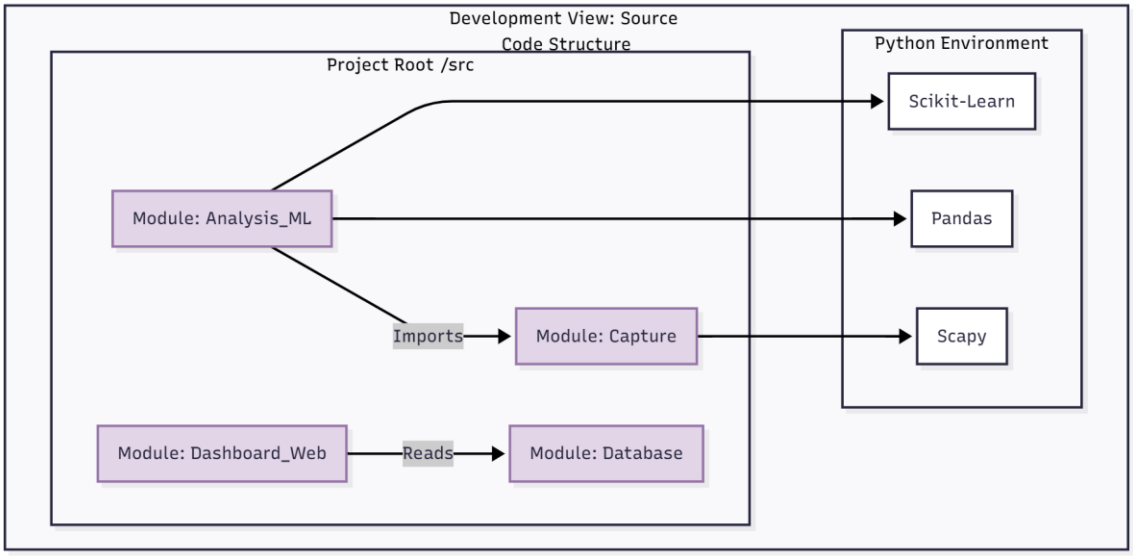


Figure 10: ML – NMS Development View Diagram (High Level)

5.4. Process View

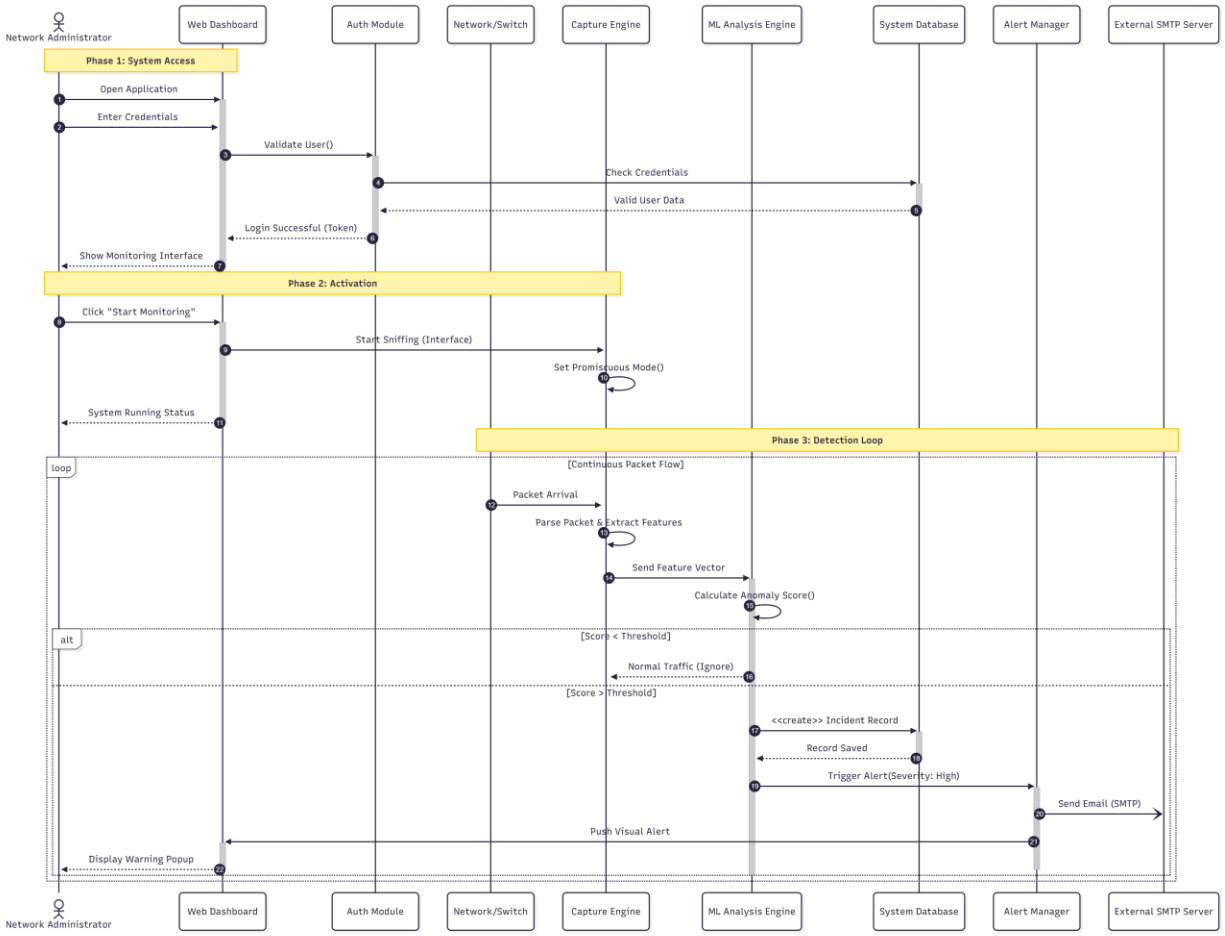


Figure 11: Process View of ML – NMS

5.5. Physical View

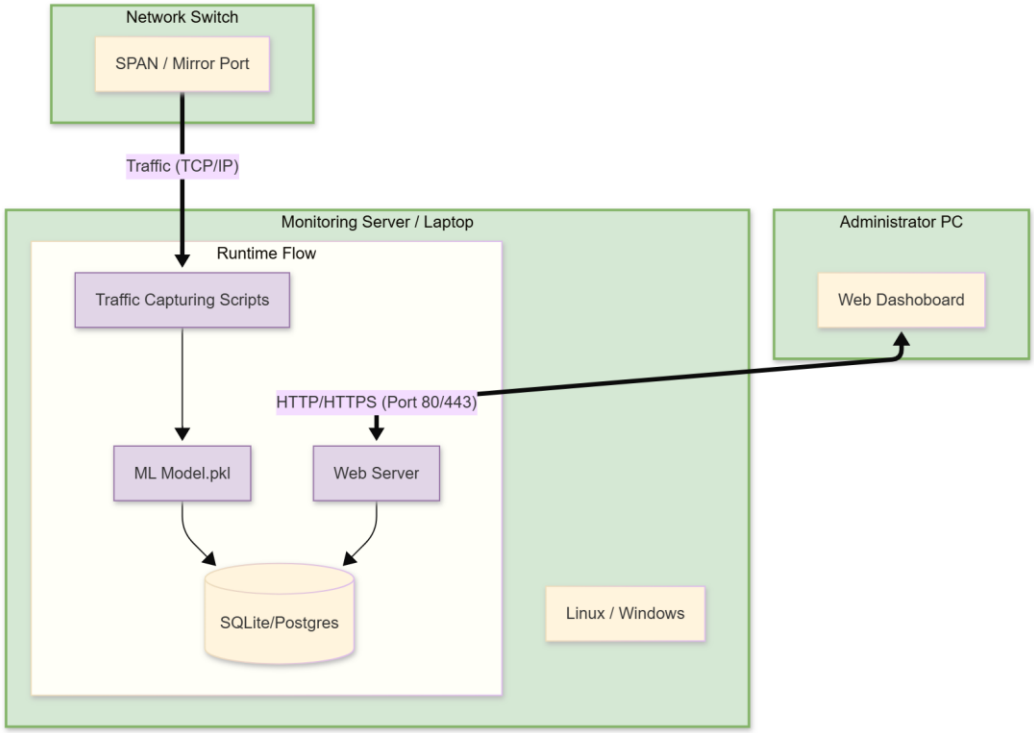


Figure 12: ML – NMS Physical View Diagram

5.6. User Interface Design

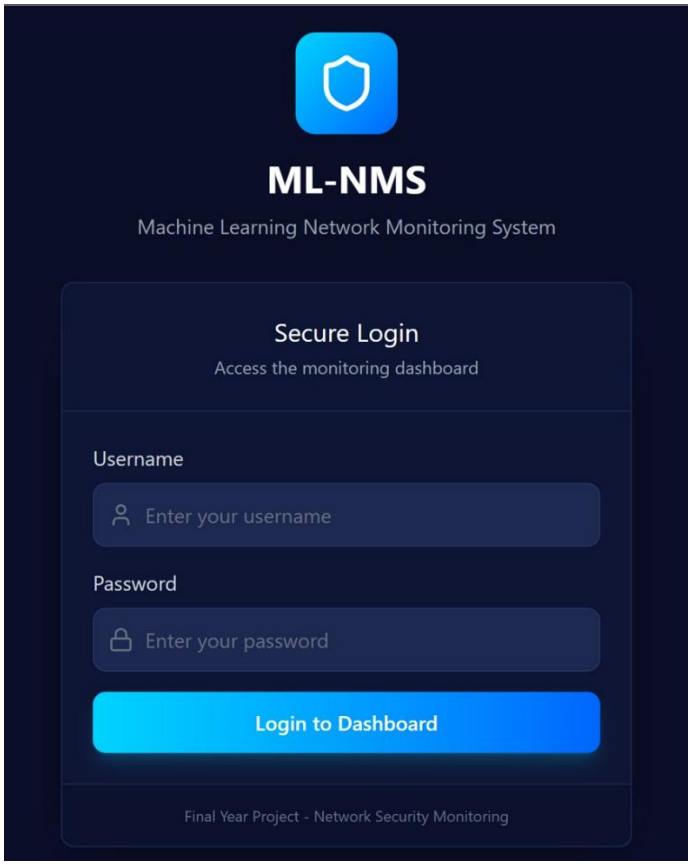


Figure 13: ML- NMS Dashboard Login

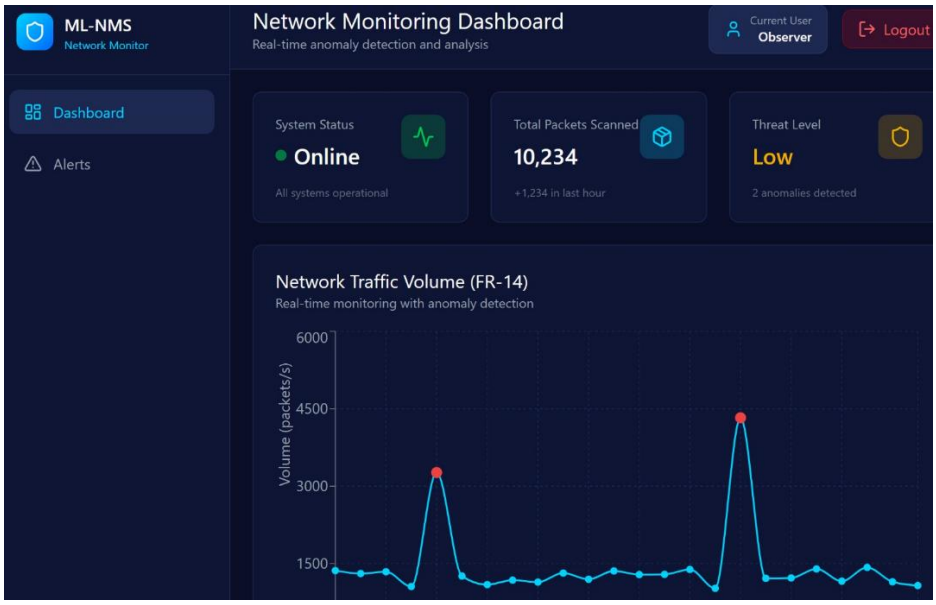


Figure 14: Network Monitoring Dashboard

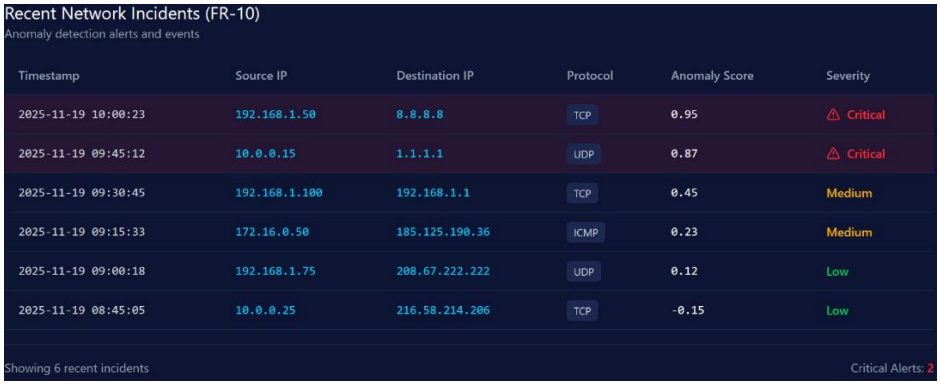


Figure 15: Recent Network Monitoring and Incidents Dashboard