



TITLE DEFENCE PRESENTATION

ML-BASED NETWORK MONITORING SYSTEMS

AGHA HASAN ABEDI AUDITORIUM

MUHAMMAD UMAR MAQSOOD 2022447

SHAMINA DURRANI 2022543

MUHAMMAD YOUNAS 2022456

SUPERVISOR:

DR. MUHAMMAD ZAIN SIDDIQI

CO-SUPRVISOR:

DR. KHURRAM JADOON

Ms Beenish, Lecturer

Recap

Project Goal:

Develop a **Machine Learning-Based Network Monitoring System** capable of **Anomaly Detection**.

The Problem

- **Evolving Threats:** Global cyber threats are increasing yearly by 25–30%. There is a significant rise in polymorphic and zero-day attacks.
- **Failure of Traditional Systems:** Traditional Signature-Based Intrusion Detection Systems (IDS) are static, rely on known attack signatures, and are abortive against novel/zero-day threats.
- **Operational Inefficiency:** Existing systems often produce too many incorrect alerts, which degrades trust in the Security Operation Center (SOC) and reduces operational efficiency.
- **The Need:** A requirement for a system that can evolve as the threat landscape changes, particularly given the high exposure of critical infrastructures.

Problem Statement

We aim to overcome the limitations of traditional intrusion detection systems such :

- 1 Static, signature based, can not detect novel attacks.
- 2 Too many incorrect alerts weaken security operation center (SOC) trust and reduce operational efficiency.
- 3 Remain static after deployment, cannot adopt to evolving threat landscape.

Requirement Gathering Methodology

1

Literature Review: To understand the limitations of current ML intrusion detection and validate the feasibility of using unsupervised learning and autoencoders

2

Analysis of Existing Systems: To identify industry-standard features (e.g., Real-time dashboards, Anomaly Scoring) and ensure the FYP matches current "State of the Art" capabilities.

3

Domain Analysis: The initial scope was defined to address the specific problem of "unknown attack patterns" in LAN environments, limiting the scope to metadata analysis (privacy constraint) and specific protocols.

Project - Constraints

Timeframe: The core development is strictly limited to **2-3 months** as this is a simplified scope for a Final Year Project (FYP).

Computational Resources: The system must operate on **standard commodity hardware** (e.g., a high-spec laptop or standard server), rather than requiring specialized high-performance computing clusters.

Privacy: The system is constrained to analyzing **metadata only** (packet size, timestamps, headers). It **cannot** inspect or store packet payloads (content) to ensure user privacy.

Connectivity: The system operates as a passive monitor and is constrained by the network topology; it **requires** a specific configuration (Port Mirroring/SPAN) to actually "see" the traffic.

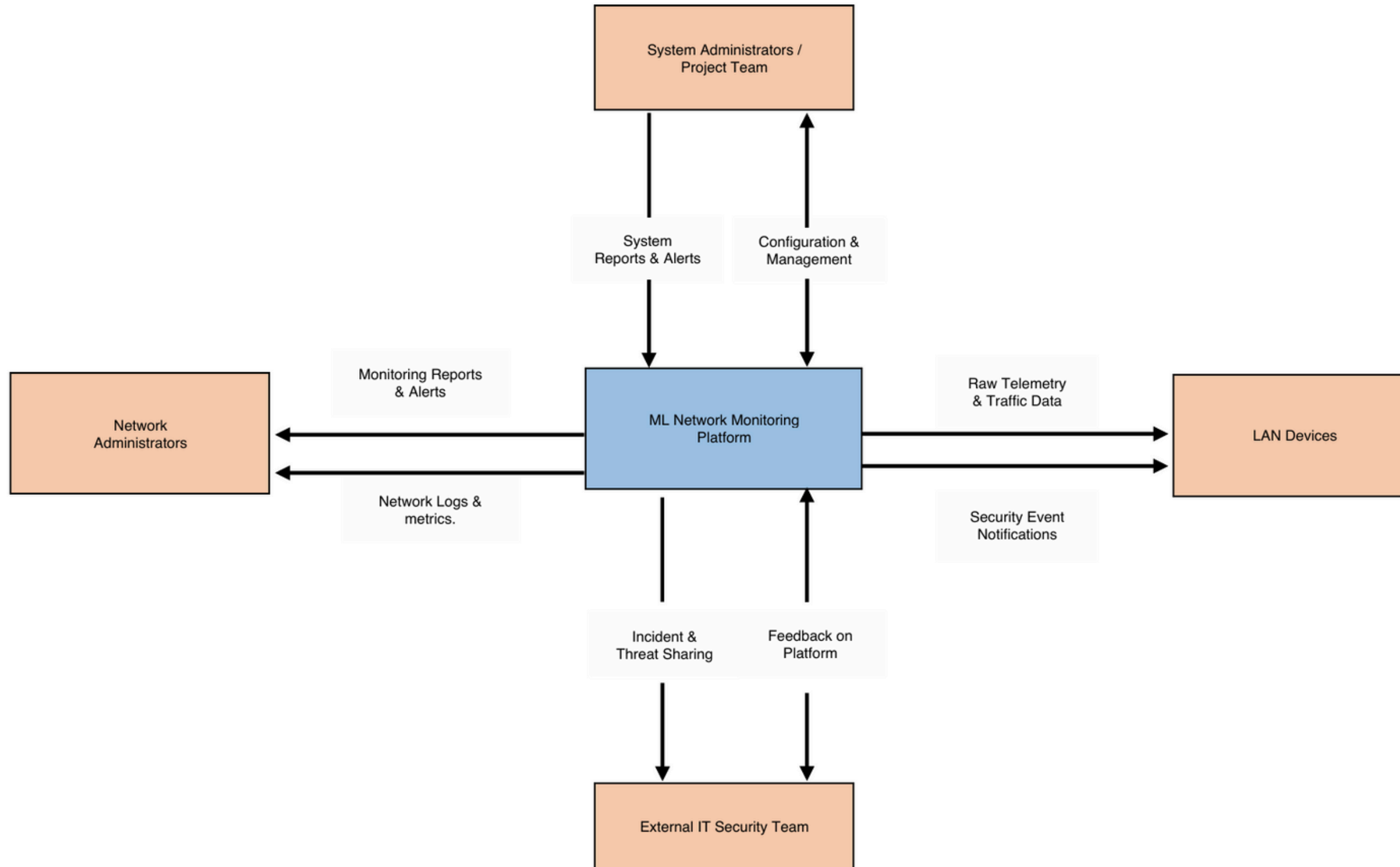
Project - Assumptions

Network Infrastructure: It is assumed that the existing network switch or router actually **supports SPAN or Port Mirroring**. Without this, the system cannot capture the necessary traffic.

Training Data: It is assumed that a "clean" dataset of **normal network traffic is available** (or can be captured) to train the initial Machine Learning baseline.

Dependencies: The project relies on the availability and stability of specific Python libraries (**Scikit-learn, Pandas**) and standard network drivers for packet capture

Context Diagram



Feature Set Matrix

Feature	Priority Of Development	Dependencies
Network Traffic Capture	High	
ML anomaly detection engine	High	
Data Storage & Management		
Alert Mechanism		
Web Dashboard		
User Management		

Distinguishing Functional Requirements

1. Network Traffic Ingestion

Real-Time Capture: Ingests LAN packets using Promiscuous Mode to monitor the entire segment.

Feature Extraction: Parses raw binary (TCP/UDP/ICMP) into ML-ready metadata (Packet Size, Flow Duration, IPs).

Vectorization: Converts network flows into numerical feature vectors for the ML model.

2. ML Detection Engine

Anomaly Scoring: Calculates a specific severity score (e.g., -1.0 to 1.0) for every packet/flow.

Dynamic Thresholding: Classifies traffic as "Anomalous" only when the score exceeds a configurable limit.

Real-Time Inference: Feeds live data into the model for immediate prediction, not post-event batching.

- **Feature 3: Data Storage & Management**
- **FR-10:** The system shall store detailed records of all detected anomalies (Time, IP, Score) in a dedicated "Incidents" table.
- **Feature 4: Alert Mechanism**
- **FR-12:** The system shall send an email notification to the administrator specifically containing the **Source IP and Anomaly Score** upon detection.
- **Feature 5: Web Dashboard & Visualization**
- **FR-14:** The dashboard shall display a real-time line graph explicitly showing **traffic volume and anomaly spikes**.

Distinguishing Non-Functional Requirements

1. Detection Latency (Real-Time Processing)

Metric: Anomaly scoring must complete within **1 second** of data ingestion.

Why it's distinguishing: Critical for an IDS; delays render security alerts useless.

2. Resource Efficiency (Lightweight Agent)

Metric: The system agent must utilize $\leq 10\%$ of **CPU** resources on the host machine.

Why it's distinguishing: As a background monitoring process, it cannot disrupt the primary function of the server/network it is protecting.

3. Dashboard Responsiveness

Metric: Page load times must be **< 3 seconds** under normal load.

Why it's distinguishing: Ensures the administrator can view "Network Health" without lag during an active incident.

User Interface Requirements

Web Dashboard (Main Interface)

- **Requirement:** Must be a "clean, responsive interface"
- **Visual Elements Required:**

Traffic Graph: A real-time line graph plotting **Traffic Volume vs. Anomaly Spikes**

Incident List: A table or list displaying recent anomalies

System Status: Indicators of "Network Health" (e.g., Monitoring Active)

- **Access:** Accessible via Chrome/Edge

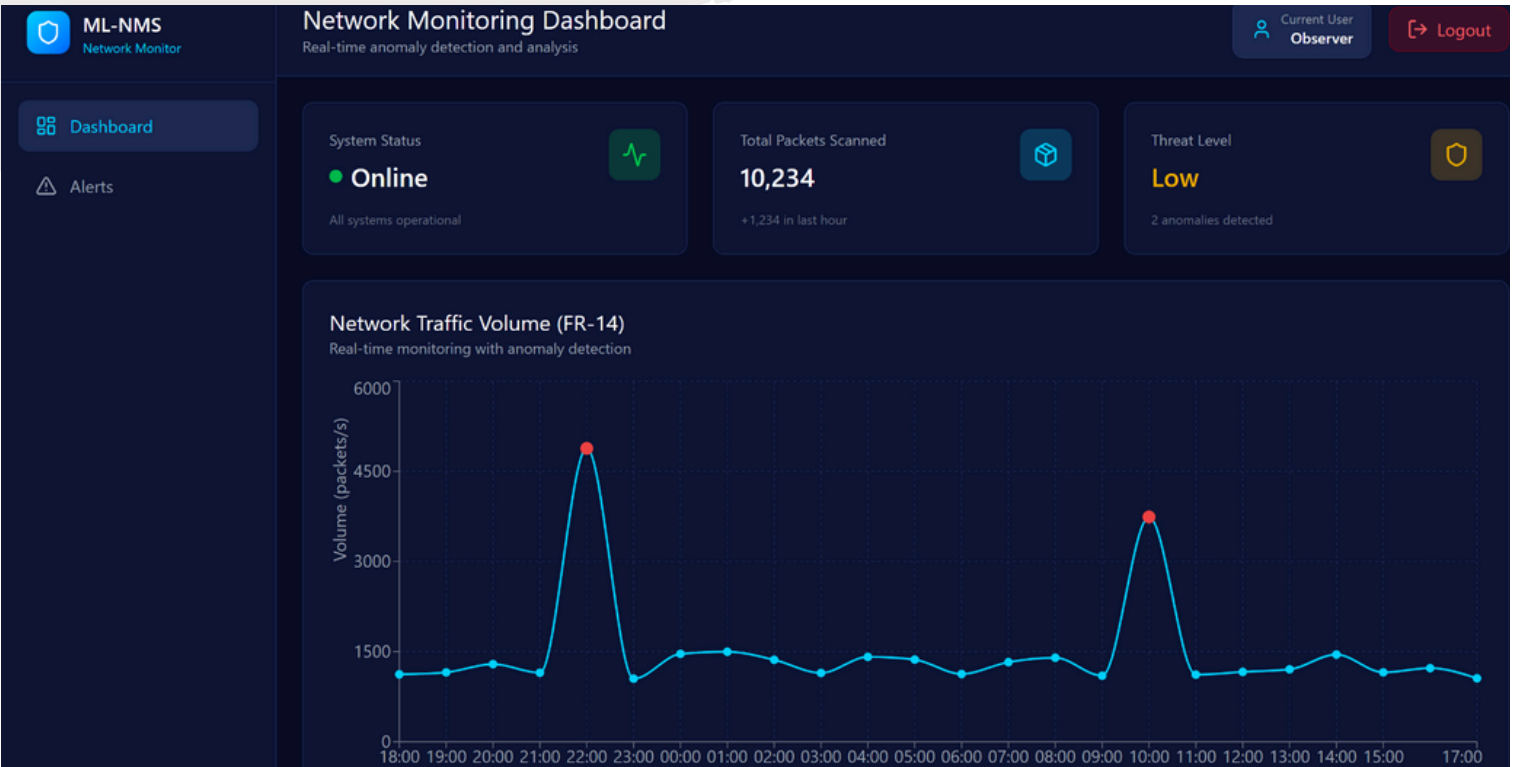


Figure: Network Monitoring Dashboard

Network Incidents (FR-10)					
Detection alerts and events					
Time	Source IP	Destination IP	Protocol	Anomaly Score	Severity
-19 10:00:23	192.168.1.50	8.8.8.8	TCP	0.95	Critical
-19 09:45:12	10.0.0.15	1.1.1.1	UDP	0.87	Critical
-19 09:30:45	192.168.1.100	192.168.1.1	TCP	0.45	Medium
-19 09:15:33	172.16.0.50	185.125.190.36	ICMP	0.23	Medium
-19 09:00:18	192.168.1.75	208.67.222.222	UDP	0.12	Low
-19 08:45:05	10.0.0.25	216.58.214.206	TCP	-0.15	Low
recent incidents					

Figure: Network Incident List

User Interface Design

Web Interface: Secure Access & Authentication

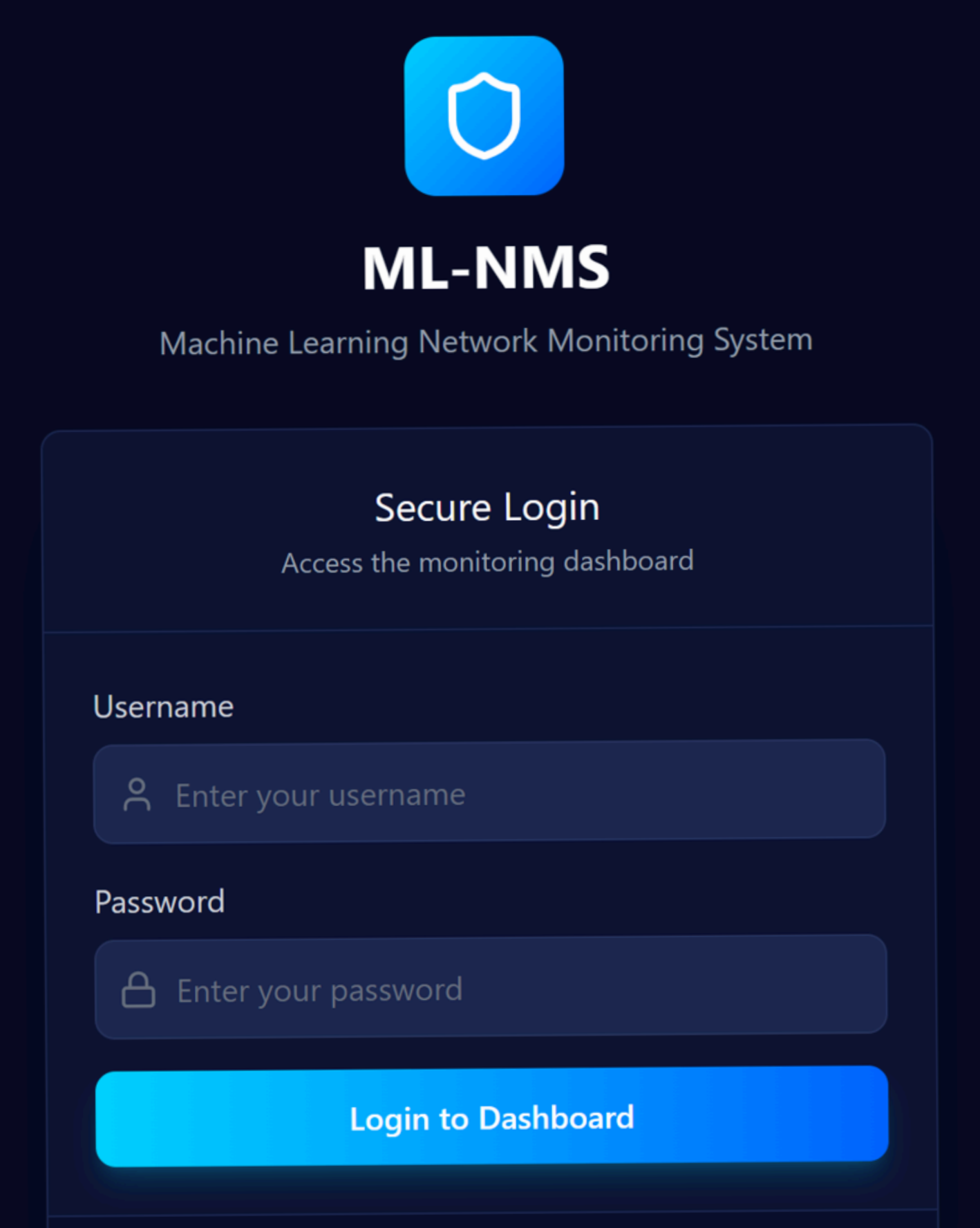
Key Features

Mandatory Authentication : The system enforces a strict "Login First" policy to prevent unauthorized personnel from viewing sensitive network traffic data.

Role-Based Gateway : This interface acts as the decision point for **RBAC**. Upon login, the backend determines if the user is an "Administrator" (Full Access) or "Observer" (Read-Only) and renders the appropriate dashboard view.

Security-Centric Design: Designed with a "Dark Mode" aesthetic standard in Security Operations Centers (SOCs) to reduce eye strain during long monitoring sessions.

Modern Tech Stack: Built using **React.js** for a responsive, lag-free experience, ensuring quick access to the monitoring tools



The image displays a user interface for a system named ML-NMS (Machine Learning Network Monitoring System). At the top, there is a blue shield icon inside a rounded square. Below the icon, the text "ML-NMS" is written in a large, bold, white font, followed by "Machine Learning Network Monitoring System" in a smaller, lighter font. The main section of the interface is titled "Secure Login" in a bold white font, with the subtitle "Access the monitoring dashboard" below it. This section contains two input fields: "Username" and "Password". Each field has a label and a placeholder text "Enter your username" and "Enter your password" respectively, accompanied by a small icon (a person for username and a lock for password). At the bottom of the login section is a large, bright blue button with the text "Login to Dashboard" in white.

System Architecture

Use Case Diagram

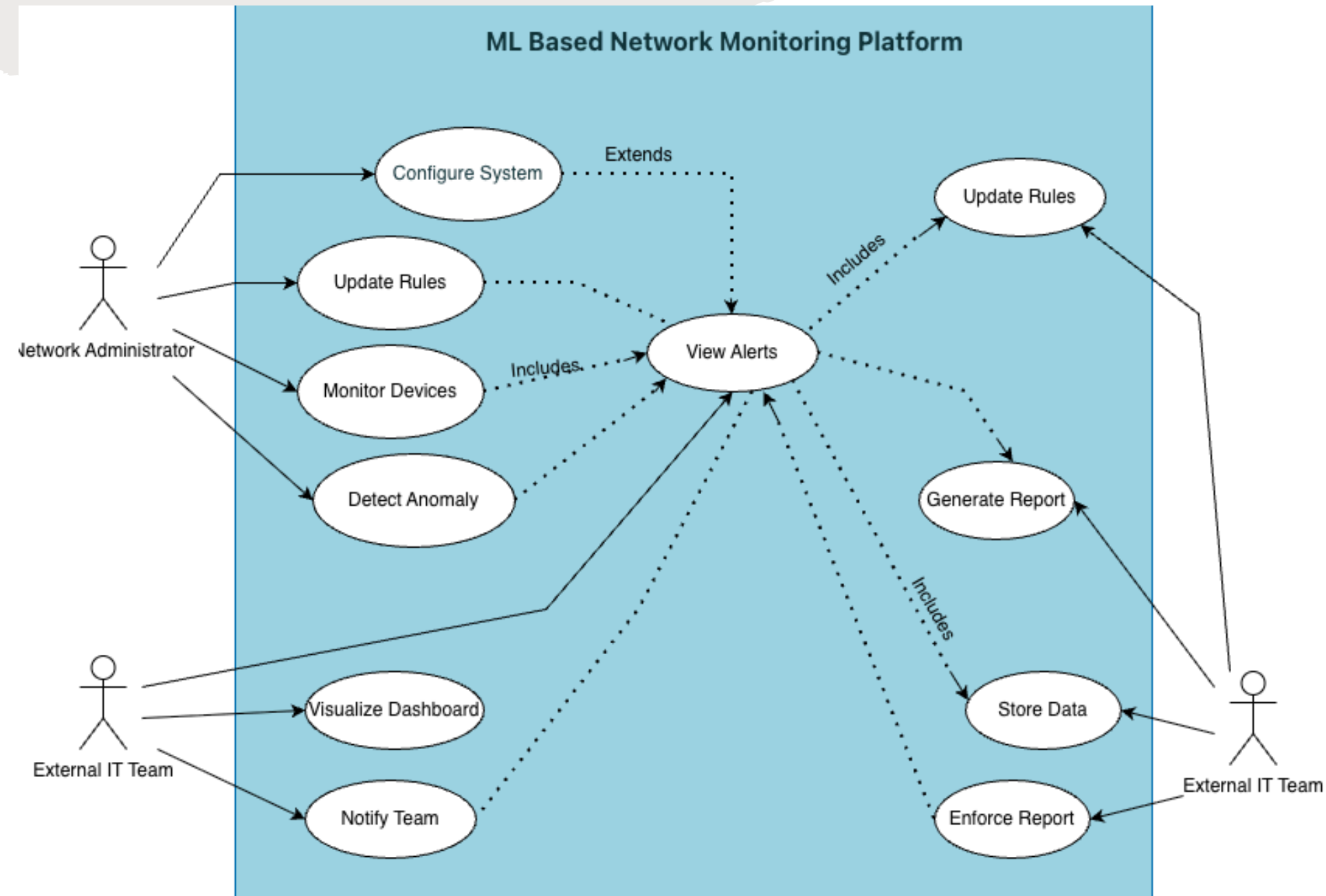


Figure: Use Case Diagram

Use Case Diagram

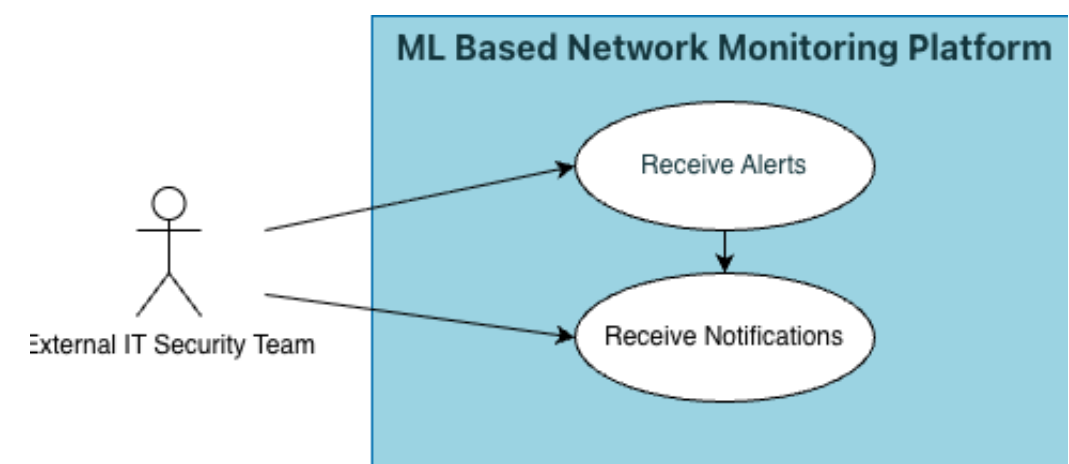


Figure: External IT Team Use Case

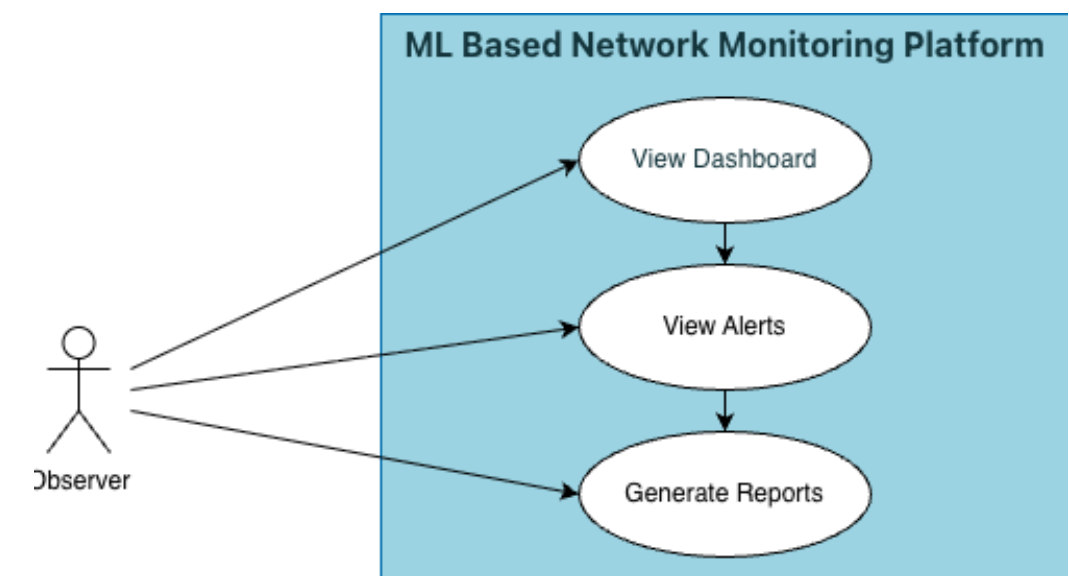


Figure: Observer Use Case

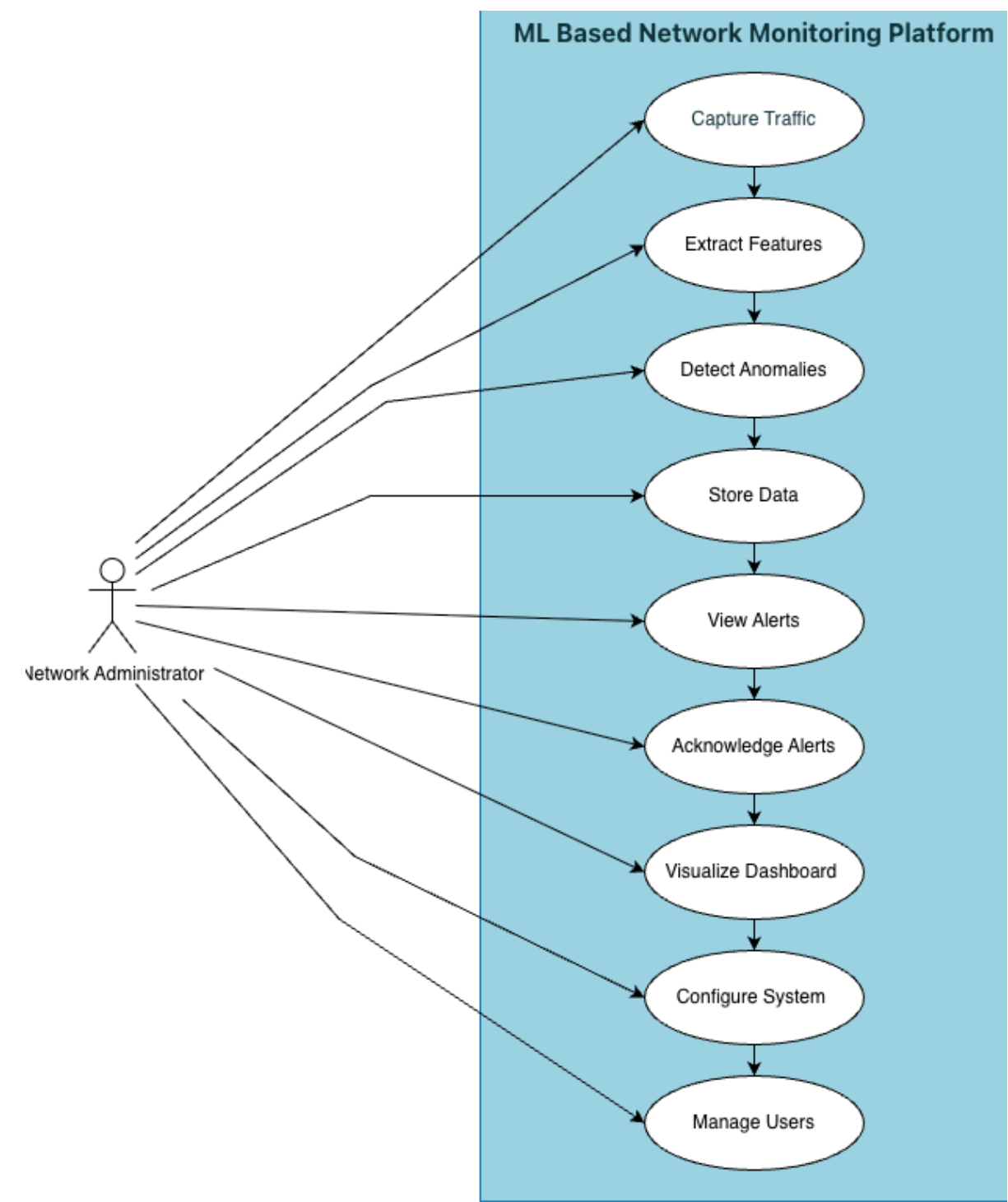


Figure: Network Admininterator Use Case

Class Diagram

Modular Design: The system is decoupled into distinct modules for capturing (PacketCapture), processing (FeatureExtractor), and analysis (MLEngine) to ensure scalability.

Centralized Control: The SystemController acts as the orchestrator, managing the lifecycle of all core components using the Composition pattern.

Role-Based Security: Implements a secure hierarchy (User inheritance) to distinguish between Administrators (configuration access) and Observers (read-only access).

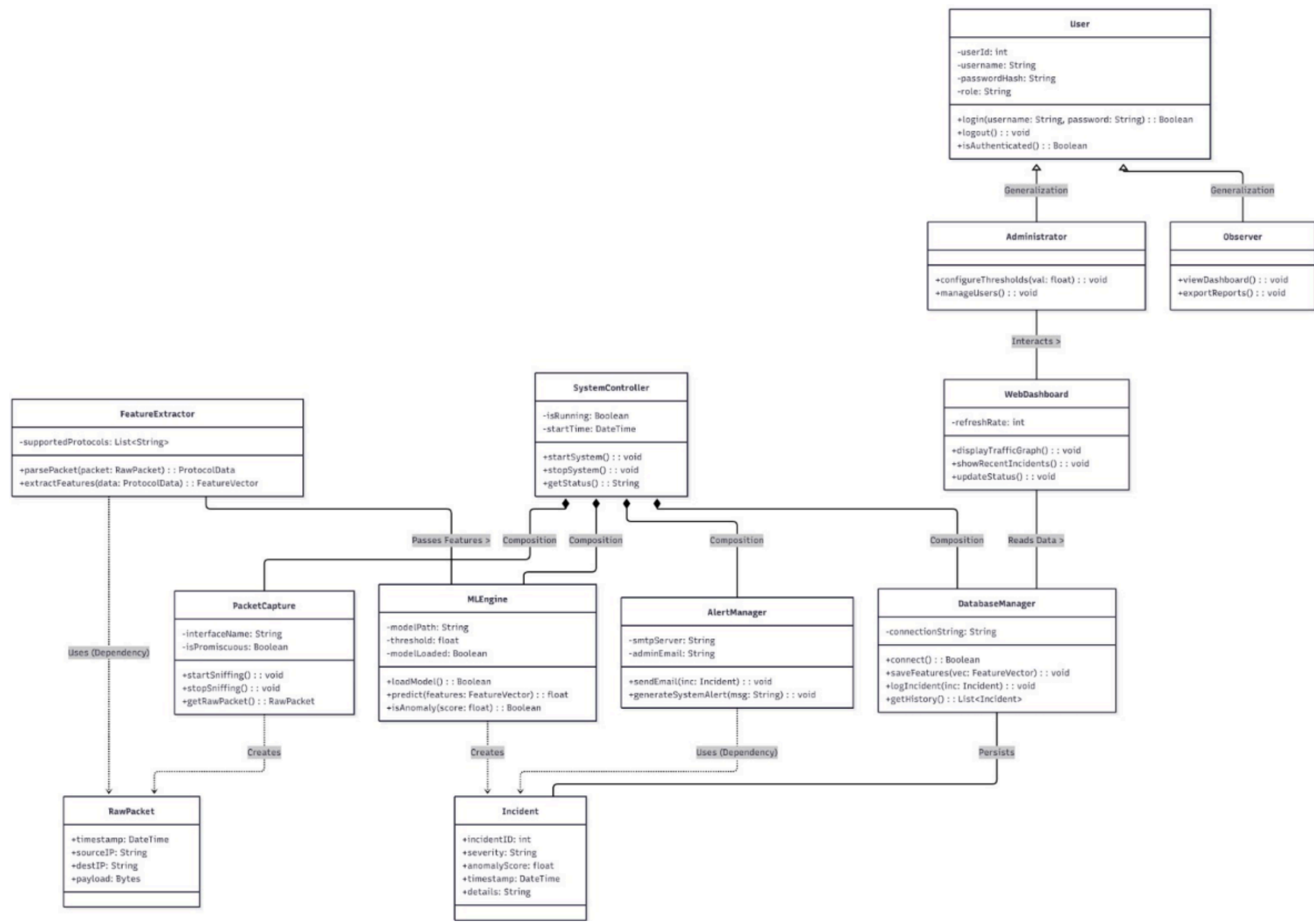
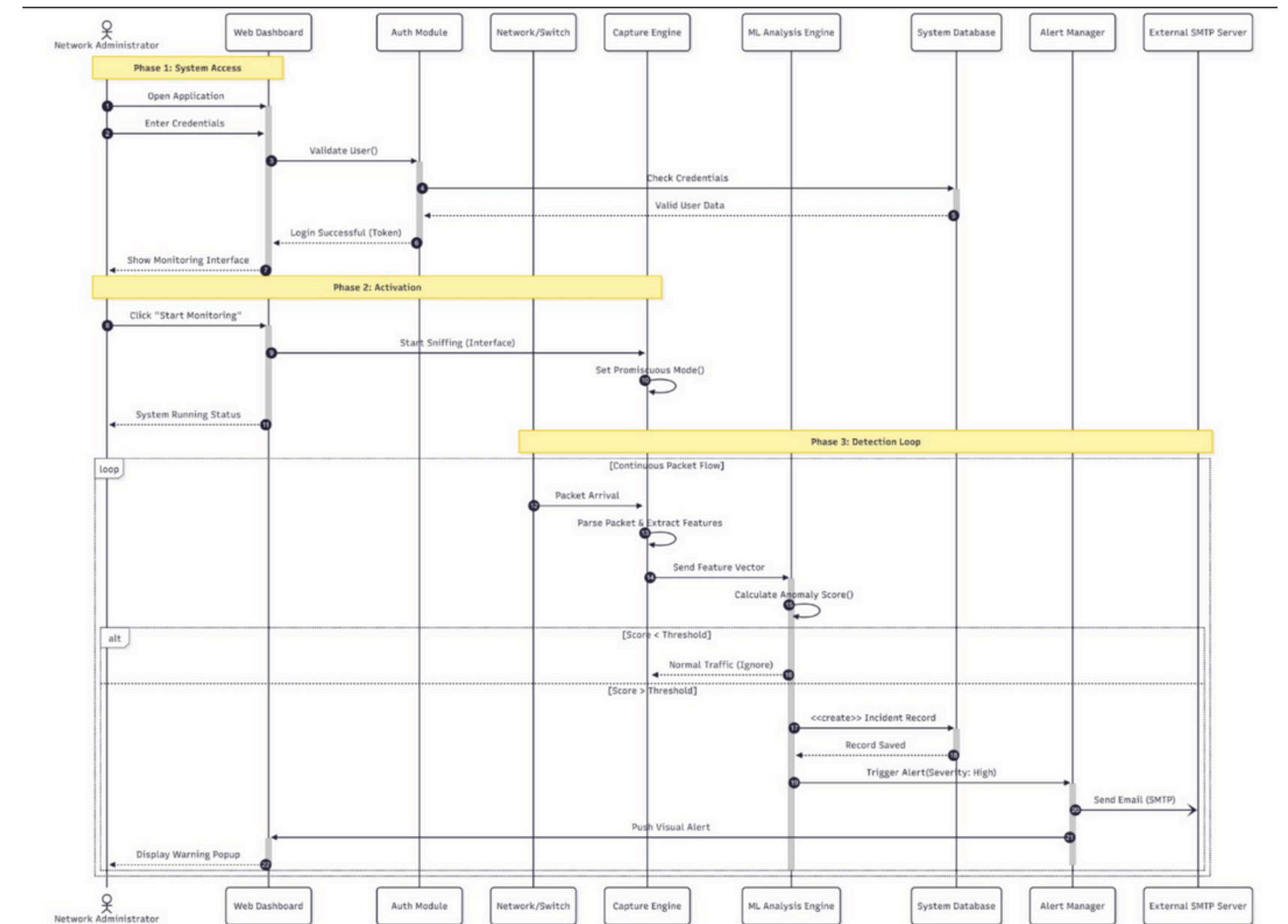


Figure: Class Diagram

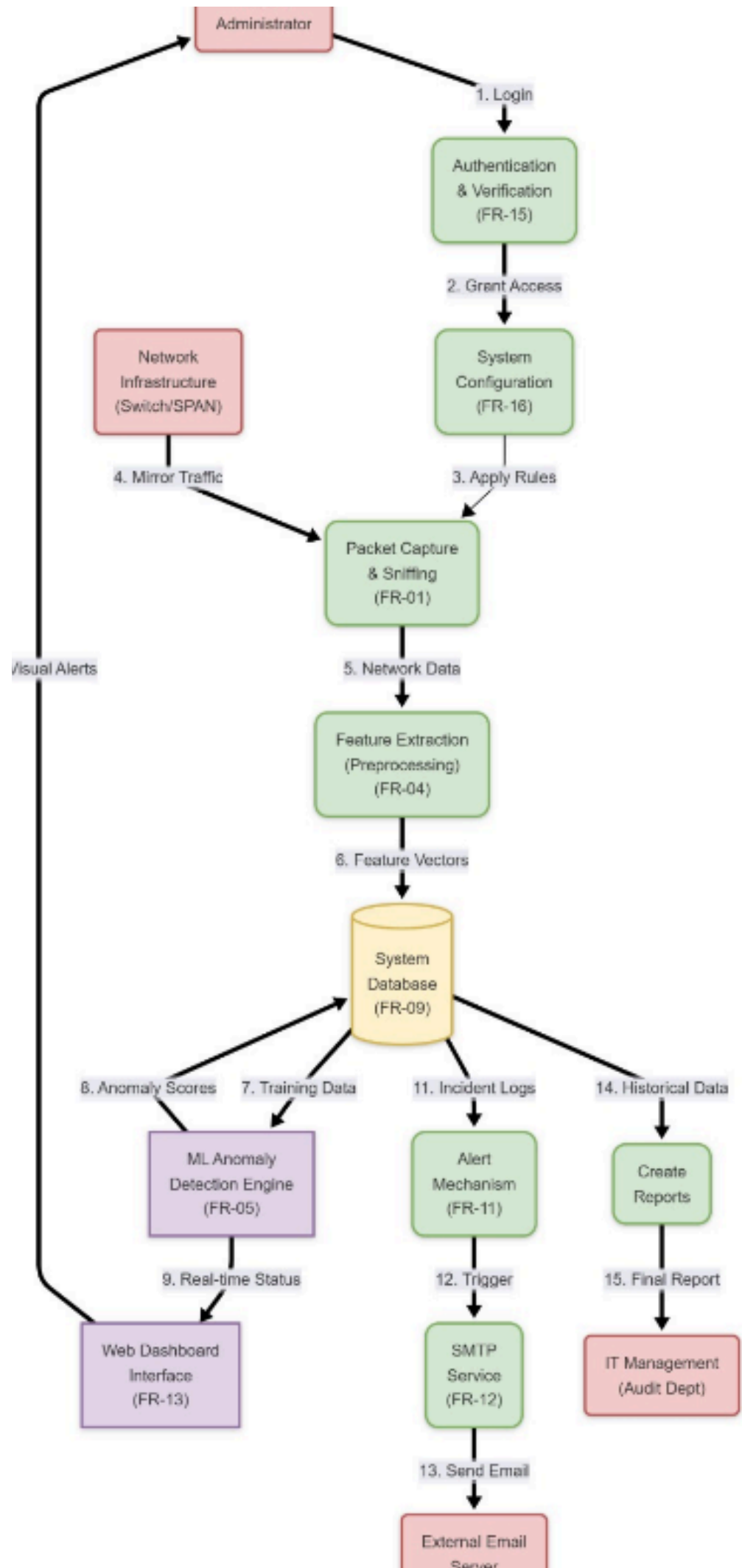
Sequence Diagram

Phase-Based Approach (Matches the yellow labels)

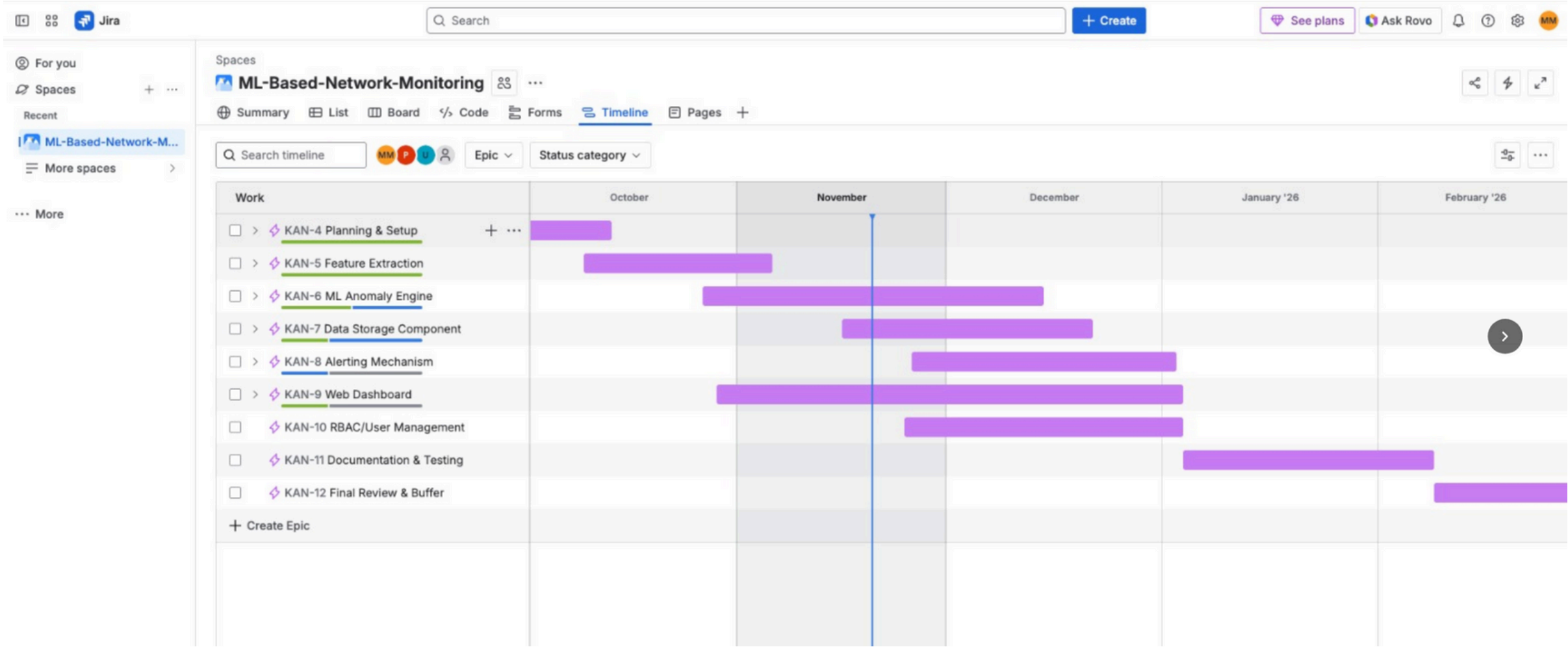
- Phase 1: Access Control: Secure authentication flow where the Auth Module validates credentials against the System Database before granting dashboard access.
- Phase 2: Initialization: Upon administrator command, the Capture Engine activates Promiscuous Mode to begin sniffing all network traffic.
- Phase 3: Real-Time Detection: A continuous loop where packets are parsed and sent to the ML Engine.
 - Normal Traffic: Ignored to save resources.
 - Anomalies: Triggers a 3-step response: Log to DB, Send Email (SMTP), and Push Visual Alert



Data Flow Diagram



Jira Dashboard



Thank You

FOR LISTENING.