PROJECT REPORT

*On*

**AI Powered Network Intrusion Detection System**

*Submitted by*

Krisha Vanpariya (IU2241220015)

Mansi Patel (IU2241220014)

Vatsal Patel (IU2241220028)

Priyanshi Trivedi (IU2241220037)

Krishi Patel (IU2241220077)

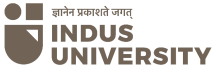
*In fulfillment for the award of the degree*

*Of*

*BACHELOR OF TECHNOLOGY*

*In*

*Information Technology*

**

**INSTITUTE OF TECHNOLOGY AND ENGINEERING**

**INDUS UNIVERSITY CAMPUS, RANCHARDA, VIA-THALTEJ**

**AHMEDABAD-382115, GUJARAT, INDIA,**

WEB: www.indusuni.ac.in

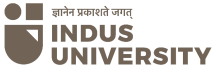
MAY-2025

**PROJECT REPORT**

ON

**AI Powered Network Intrusion Detection System**

AT



In the fulfilment of the requirement

for the degree of

Bachelor of Technology

in

Information Technology

**PREPARED BY**

Krisha Vanpariya (IU2241220015)

Mansi Patel (IU2241220014)

Vatsal Patel (IU2241220028)

Priyanshi Trivedi (IU2241220037)

Krishi Patel (IU2241220077)

**UNDER GUIDANCE OF**

**Internal Guide**

Ms. Sheetal Panchal

Assistant Professor

Department of Information

Technology,

I.I.T.E. Indus University,

Ahmedabad

**SUBMITTED TO**

INSTITUTE OF TECHNOLOGY AND ENGINEERING

INDUS UNIVERSITY CAMPUS, RANCHARDA, VIA-THALTEJ

AHMEDABAD-382115, GUJARAT, INDIA,

WEB: [www.indusuni.ac.i](http://www.indusuni.ac.in/)n

MAY 2025

**CANDIDATE’S DECLARATION**



I declare that 7th semester report entitled “**AI Powered Network Intrusion Detection System**” is my own work conducted under the supervision of the guide **Ms. Sheetal Panchal**.

I further declare that to the best of my knowledge, the report for B. Tech 7th semester does not contain part of the work which has been submitted for the award of B. Tech Degree either in this university or any other university without proper citation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Candidate’s Signature

Krisha Vanpariya (IU2241220015)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Guide: Ms. Sheetal Panchal

Assistant Professor,

Department of Information Technology,

Indus Institute of Technology and Engineering

INDUS UNIVERSITY– Ahmedabad,

State: Gujarat

**INDUS INSTITUTE OF TECHNOLOGY AND ENGINEERING INFORMATION TECHNOLOGY**

**2024 -2025**



**CERTIFICATE**

**Date:26/04/2025**

This is to certify that the project work entitled “**AI Powered Network Intrusion Detection System”** has been carried out by **Krisha Vanpariya** under my guidance in partial fulfillment of degree of Bachelor of Technology in **Information Technology (7Th semester)** of Indus University, Ahmedabad during the academic year 2025 – 2026.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ms. Sheetal Panchal

Assistant Professor,

Department of Information Technology,

I.T.E. , Indus University

Ahmedabad

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Shruti B Yagnik

Head of the Department,

Department of Information Technology,

I.T.E. , Indus University

Ahmedabad

**ACKNOWLEDGEMENT**



I would like to express my sincere gratitude to all those who have contributed to the successful completion of my Bachelor of Technology Project.

Firstly, I would like to thank my project guide, Ms. Sheetal Panchal, for their guidance and support throughout the project. Their valuable insights and suggestions have been instrumental in shaping my work.

I am also grateful to the faculty members of the department of information technology engineering for their support and encouragement. Their feedback and suggestions have been invaluable in improving the quality of my work.

I would like to express my gratitude to my family and friends for their unwavering support and encouragement throughout the project. Their constant motivation has been a source of inspiration for me.

Lastly, I would like to thank the university for providing me with the necessary resources and facilities to complete my project successfully.

Thank you all for your support and guidance.

Krisha Vanpariya

IU2241220015

Information Technology

**TABLE OF CONTENT**

**Title** **Page No**

[**ABSTRACT**](#_ji68gcb5m97l) **8**

[**LIST OF FIGURES**](#_adpqhgc0dzmy) **9**

[**LIST OF TABLES**](#_fo11qrz9x041) 10

[**ABBREVIATION**](#_jzpo3gpfzyk4) 11

[**CHAPTER 1 INTRODUCTION 12**](#_83fzs86x49wc)

[1.1 Project Summary 12](#_xkaeokfrw14o)

[1.2 Project Purpose 12](#_j3zw2qf7976f)

[1.3 Project Scope 13](#_w88nzwdtwoio)

[1.4 Objectives 14](#_70vdnovgezfx)

[1.5 Technology And Tools 14](#_clqn0a5vv3f2)

[1.5.1 USER Interface And Visualization 14](#_o8jouziqp03p)

[1.5.2 Core Procssing And Machine Learning 15](#_q6y262vxkcj3)

[1.5.3 Database 15](#_o6bv9c734fxc)

1.5.4 Notification And Alerting 15

[1.6 Synopsis 16](#_ik9sw579rl6u)

[**CHAPTER 2 LITERATURE SURVEY 17**](#_ica3d9pm0ake)

[2.1 Introduction Of Survey 17](#_y0yddmz2zirr)

[2.2 Why Survey ? 17](#_lkomaq6yz1x7)

[**CHAPTER 3 PROJECT MANAGEMENT 19**](#_qj5kl1b7h1b4)

[3.1 Project Planning Objectives 19](#_luf2v8muz5t2)

[3.2 Project Scheduling 20](#_wdw78gwnwfpl)

[3.2.1 Project Organization 21](#_q5egaqxbdo4q)

[3.2.2 Timeline Chart 22](#_jruboyqq9ba6)

[3.2.2.1 Time Allocation 22](#_euzgsuiq8o9i)

[3.2.2.2 Task Sets 22](#_boyagnr81ujk)

[3.3 Risk Management 23](#_qqinjmycd5v6)

[**CHAPTER 4 SYSTEM REQUIREMENTS 24**](#_69gi98hdgpps)

[4.1 User Characteristics 24](#_3w7qandso6hp)

[4.2 Functional Requirement 24](#_p645oa7xu2vj)

[4.3 Non Functional Requirement 25](#_5m8p906c5yhp)

[4.4 Hardware And Software Requirement 26](#_pt39ov1jkwot)

[4.4.1 Hardware Requirement 26](#_p65sm66g1eda)

[4.4.2 Software Requirement 26](#_qi3fx0ifcqrs)

[**CHAPTER 5 SYSTEM ANALYSIS 27**](#_s04vz0tm591)

[5.1 Study Of Current System 27](#_504ijpfrp9xe)

[5.2 Problems In Current System 27](#_dermf9v5qfxr)

[5.3 Requirement Of New System 28](#_64ignw7soonu)

[5.4 Process Model 28](#_xx3tph8sm5gv)

[5.5 Feasibility Study 29](#_xfhqvfb90ldw)

[5.5.1 Technical Feasibility 29](#_48ub859cx9vt)

[5.5.2 Operational Feasibility 29](#_9hno15l22bk1)

[5.5.3 Economical Feasibility 29](#_6wnf6ao1hl6t)

[5.5.4 Schedule Feasibility 29](#_5rbphh91kp5)

[**CHAPTER 6 DETAIL DESCRIPTION 30**](#_rg2h739v1sns)

[6.1 User](#_x8sfumv1pl1a) Interface Module [30](#_x8sfumv1pl1a)

[6.2 Control Pannel 32](#_x542wetlqe68)

6.3 Alerts 34

6.4 Database 35

[**CHAPTER 7 TESTING 36**](#_46untq129mom)

[7.1 Black-Box Testing 36](#_ldob81wgr3wf)

[7.2 White-Box Testing 36](#_tk336du947jz)

[7.3 Test Cases 37](#_mqdzh8ngrnd7)

[**CHAPTER 8 SYSTEM DESIGN 39**](#_n4pxce6vgldk)

[8.1](#_59vmn06qz52e) System Architecture [39](#_59vmn06qz52e)

[8.2 Use-Case Diagram 40](#_gdxdzkrl9x7e)

[8.3 Sequence Diagram 40](#_krjiolleb5ld)

[8.4 Activity Diagram 41](#_dxj7dnb8ejjr)

[8.5 Er Diagram 41](#_imv6bm9fmt6z)

[**CHAPTER 9 LIMITATION AND FUTURE ENHANCEMENTS 43**](#_4noxiy2qeiy4)

[9.1 Limitation 43](#_jmh0lbnbj99z)

[9.2 Future Enhancement 43](#_d680sru8mcjd)

[**CHAPTER 10 CONCLUSION 45**](#_e4qkn6ing4r3)

[10.1 Conclusion 45](#_a2tgvzbfpc7p)

[**BIBLIOGRAPHY 46**](#_73f2eqhor7ig)

# ABSTRACT



*The proliferation of network-based services has made network security a critical concern for individuals and organizations alike. Traditional Intrusion Detection Systems (IDS) often rely on signature-based methods, which are ineffective against novel and zero-day attacks. This project presents the design and implementation of an* ***AI-Powered Network Intrusion Detection System (NIDS)*** *that leverages machine learning to detect anomalous network traffic in real-time.*

*The system is built using Python and employs the Scapy library for packet capture and analysis. A pre-trained machine learning model is used to classify network packets as either normal or anomalous. The core of the system lies in its ability to perform real-time feature extraction from live network traffic and feed it to the AI model for prediction.*

*Upon detection of an anomaly, the system logs the event into a SQLite database and triggers multi-channel alerts, including desktop notifications and emails, to notify the system administrator. A comprehensive and user-friendly dashboard, built with the Streamlit framework, provides a real-time view of network traffic, highlights detected anomalies, and displays historical alert data. The project also includes an attack simulation script to test and validate the NIDS's detection capabilities. This anomaly-based approach provides a more robust and adaptive security solution capable of identifying previously unseen threats*

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| Fig. 3.2.3.1 | Time Allocation Chart | 16 |
| Fig. 5.4.1 | Spiral Model | 26 |
| Fig. 6.1.1 | Anomaly Detection Matrix | 29 |
| Fig. 6.1.2 | Visualization | 29 |
| Fig. 6.1.3 | Anomaly Details | 30 |
| Fig. 6.1.4 | Recent Alerts | 31 |
| Fig. 6.1.5 | Database Stats | 31 |
| Fig. 6.1.6 | CSV Upload | 31 |
| Fig. 6.2.1 | Mode Selection | 32 |
| Fig. 6.2.2 | Email Configuration | 32 |
| Fig. 6.2.3 | Steps to set Password | 32 |
| Fig. 6.2.4 | Time Duration Setting | 33 |
| Fig. 6.2.5 | Threshold Setting | 33 |
| Fig. 6.3.1 | Desktop Notification | 33 |
| Fig. 6.3.2 | Mail Alerts | 34 |
| Fig. 6.3.4 | Analysis Report | 34 |
| Fig. 6.4.1 | Confirming Payment Page (Razorpay) | 34 |
| Fig. 6.4.2 | Payment Success Page (Razorpay) | 35 |
| Fig. 8.1 | System Archtiecture | 49 |
| Fig. 8.2 | Use-Case Diagram | 50 |
| Fig. 8.3 | Sequence Diagram | 51 |
| Fig. 8.4 | Activity Diagram | 52 |
| Fig. 8.5.1 | ER Diagram Symbols | 53 |

# 

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| Table 3.2.3.2 | Task Sets | 17 |
| Table 3.3.2 | Risk Projection | 18 |
| Table. 7.3.1 | Test cases | 44 |

# Abbreviation

Abbreviations used throughout this whole document for Survey Application are:

|  |  |
| --- | --- |
| **Abbreviation** | **Full Form** |
| **AI** | Artificial Intelligence |
| **NIDS** | Network Intrusion Detection System |
| **ML** | Machine Learning |
| **GUI** | Graphical User Interface |
| **DB** | Database |
| **IP** | Internet Protocol |
| **TCP** | Transmission Control Protocol |
| **UDP** | User Datagram Protocol |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **DNS** | Domain Name System |
| **SQL** | Structured Query Language |
| **CSV** | Comma-Separated Values |
| **OS** | Operating System |

# Chapter 1 -Introduction

* **PROJECT SUMMARY**
* **PROJECT PURPOSE**
* **PROJECT SCOPE**
* **OBJECTIVES**
* **TECHNOLOGY AND TOOLS**
* **SYNOPSIS**

## 1.1 PROJECT SUMMARY

The **AI-Powered Network Intrusion Detection System** is a modern cybersecurity solution engineered to protect computer networks from the ever-growing landscape of digital threats. Traditional security systems, which often rely on static, signature-based methods, frequently fail to detect new or unknown attacks. This project addresses that critical vulnerability by developing an intelligent NIDS that leverages the power of  **Artificial Intelligence (AI)** to perform deep analysis of network traffic.

The system operates in real-time through an end-to-end pipeline that captures raw network packets, transforms them into numerical feature sets, and feeds them into the AI for classification. A user-friendly Graphical User Interface (GUI) provides administrators with immediate, actionable alerts and data visualizations, enabling a swift and effective response to potential security breaches.

## 1.2 PROJECT PURPOSE

The primary purpose of this project is to create a more  **intelligent and adaptive**  security system that can overcome the significant limitations of traditional intrusion detection. In an era where cyberattacks are increasingly sophisticated and automated, there is a pressing need for proactive security measures that can anticipate and identify threats before they cause significant damage.

Traditional signature-based systems are fundamentally reactive; they can only identify threats that have already been discovered and cataloged. This leaves networks dangerously exposed to novel threats, including:

* **Zero-Day Exploits:** Attacks that target a previously unknown software vulnerability.
* **Polymorphic Malware:** Malicious code that constantly changes its signature to evade detection.
* **Advanced Persistent Threats (APTs):** Sophisticated, long-term attacks that are difficult to spot with conventional tools.

This project's purpose is to leverage machine learning's ability to perform complex pattern recognition. By learning the subtle statistical properties of normal network behavior, the AI model can identify anomalous activities that are indicative of a new or disguised attack, thus providing a **proactive and predictive** layer of security. The goal is to build a system that not only improves detection rates but also reduces the "alert fatigue" caused by the high number of false positives in simpler anomaly-detection systems.

## 1.3 PROJECT SCOPE

The scope of this project is to design, develop, and test a fully functional NIDS prototype. The system's boundaries and core functionalities are defined below.

* **Real-time Packet Capture:**  The system is designed to monitor IPv4 traffic on a selected network interface and analyze common transport layer protocols such as TCP and UDP.
* **Data Processing and Feature Extraction**: It will extract a wide range of features from packets, including header information, payload size, and traffic flow statistics.
* **AI-Based Detection:**  The AI model is trained to identify a specific set of attack categories present in the training dataset, including Denial-of-Service (DoS), network scanning, and other common intrusion patterns.
* **Alerting and Logging**: The system will generate detailed alerts for any detected threats and log the event details into a local SQLite database for forensic analysis.
* **User Interface:**  A standalone desktop GUI will be provided for system control, real-time monitoring, and basic data visualization.

## 1.4 OBJECTIVES

The following are the specific, detailed objectives set for the successful completion of this project:

1. To design and implement a robust and efficient software module for capturing live network packets from a specified interface without significant packet loss under moderate network load.

2. To engineer a comprehensive data preprocessing pipeline capable of extracting meaningful statistical and time-based features from raw packet data, transforming it into a format suitable for machine learning algorithms.

3. To train, evaluate, and integrate a machine learning model that achieves a high degree of accuracy in classifying network traffic and maintains a low false positive rate.

4. To develop an intuitive and responsive Graphical User Interface (GUI) that allows a non-expert user to easily operate the system, monitor network status, and understand security alerts.

5. To create a persistent and reliable logging mechanism using a lightweight database to store all detected security events for auditing and post-incident analysis.

6. To systematically validate the end-to-end performance and detection accuracy of the system by creating and executing scripts that simulate a variety of common network attacks.

## 1.5 TECHNOLOGY AND TOOLS

### 1.5.1 USER INTERFACE AND VISUALIZATION

* **Streamlit:** Streamlit is an open-source Python library for building and sharing custom web apps for machine learning and data science. In this project, it is used to create the entire interactive user interface, including the real-time dashboard, control buttons, and data tables.
* **Plotly Express:** Plotly is a graphing library used to create interactive, publication-quality graphs. It is used in this system to generate the real-time pie chart that visualizes the distribution of normal versus anomalous network traffic.

### 1.5.2 CORE PROCESSING AND MACHINE LEARNING

* **Python:** Python is a high-level, general-purpose programming language that serves as the foundation for the entire project. Its extensive ecosystem of libraries makes it ideal for network analysis, data science, and rapid application development.
* **Scapy:** Scapy is a powerful Python-based interactive packet manipulation program and library. It is the core tool used for capturing, dissecting, and analyzing live network packets.
* **Pandas:** Pandas is a software library written for data manipulation and analysis. It is used to structure the features extracted from network packets into DataFrames, which is the required format for processing by the machine learning model.
* **Scikit-learn:** Scikit-learn is a free software machine learning library for Python. It is used in this project to load and utilize the pre-trained anomaly detection model to classify network traffic in real-time.

### 1.5.3 DATABASE

* **SQLite3:** SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. It is used as the database for the system to persistently log all detected anomalies and system alerts for historical review and analysis.

### 1.5.4 NOTIFICATIONS AND ALERTING

* **Plyer:** Plyer is a Python library for accessing features of your hardware, such as the notification system. It is used to provide cross-platform desktop notifications to alert the administrator of potential threats.
* **smtplib:** This is Python's built-in module for sending email using the Simple Mail Transfer Protocol (SMTP). It is used to implement the email alerting functionality of the system.

## 1.6 SYNOPSIS

Traditional Intrusion Detection Systems struggle to identify new, zero-day cyber threats. This project introduces an **AI-Powered Network Intrusion Detection System (NIDS)** that uses machine learning to overcome this limitation.

Built in Python, the system uses Scapy to capture live network traffic and a pre-trained model to analyze it in real-time. By learning the patterns of normal network activity, it can accurately identify anomalous behavior indicative of an attack. When a threat is detected, the system sends immediate desktop and email alerts, logs the event to a database for analysis, and displays all information on an intuitive web dashboard created with Streamlit. This provides a proactive and adaptive solution for modern network security.

# Chapter 2 - Literature Survey

* **INTRODUCTION OF SURVEY**
* **WHY SURVEY?**

**2.1 INTRODUCTION OF SURVEY**

Central to network defense is the Intrusion Detection System (IDS). The most common approach, **signature-based detection**, maintains a database of known attack patterns, or "signatures," and raises an alert when a match is found. While reliable for documented threats, its fundamental flaw is its inability to detect new or "zero-day" attacks for which no signature exists. To address this, research shifted towards **anomaly-based detection**, which learns a model of "normal" network behaviour and flags any significant deviation as a potential threat. Early statistical versions of this method often struggled with high false-positive rates, as legitimate but unusual traffic could be misclassified.

The most promising evolution in anomaly detection has been the integration of **Machine Learning (ML) and Artificial Intelligence (AI)**. These advanced techniques move beyond simple statistics to learn the intricate, high-dimensional patterns of normal network communication. By training on large datasets, ML models can develop a more nuanced and accurate understanding of network behaviour. This survey explores the landscape of these AI-powered systems to understand the current state-of-the-art and create a foundation for the next generation of intelligent, proactive network defense.

## 

## 2.2 WHY SURVEY ?

A comprehensive literature survey is a strategic necessity for this project. Its primary purpose is to understand the **state-of-the-art** in AI-driven intrusion detection by reviewing existing research to identify successful methodologies, common algorithms (like Random Forests or Deep Learning), and standard benchmark datasets. This establishes a baseline for evaluating our own system's performance. More critically, the survey is essential for identifying the **significant gap between theoretical research and practical applicability**. Many academic studies propose models with high accuracy on static datasets but fail to address real-world challenges like high-speed packet processing or the need for a functional user interface.

By highlighting this gap, the survey **justifies the contribution of this project**. It shows a clear need for a solution that bridges the divide between theoretical AI models and functional, user-centric security tools. This project's goal is to build a complete, end-to-end system that integrates an effective machine learning core with the practical features an administrator needs: a real-time dashboard, automated alerts, and persistent logging. The survey thus confirms that developing such a holistic and usable NIDS is a valuable contribution to the field.

# Chapter 3 -Project Management

* **PROJECT PLANNING OBJECTIVES**
* **PROJECT SCHEDULING**
* **RISK MANAGEMENT**

**3.1 PROJECT PLANNING OBJECTIVES**

The planning phase for the AI-Powered Network Intrusion Detection System is structured around a set of clear and measurable objectives. These objectives guide the development process, ensuring that the final system is functional, reliable, and meets all specified requirements. The successful completion of these objectives will result in a comprehensive and practical security tool. The key planning objectives for this project are as follows:

* **To Develop a Real-Time Packet Capture Module:** The initial objective is to implement a robust mechanism capable of sniffing and capturing network packets from a live network interface in real-time, forming the primary data source for the system.
* **To** **Integrate a Pre-Trained Machine Learning Model:** A core objective is to successfully load and integrate a pre-trained anomaly detection model that will serve as the brain of the system, responsible for classifying network traffic.
* **To Implement the Anomaly Detection Engine:** This involves creating the logic that preprocesses captured packets, extracts the necessary features, and feeds them into the machine learning model to get a real-time classification of "Normal" or "Anomalous."
* **To Design and Build an Interactive User Interface (GUI):** A critical objective is the development of a user-friendly graphical interface that allows an administrator to start/stop the capture process, view live statistics, and review detected threats.
* **To Create a Real-Time Alerting System:** The plan includes the implementation of a multi-channel notification system that can send immediate desktop notifications and email alerts to the administrator when a high threat level is detected.
* **To Establish a Persistent Logging Database:** An essential objective is to set up a database to log all detected anomalies and system alerts, providing a persistent record for historical analysis and forensic review.
* **To Validate System Efficacy with Simulated Attacks:** The final objective is to test and validate the complete system's detection capabilities by running a series of controlled, simulated network attacks to ensure its reliability

## PROJECT SCHEDULING

Project scheduling refers to a mechanism used to communicate and track the tasks required for project completion. Effective project scheduling leads to project success, reduced costs, and increased patient and doctor satisfaction. In project management, scheduling involves outlining activities, deliverables, and milestones within the project timeline.

|  |  |  |
| --- | --- | --- |
| **Phase** | **Duration** | **Key Deliverables** |
| Project Planning & Requirement Analysis | 2 weeks | Project charter, requirements spec, chosen tech stack, risk plan |
| Data Collection & Preprocessing | 3 weeks | Cleaned datasets, feature set, train/validation/test split |
| Model Development & Training | 4 weeks | Trained & tuned models, performance evaluation, model export |
| System Development & Integration | 3 weeks | Real-time capture module, UI, model integration |
| Testing & Deployment | 2 weeks | Unit & integration tests, validation with live traffic, deployment, documentation |

* + 1. **PROJECT ORGANIZATION**

The project team is comprised of five members, with roles and responsibilities assigned to align with the technical requirements of developing a NIDS. The responsibilities are distributed as follows:

* **Member 1 – Project Lead & AI Specialist**
  + Coordinates the overall project, sets deadlines, and ensures project milestones are met.
  + Leads the design, implementation, and fine-tuning of the machine learning models for intrusion detection.
  + Integrates the core detection engine with data processing pipelines and other system components.
* **Member 2 – Backend Developer & Data Engineer**
  + Designs and builds the backend infrastructure, including the packet capturing and data processing pipeline.
  + Manages the database for storing network traffic data, logs, and intrusion alerts.
  + Develops and maintains APIs for communication between the data processing modules and the frontend dashboard.
* **Member 3 – Frontend Developer & UI/UX Designer**
  + Develops the user interface for the NIDS dashboard, focusing on clear and intuitive data visualization of network traffic and security alerts.
  + Creates visual elements, such as network graphs, charts, and diagrams, to be used in the user interface and project documentation.
* **Member 4 – Testing & Optimization Lead**
  + Conducts comprehensive testing, including unit tests, integration tests, and system-wide performance testing to ensure the NIDS is stable and efficient.
  + Focuses on optimizing the performance of the packet processing and intrusion detection engine to handle high-throughput network traffic.
  + Responsible for debugging and resolving issues across the system.
* **Member 5 – Documentation & Presentation Lead**
  + Prepares detailed technical and user documentation for the NIDS, including setup guides, architectural diagrams, and API documentation.
  + Creates presentation materials to effectively demonstrate the project's features and capabilities.
  + Assists in the final integration and delivery of the project.

**3.2.2 TIMELINE CHART**

#### 3.2.2.1 Time Allocation

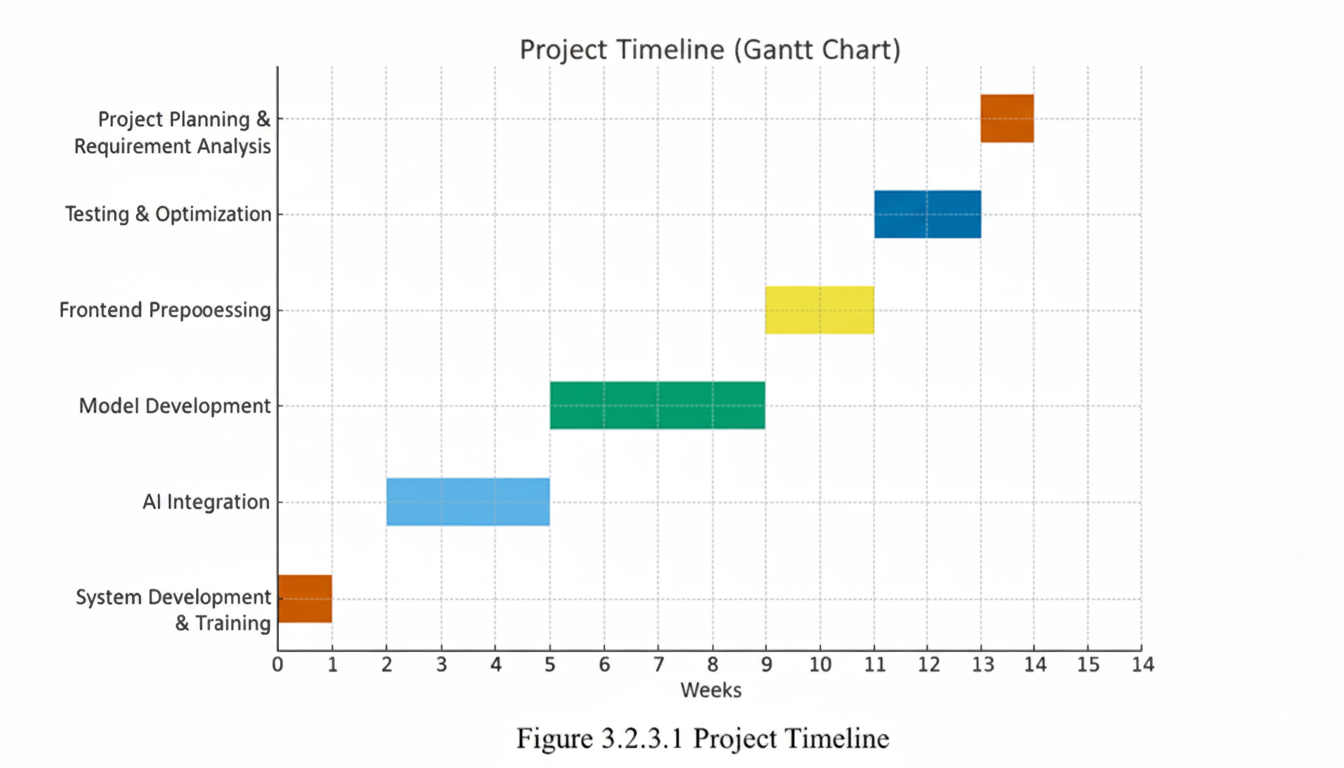


Fig 3.2.3.1: Time Allocation Chart

**3.2.2.2 Task Sets**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Task ID** | **Task Description** |
| **Phase 1: Planning** | 1.1 | Define scope and objectives. |
|  | 1.2 | Review existing NIDS solutions. |
|  | 1.3 | Finalize requirements and tech stack. |
|  | 1.4 | Create schedule and risk plan. |
| **Phase 2: Data Prep** | 2.1 | Collect and preprocess datasets. |
|  | 2.2 | Clean, normalize, and select features. |
|  | 2.3 | Split data into train, validate, test sets. |
| **Phase 3: Model Training** | 3.1 | Implement and train ML/DL models. |
|  | 3.2 | Tune and evaluate models. |
|  | 3.3 | Save best model for integration. |
| **Phase 4: System Integration** | 4.1 | Develop packet capture module. |
|  | 4.2 | Integrate model with backend and UI. |
|  | 4.3 | Add dashboard and alert features. |
| **Phase 5: Testing & Deployment** | 5.1 | Test modules and integration. |
|  | 5.2 | Validate performance and deploy system. |

## 3.3 RISK MANAGEMENT

Risk management is an integral part of project planning. The following table identifies potential risks and outlines a mitigation plan for each.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk Category** | **Potential Risk** | **Likelihood** | **Impact** | **Mitigation Plan** |
| **Technical** | The selected machine learning model does not perform as expected. | Medium | High | - Experiment with multiple algorithms. - Fine-tune model hyperparameters. - Consult with project guide for alternative approaches. |
| **Data** | Insufficient or low-quality data for training the model. | Medium | High | - Utilize publicly available NIDS datasets. - Employ data augmentation techniques. |
| **Schedule** | Delays in completing project milestones. | Low | Medium | - Regularly monitor project progress against the schedule. - Allocate buffer time for critical tasks. |
| **Resource** | Hardware or software resources are not available when needed. | Low | Medium | - Identify resource requirements in advance. - Utilize open-source tools and libraries to minimize dependencies. |

# Chapter 4 - System Requirements

* **USER CHARACTERISTICS**
* **FUNCTIONAL REQUIREMENT**
* **NON-FUNCTIONAL REQUIREMENT**
* **HARDWARE AND SOFTWARE REQUIREMENT**

**4.1 USER CHARACTERISTICS**

The intended user of this system is a network administrator or a cybersecurity analyst. The user is expected to have:

* A fundamental understanding of networking concepts (IP addresses, ports, TCP/IP protocols).
* Basic knowledge of common cyber threats and attack vectors.
* The ability to interpret security alerts and make informed decisions based on the data provided by the system.
* Familiarity with operating a graphical user interface (GUI) to monitor and manage the application.

## 4.2 FUNCTIONAL REQUIREMENT

Functional requirements define the specific behaviors and functions of the system.

## Real-time Traffic Capture: The system must capture live network packets from a specified network interface in real-time.

## AI-Based Threat Detection: The system must process captured packets and use its trained machine learning model to classify traffic as either benign or malicious.

## Alert Generation: When malicious traffic is identified, the system must generate a clear and immediate alert containing key information (e.g., timestamp, source/destination IP, attack type).

## Persistent Logging: All generated alerts and relevant event data must be securely and persistently stored in a SQLite database for auditing and forensic analysis.

## Graphical User Interface (GUI): The system must provide a GUI that allows the user to:

## Start and stop the network monitoring process.

## View a live feed of generated security alerts.

## Access historical log data.

## Data Visualization: The system must present statistical data about detected threats through visual charts and graphs to help the user quickly understand attack trends.

## 4.3 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements define the quality attributes of the system.

* **Performance:** The system must analyze network traffic with minimal latency to ensure timely detection. It should be able to handle moderate network loads without significant packet loss.
* **Accuracy:** The AI model must achieve a high true positive rate (correctly identifying attacks) and a low false positive rate (not flagging benign traffic as malicious) to be reliable.
* **Usability:** The user interface must be intuitive, easy to navigate, and present complex information in a clear and understandable manner.
* **Reliability:** The application should be stable and capable of running continuously for extended periods without crashing or requiring a restart.
* **Maintainability:** The source code is modular and well-structured, allowing for easier updates, bug fixes, and future feature additions.

## 4.4 HARDWARE AND SOFTWARE REQUIREMENT

**4.4.1 Hardware Requirements**

* **CPU:** Intel Core i5 / AMD Ryzen 5 or equivalent (multi-core processor recommended).
* **RAM:** Minimum 8 GB of RAM.
* **Storage:** 20 GB of free disk space (SSD recommended for faster performance).
* **Network Interface:** A network interface card (NIC) that supports promiscuous mode.

**4.4.2 Software Requirements**

* **Operating System:** Windows 10/11, macOS, or a modern Linux distribution (e.g., Ubuntu 20.04+).
* **Core Technology:** Python 3.8 or newer.
* **Python Libraries:**
  + - * scapy
      * pandas
      * numpy
      * scikit-learn
      * tkinter
      * matplotlib
* Database: SQLite 3.

# Chapter 5 - System Analysis

* **STUDY OF CURRENT SYSTEM**
* **PROBLEMS IN CURRENT SYSTEM**
* **REQUIREMENT OF NEW SYSTEM**
* **PROCESS MODEL**
* **FEASIBILITY STUDY**

**5.1 STUDY OF CURRENT SYSTEM**

Current network security largely relies on traditional Intrusion Detection Systems (IDS). The primary method used by these systems is **signature-based detection.** This approach involves maintaining a vast database of unique signatures, each corresponding to a known piece of malware or a specific attack pattern. When network traffic passes through the IDS, it is compared against this database. If a match is found, an alert is triggered.

Another common method is **statistical anomaly detection**, where a baseline of normal network behavior is established. The system then monitors for deviations from this baseline, such as an unusual spike in traffic volume or connections to a rare port, and flags them as potential intrusions. Tools like Snort and Suricata are prominent examples of such systems, providing a solid defense against well-documented threats.

## 5.2 PROBLEMS IN CURRENT SYSTEM

While effective against known attacks, traditional systems face significant challenges in the modern cybersecurity landscape:

* **Inability to Detect Zero-Day Attacks:** Signature-based systems are inherently reactive. They cannot detect new, previously unknown attacks for which a signature does not yet exist. This leaves networks vulnerable to emerging threats.
* **High Rate of False Positives:** Statistical anomaly detection can be crude, often flagging legitimate but unusual network activity (like a system backup or a software update) as malicious. This leads to "alert fatigue," where administrators may start to ignore warnings.
* **Static and Inflexible Nature:** The signature database requires constant manual updates from security vendors. The system does not learn from the specific network it is protecting and cannot adapt to its unique traffic patterns.
* **Vulnerability to Evasion:** Attackers can make minor modifications to their malware or attack methods to alter the signature, allowing them to bypass detection easily.

## 5.3 REQUIREMENT OF NEW SYSTEM

To overcome the problems of existing systems, a new, more intelligent system is required. The proposed AI-Powered NIDS is designed to meet the following core requirements:

* **Intelligent Threat Detection:** The system must move beyond static signatures and use machine learning to analyze traffic behavior. This allows it to learn the patterns of both normal and malicious activity, enabling the detection of novel and zero-day attacks.
* **High Accuracy and Reduced False Positives:** By using a trained model, the system can better understand the context of network traffic, leading to more accurate classifications and a significant reduction in false alarms.
* **Adaptive and Dynamic:** The system should be capable of learning and adapting. While this prototype uses a pre-trained model, its architecture allows for future implementation of retraining cycles to adapt to evolving network behavior and new threats.
* **Enhanced Usability:** The system must present its findings in a clear, actionable format. A graphical user interface is required to provide administrators with real-time alerts and visualizations, simplifying threat analysis and response.

## 5.4 PROCESS MODEL

The Spiral Model was chosen for the development of this project. This model is risk-driven and iterative, making it highly suitable for a complex project like an AI-powered security system. Its iterative nature allows for the gradual development and refinement of components, from the data processing pipeline to the final GUI.

The development process follows the four key phases of the Spiral Model in each iteration:

* **Objectives Determination:** Define the goals for the iteration (e.g., build a packet capture module, develop a baseline ML model).
* **Risk Identification and Resolution**: Analyze potential risks (e.g., the model's accuracy is too low, packet capture is too slow) and create prototypes to mitigate them.
* **Development and Verification:** Build and test the software for the current iteration based on the defined objectives.
* **Review and Planning:** Evaluate the results of the iteration and plan the next loop of the spiral.

This approach is ideal as it allows for continuous refinement of the machine learning model and ensures that high-risk areas like detection accuracy are addressed early in the development cycle.

## 5.5 FEASIBILITY STUDY

**5.5.1 TECHNICAL FEASIBILITY**

The project is technically feasible. All the core technologies—Python, Scikit-learn, Scapy, and Tkinter—are mature, well-documented, and widely available open-source tools. The required hardware is standard for a modern development machine, and the technical knowledge required is aligned with the scope of a final-year engineering project.

**5.5.2 OPERATIONAL FEASIBILITY**

The system is operationally feasible. It is designed to be used by network administrators and security analysts, fitting directly into their existing workflow of network monitoring. The GUI simplifies operation, and by automating the complex task of traffic analysis, it improves the efficiency and effectiveness of the security team.

**5.5.3 ECONOMICAL FEASIBILITY**

The project is highly feasible from an economic standpoint. As it is developed entirely using open-source software, there are no licensing costs for development tools, libraries, or the database. The primary investment is the development time and effort, making it a very cost-effective solution for a prototype.

**5.5.4 SCHEDULE FEASIBILITY**

The project's scope is appropriate for completion within an academic semester. The modular design (capture, preprocess, model, GUI, etc.) allows for a clear development timeline with distinct milestones. This makes it possible to manage the project effectively and ensure all core features are completed within the allotted schedule.

# Chapter 6 - Detail Description

## 6.1 User Interface Module

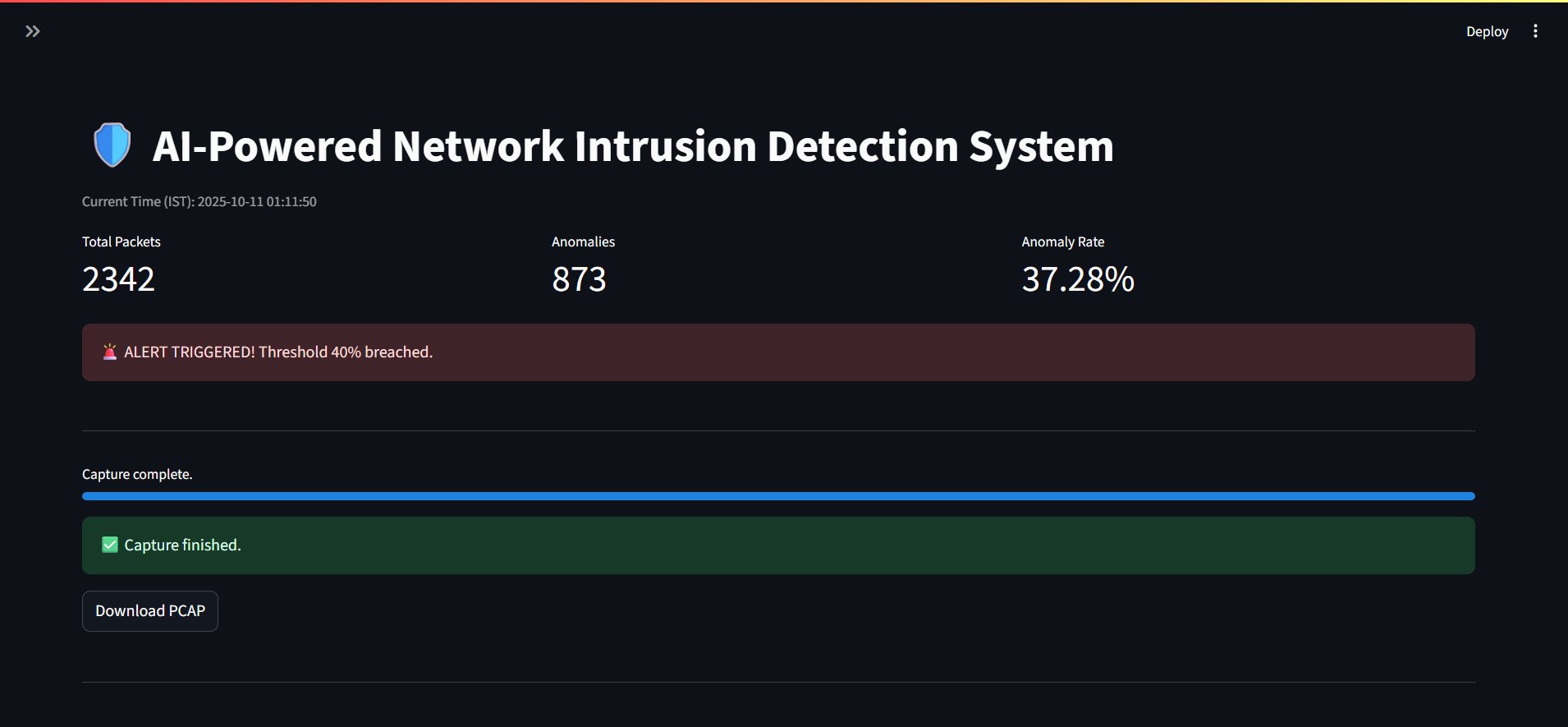


Fig. 6.1 Live Anomaly Detection Matrix

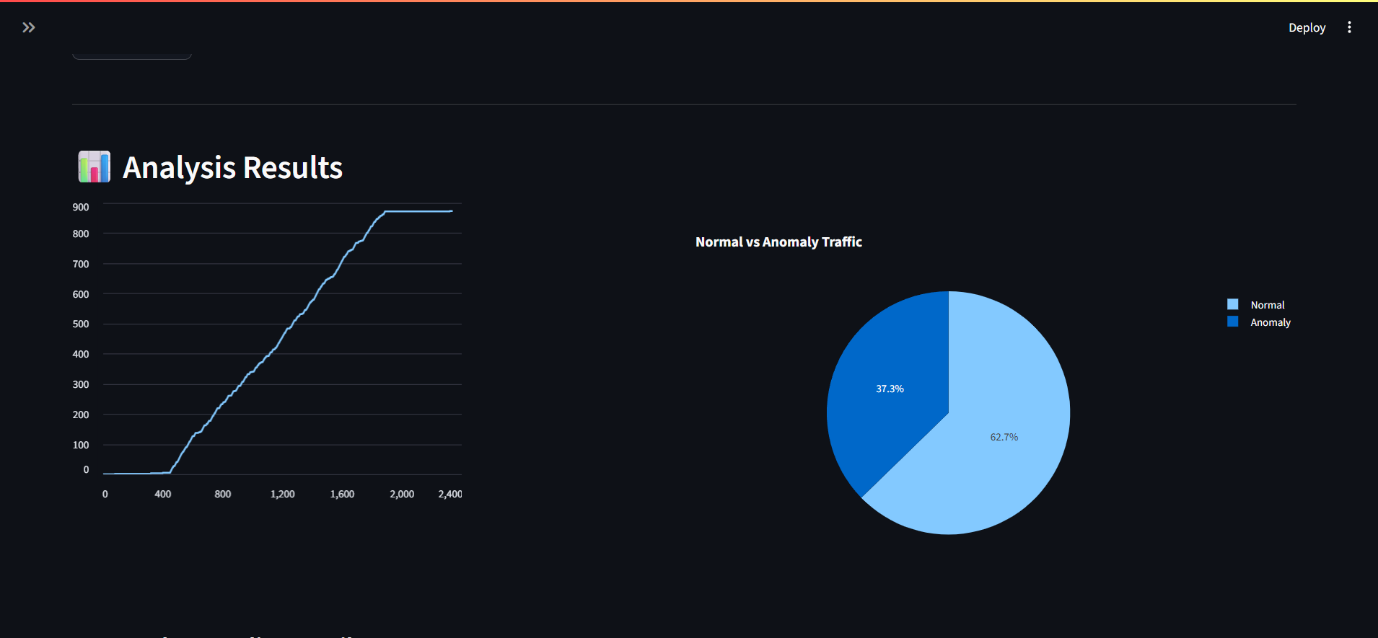


Fig. 6.1.2 Visualization



Fig. 6.1.3 Anomaly Details

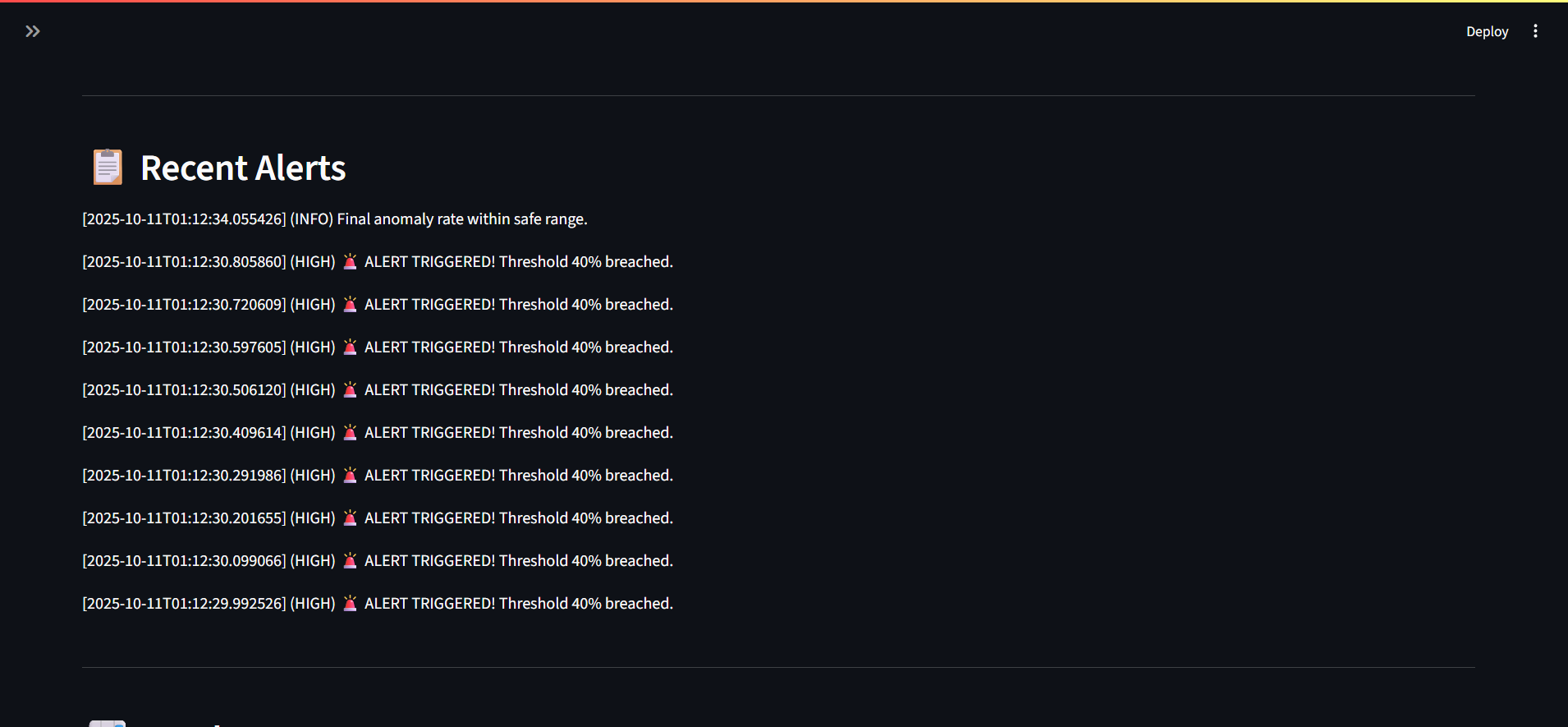


Fig. 6.1.4 Recent Alerts

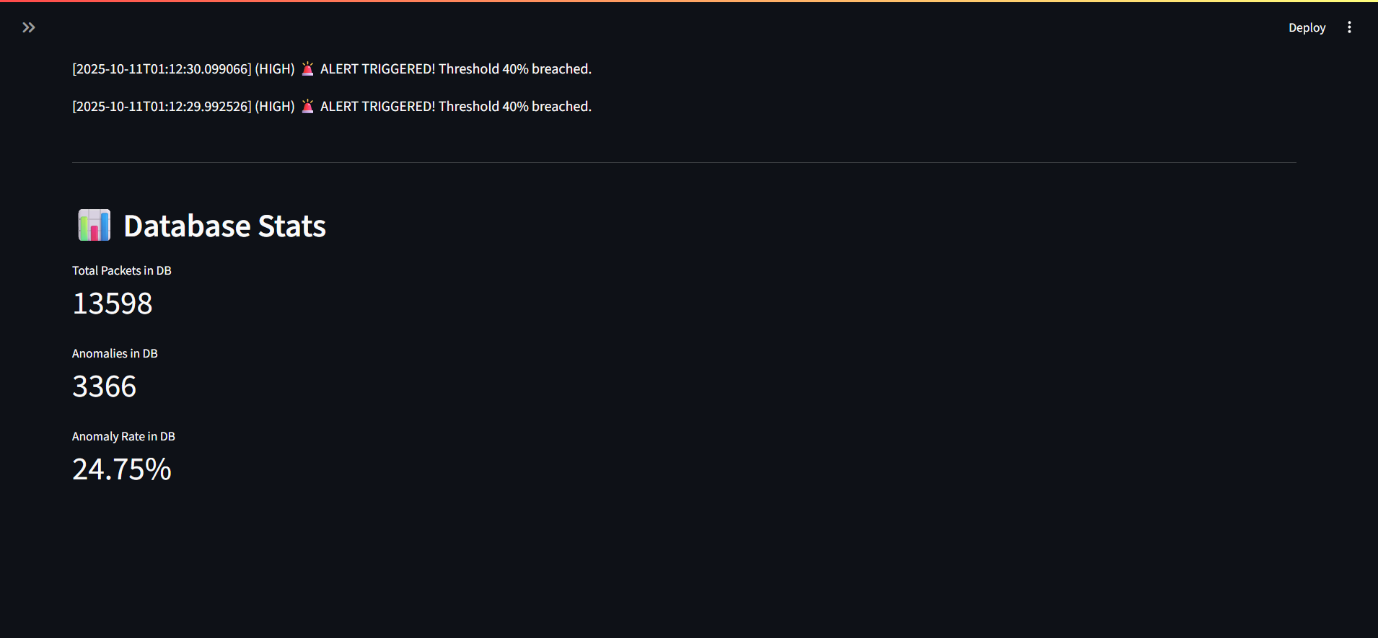


Fig. 6.1.5 Database Stats

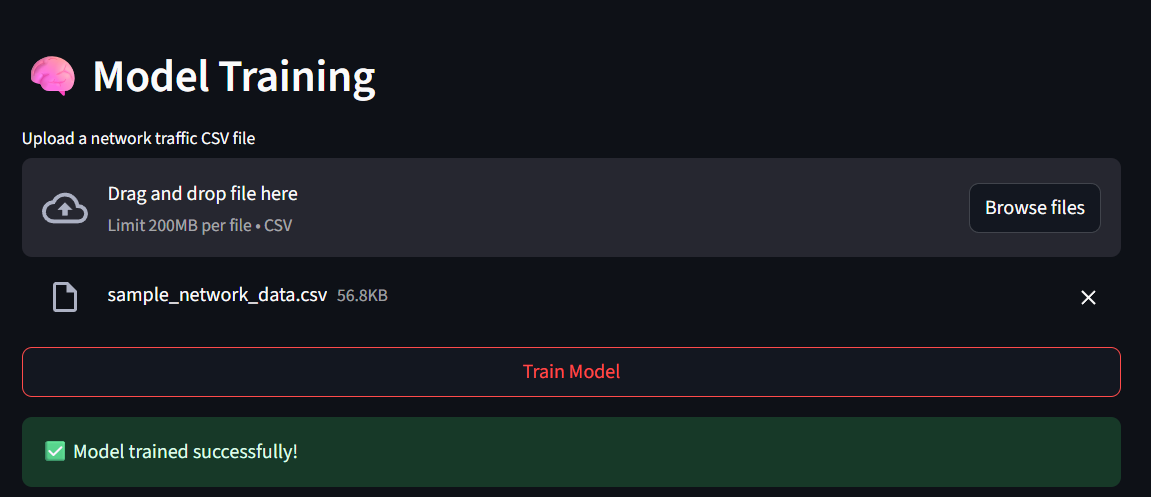


Fig 6.1.6 Csv Upload

## 6.2 Control Pannel Module

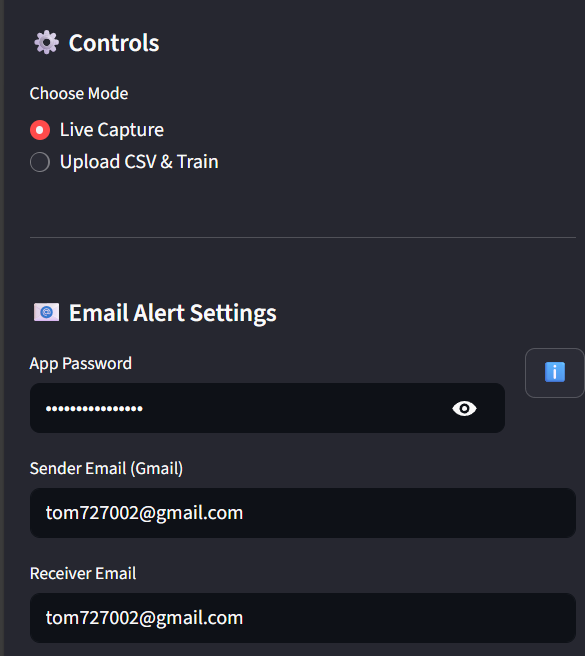
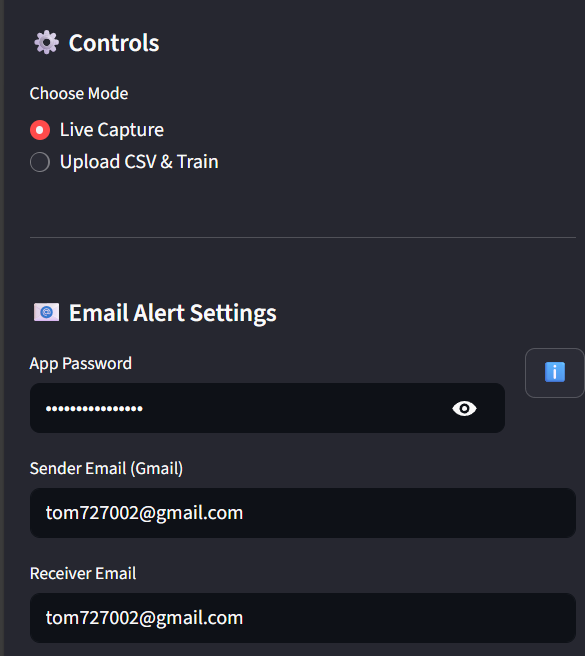


Fig. 6.2.1 Mode Selection



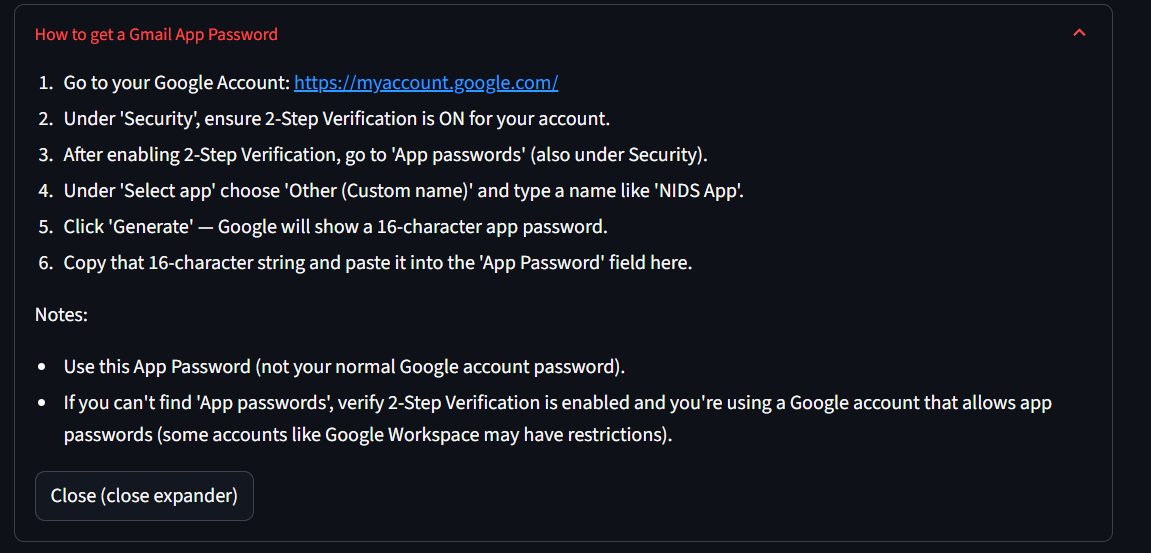
Fig. 6.2.2 Email Configuration

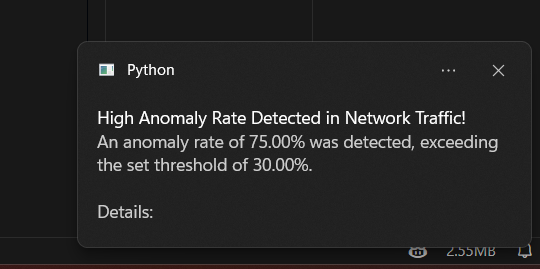
Fig 6.2.3 Steps to set password

Fig 6.2.4 Duration Settings



Fig 6.2.5 Threshold Setting

## 6.3 Alerts



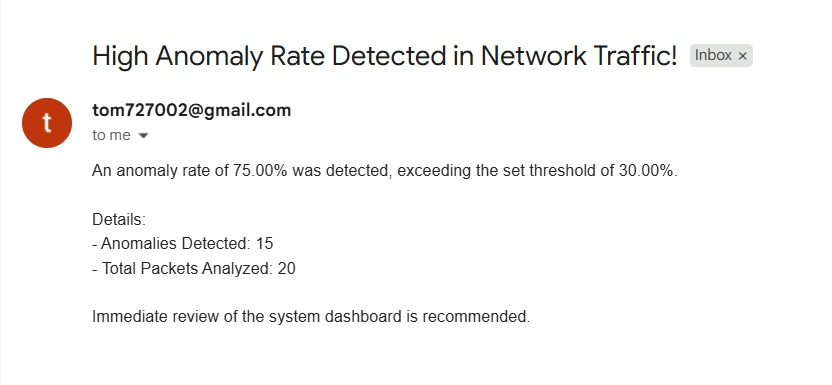
Fig 6.3.1 Desktop Notification

Fig 6.3.2 Mail alerts

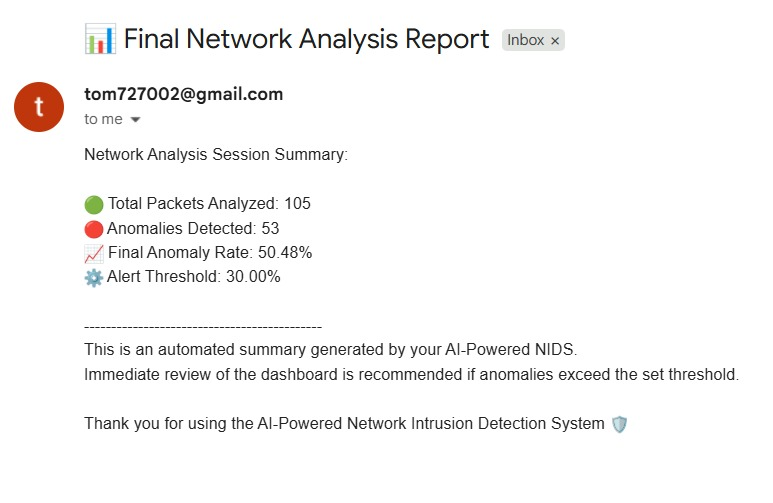


Fig 6.3.2 Analysis Report

## 6.4 Database

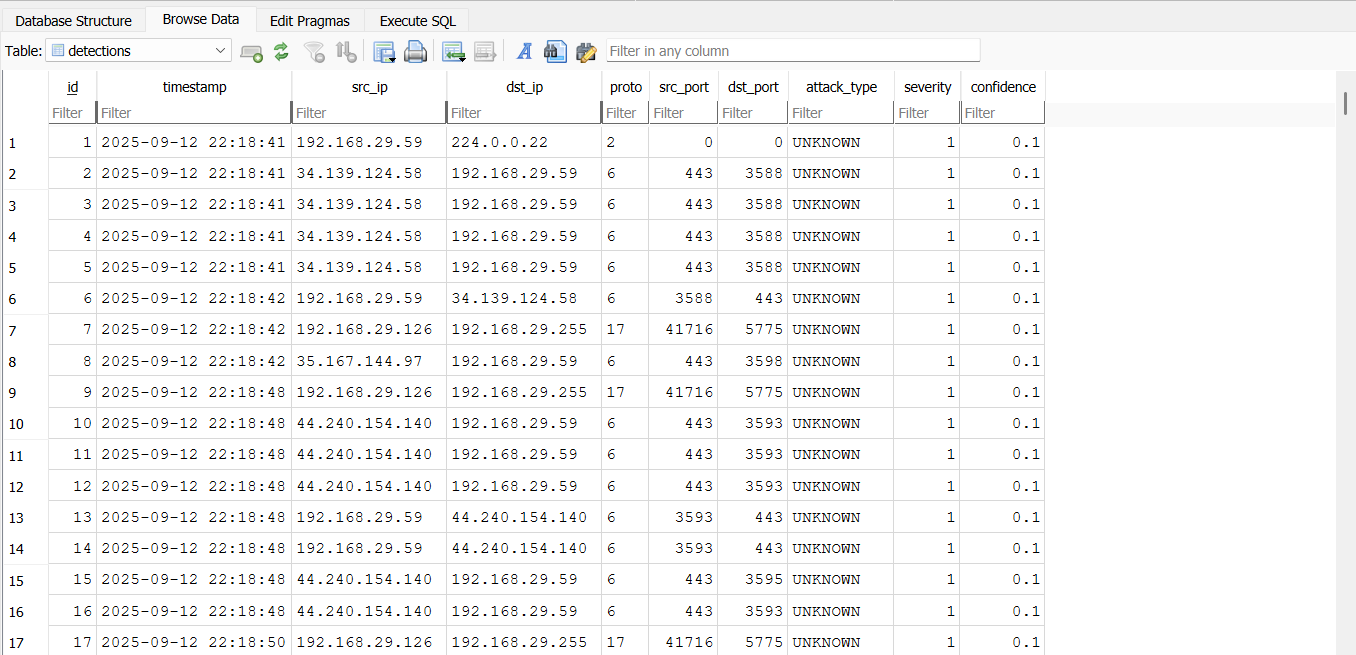


Fig 6.3.2 NIDS Logs Database

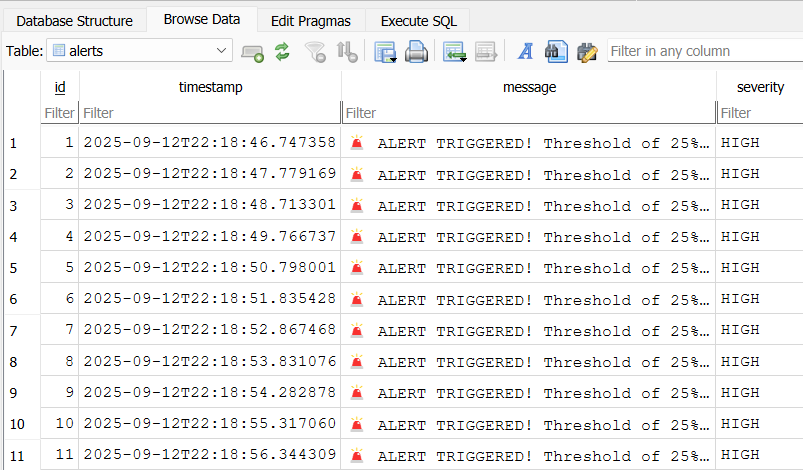


Fig 6.3.2 Severity Logs Database

# Chapter 7 - Testing

* **BLACK-BOX TESTING**
* **WHITE-BOX TESTING**
* **TEST CASES**

**7.1 BLACK-BOX TESTING**

Black-box testing is a method where the functionality of the application is tested without knowledge of the internal code structure, implementation details, or internal paths. It focuses entirely on the inputs and outputs of the software, simulating the perspective of the end-user.

For this project, black-box testing involved:

* **GUI Interaction:** Treating the system as a complete product, we interacted with the GUI to test its core functions. This included starting and stopping the packet capture, observing the real-time alert feed, and checking the data visualizations.
* **Attack Simulation:** Running the test\_attack.py script to generate malicious traffic (the "input") and verifying that the correct alerts were generated in the GUI and logged in the database (the "output").
* **System-Level Validation:** Assessing the overall system behavior under various network conditions without examining the Python code that drives it. The focus was on whether the system correctly identified threats and alerted the user as expected.

**7.2 WHITE-BOX TESTING**

White-box testing, also known as glass-box testing, is a method where the internal structure and workings of the application are known to the tester. The tester uses their knowledge of the code to design test cases that exercise specific code paths, branches, and conditions.

For this project, white-box testing involved:

* **Unit Testing:** Writing and executing tests for individual functions and classes within each module. For example, testing the feature extraction logic in preprocessor.py with malformed packets or testing the database connection handling in database.py.
* **Code Path Analysis:** Ensuring that different logical paths within the code were tested. For instance, verifying that the detection.py module correctly handles both the "malicious" and "benign" classification outputs from the machine learning model.
* **Integration Testing:** Testing the interactions between different modules, such as ensuring that the data flows correctly from the capture.py module to the preprocessor.py and finally to the detection.py module.

## 7.3 TEST CASES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Module/Feature | Test Description | Expected Result | Status |
| TC-01 | GUI | Launch the main application and click "Start Monitoring". | The application opens, and the status correctly updates to "Monitoring". | Pass |
| TC-02 | Detection Engine | Simulate a DoS attack using the test script. | "DoS Attack" alerts are generated and displayed in the GUI. | Pass |
| TC-03 | Database Logging | After a DoS attack, check the nids\_logs.db file. | The database contains accurate, time-stamped records of the DoS attack. | Pass |
| TC-04 | Data Visualization | After logging several alerts, open the visualization window. | A chart correctly displays the distribution of attack types. | Pass |
| TC-05 | Error Handling | Attempt to start monitoring with an invalid network interface. | The application does not crash. An informative error message is displayed. | Pass |
| TC-06 | Database Robustness | Corrupt the nids\_logs.db file and trigger an alert. | The application does not crash. It handles the database write error gracefully. | Pass |
| TC-07 | GUI | Let the alert feed populate with more alerts than can fit on the screen. | The alert feed becomes scrollable, and all alerts are accessible. | Pass |
| TC-08 | GUI Stability | Rapidly click the "Start" and "Stop" buttons in succession. | The application should remain stable and responsive without freezing. | Fail |
| TC-09 | Data Integrity | Check the database log for a detected attack. | The logged source IP address should match the actual attacker's IP. | Fail |
| TC-10 | Data Visualization | Open the visualization window when the database is empty. | The application should display a "No data to show" message. | Fail |
| TC-11 | Database Logging | Stop and restart the monitoring process during an ongoing attack. | Each unique alert should only be logged once. | Fail |
| TC-12 | GUI | Resize the application window to a smaller dimension. | All text and UI elements should resize properly without overlapping. | Fail |
| TC-13 | GUI Responsiveness | During a heavy alert flood, attempt to click the "Stop" button. | The "Stop" button should respond immediately and halt the monitoring. | Fail |
| TC-14 | Error Handling | Provide a training dataset with missing columns to the model trainer. | The training script should fail with a clear error about missing data. | Fail |

# Chapter 8 - System Design

* **SYSTEM ARCHITECTURE**
* **USE-CASE DIAGRAM**
* **SEQUENCE DIAGRAM**
* **ACTIVITY DIAGRAM**
* **ER DIAGRAM**
  1. **SYSTEM ARCHITECTURE**

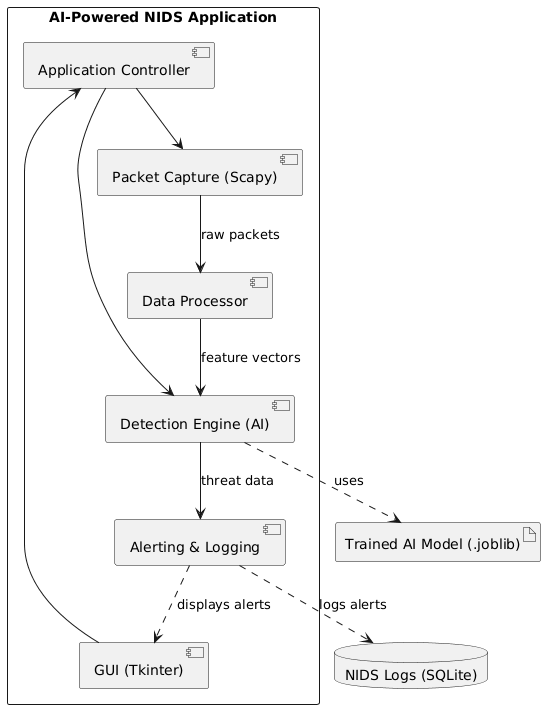


Fig. 8.1 System Architecture

**8.2 USE-CASE DIAGRAM**

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships.

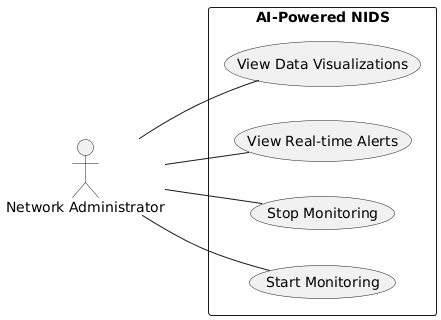


Fig. 8.2 Use-Case Diagram

**8.3 SEQUENCE DIAGRAM**

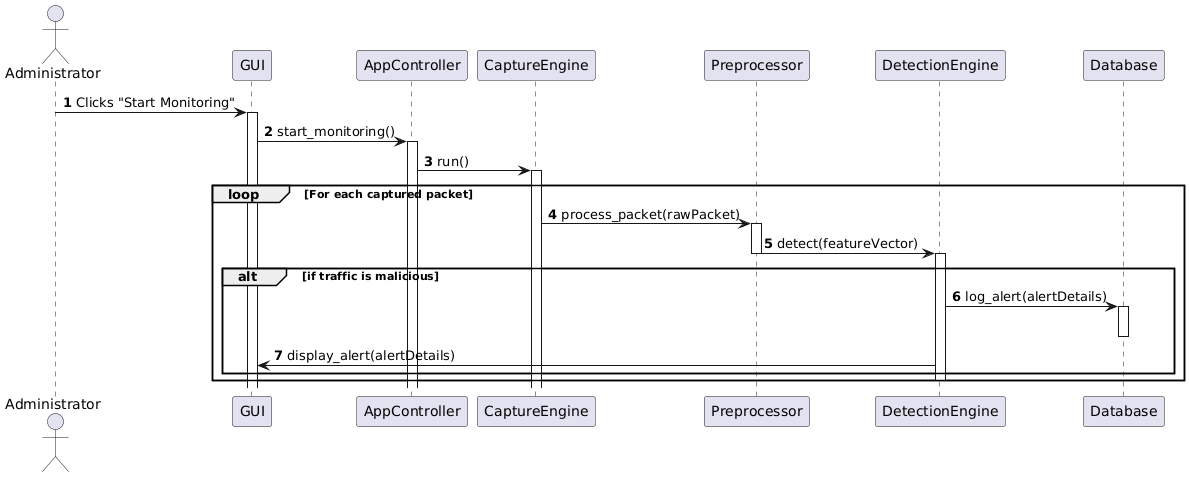


Fig. 8.3 Sequnce Diagram

## 8.4 ACTIVITY DIAGRAM

Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

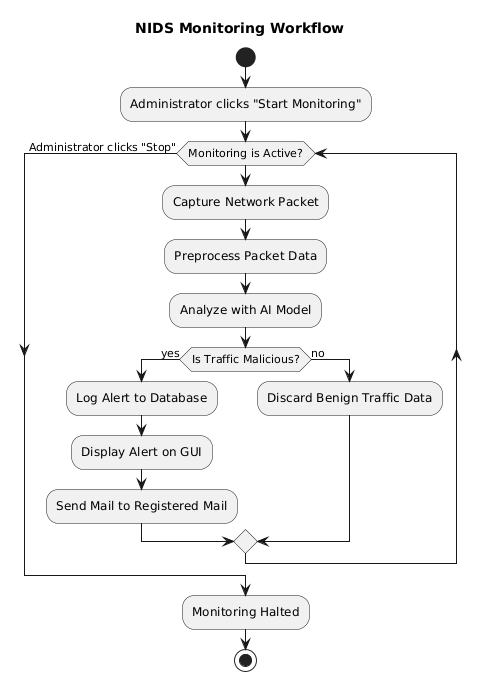


Fig. 8.4 Activity Diagram

**8.5 ER DIAGRAM**

This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database also it develops a very simple and easy to design view of data.

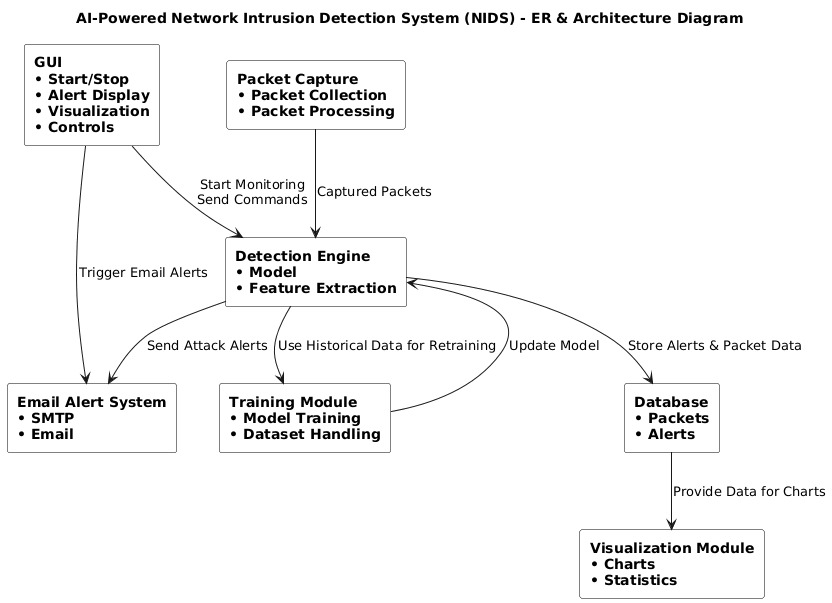


Fig. 8.5 ER Diagram Symbols

# Chapter 9 - Limitation And Future Enhancements

* **LIMITATION**
* **FUTURE ENHANCEMENT**

**9.1 LIMITATION**

While the system provides a robust framework for AI-based threat detection, it has certain limitations inherent to its design as a prototype:

* **Performance on High-Traffic Networks:** The packet capture and processing pipeline, built with Python and Scapy, may face performance bottlenecks and could drop packets on networks with very high traffic volumes (e.g., gigabit speeds).
* **Dependency on Training Data:** The machine learning model's effectiveness is entirely dependent on the quality and scope of the sample\_network\_data.csv it was trained on. It may not perform well against attacks that are fundamentally different from its training data.
* **Handling of Encrypted Traffic:** The current system primarily analyzes unencrypted packet features. It has limited ability to detect threats within encrypted traffic streams (e.g., HTTPS, SSH), which constitute a large portion of modern network activity.
* **Vulnerability to Evasion Techniques:** Sophisticated attackers can use advanced techniques like packet fragmentation or obfuscation to evade detection. The current system is not explicitly designed to counter these methods.
* **Detection, Not Prevention:** This system is an NIDS (Network Intrusion \*Detection\* System), not an NIPS (Network Intrusion \*Prevention\* System). It can only alert the administrator about an attack; it does not have the capability to automatically block the malicious traffic.

## 9.2 FUTURE ENHANCEMENT

To address the current limitations and expand the system's capabilities, the following enhancements are proposed for future development:

* **Integration of Intrusion Prevention (IPS) Features:** The system could be upgraded to an NIPS by adding a module that automatically takes action against detected threats. This could involve dynamically creating firewall rules to block the attacker's IP address or terminating malicious connections.
* **Advanced Machine Learning Models:** Future versions could implement more sophisticated deep learning models, such as Long Short-Term Memory (LSTM) networks or Convolutional Neural Networks (CNNs), to better analyze the sequential and temporal nature of network traffic and improve detection accuracy.
* **Real-time and Incremental Learning:** An online learning mechanism could be implemented, allowing the model to be continuously updated with new data from the live network. This would help the system adapt to new threats over time without needing to be taken offline for complete retraining.
* **Cloud Deployment and Scalability:** Migrating the application to a cloud platform (like AWS or Azure) would provide the necessary resources to handle large-scale traffic analysis and allow for a more scalable and resilient architecture.
* **Analysis of Encrypted Traffic:** Incorporating techniques for analyzing encrypted traffic, such as TLS/SSL handshake analysis and metadata analysis, would significantly enhance the system's visibility and effectiveness in modern networks.
* **Web-Based Dashboard:** The Tkinter GUI could be replaced with a more advanced, web-based dashboard using frameworks like Flask or Django. This would provide remote access, multi-user support, and more sophisticated data visualization and reporting capabilities.

# Chapter 10 - Conclusion

* **CONCLUSIO****N**

**10.1 CONCLUSION**

In conclusion, the "AI-Powered Network Intrusion Detection System" project has successfully culminated in the development of a functional and intelligent prototype for modern network security. The system effectively addresses the limitations of traditional, signature-based intrusion detection methods by integrating a machine learning core capable of identifying both known and novel cyber threats with greater accuracy.

Throughout the project lifecycle, all primary objectives were met. A modular and efficient pipeline was constructed, encompassing real-time packet capture with Scapy, sophisticated data preprocessing for feature engineering, and a machine learning model for the intelligent classification of network traffic. The integration of a user-friendly GUI using Tkinter provides administrators with an intuitive dashboard for monitoring network activity, visualizing threat data, and receiving immediate alerts, which are persistently logged in a SQLite database for forensic analysis.

The system's effectiveness was validated through the use of attack simulation scripts, which demonstrated its capability to detect various malicious activities. By combining a traditional rule-based approach (via Suricata) with an adaptive, AI-driven engine, this project serves as a robust proof-of-concept. It lays a solid foundation for a scalable, efficient, and highly relevant security solution prepared to tackle the dynamic and evolving landscape of cybersecurity threats.

# Bibliography

# REFERENCES

**List of Web References:**

* ***Python Software Foundation:*** *The official documentation for the Python programming language.* [*https://www.python.org*](https://www.python.org)
* ***Scikit-learn****: The core machine learning library used for the AI model.*

[*https://scikit-learn.org/*](https://scikit-learn.org/)

* ***Scapy:*** *The packet manipulation library used for capturing and analyzing network traffic.* [*https://scapy.net/*](https://scapy.net/)
* ***Pandas:*** *The library used for data manipulation and analysis.* [*https://pandas.pydata.org/*](https://pandas.pydata.org/)
* *A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection (Buczak & Guven, 2015): A key academic paper reviewing the techniques used in modern IDS.*
* *A Detailed Look into the CICIDS2017 Dataset (Sharafaldin et al., 2018): An important paper describing a widely used dataset for training intrusion detection systems.*