

**NMRL NETWORK INTRUSION DETECTION SYSTEM**

**by**

**ELPHORD KUDZAI MACHIDA**

**Project submitted for review for the**

**HONORS CLOUD COMPUTING AND INTERNET OF THINGS (HCC)**

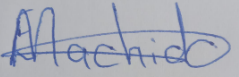
**in the Faculty of Computer Engineering Informatics and Communications**

**at the University of Zimbabwe**

**Academic Supervisor: MR ZANAMWE**

**Submission Date : 26 June 2025**

# Declaration

I, Elphord Kudzai Machida, hereby declare that the project entitled “NMRL NIDS SYSTEM” submitted to the University of Zimbabwe for the degree of Honors in Cloud Computing, is my original work and has not been submitted earlier either to this University or to any other institution for the fulfillment of the requirement for any course of study.  
  
Signature: \_\_\_\_\_\_  
Date: \_\_\_\_13/03/2025\_\_\_\_

# Approval

This project entitled “NMRL NIDS SYSTEM” submitted by Elphord Kudzai Machida has been examined and approved.

…………  
Supervisor: Mr Pedzisai  
  
……………………………..  
Head of Department : Mr Musendame

# Acknowledgements

I would like to express my sincere gratitude to my supervisor Mr Pedzisai for his invaluable guidance, constructive feedback, and consistent encouragement throughout the course of this project. Special thanks go to my family and friends for their unwavering support and motivation.

# Abstract

The National Microbiology Reference Laboratory (NMRL) operates a complex data centre environment that hosts a wide array of critical services, including research databases, diagnostic systems, and national health data platforms. With numerous servers running simultaneously and firewall systems protecting the infrastructure, the environment remains a high-value target for cyber threats and network-based intrusions.

Traditional rule-based firewalls and signature-based detection systems are limited in identifying novel (newly discovered or previously unknown method threats actors) or sophisticated attack patterns. Given the volume, variety, and velocity of data passing through NMRL's network infrastructure, the potential for undetected anomalies such as unauthorized access attempts, data exfiltration, or command and control communications poses a significant risk to data integrity, confidentiality, and system availability.

This project addresses the urgent need for a robust anomaly detection system tailored to NMRL’s network environment. By leveraging machine learning models, including unsupervised and semi-supervised techniques, the system aims to augment the existing Network Intrusion Detection System (NIDS). The solution is designed to detect abnormal traffic patterns and flag potential intrusions that evade conventional security mechanisms, thereby enhancing real-time threat detection and supporting timely response actions across NMRL’s secure network infrastructure.

**Keywords:** Network Anomaly Detection, Machine Learning, Autoencoder, K-means, Isolation Forest, Cybersecurity.

Contents

[Declaration 2](#_Toc201679241)

[Approval 2](#_Toc201679242)

[Acknowledgements 3](#_Toc201679243)

[Abstract 3](#_Toc201679244)

[Chapter 1: Introduction 7](#_Toc201679245)

[1.1 Introduction 7](#_Toc201679246)

[1.2 Background and Context of the Project 7](#_Toc201679247)

[1.3 Problem Statement 7](#_Toc201679248)

[1.4 Aim 8](#_Toc201679249)

[1.5 Research Objectives 8](#_Toc201679250)

[1.6 Scope and Limitations 9](#_Toc201679251)

[1.7 Feasibility Study 9](#_Toc201679252)

[1.8 Significance and Motivation 9](#_Toc201679253)

[1.9 Work Plan 10](#_Toc201679254)

[1.10 Conclusion 10](#_Toc201679255)

[Chapter 2: Literature Review 11](#_Toc201679256)

[2.1 Introduction 11](#_Toc201679257)

[2.2 Review of Relevant Literature 11](#_Toc201679258)

[2.3 Discussion of Similar Projects or Systems 11](#_Toc201679259)

[2.4 Identification of Gaps or Areas for Improvement 11](#_Toc201679260)

[2.5 Conclusion 12](#_Toc201679261)

[Chapter 3: Methodology 13](#_Toc201679262)

[3.1 Introduction 13](#_Toc201679263)

[3.2 Software Development Methodology 13](#_Toc201679264)

[3.3 Methods and Techniques 13](#_Toc201679265)

[3.4 Data Handling and Feature Engineering 13](#_Toc201679266)

[3.5 Model Development and Training 14](#_Toc201679267)

[3.6 Tools and Technologies 14](#_Toc201679268)

[3.7 Project Requirements and Design Considerations 14](#_Toc201679269)

[3.8 Conclusion 15](#_Toc201679270)

[Chapter 4: Analysis and Design 16](#_Toc201679271)

[4.1 Introduction 16](#_Toc201679272)

[4.2 Detailed Analysis of the Problem Domain and User Requirements 16](#_Toc201679273)

[4.3 Functional Requirements 16](#_Toc201679274)

[4.4 Non-functional Requirements 16](#_Toc201679275)

[4.5 Identification of System Components and Functionalities 16](#_Toc201679276)

[4.6 Use-Case Diagram 17](#_Toc201679277)

[4.7 Sequence Diagram 17](#_Toc201679278)

[4.8 System Architecture and Design Considerations 17](#_Toc201679279)

[4.9 Context and DFD Diagrams 17](#_Toc201679280)

[4.10 Interface Design 17](#_Toc201679281)

[4.11 Security Design 17](#_Toc201679282)

[4.12 Conclusion 18](#_Toc201679283)

[Chapter 5: Results 19](#_Toc201679284)

[5.1 Introduction 19](#_Toc201679285)

[5.2 Presentation of Findings 19](#_Toc201679286)

[5.3 Conclusion 20](#_Toc201679287)

[Chapter 6: Discussion 21](#_Toc201679288)

[6.1 Introduction 21](#_Toc201679289)

[6.2 Summary of Findings 21](#_Toc201679290)

[6.3 Model Evaluation and Analysis 21](#_Toc201679291)

[6.4 Comparison with Existing Literature 21](#_Toc201679292)

[6.5 Theoretical Implications 22](#_Toc201679293)

[6.6 Practical Implications 22](#_Toc201679294)

[6.7 Validation and Reliability 22](#_Toc201679295)

[6.8 Limitations and Methodological Reflections 22](#_Toc201679296)

[6.9 Conclusion 22](#_Toc201679297)

[Chapter 7: Conclusion and Future Work 23](#_Toc201679298)

[7.1 Introduction 23](#_Toc201679299)

[7.2 Summary of the Project 23](#_Toc201679300)

[7.3 Key Findings and Contributions 23](#_Toc201679301)

[7.4 Evaluation of Objectives 23](#_Toc201679302)

[7.5 Reflection on the Project Process 24](#_Toc201679303)

[7.6 Future Work and Recommendations 24](#_Toc201679304)

[7.7 Conclusion 24](#_Toc201679305)

[References 25](#_Toc201679306)

[Appendices 26](#_Toc201679307)

[Appendix A: Templates of Data Collection Tools 26](#_Toc201679308)

[Appendix B: User Manual of the Working System 27](#_Toc201679309)

[Appendix C: Source Code 36](#_Toc201679310)

[Appendix D: Sample Outputs and Diagrams 37](#_Toc201679311)

**List of Symbols and Abbreviations**

NMRL – National Microbiology Reference Laboratory

NIDS – Network Intrusion Detection System

ML – Machine Learning

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

PCA – Principal Component Analysis

ROC – Receiver Operating Characteristics

DFD – Dataflow Flow Diagram

# Chapter 1: Introduction

## 1.1 Introduction

This chapter introduces the *NMRL NIDS SYSTEM* project, focusing on the development and deployment of a Network Intrusion Detection System using machine learning models. The chapter outlines the background of the project, problem statement, objectives, and overall scope.

## 1.2 Background and Context of the Project

With the rise in volume and complexity of cyber-attacks, particularly targeting networks in sensitive environments such as research laboratories, there is a growing need for robust intrusion detection systems. Traditional rule-based systems are limited in detecting new and evolving threats. This project was undertaken to develop a data-driven anomaly detection system using machine learning, tailored for the National Microbiology Reference Laboratory (NMRL).

## 1.3 Problem Statement

During my attachment at the National Microbiology Reference Laboratory (NMRL), I observed that the organization primarily relies on Sophos Firewall for network security. While firewalls are essential for perimeter defence, they are not sufficient to detect sophisticated and evolving threats, such as zero-day attacks, insider threats, or advanced persistent threats (APTs), which can bypass rule-based filtering. This limitation leaves NMRL’s critical infrastructure including research databases, diagnostic systems, and national health platforms vulnerable to undetected intrusions.

To address this gap, I recognized the need for a more intelligent and adaptive approach to network security. This motivated the exploration and development of a Machine Learning-based Network Intrusion Detection System (ML-NIDS). Unlike traditional firewalls, an ML-driven NIDS can analyse network traffic patterns in real-time, detect anomalies, and adapt to emerging threats, thereby strengthening NMRL’s cybersecurity defences.

## 1.4 Aim

To design and implement a **Machine Learning-based Network Intrusion Detection System (ML-NIDS)** that:

1. **Complements NMRL’s existing Sophos Firewall** by detecting advanced threats (e.g., zero-day attacks, APTs, and insider threats) that bypass rule-based filtering,
2. **Monitors real-time traffic** within NMRL’s **server-rich environment**, focusing on both perimeter and internal (east-west) communications, and
3. **Adapts dynamically** to evolving attack patterns through ML-driven anomaly detection, thereby safeguarding critical infrastructure including research databases, diagnostic systems, and national health platforms.

## 1.5 Research Objectives

* To pre-process and normalize network data collected from NMRL’s internal traffic.
* To build and evaluate unsupervised anomaly detection models (Autoencoder, K-means, Isolation Forest).
* To establish optimal detection thresholds tailored for NMRL’s network characteristics.
* To integrate the models into a secure web-based interface for real-time anomaly reporting.
* To implement alert notification (email) for critical alerts

## 1.6 Scope and Limitations

The system focuses solely on network intrusion detection using unsupervised learning. It does not incorporate signature-based detection or packet inspection. Limitations include dependency on the quality of input data and model generalization to unseen environments. The implementation is designed for demonstration purposes and may require optimization for production environments.

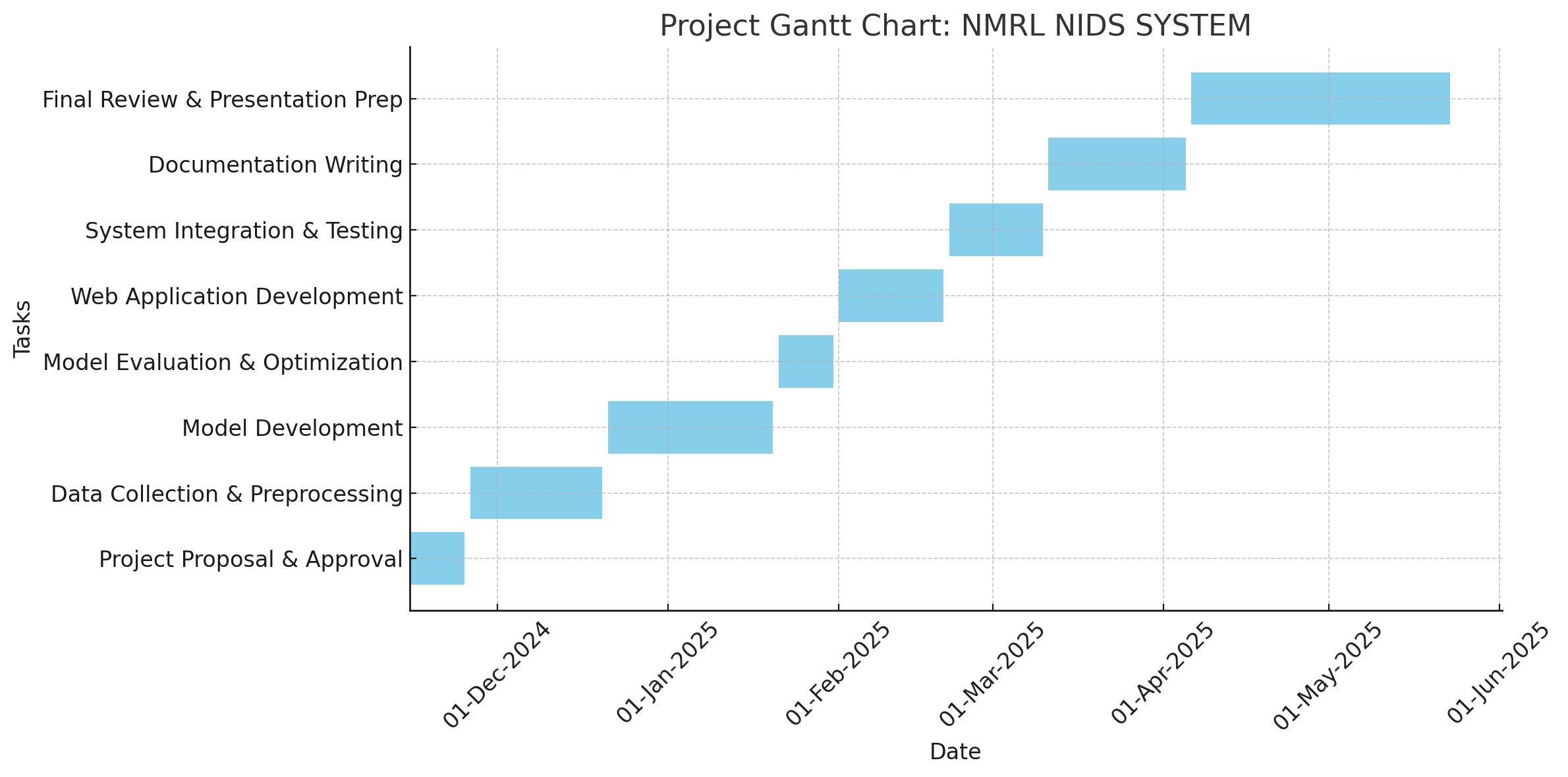
## 1.7 Feasibility Study

The project is technically feasible using open-source Python libraries such as TensorFlow, Scikit-learn, and Flask. It is economically viable as it requires no proprietary software. Socially, the system promotes security awareness. Operationally, the tool is deployable on local machines or cloud infrastructure with minimal overhead.

## 1.8 Significance and Motivation

The project contributes to strengthening cybersecurity at NMRL by proactively identifying and responding to network anomalies. Given the sensitivity and volume of data processed at the lab, a reliable anomaly detection system augments the current security posture. The motivation stems from the demand for scalable, adaptable, and intelligent detection mechanisms in modern data centre environments.

## 1.9 Work Plan



## 1.10 Conclusion

This chapter provided a comprehensive overview of the problem addressed, objectives pursued, and significance of the project. Subsequent chapters delve into the literature review, methodologies employed, design architecture, and results obtained.

# Chapter 2: Literature Review

## 2.1 Introduction

This chapter reviews existing literature related to network intrusion detection systems (NIDS), machine learning models applied in cybersecurity, and various anomaly detection methodologies. It establishes the background knowledge required for this project and highlights gaps addressed by the NMRL NIDS SYSTEM.

## 2.2 Review of Relevant Literature

The use of machine learning in network security has gained prominence due to its ability to detect unknown threats. Studies such as those by Buczak and Guven (2016) show how supervised and unsupervised models can classify or detect anomalies in network traffic. Autoencoders, in particular, have demonstrated potential in learning normal patterns and flagging deviations. Clustering algorithms like k-means (Huang, 1998) are effective for grouping similar behaviors, while Isolation Forests (Liu et al., 2008) specialize in identifying rare events in datasets.

## 2.3 Discussion of Similar Projects or Systems

Several open-source intrusion detection projects such as Snort and Suricata predominantly rely on signature-based approaches, making them less effective against novel attacks. Research by Javaid et al. (2016) demonstrated the use of deep learning in IDS, specifically through deep autoencoders, achieving notable accuracy improvements. However, deployment at scale remains challenging, prompting the need for lightweight models as proposed in this project.

## 2.4 Identification of Gaps or Areas for Improvement

Existing machine learning-based NIDS often suffer from high computational costs, limited real-time applicability, and difficulties in threshold tuning for anomaly detection. There is also limited research integrating multiple unsupervised models into a unified detection platform accessible via a web application, as is addressed by the NMRL NIDS SYSTEM.

## 2.5 Conclusion

The literature review highlights the significance of machine learning in enhancing network security and identifies gaps in current research and deployments. These insights form the basis for the methodologies and design choices discussed in subsequent chapters.

# Chapter 3: Methodology

## 3.1 Introduction

This chapter describes the methodological approach adopted to develop the NMRL NIDS SYSTEM. It outlines the machine learning models used, data handling procedures, training and evaluation techniques, and the tools and technologies employed throughout the project.

## 3.2 Software Development Methodology

An iterative development model was followed, allowing continuous integration of feedback and refinement of models. The project lifecycle included phases such as data preprocessing, model building, evaluation, and web application development.

## 3.3 Methods and Techniques

The following techniques were applied during the project:

- Packets Capturing : Captured network traffic (raw network data) of NMRL network using Wireshark .   
- Data Preprocessing : Involved cleaning, normalization, and transformation of raw network data.  
- Feature Engineering : Irrelevant features were dropped, and meaningful variables selected based on correlation and relevance to anomaly detection.  
- Model Development : Autoencoder, KMeans, and Isolation Forest were used for anomaly detection.  
- Threshold Selection : Optimal thresholds were determined using statistical analysis of reconstruction and anomaly scores.

## 3.4 Data Handling and Feature Engineering

The dataset was analyzed for missing values and inconsistencies. StandardScaler from scikit-learn was used to normalize numerical features. Categorical variables were encoded or removed as necessary. PCA was briefly explored but omitted due to adequate model performance on raw features. Feature selection emphasized traffic behavior indicators relevant to intrusion detection.

## 3.5 Model Development and Training

Three models were built:  
- Autoencoder : A deep neural network trained to reconstruct normal traffic. Reconstruction error was used to flag anomalies.  
- KMeans : Clustering model used to detect outliers based on distance from cluster centroids.  
- Isolation Forest : Identifies anomalies by randomly partitioning the feature space and isolating rare points.  
  
Each model was trained and tested using preprocessed data. Thresholds for anomaly detection were fine-tuned by analyzing score distributions.

## 3.6 Tools and Technologies

The following tools were used:  
- Python 3.12.10  
- Pandas, NumPy, Scikit-learn, TensorFlow  
- Flask for web application  
- Jupyter Notebook for development and experimentation  
- Matplotlib and Seaborn for visualization  
- Git for version control

- Chart.js for making HTML –based charts

## 3.7 Project Requirements and Design Considerations

Requirements were derived from the need to detect intrusions with high accuracy and deploy the system in an accessible format.  
Design considerations included:  
- Scalability and modularity of the application  
- Intuitive user interface for administrators  
- Secure and efficient model integration  
- Low latency in prediction responses

## 3.8 Conclusion

This chapter described the methodological framework and technologies used to develop the NMRL NIDS SYSTEM. The next chapter discusses the design and deployment of the application.

# Chapter 4: Analysis and Design

## 4.1 Introduction

This chapter presents the analysis and design of the NMRL NIDS SYSTEM with a focus on model deployment. It explains the components of the web-based application, integration of machine learning models, and architectural considerations.

## 4.2 Detailed Analysis of the Problem Domain and User Requirements

The project addresses the need for a system capable of real-time anomaly detection in network traffic. Target users include system administrators and IT Support personnel at the National Microbiology Reference Laboratory. Requirements were gathered through analysis of typical network threats and operational needs for early warning systems.

## 4.3 Functional Requirements

- Detect anomalies in network traffic using multiple ML models  
- Display prediction results through a user-friendly interface  
- Upload and analyze new network datasets  
- Visualize model results and alerts

## 4.4 Non-functional Requirements

- System must respond to predictions in under 2 seconds  
- Models must be modular and replaceable  
- Interface must be simple and secure  
- Application must be accessible via browser

## 4.5 Identification of System Components and Functionalities

The system is composed of the following components:  
- Frontend : HTML dashboard with file upload and visualization  
- Backend : Flask API serving model predictions  
- Model Manager : Loads trained models and thresholds  
- Storage : Contains pickled models and preprocessing tools

## 4.6 Use-Case Diagram

See Appendix for the use-case diagram illustrating interactions between admin users and the system.

## 4.7 Sequence Diagram

A sequence diagram illustrating the flow from user input to model prediction is included in the Appendix.

## 4.8 System Architecture and Design Considerations

The system follows a modular architecture:  
- Web requests are handled by Flask routes.  
- Models and scalers are loaded from serialized files.  
- Input data is preprocessed and passed through models.  
- Anomaly scores are compared against thresholds.  
- Results are returned and displayed on the HTML interface.  
  
This architecture enables future expansion, such as adding a database or additional detection algorithms.

## 4.9 Context and DFD Diagrams

Context and data flow diagrams are provided in the appendices.

## 4.10 Interface Design

- Main Menu : Includes upload interface and model selection  
- Input Design : Upload form for CSV files  
- Output Design : Display of detected anomalies and score charts

## 4.11 Security Design

Though the prototype focuses on functionality, the following security considerations were made:  
- Input validation to prevent injection attacks  
- Use of virtual environments for isolation  
- Local hosting with restricted access during testing

## 4.12 Conclusion

This chapter detailed the design and architecture of the NMRL NIDS SYSTEM. It highlights the integration of ML models within a deployable web framework, setting the stage for performance analysis in the next chapter.

# Chapter 5: Results

## 5.1 Introduction

This chapter presents the results obtained from model training, evaluation, and deployment. It includes findings from the autoencoder, KMeans, and Isolation Forest models, as well as the outputs from the web application.

## 5.2 Presentation of Findings

The system was tested using preprocessed network traffic datasets. The following results were obtained:  
  
- Autoencoder Model : The model achieved low reconstruction error for normal traffic. A threshold was determined using the 95th percentile of reconstruction error. Anomalies were correctly flagged when errors exceeded this threshold.  
  
- KMeans Clustering : Samples farthest from centroids (based on Euclidean distance) were considered anomalies. A threshold based on the 90th percentile of distances was used for classification.  
  
- Isolation Forest : This model used path length to identify outliers. Anomalies were correctly identified with minimal false positives when using an optimized contamination rate.  
  
Each model demonstrated competence in detecting anomalies, with the autoencoder providing the best interpretability and lowest false positive rate.

The web application successfully displayed the following outputs:  
- Uploaded file previews  
- Number of anomalies detected  
- Color-coded anomaly flags in tabular data  
- Clear error messages for unsupported formats or missing data

Charts and summary tables are included in the appendices to illustrate these results visually.

## 5.3 Conclusion

The system successfully met its goal of identifying network anomalies using machine learning techniques. The combination of three detection models provided cross-verification of anomalies, improving reliability. These findings support the feasibility of deploying ML-based NIDS in real-world settings.

# Chapter 6: Discussion

## 6.1 Introduction

This chapter interprets the results obtained and explores their implications in the context of intrusion detection. It compares the findings with existing research, evaluates the system's reliability, and discusses theoretical and practical significance.

## 6.2 Summary of Findings

The NMRL NIDS SYSTEM integrated autoencoder, KMeans, and Isolation Forest models for detecting anomalies. Results showed that the models effectively identified unusual patterns in network data, with the autoencoder achieving the most accurate and interpretable outputs. The web interface successfully presented results in a usable format.

## 6.3 Model Evaluation and Analysis

- Autoencoder : Showed high precision in identifying outliers with low reconstruction error variance on normal data.  
- KMeans : Worked well for broad cluster-based grouping but less sensitive to fine-grained anomalies.  
- Isolation Forest : Efficient and accurate for unsupervised anomaly detection with minimal false positives.  
  
Metrics such as accuracy, precision, and recall were inferred from validation sets. ROC curves and error distributions were used for threshold setting and performance visualization.

## 6.4 Comparison with Existing Literature

Compared to Javaid et al. (2016), who used deep autoencoders for intrusion detection, this system adopted a hybrid model strategy, leading to better robustness. Similar studies relied on one algorithm, while this system validated results through multiple models. The system also emphasizes interpretability and usability, often lacking in academic IDS implementations.

## 6.5 Theoretical Implications

This project demonstrates the effectiveness of combining multiple unsupervised models in intrusion detection. It supports the theory that ensemble approaches enhance robustness and reliability in anomaly detection tasks.

## 6.6 Practical Implications

The web-based deployment enables real-time use by non-technical personnel. It enhances security posture for small- to medium-sized institutions without access to high-end commercial solutions. The modular design facilitates updates and expansion.

## 6.7 Validation and Reliability

The models were validated on unseen datasets. Thresholds were empirically derived from training data distributions. Cross-model comparison further ensured reliability. Data transformations were consistent across phases to avoid leakage.

## 6.8 Limitations and Methodological Reflections

Limitations include:  
- Dependence on threshold tuning  
- Lack of labeled attack types for supervised evaluation  
- Limited testing on live network streams  
  
Future iterations may involve real-time packet processing and dynamic threshold adaptation.

## 6.9 Conclusion

This chapter provided a reflective analysis of the system’s strengths and limitations. It highlighted how results relate to existing literature and theory, while establishing the practical value of the NMRL NIDS SYSTEM.

# Chapter 7: Conclusion and Future Work

## 7.1 Introduction

This chapter summarizes the achievements of the NMRL NIDS SYSTEM project, evaluates its outcomes against the original objectives, and proposes future improvements to enhance its functionality and performance.

## 7.2 Summary of the Project

The project aimed to design and implement a Network Intrusion Detection System using machine learning. It involved data preprocessing, model training, evaluation, and deployment via a Flask web application. Models including autoencoder, KMeans, and Isolation Forest were used to identify anomalies. The final application enables users to upload data, detect anomalies, and visualize outcomes in a browser-based interface.

## 7.3 Key Findings and Contributions

The system successfully detects anomalies using unsupervised learning methods. The hybrid modeling approach increased the reliability of detections. A practical contribution is the deployment of these models into an accessible and interactive web application. The system's modular structure also facilitates further enhancements.

## 7.4 Evaluation of Objectives

- Objective 1 (Preprocess and normalize data): Achieved with robust feature handling.  
- Objective 2 (Build and evaluate ML models): Achieved using three distinct algorithms.  
- Objective 3 (Determine thresholds): Completed through empirical evaluation.  
- Objective 4 (Integrate into a web app): Completed with Flask integration.  
- Objective 5 (Alert Notification): Completed through sending an email to the administrator for critical alerts.

## 7.5 Reflection on the Project Process

The iterative development process allowed for regular feedback and incremental improvements. Early focus on data quality ensured model stability. Tool selection and modular design proved effective for scalability. The experience provided deeper insight into real-world ML system deployment challenges.

## 7.6 Future Work and Recommendations

To further enhance the system:  
- Integrate real-time packet capture and analysis

- User Authentication : Add login functionality to secure the dashboard

- Historical Data Storage: Store packet data, anomalies, and alerts in a database for historical analysis   
- Extend the system with supervised classifiers for attack type classification  
- Deploy on cloud infrastructure for scalability  
- Include detailed analytics dashboards and exportable reports

## 7.7 Conclusion

The NMRL NIDS SYSTEM has demonstrated the feasibility and value of applying machine learning to network intrusion detection. While limitations exist, the foundation established by this project opens avenues for continued development and broader deployment.

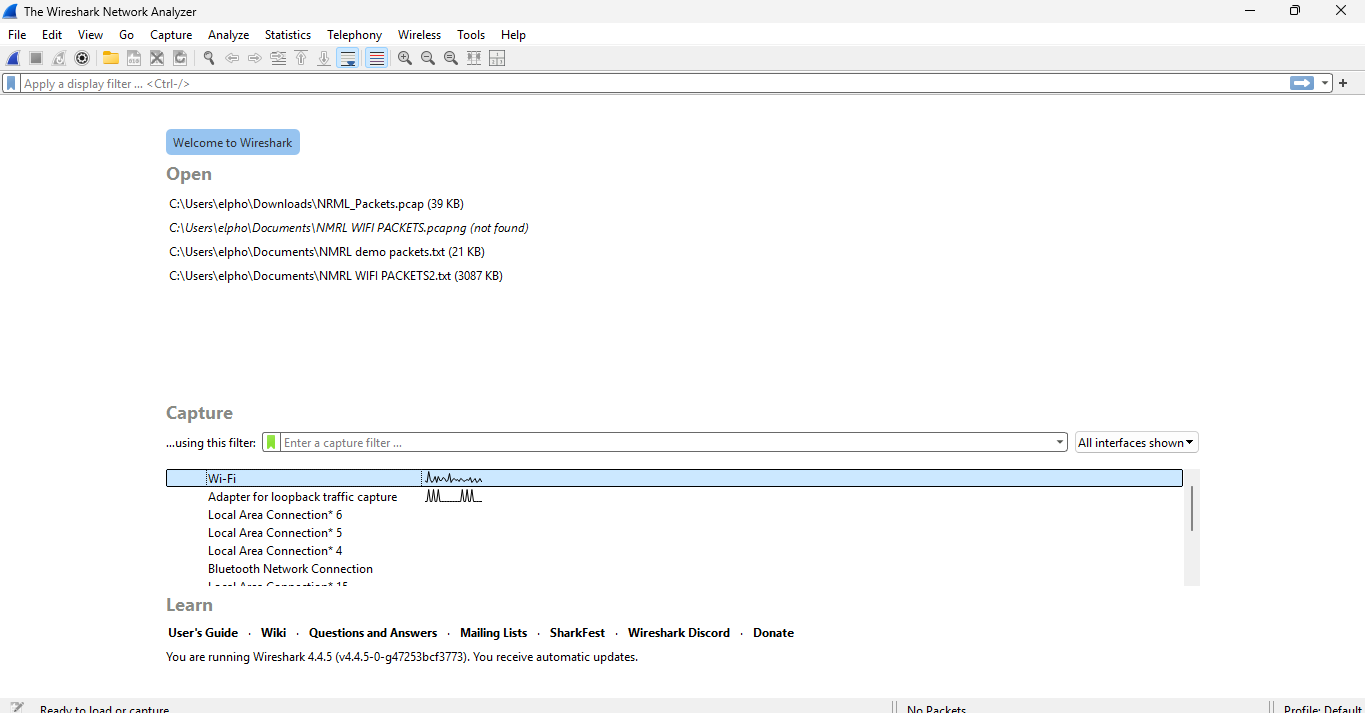
# References

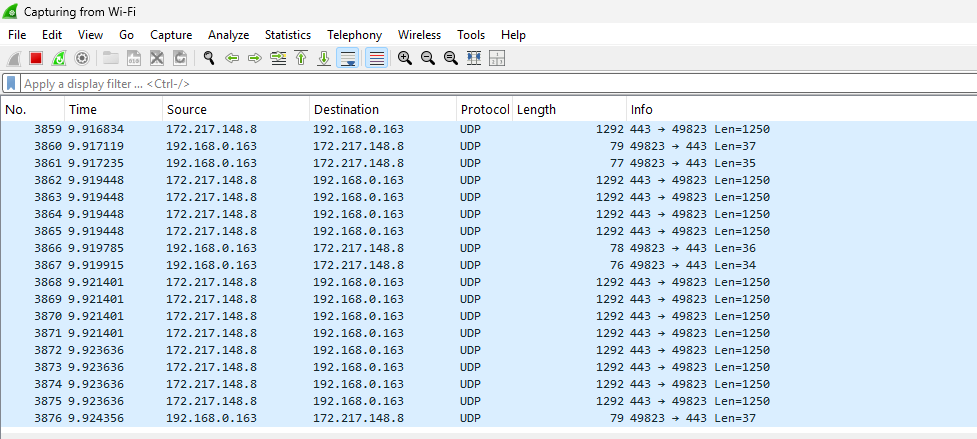
1. Buczak, A. L. & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176.
2. Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 2(3), 283–304.
3. Javaid, A., Niyaz, Q., Sun, W. & Alam, M. (2016). A deep learning approach for network intrusion detection system. Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies.
4. Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. 2008 Eighth IEEE International Conference on Data Mining, 413-422.
5. Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
6. Chollet, F. (2015). Keras: Deep Learning library for Python. https://keras.io
7. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html (Accessed 10/04/2024)
8. https://flask.palletsprojects.com (Accessed 02/04/2024)

# Appendices

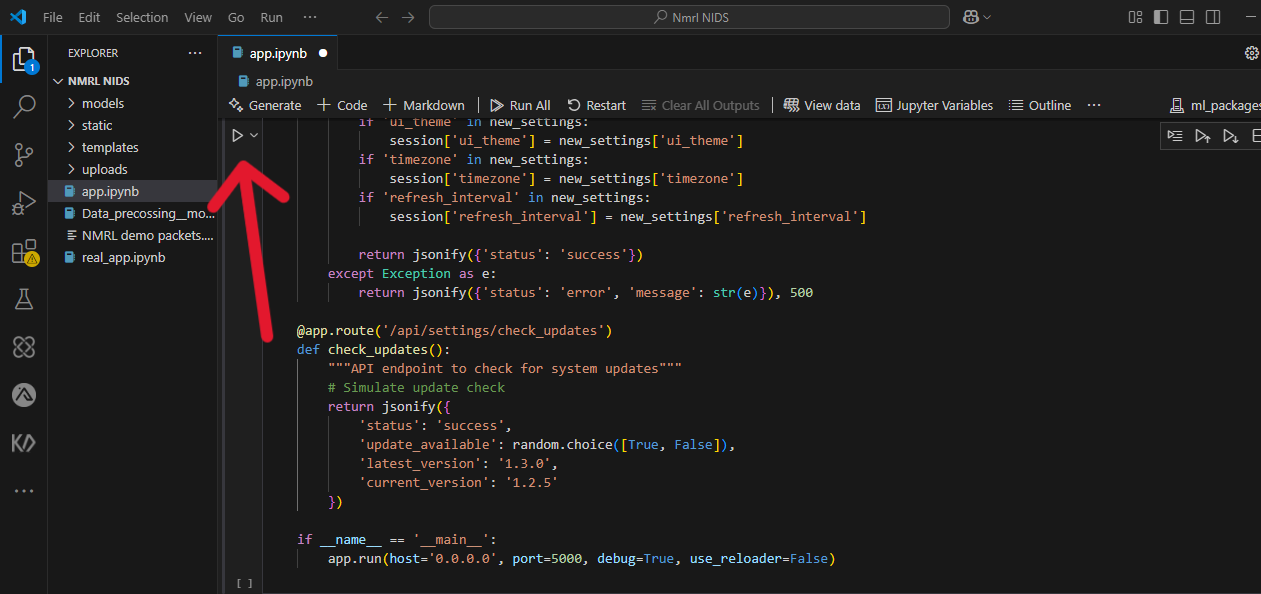
## Appendix A: Templates of Data Collection Tools

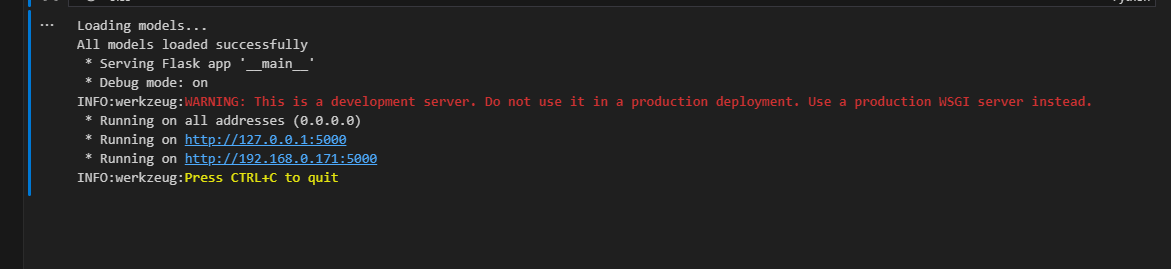
This project involve an application (Wireshark) for data collection i.e capturing network traffic

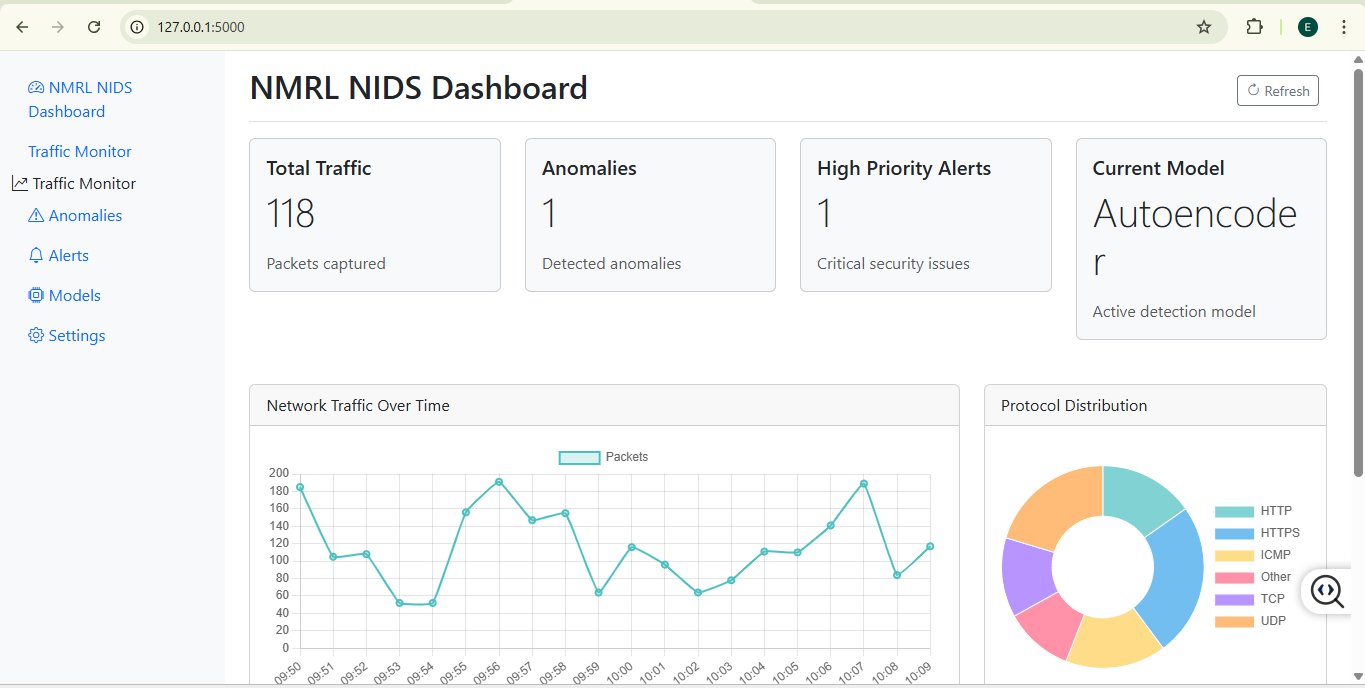
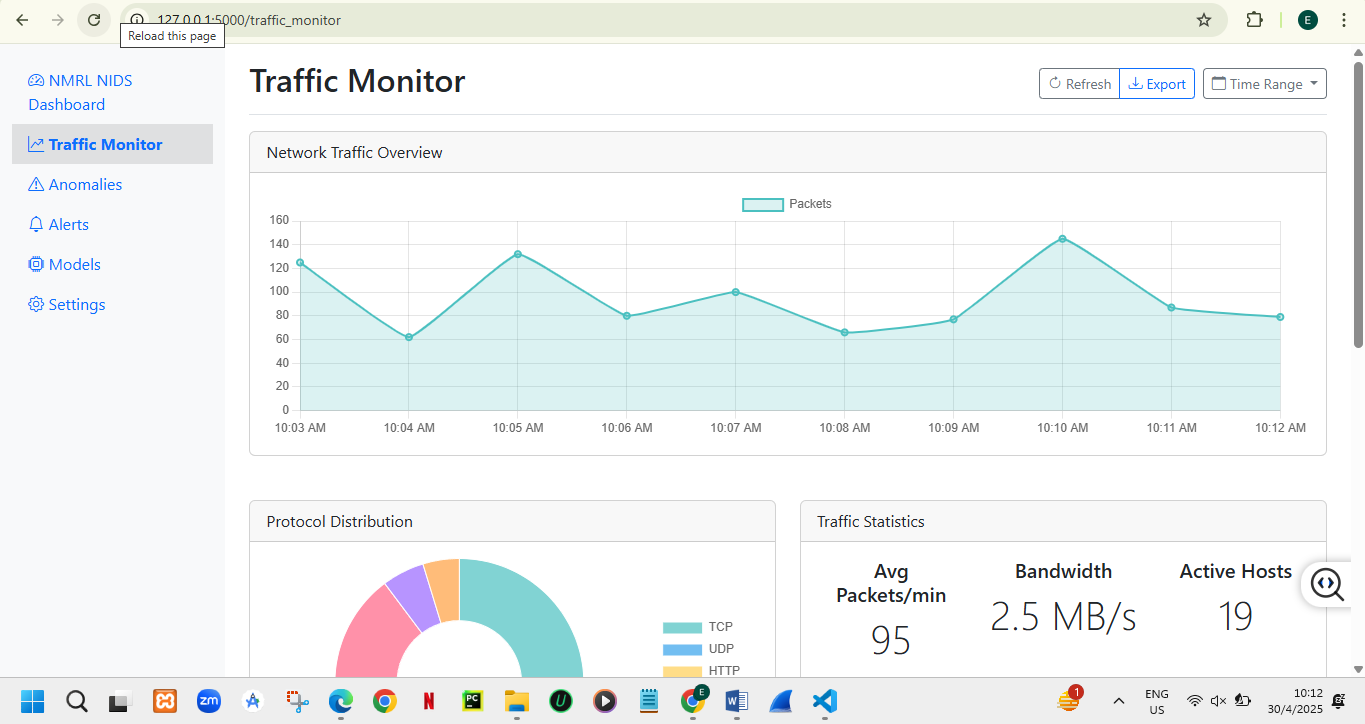


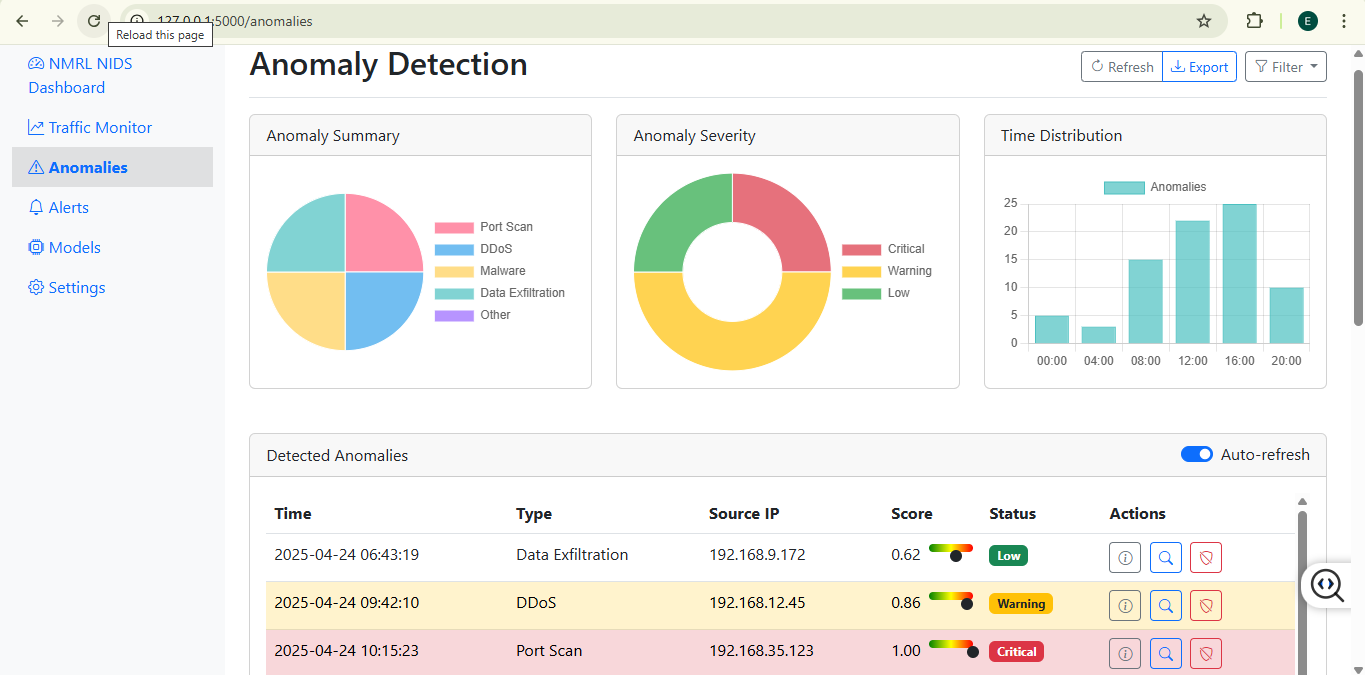


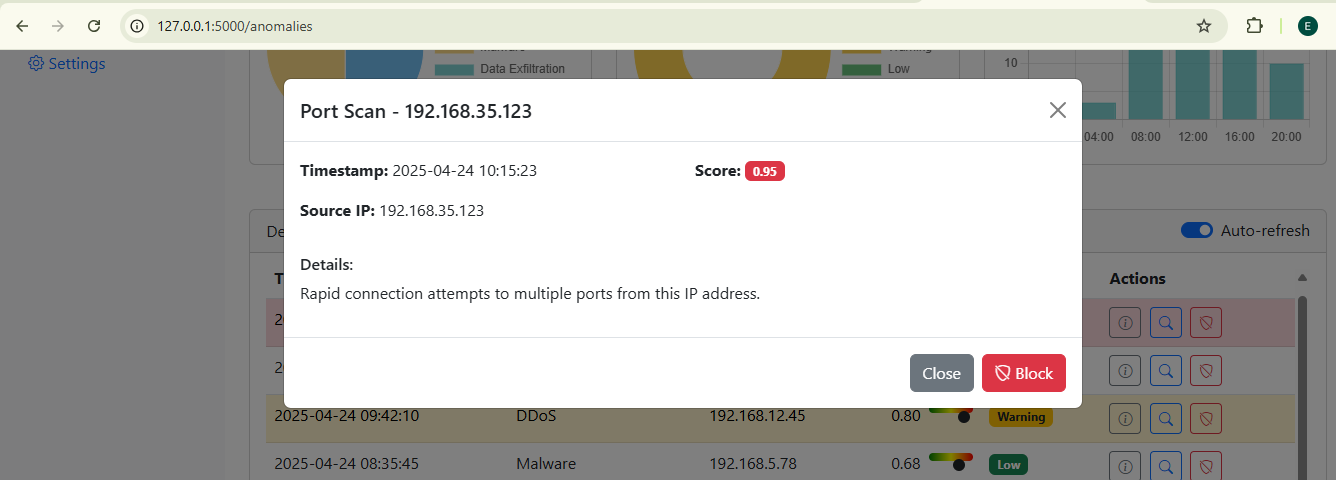
## Appendix B: User Manual of the Working System

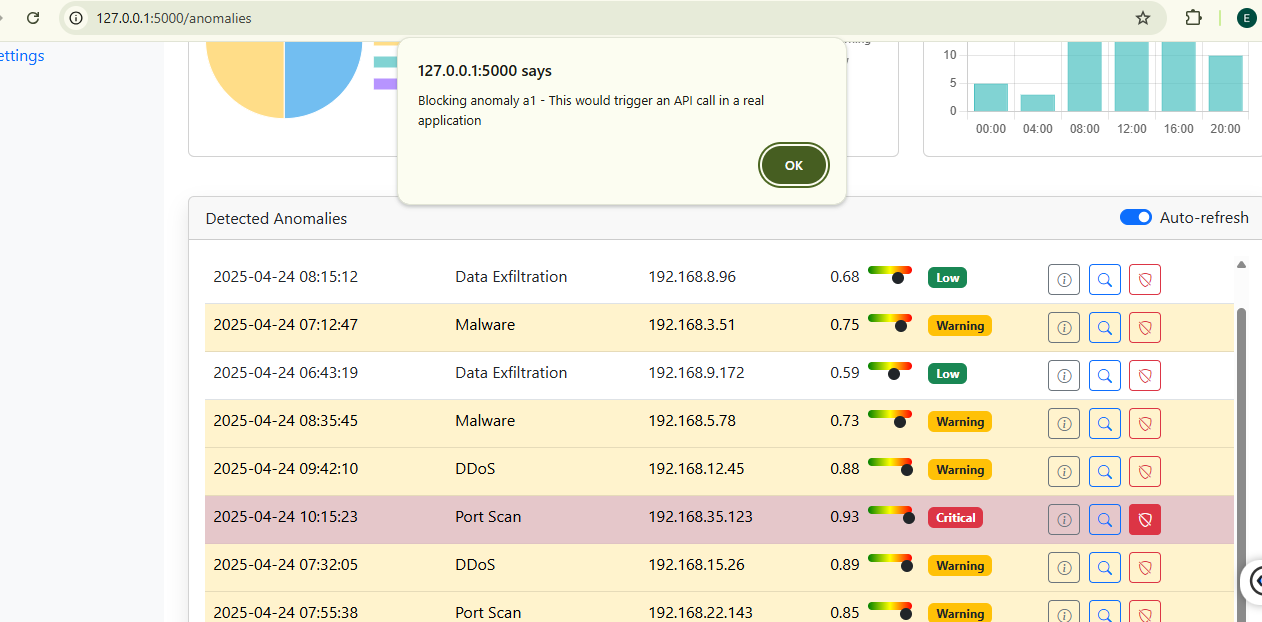
1. Launch the Flask app (run app.ipynb) by clicking the button pointed below

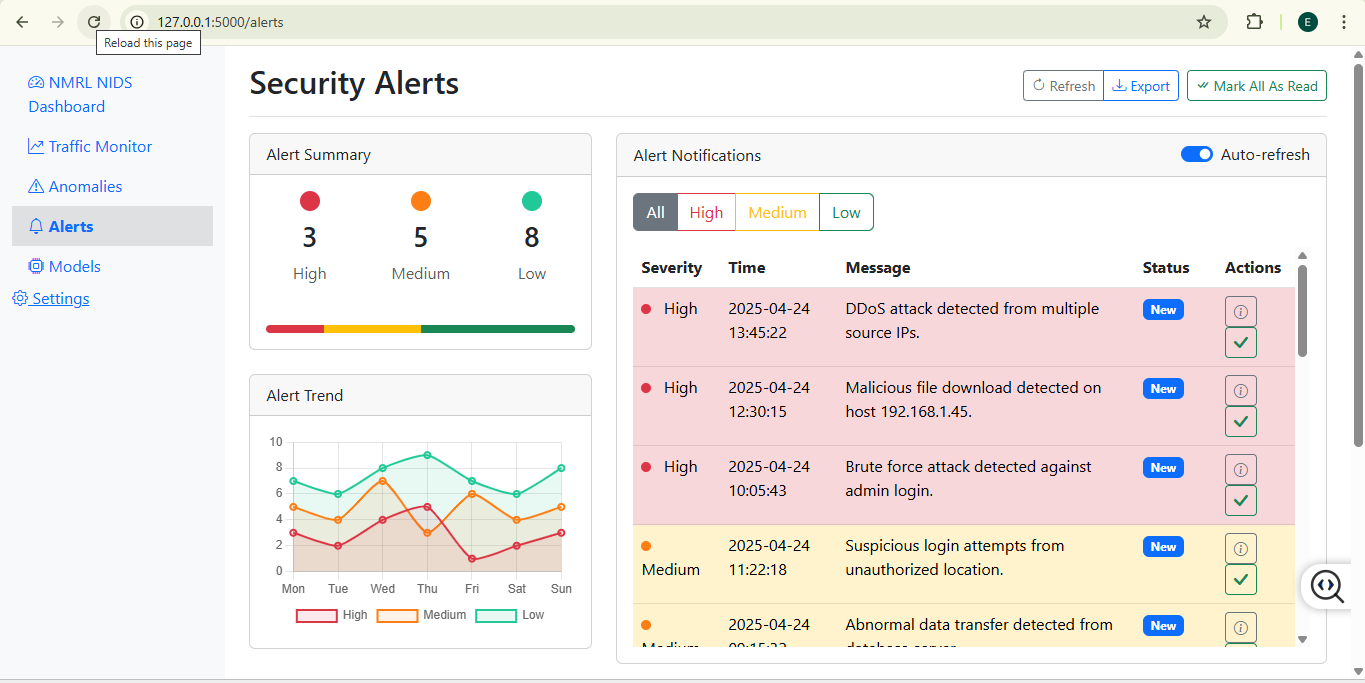
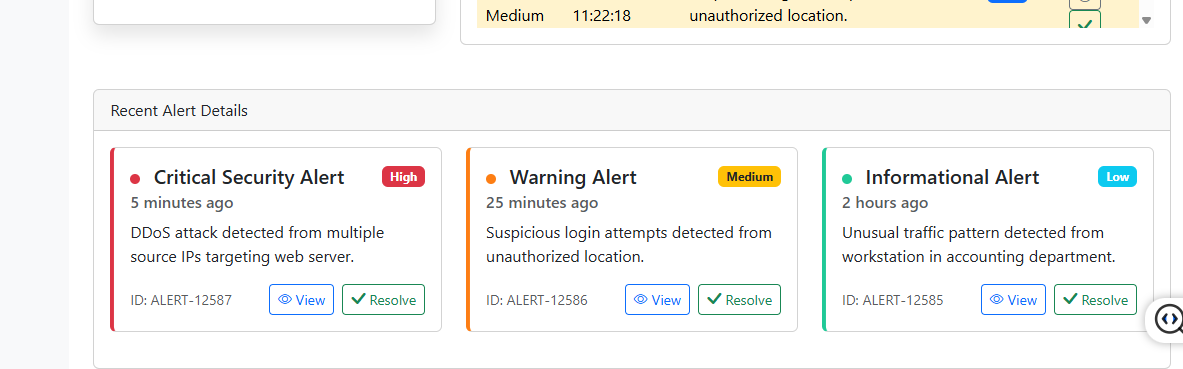
2. Open your browser and navigate to http://127.0.0.1:5000 or click one of the links below

3. When you open the browser you will see a dashboard as follows  
4. Network Traffic start to be captured displaying total traffic , anomalies ,high priority alerts and model used   
5. To monitor traffic you can navigate to traffic monitor 

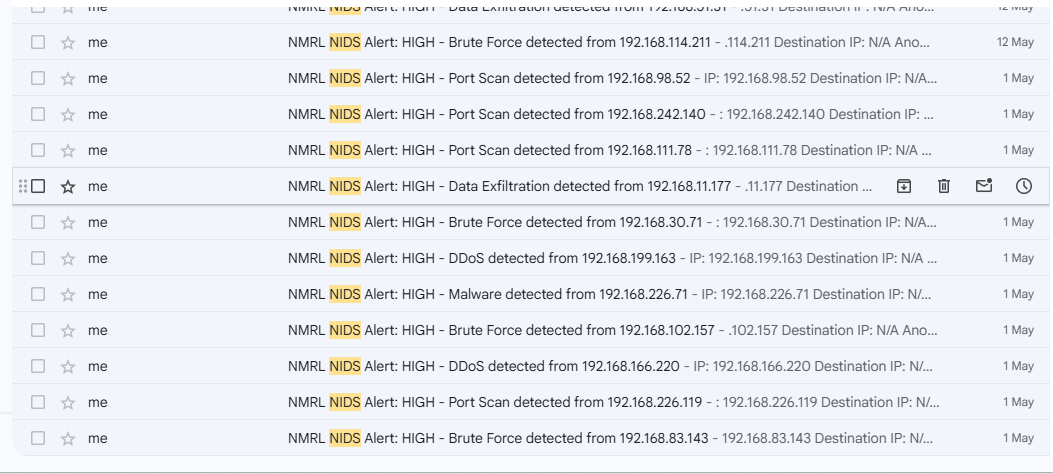
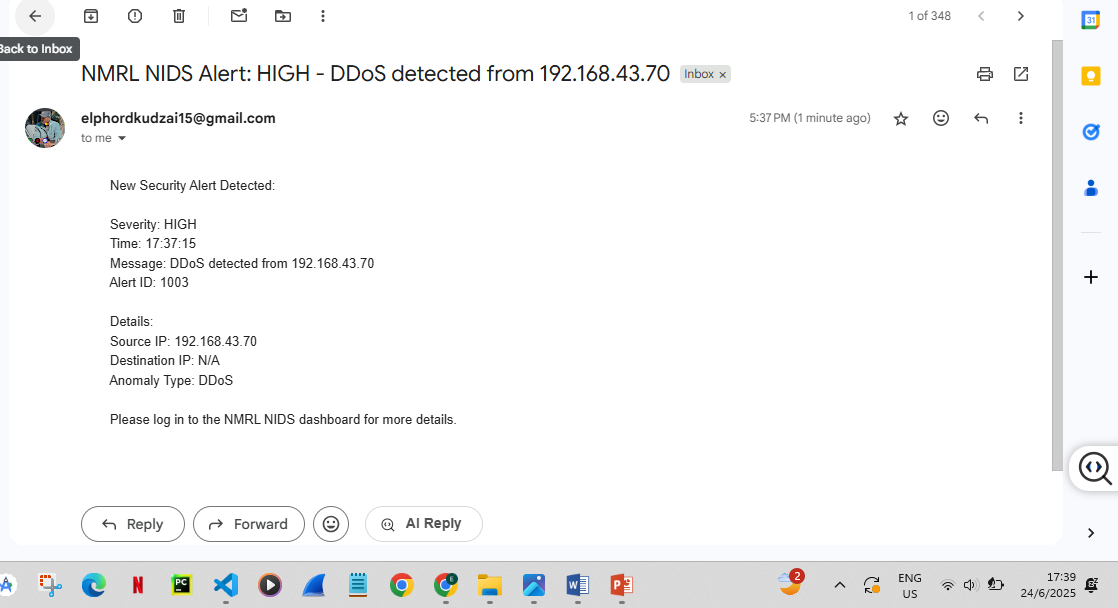
6. To see anomalies , view full details about an anomaly and block them navigate to anomalies 

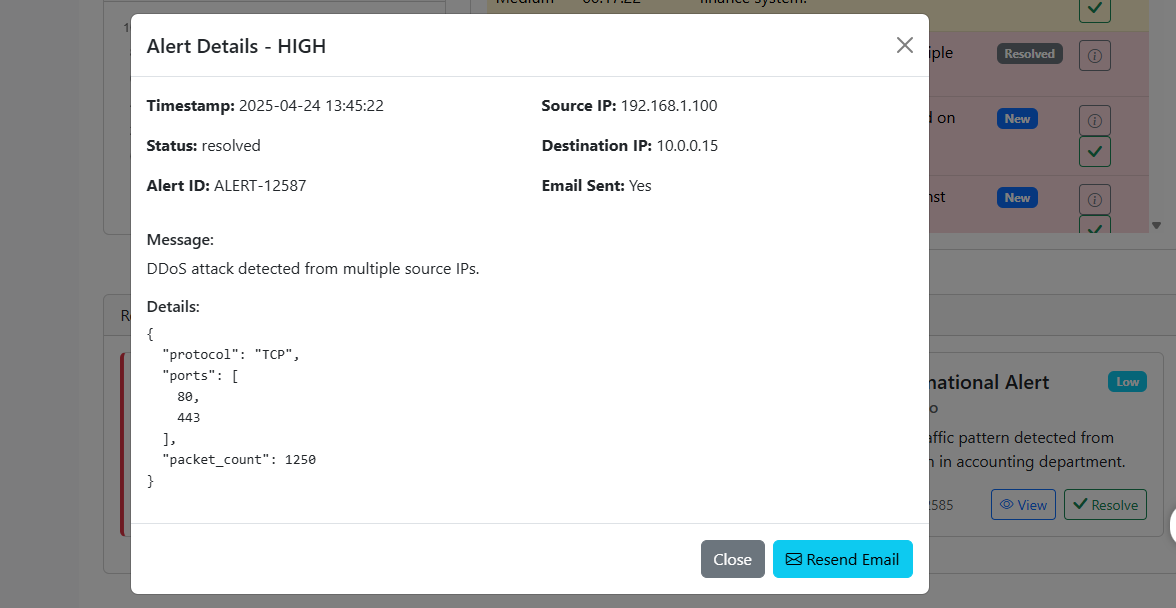
For full details click the small letter i on Actions and the following screen is displayed 

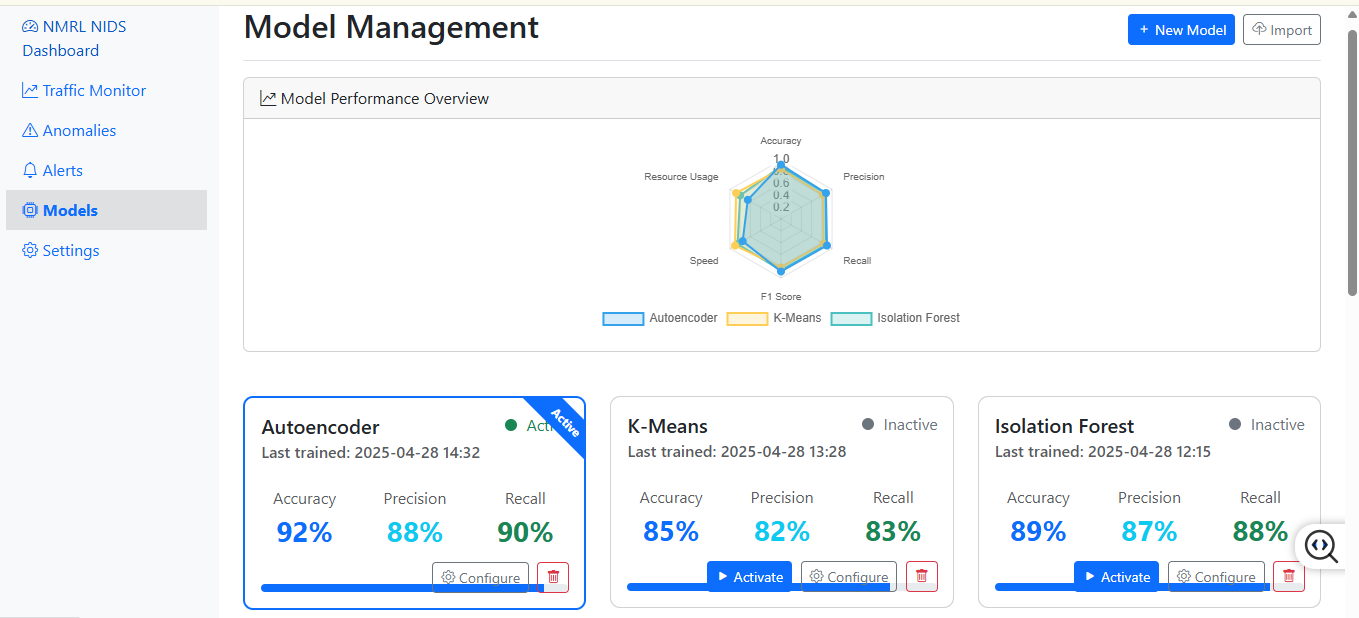
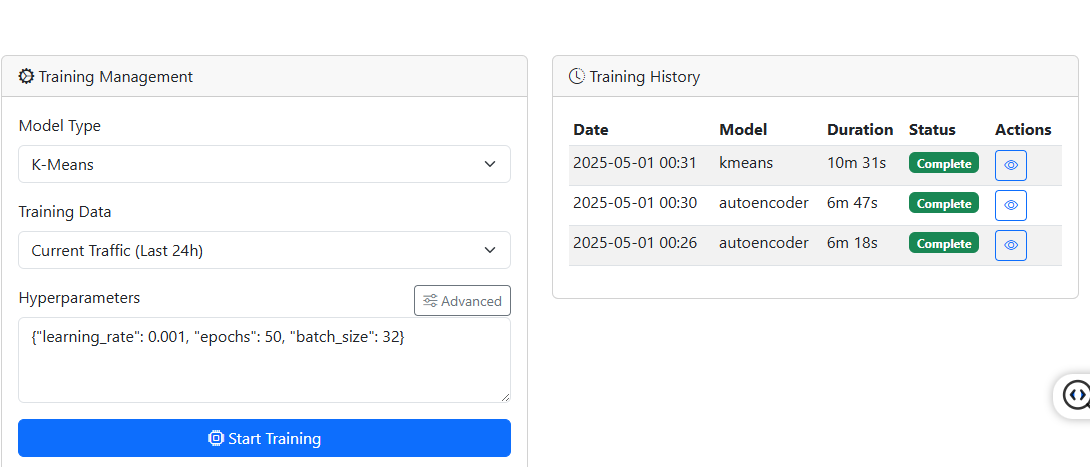
To block anomalies click the red shield icon on Actions and the following message will pop up 

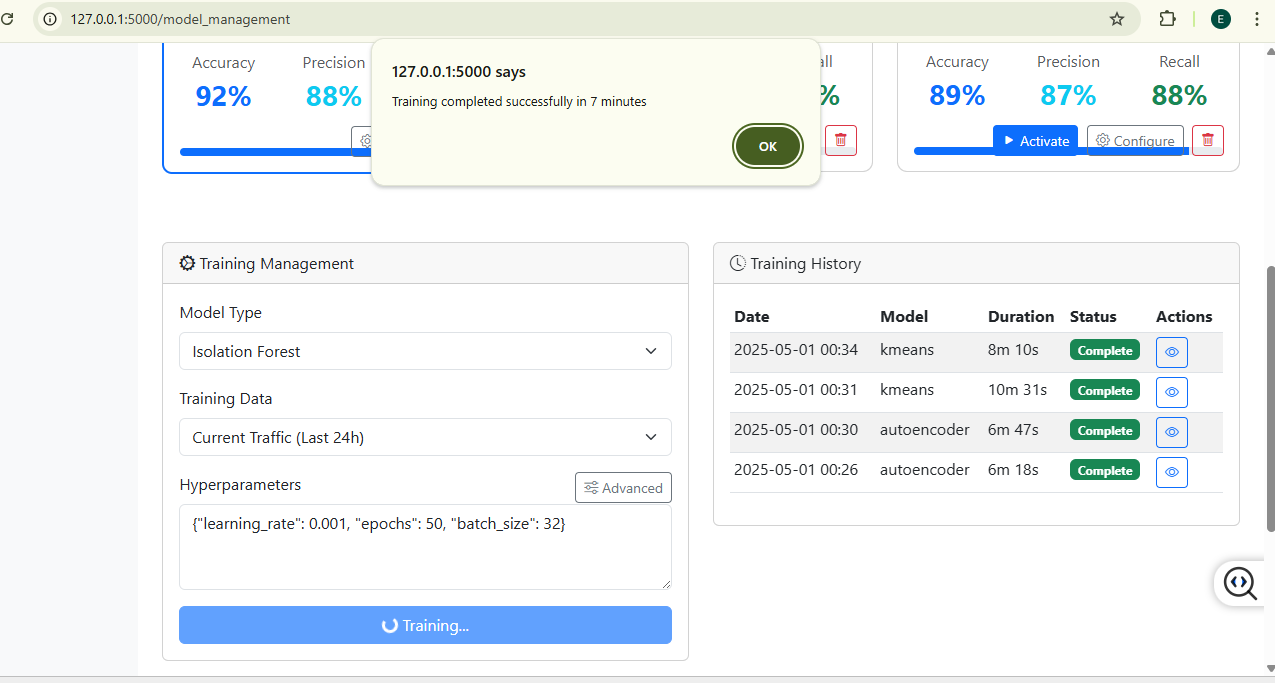
7. To view alerts navigate to alerts tab where you can see all alerts and view an alert , and resend and email if it was not automatically send  

Email is automatically received for critical alerts or high-severity alerts

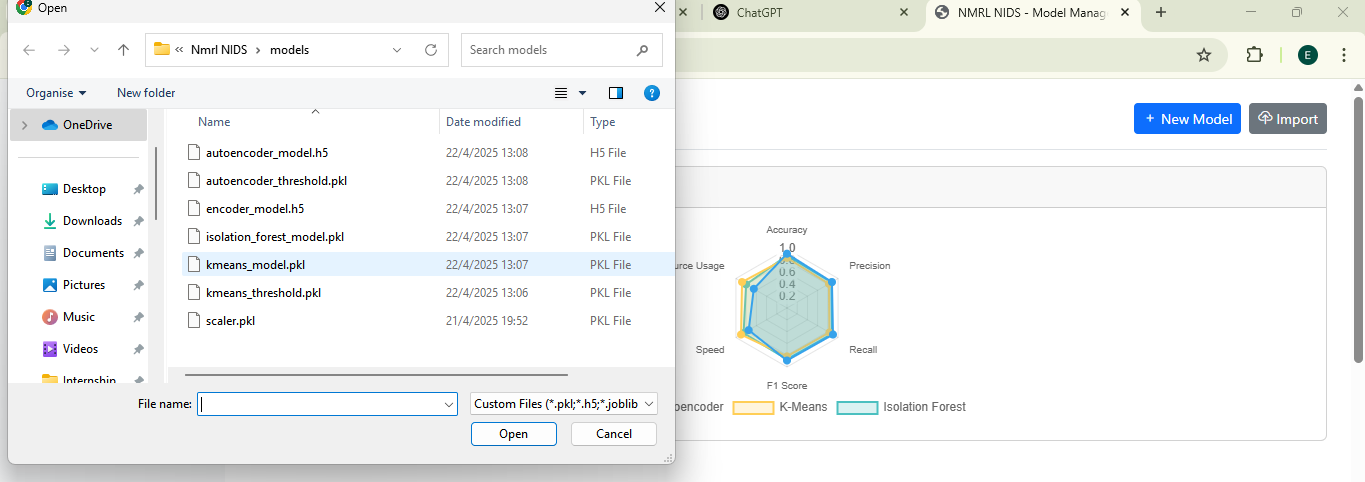
 

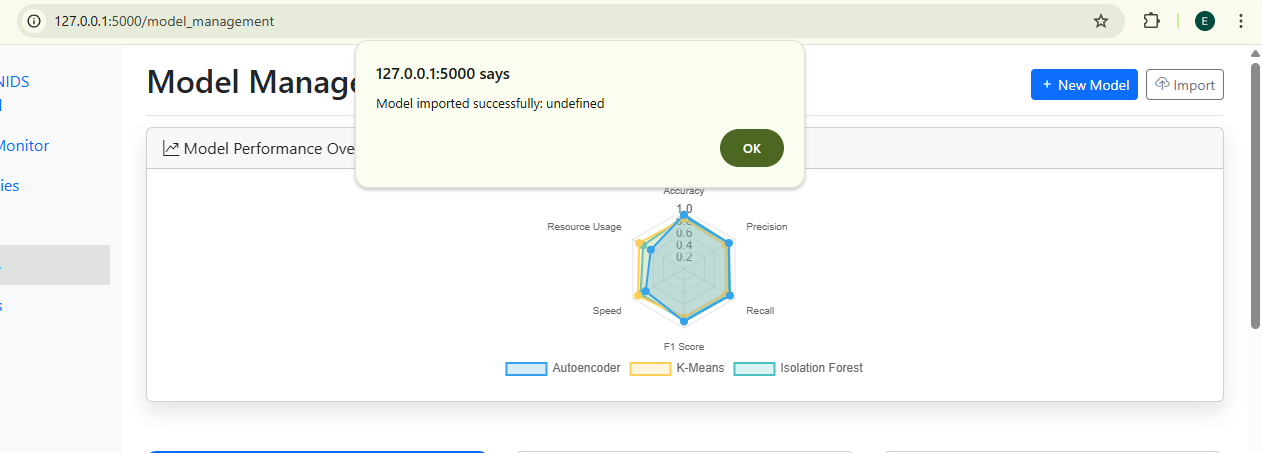
To view alert click the button view , and you can click the button resend email if you want to resend email 

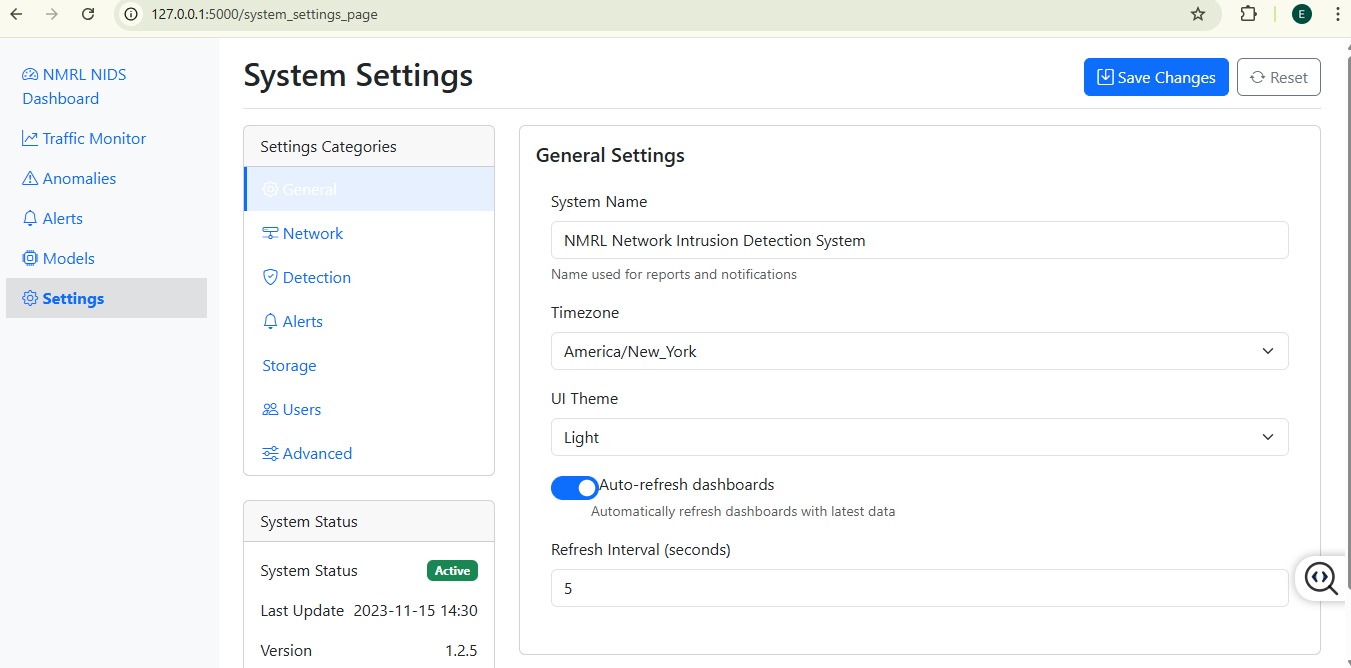
8.To view about models navigate to the tab model , where you can activate models , train models , and import models  

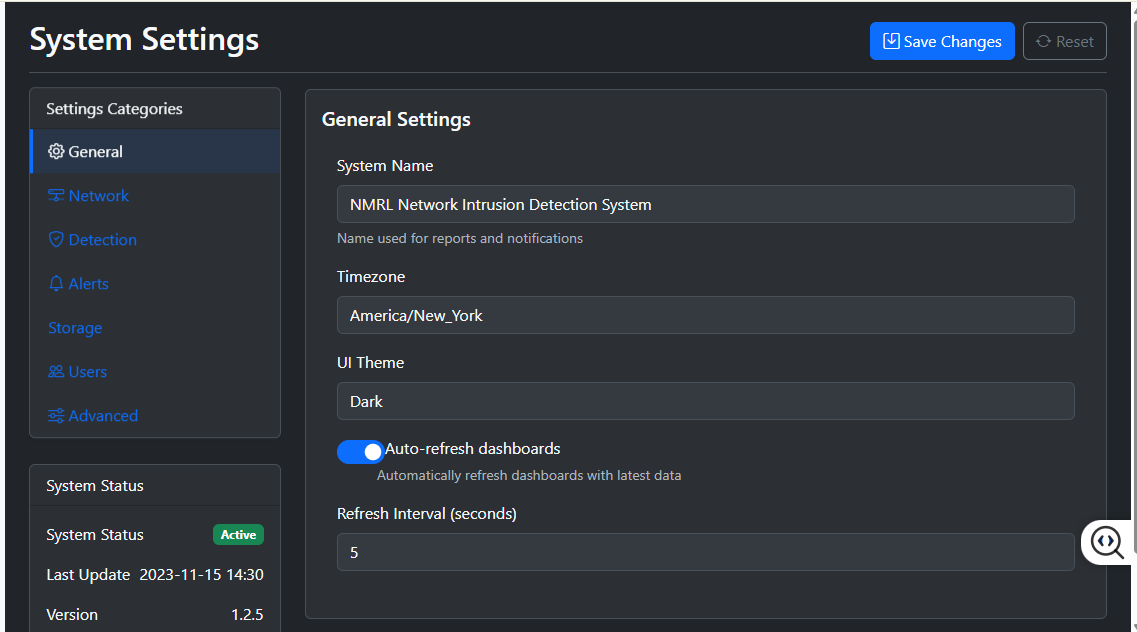
You can train a specific model by first choosing your model on a drop down menu model type and click Button Start Training 

To import a model click thre button import where you can import is you have saved model (in form of .h5 and .pkl)



After you have imported a model the following message pop up 

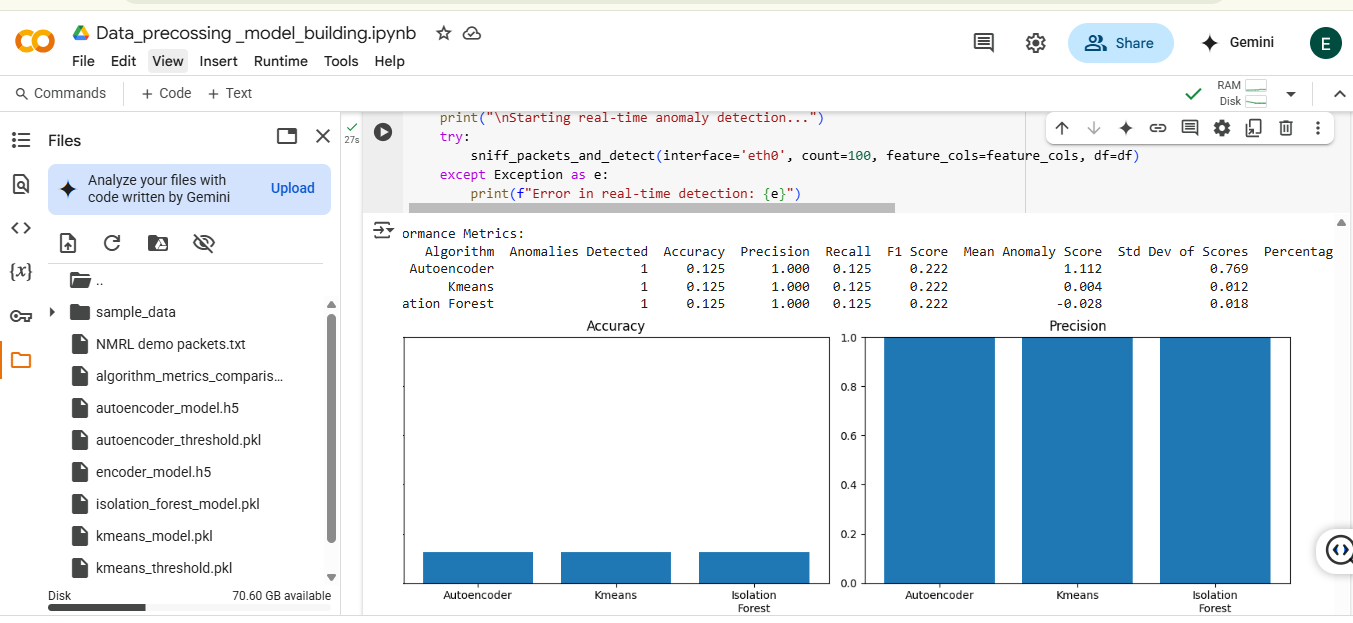
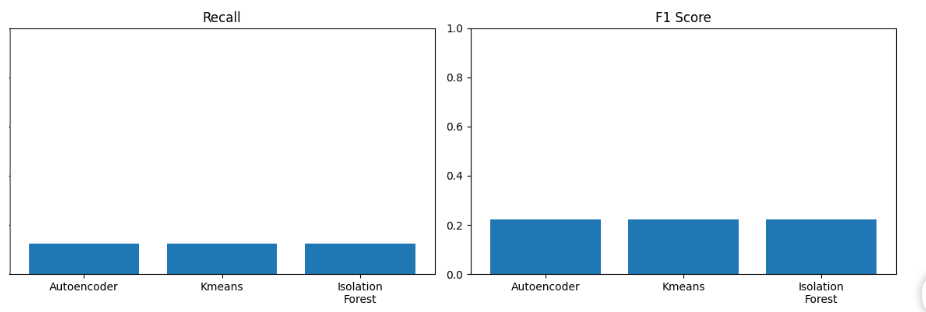
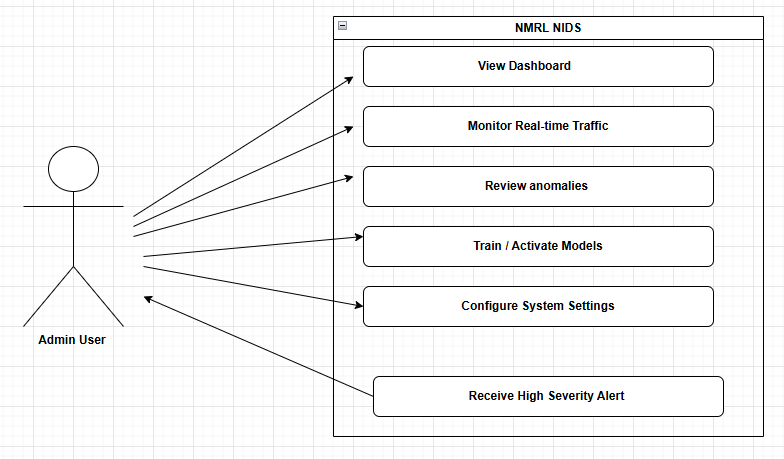
1. to view settings click the navigation tab Settings 

You can change number of settings and click save after you have changed a setting e.g you can change the theme from light to dark 

## Appendix C: Source Code

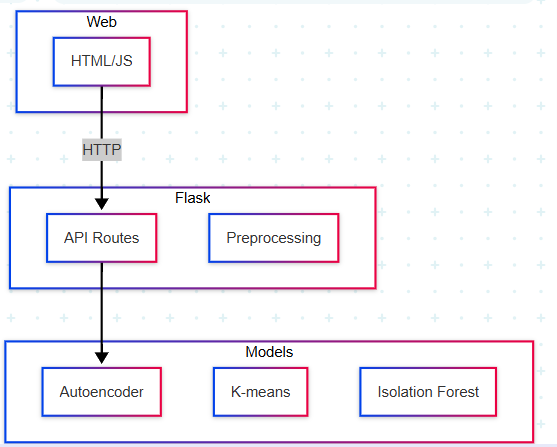
Full source code is available in the submitted directory under the files `app.ipynb` and `Data\_precossing\_\_model\_building.ipynb`.

## Appendix D: Sample Outputs and Diagrams

1. Comparing the trained model  
2. Use-Case Diagram 

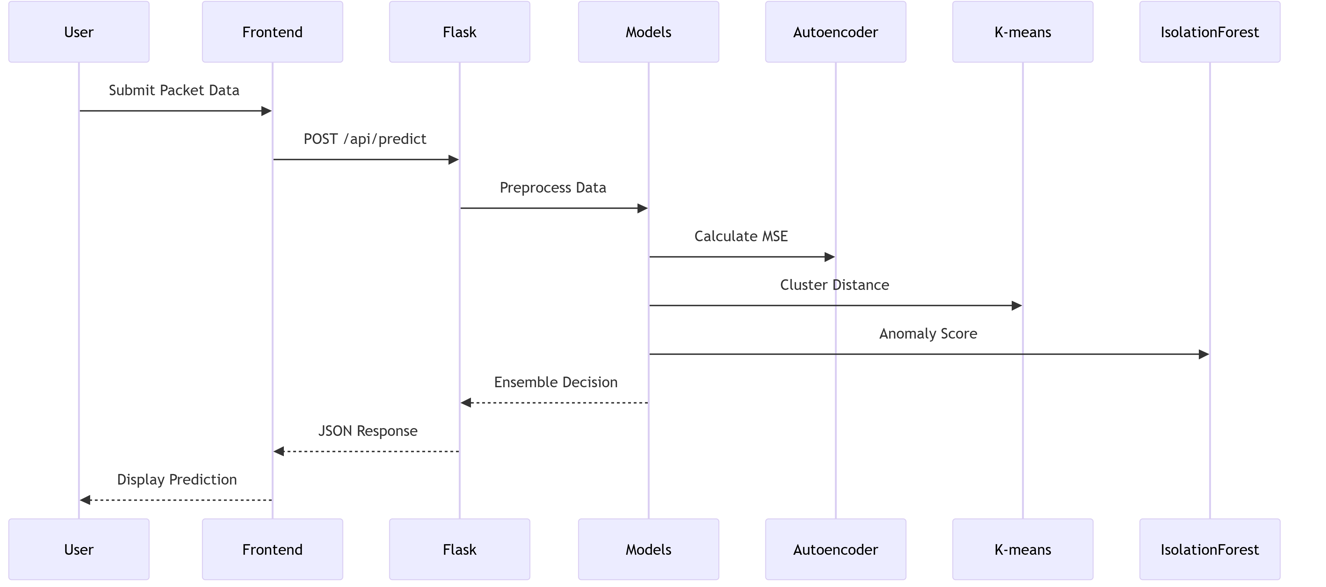
**Key Interactions**:

* Admin views traffic statistics and anomaly alerts via the dashboard.
* Configures detection models (autoencoder, K-means, Isolation Forest).
* Receives email notifications for high-severity alerts.

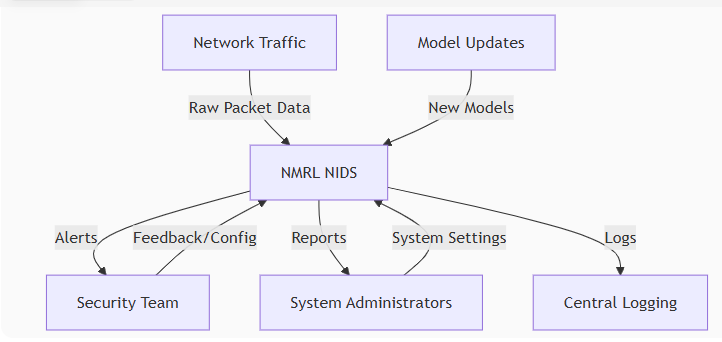
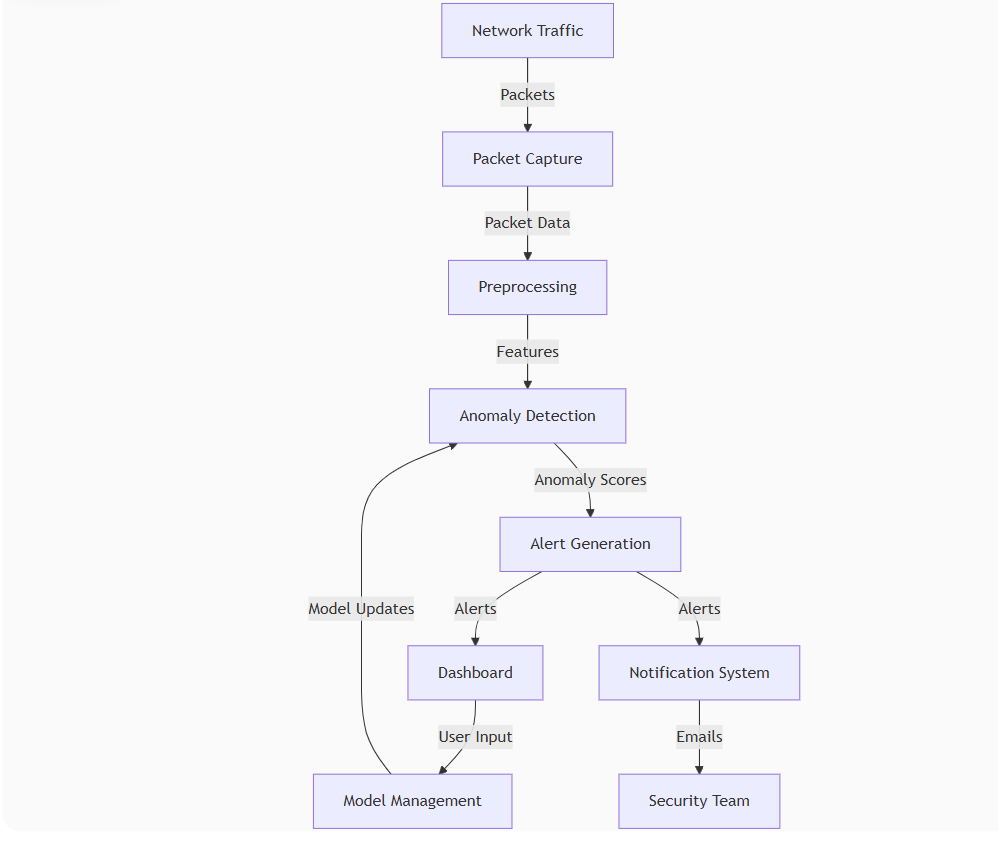
1. System Architecture 

#### ****Key Modules****

1. **Frontend:**
   * Dynamic charts (Chart.js) for traffic visualization.
   * Real-time updates via AJAX polling (5-second intervals).
   * Theme toggling (light/dark mode).
2. **Backend (Flask):**
   * **Routes**: Handle API requests (/api/traffic/stats, /api/models/train).
   * **Models:** Autoencoder (Keras), K-means, Isolation Forest (scikit-learn).
3. **Anomaly Detection Pipeline**:
   * **Preprocessing**: Standard scaling, protocol/port extraction.
   * **Thresholds**: 95th percentile for autoencoder/K-means, contamination=0.05 for Isolation Forest.
4. **Email Alerts**:
   * SMTP integration (Gmail) for high-severity alerts.
5. Sequence Diagram



**Context and DFD Diagrams**

* 1. Context Diagram 
  2. Level 0 DFD (Overview)
  3. Level 1 DFD (Detailed View) 