**<** **SecureNet IDS>**

**Design Document**

**Version 1.6**



**Group Id: <** F24PROJECTCE3A6 >

**Supervisor Name: <**Laraib Sana>

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 18/12/2024 | 1.0 | Introduction of the project | BC210424643 |
| 09/01/2025 | 1.1 | Created Entity Relationship Diagram (ERD) | BC210401276 |
| 13/01/2025 | 1.2 | Developed Database Design Diagram with schema details | BC210424643 |
| 20/01/2025 | 1.3 | Created Sequence Diagrams for system workflows | BC210401276 |
| 28/02/2025 | 1.4 | Designed Architecture Design Diagram outlining system layers | BC210424643 |
| 26/02/2025 | 1.5 | Developed Class Diagram defining system objects & interactions | BC210424643 |
| 28/02/2025 | 1.6 | Completed Test Cases for system validation | BC210424643 |

**Table of Contents**

1. [Introduction of Design Document](#One)
2. [Entity Relationship Diagram (ERD)](#ERD)
3. [Sequence Diagrams](#Six)
4. [Architecture Design Diagram](#Seven)
5. [Class Diagram](#class)
6. [Database Design](#databasedesign)
7. [Interface Design](#interfacedesign)
8. [Test Cases](#testcases)

1. **Introduction of Design Document**

The **Design Document** for the **SecureNet IDS** serves as a detailed blueprint that guides the implementation of the system. This document provides a structured approach to understanding the software architecture, data flow, system interactions, and database schema. It ensures that all stakeholders, including developers, designers, and testers, have a unified vision of the system's design before moving to the development phase.

This document consists of the following key components:

**1. Entity Relationship Diagram (ERD)**

The **ERD** provides a visual representation of the system’s data model by defining key entities and their relationships. It includes:

* **User:** Stores user credentials and role-based access control.
* **TrafficData:** Captures real-time network traffic, including source and destination IPs, protocols, and timestamps.
* **Model:** Stores details about trained machine learning models used for intrusion detection, including accuracy and precision metrics.
* **AnalysisResult:** Stores the classification results of analyzed network traffic, including detected threats and confidence scores.

The **ERD ensures proper data structuring** by defining primary and foreign keys and enforcing referential integrity.

**2. Sequence Diagrams**

Sequence diagrams depict the dynamic behavior of the system by illustrating interactions between different components. The diagrams included in this document cover:

* **Capturing Network Traffic:** Details how network packets are captured using Wireshark and stored for analysis.
* **Uploading Data for Analysis:** Shows the interaction between the user, file upload module, and server validation.
* **Preprocessing & Feature Engineering:** Demonstrates how network data is cleaned, encoded, and prepared for machine learning models.
* **Model Training & Evaluation:** Outlines the process of training machine learning models with preprocessed data and evaluating their performance.
* **Real-Time Intrusion Detection Analysis:** Shows how new network traffic is classified using trained models.
* **Display Results:** Depicts how classification results are retrieved from the database and displayed to the user.
* **Save or Export Results:** Details how the system allows users to export intrusion detection reports.

These diagrams **ensure a clear understanding of system interactions**, facilitating better communication among developers and stakeholders.

**3. Architecture Design Diagram**

The **system architecture** follows a **three-tier architecture** that ensures modularity, scalability, and maintainability:

* **Presentation Layer:** The web-based UI that allows users to upload network data and view intrusion detection results.
* **Business Logic Layer:** Contains core processing modules, including file validation, data preprocessing, model training, and real-time intrusion detection.
* **Data Layer:** Stores network traffic data, trained models, and analysis results in structured databases.  
  This architecture design ensures that **each layer is independent**, allowing future enhancements without affecting other system components.

**4. Class Diagram**

The **Class Diagram** defines the object-oriented structure of the system, including:

* **User Class:** Manages user authentication and profile updates.
* **TrafficData Class:** Represents captured network traffic and its associated attributes.
* **Model Class:** Stores information about trained machine learning models.
* **AnalysisResult Class:** Stores the classification results and provides methods to generate reports.  
  Each class defines its **attributes and operations**, ensuring a well-structured codebase.

**5. Database Design**

The database schema includes:

* **Primary Tables:** User, TrafficData, Model, AnalysisResult.
* **Relationships:**
  + TrafficData has a **one-to-many** relationship with AnalysisResult.
  + Model has a **one-to-many** relationship with AnalysisResult.
  + User can be linked to TrafficData (if needed) for user-specific traffic analysis.
* **Constraints:**
  + Primary keys (PK) ensure unique identification of records.
  + Foreign keys (FK) maintain **referential integrity** between related tables.  
    This database design ensures **efficient data storage and retrieval**, reducing redundancy while maintaining flexibility.

**6. Interface Design**

The document includes **mockups and UI designs** for the following system interfaces:

* Login & User Authentication Page.
* File Upload & Network Traffic Data Input Page.
* Intrusion Detection Dashboard displaying real-time analysis results.
* Report Export Page for saving results in CSV/PDF format.  
  These interfaces ensure an intuitive user experience while maintaining system functionality.

**7. Test Cases**

The document includes **test case scenarios** to ensure the system functions as expected. Each use case is tested for:

* Successful file uploads and validation.
* Accurate preprocessing and transformation of network data.
* Correct model training and evaluation.
* Accurate classification of network traffic as normal or malicious.
* Proper storage and retrieval of analysis results.
* Secure access control and authentication mechanisms.  
  These test cases help in **identifying potential issues early**, improving system reliability.

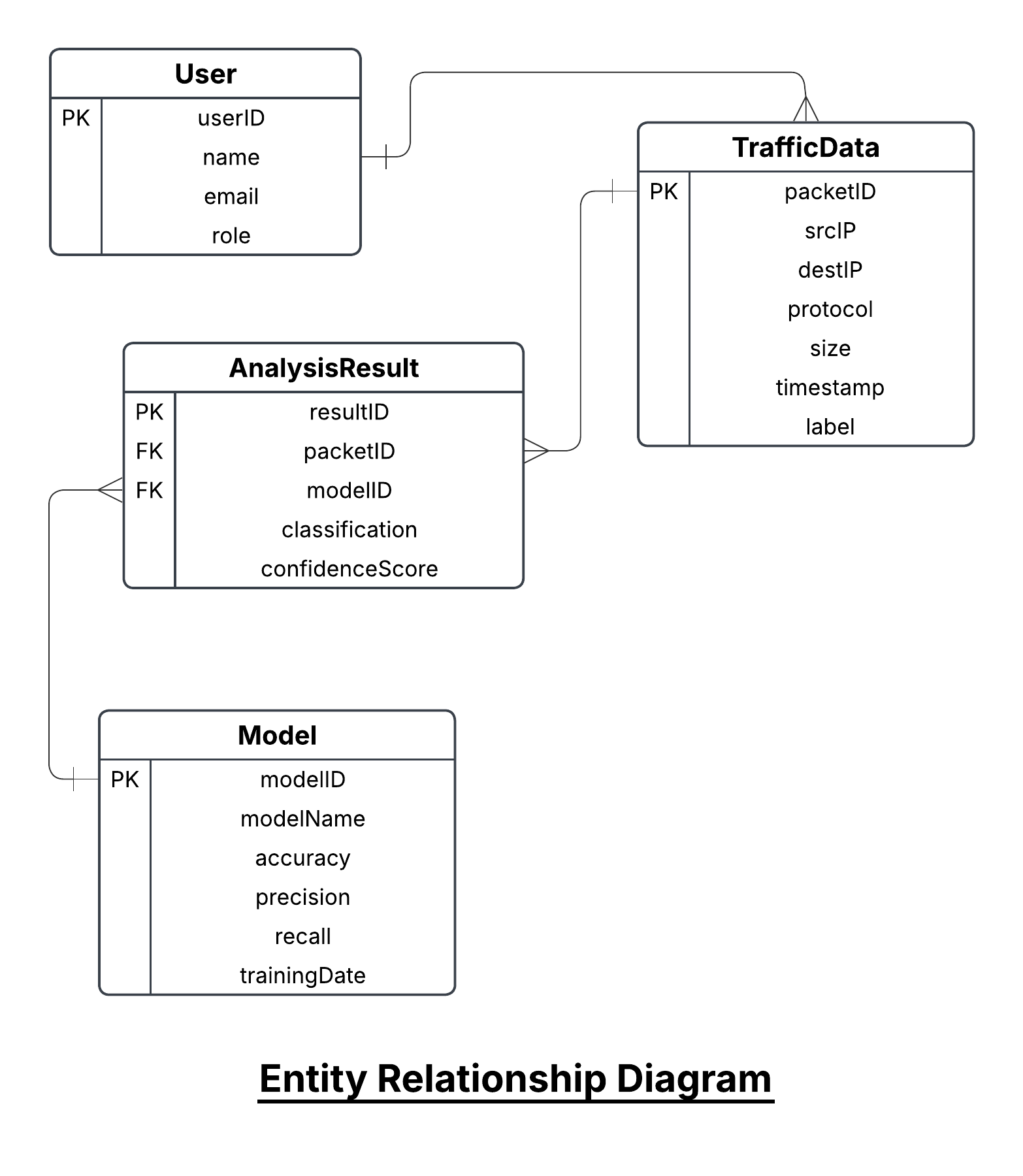
**Purpose and Benefits of the Design Phase**

The **design phase is a critical step** in the software development lifecycle (SDLC) as it provides a structured roadmap for implementation. The **key benefits** of this phase include:

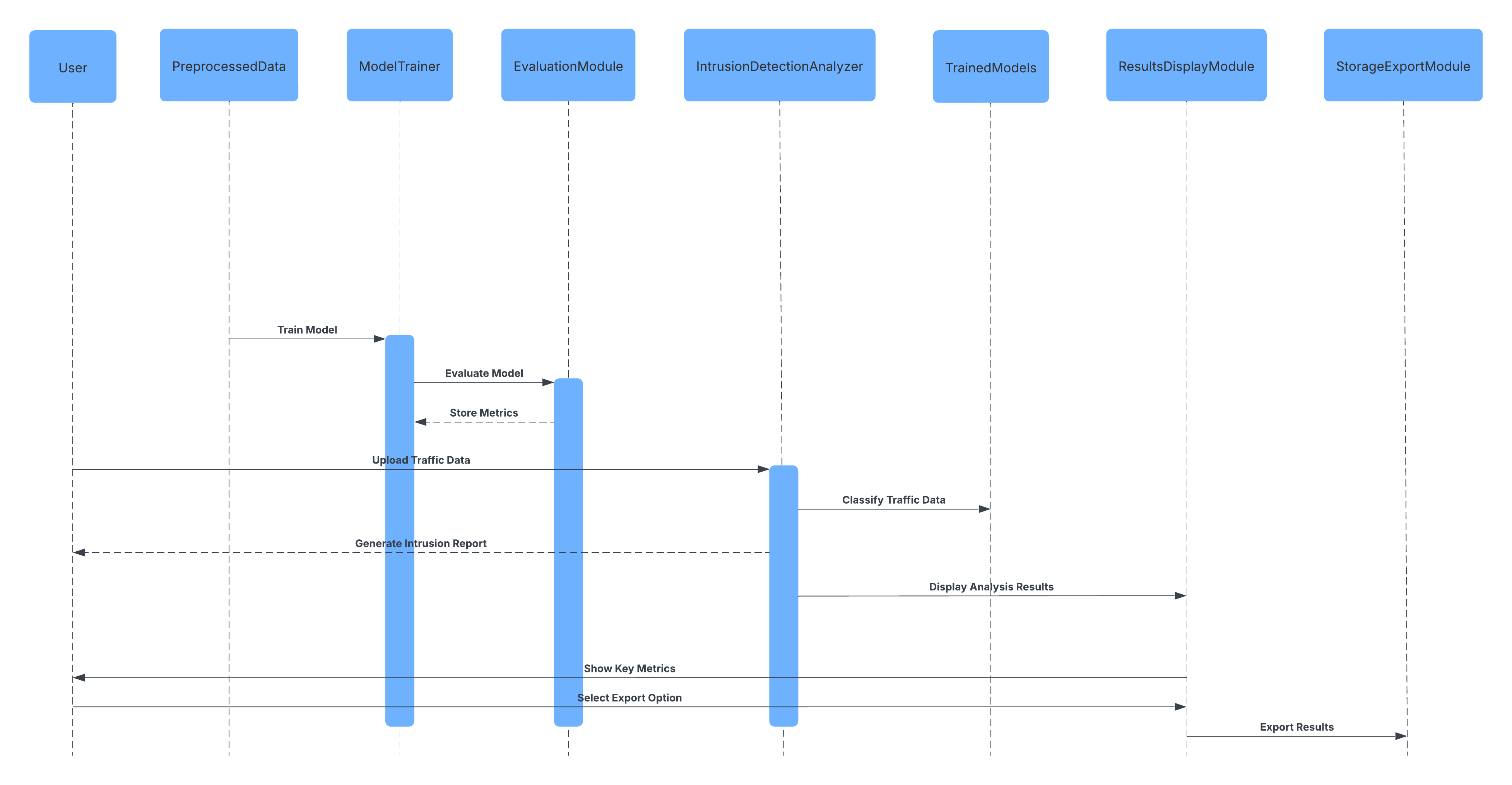
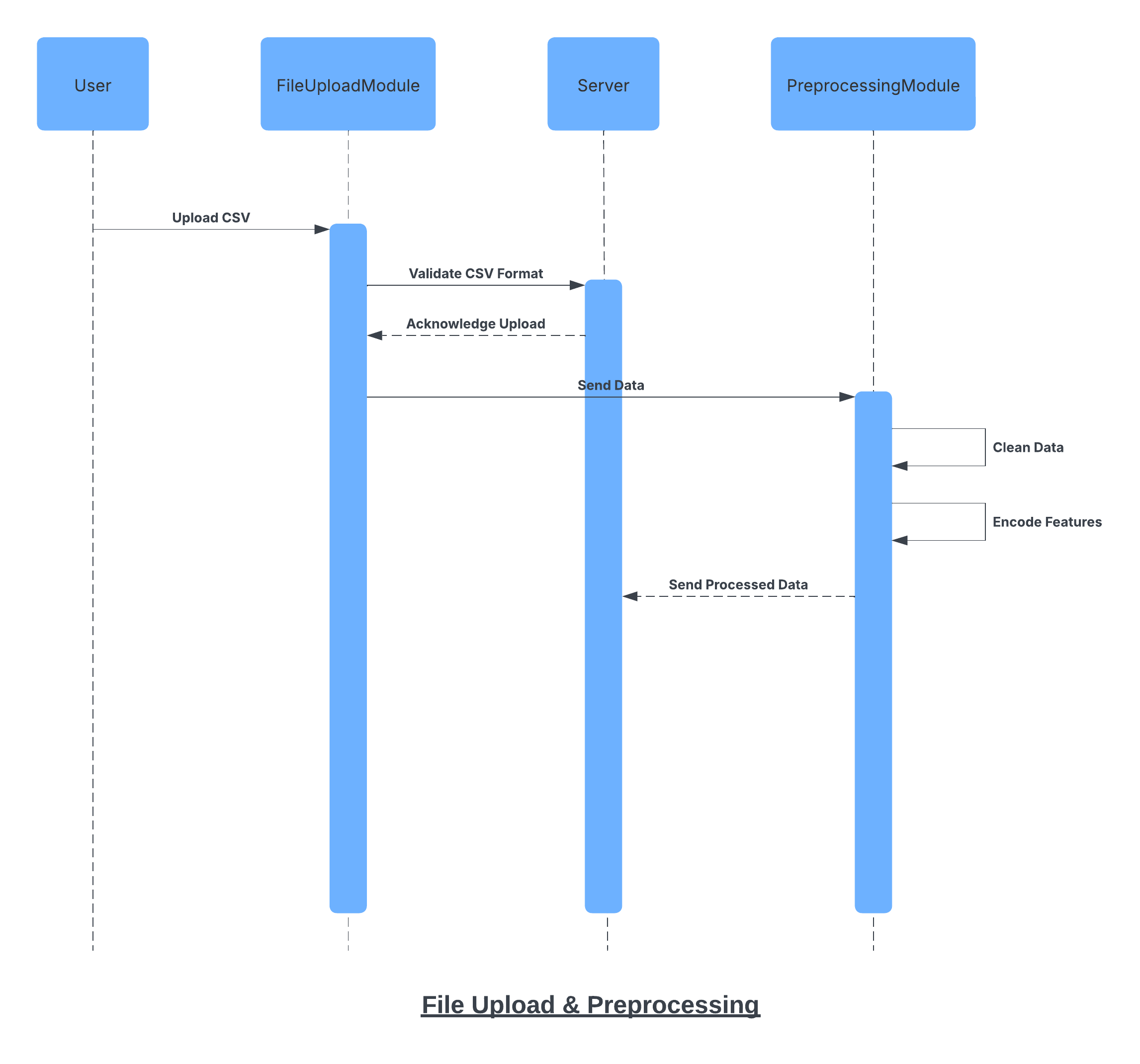
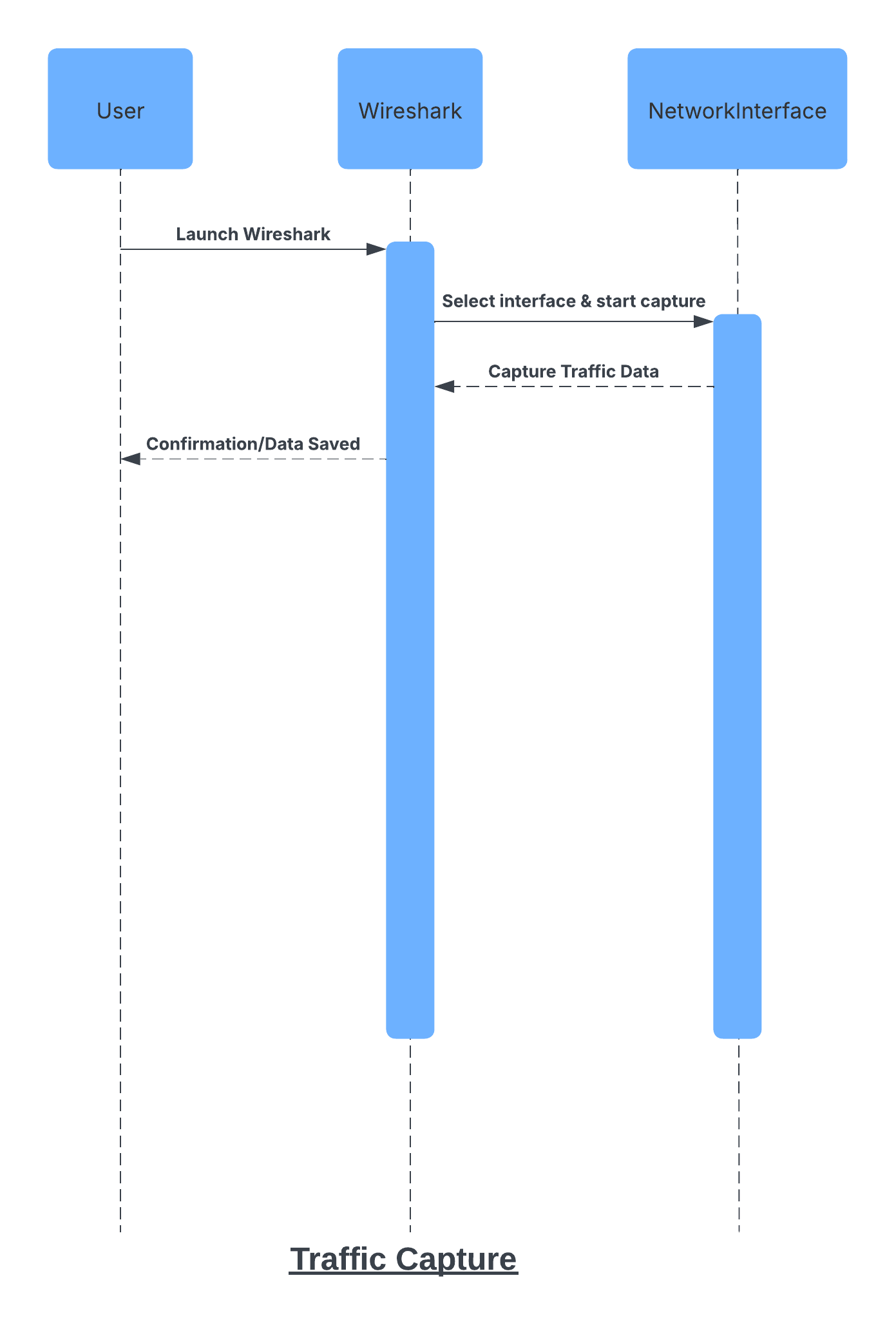
1. **Ensures System Alignment:**
   * Defines a clear **blueprint** for developers, testers, and designers, ensuring the final system meets all functional and non-functional requirements.
2. **Improves Efficiency and Maintainability:**
   * A well-structured architecture allows **easy debugging, upgrades, and scalability**, making future modifications smoother.
3. **Reduces Development Risks:**
   * Identifies potential **bottlenecks** or **conflicts** before implementation, reducing **unexpected errors** in later phases.
4. **Enhances Security & Data Integrity:**
   * Ensures **proper data structuring** (ERD, database schema) to **prevent security vulnerabilities** such as **unauthorized access** or **data inconsistency**.
5. **Improves Communication:**
   * Serves as a **reference document** for all stakeholders, ensuring a unified understanding of system design.

By following a **structured design approach**, this document guarantees that the we adhere to industry **best practices**, ultimately leading to a **robust, scalable, and secure** intrusion detection system.

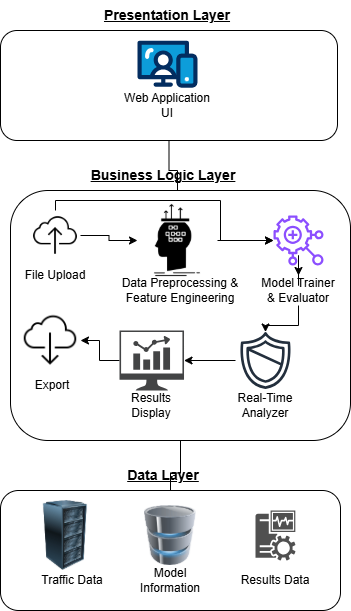
1. **Entity Relationship Diagram (ERD)**



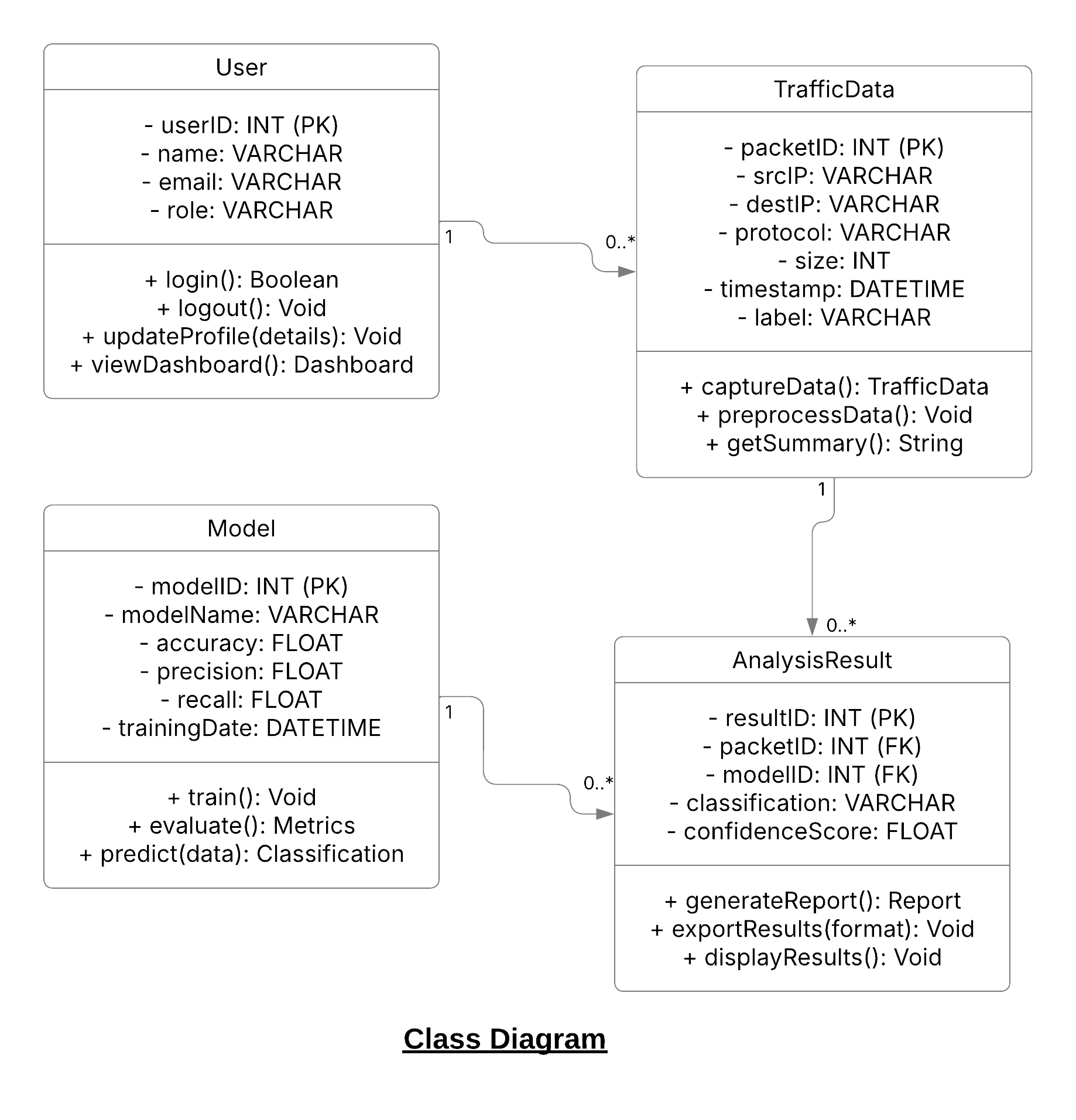
1. **Sequence Diagrams**



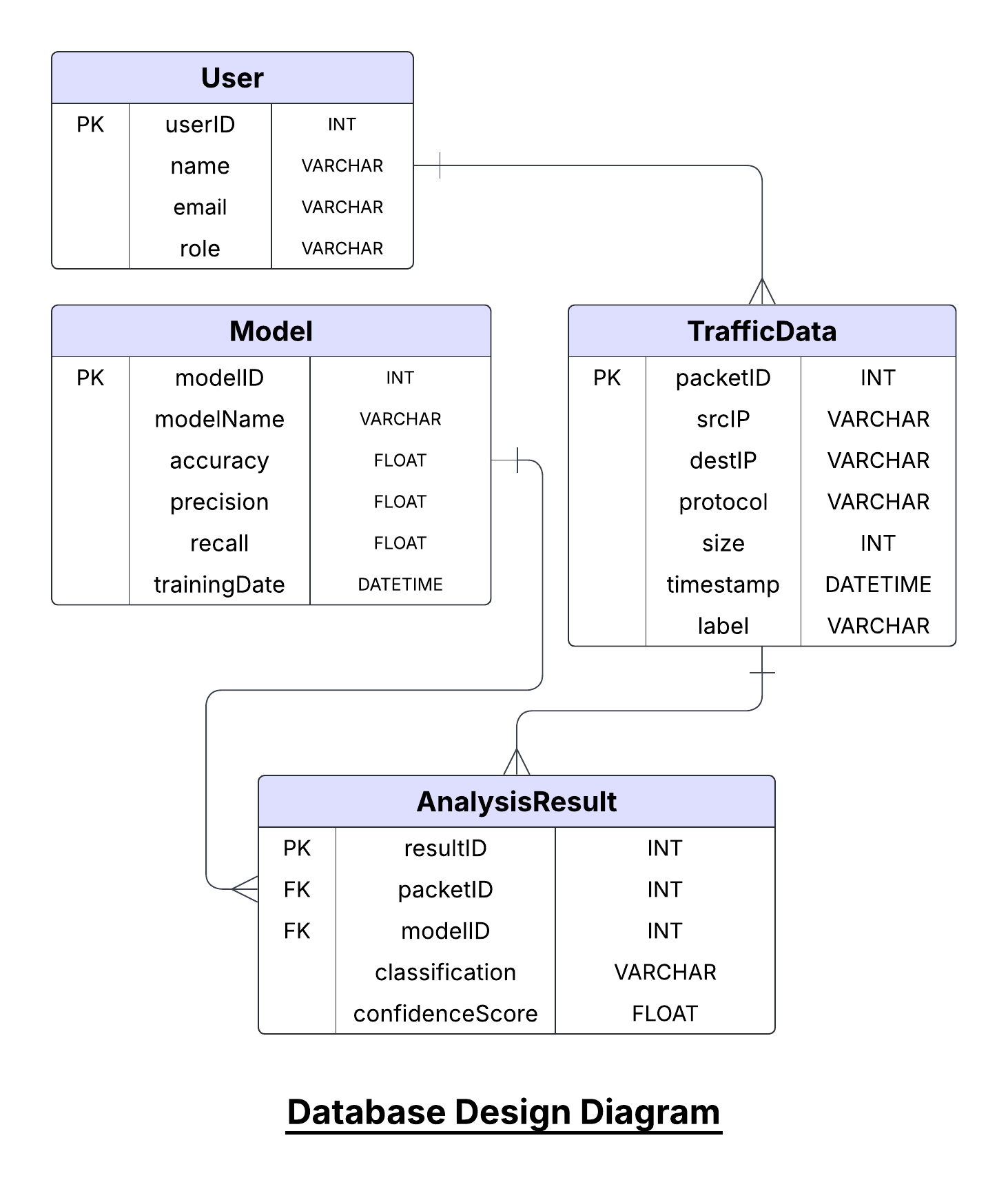
1. **Architecture Design Diagram**



1. **Class Diagram**

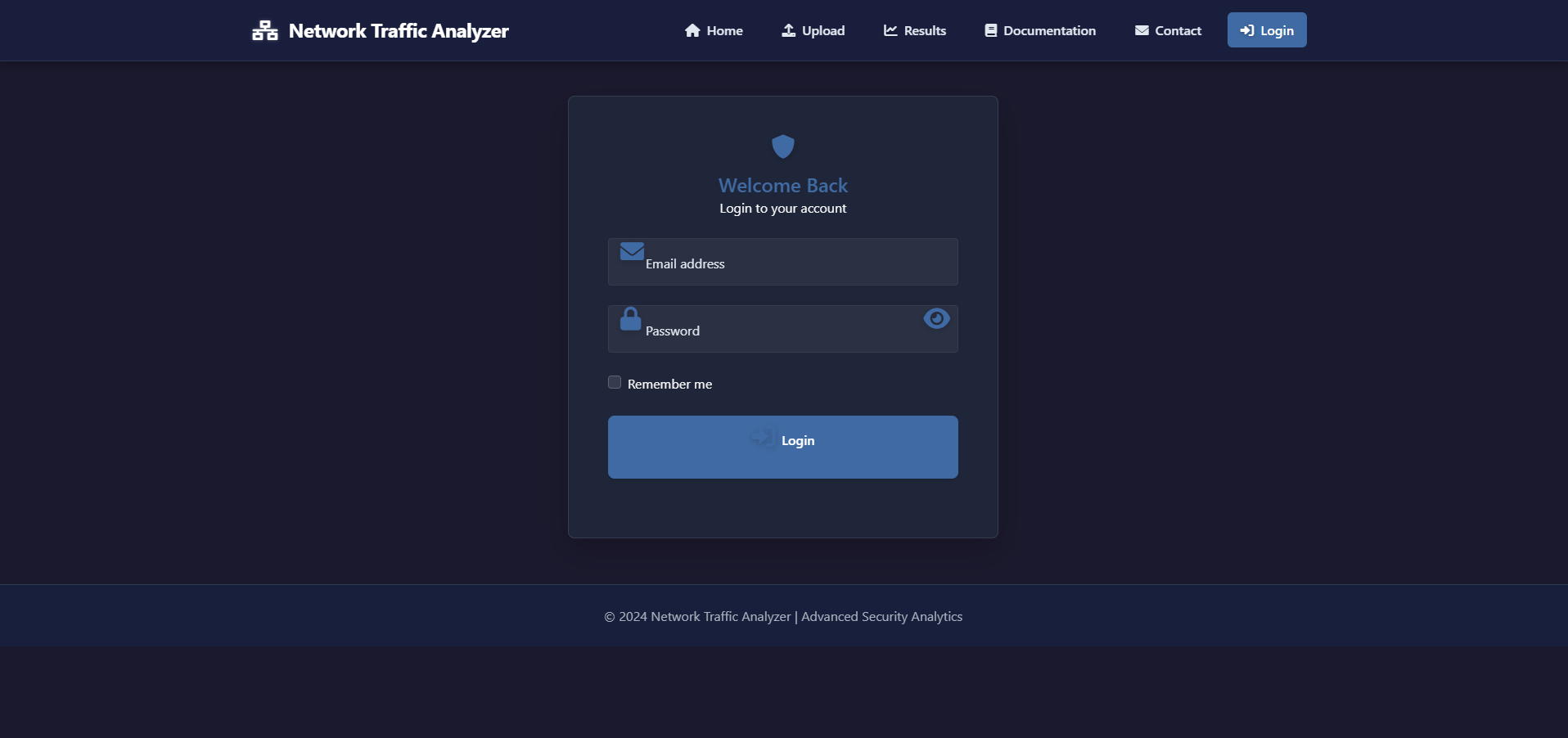


1. **Database Design**

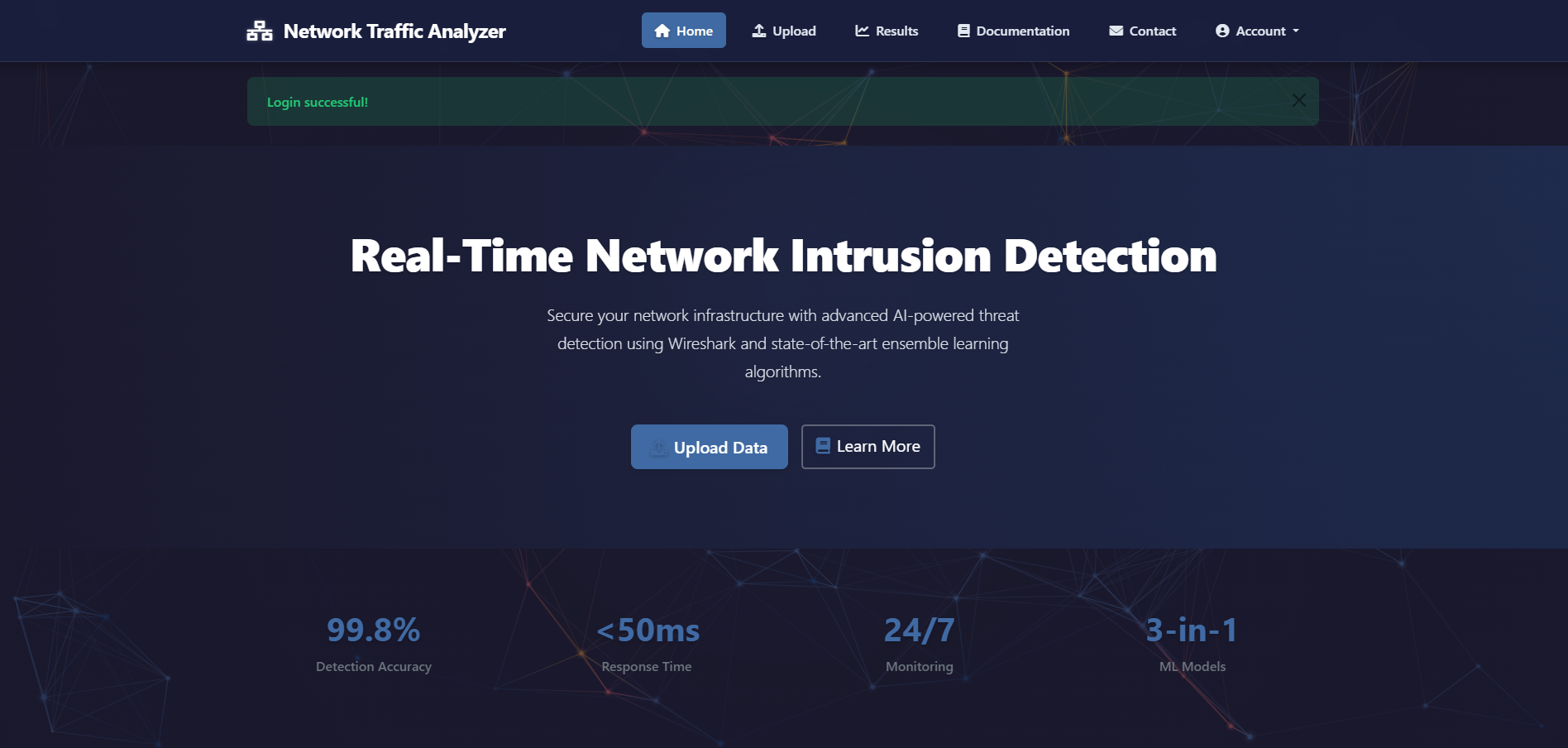


1. **Interface Design**

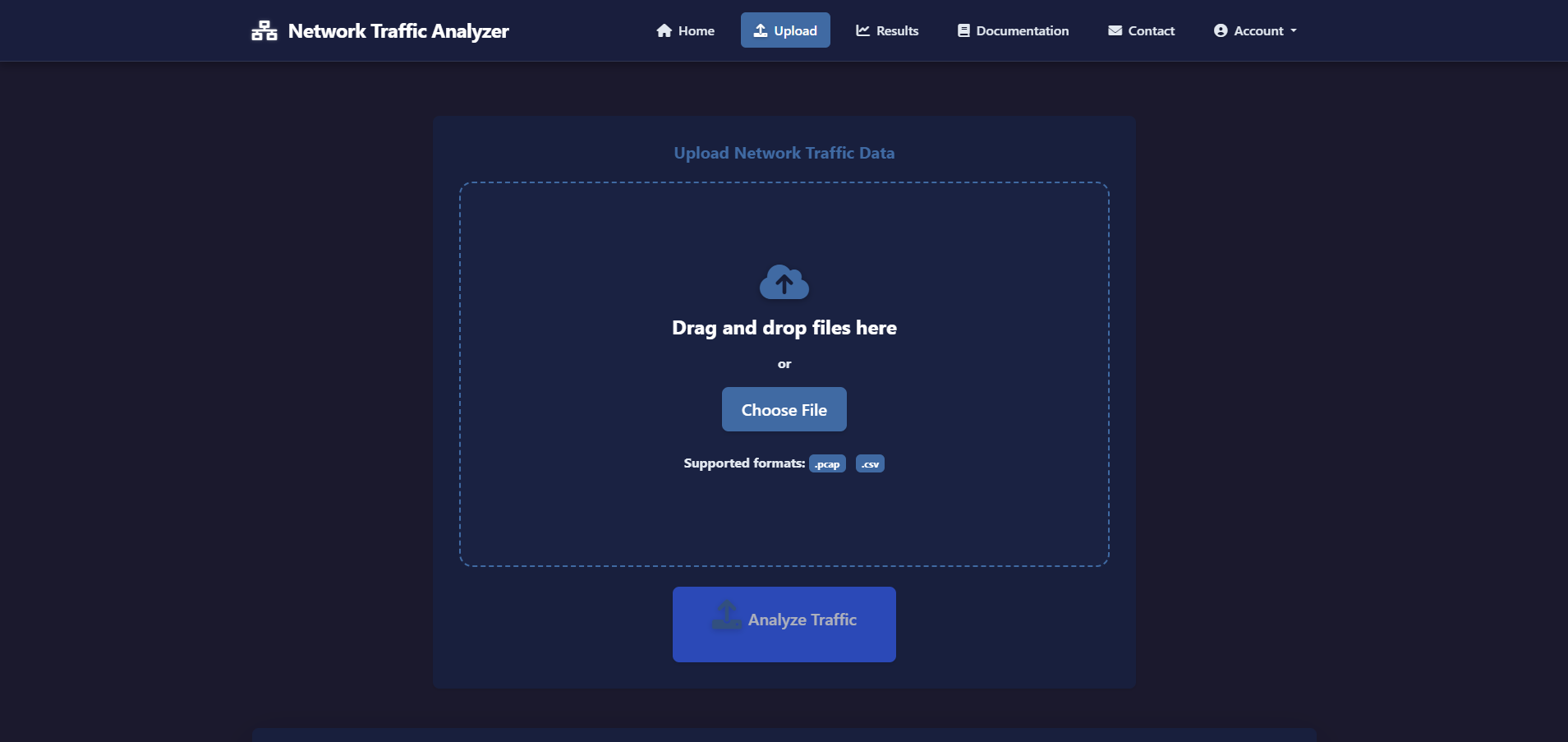
Login Page:



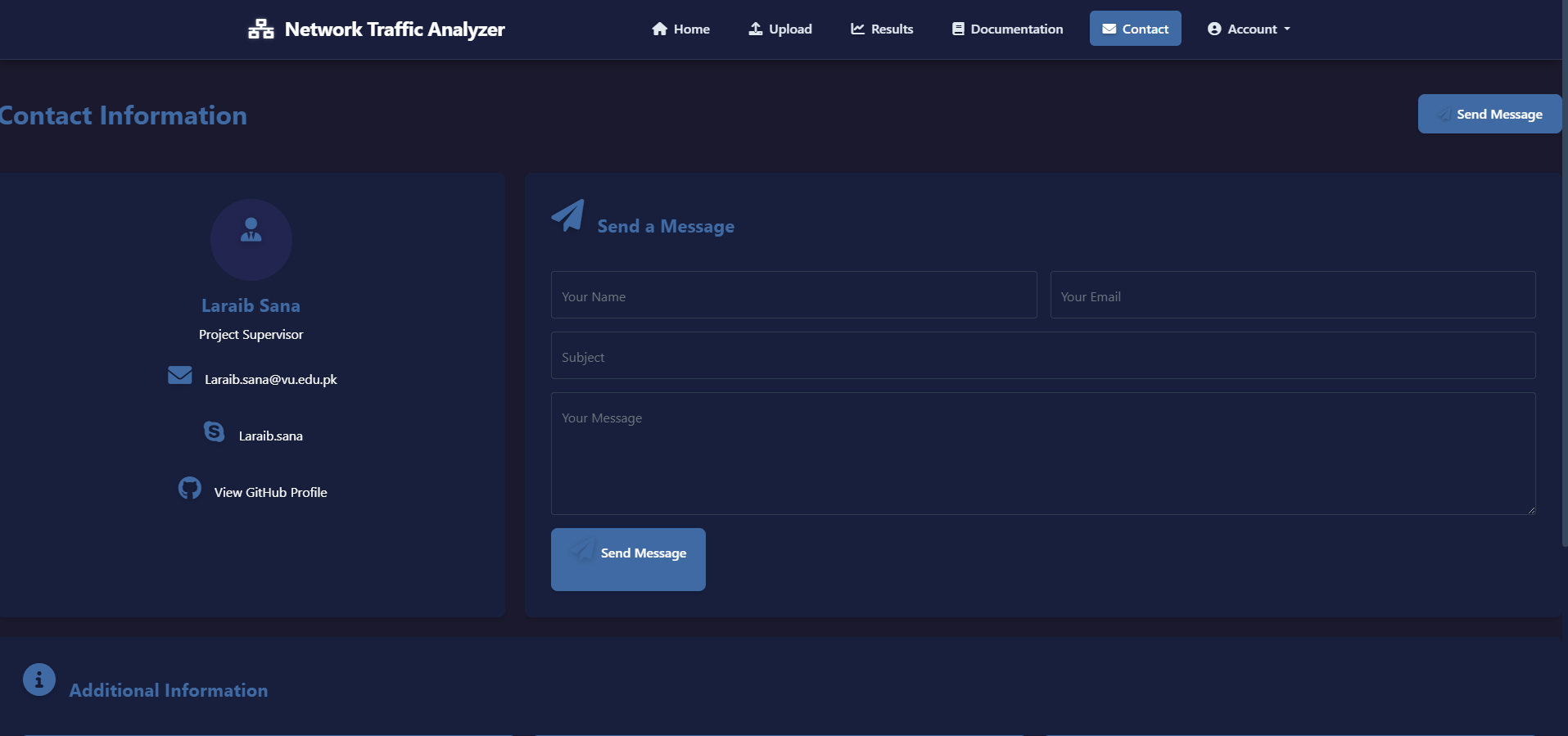
Home Page:



File Upload Page:



Contact Us Page:



1. **Test Cases**

Below are detailed test cases for each scenario of our project **Real-Time Network Intrusion Detection System**. Each test case includes Test Case ID, Description, Preconditions, Test Steps, Expected Output, and Pass/Fail Criteria.

1. **Capturing Network Traffic (Using Wireshark)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC01 | Verify Wireshark captures live network traffic | Wireshark installed & active network interface available | 1. Launch Wireshark  2. Select network interface  3. Start capture  4. Stop capture after some time | Wireshark captures and displays live packets | Capture is visible in Wireshark with correct packet details |
| TC02 | Verify captured traffic is saved in CSV format | Wireshark installed | |  | | --- | |  |  |  | | --- | | 1. Capture network traffic  2. Export captured data as CSV | | CSV file is generated with captured packet details | CSV file contains valid network data |
| TC03 | Verify Wireshark handles invalid configurations | Wireshark installed | |  | | --- | |  |  |  | | --- | | 1. Select an invalid network interface  2. Attempt to capture traffic | | Wireshark displays an error message | Error message appears, capture fails |

**2. Uploading Data for Analysis**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC04 | Verify valid CSV file is accepted | Web application running | 1. Navigate to file upload page  2. Upload a valid CSV file | File is successfully uploaded and validated | System displays “File uploaded successfully” |
| TC05 | Verify system rejects invalid file formats | Web application running | |  | | --- | |  |  |  | | --- | | 1. Attempt to upload non-CSV file (e.g., PDF, TXT) | | System rejects the file | System displays “Invalid file format” |
| TC06 | Verify system rejects empty files | Web application running | |  | | --- | |  |  |  | | --- | | 1. Attempt to upload an empty CSV file | | |  | | --- | | System rejects the file |  |  | | --- | |  | | System displays “Empty file cannot be uploaded” |

**3. Preprocessing & Feature Engineering**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC07 | Verify data preprocessing is applied correctly | Uploaded CSV file contains valid network data | 1. Upload CSV file  2. System performs preprocessing (removes missing values, encodes categorical features) | Processed data is stored in database | Data integrity is maintained after preprocessing |
| TC08 | Verify preprocessing handles missing values | Uploaded CSV file contains missing values | |  | | --- | |  |  |  | | --- | | 1. Upload CSV file  2. System detects and handles missing values | | System fills missing values or removes incomplete rows | No missing values in processed data |
| TC09 | Verify normalization of numerical features | Uploaded CSV file contains raw packet size & time interval data | |  | | --- | |  |  |  | | --- | | 1. Upload CSV file  2. System normalizes numerical features | | |  | | --- | | Normalized data stored successfully |  |  | | --- | |  | | Data within acceptable range |

**4. Model Training & Evaluation**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC10 | Verify model training completes successfully | Preprocessed dataset available | 1. Initiate model training  2. Wait for model completion | Model is trained and saved in the database | Training completes with acceptable performance metrics |
| TC11 | Verify evaluation metrics are calculated correctly | Preprocessed dataset available | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | 1. Train the model  2. Evaluate accuracy, precision, recall | | | Metrics displayed correctly | Metrics match expected values |
| TC12 | Verify system handles insufficient data for training | Small dataset provided | |  | | --- | |  |  |  | | --- | | 1. Initiate training with insufficient data | | |  | | --- | | System displays an error message |  |  | | --- | |  | | System warns about insufficient data |

**5. Real-Time Intrusion Detection Analysis**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC13 | Verify system classifies uploaded network data | Trained model available | 1. Upload new network data  2. System processes data  3. Model classifies traffic | Classification results displayed | Traffic is correctly labeled as normal/malicious |
| TC14 | Verify classification confidence score is generated | Trained model available | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | 1. Upload network data  2. System classifies traffic  3. Check confidence scores | | | Confidence scores are displayed | Confidence scores match expected values |
| TC15 | Verify incorrect or corrupted data is handled | Corrupted CSV file uploaded | |  | | --- | |  |  |  | | --- | | 1. Upload corrupted file  2. System processes file | | |  | | --- | | System detects and rejects the file |  |  | | --- | |  | | Error message displayed |

**6. Display Results**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC16 | Verify system displays classification results | Analysis completed | 1. Navigate to results page  2. View analysis results | Results are displayed correctly | UI shows results with traffic labels |
| TC17 | Verify system allows filtering results | Analysis completed | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | 1. Apply filter for malicious traffic only | | | Only malicious traffic results are displayed | Filter works correctly |
| TC18 | Verify system loads results efficiently | Large dataset analyzed | |  | | --- | |  |  |  | | --- | | 1. Load results page with 10,000+ records | | |  | | --- | | Results load within acceptable time |  |  | | --- | |  | | Page loads without lag |

**7. Save or Export Results**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Test Case ID** |  |  | | --- | |  |  |  | | --- | |  |  |  | | --- | |  | | **Test Case Description** | **Preconditions** | **Test Steps** | **Expected Output** | **Pass/Fail Criteria** |
| TC19 | Verify results can be exported as CSV | Analysis completed | 1. Click on “Export CSV” button | CSV file is downloaded successfully | File contains correct results |
| TC20 | Verify results can be exported as PDF | Analysis completed | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | 1. Click on “Export PDF” button | | | PDF report is generated | PDF matches displayed results |
| TC21 | Verify export fails for no data | No results available | |  | | --- | |  |  |  | | --- | | 1. Click on export button | | |  | | --- | | System displays “No data available” |  |  | | --- | |  | | Export is disabled if no results exist |

**Summary**

* **Total Test Cases**: 21
* **Coverage**: Capturing, Uploading, Preprocessing, Training, Classification, Displaying & Exporting Results
* **Expected Outcome**: Ensuring the system operates smoothly and handles errors effectively.