# Real-Time Network Intrusion Detection Using Wireshark and Advanced Ensemble Learning Techniques

## Software Requirements Specification

**Version 1.0**

**Group Id:** F24PROJECTCE3A6

**Supervisor Name:** Laraib Sana

# Revision History

| Date (dd/mm/yyyy) | Version | Description | Author |
|---|---|---|---|
| 08/12/2024 | 1.0 | Introduction of the project | BC210424643<br>BC210401276 |

# Table of Contents

# SRS Document

## Scope of Project:

The project aims to develop an intrusion detection system (IDS) that can analyze real-time network traffic to identify potential cyber threats. It involves using Wireshark to capture network traffic, preprocessing the data for machine learning analysis, and leveraging advanced ensemble models like TabNet, CatBoost, and LightGBM for intrusion detection. A user-friendly web application will also be created for uploading and analyzing network data in real-time.

**Key Deliverables**:

- Real-time network traffic analysis.
- Preprocessing and feature extraction pipeline.
- Implementation of machine learning models.
- Web application for user interaction and analysis visualization.

## Functional and Non-Functional Requirements:

### Functional Requirements

- **Traffic Capture:**
  - The system must use Wireshark to capture real-time network traffic data.
  - Captured data must be saved as a CSV file for preprocessing and analysis.
- **Data Upload:**
  - Users must upload CSV files containing network traffic data for analysis.
  - The system should validate the uploaded files for format and content consistency.
- **Data Preprocessing:**
  - Clean raw traffic data by handling missing or inconsistent values.
  - Encode categorical features such as protocol types and normalize numerical features like packet size and time intervals.
  - Label the dataset with normal and malicious traffic for supervised machine learning.
- **Model Training and Evaluation:**
  - Train ensemble models, including TabNet, CatBoost, and LightGBM, using the preprocessed dataset.
  - Evaluate models using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
- **Intrusion Detection:**

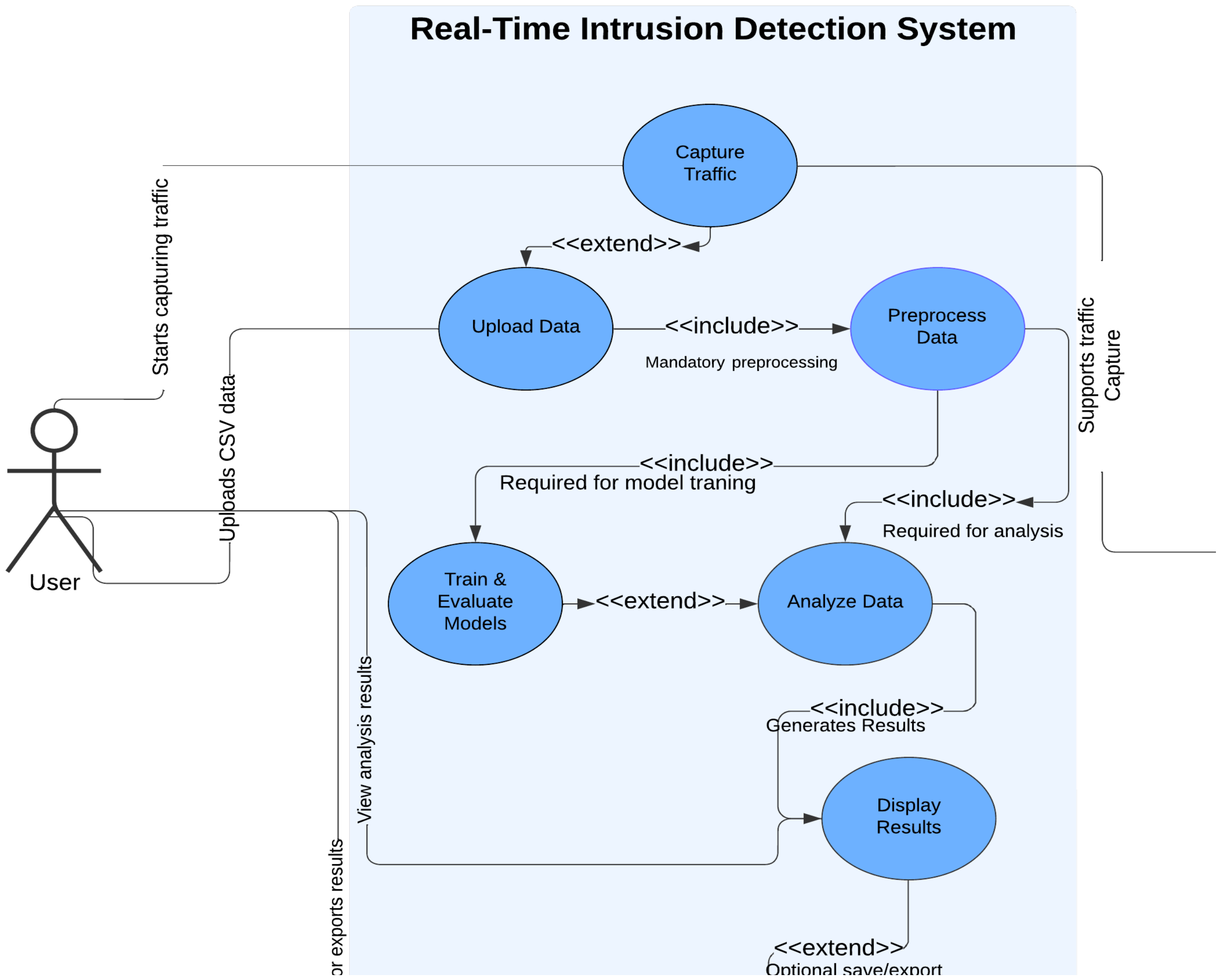
    - Analyze uploaded network traffic data using trained machine learning models to detect potential intrusions.
    - Classify network traffic as normal or malicious and generate detailed results.
- **Web Application:**
    - Provide an intuitive interface for users to upload CSV files of network traffic data.
    - Display real-time analysis results, including key metrics and influencing factors.
    - Allow users to save or export the analysis results for further review.

## Non-Functional Requirements

- **Performance:**
    - The system must process and analyze data within 2 seconds for real-time functionality.
- **Scalability:**
    - The system must handle datasets of up to 1 GB without performance degradation.
- **Usability:**
    - The web interface must be user-friendly and accessible to technical and non-technical users.
- **Reliability:**
    - The system should provide consistent results under various traffic loads.
- **Security:**
    - Ensure secure upload and processing of data to prevent unauthorized access or manipulation.
    - Validate input files to protect against malicious or corrupt data uploads.
- **Portability:**
    - The application must be compatible with multiple operating systems, including Windows and Linux.
- **Maintainability:**
    - Use a modular design to facilitate updates and additions, such as integrating new machine learning models.
- **Data Integrity:**
    - Ensure the integrity of processed and analyzed data, avoiding loss or corruption.

**These requirements provide a clear foundation for developing a robust, efficient, and user-friendly intrusion detection system**

## Use Case Diagram(s):



## Usage Scenarios:

| Use Case Title | Use Case ID | Actions | Description | Alternative Paths | Preconditions | Postconditions | Author | Exceptions |
|---|---|---|---|---|---|---|---|---|
| Capture Traffic | UC01 | 1. User launches Wireshark.<br>2. Selects the network interface.<br>3. Starts capturing network traffic.<br>4. Saves the traffic data as a CSV file. | This use case captures real-time network traffic using Wireshark and saves the captured traffic data for analysis. | 1. User selects incorrect interface and needs to restart the capture.<br>2. Save file as another format if CSV is unavailable. | Wireshark must be installed and configured. Network interface should be active. | A CSV file with captured traffic data is generated. | Wasif Nawaz | 1. Network interface unavailable.<br>2. Insufficient permissions to capture traffic.<br>3. Application crash. |
| Upload Data | UC02 | 1. User accesses the web app.<br>2. Uploads a CSV file.<br>3. System validates file format.<br>4. System preprocesses the data for analysis. | Allows users to upload captured network traffic data in CSV format to the web application for preprocessing and analysis. | User uploads an invalid file; system prompts for re-upload. Preprocessing can skip certain rows with invalid data. | A valid CSV file containing network traffic data must be available. | Data is preprocessed and ready for analysis. | Wasif Nawaz | 1. Invalid file format.<br>2. Missing or corrupt data.<br>3. Server fails to preprocess the file. |
| Preprocess Data | UC03 | 1. Clean the uploaded data.<br>2. Encode | This use case prepares the uploaded | Skip preprocessing for small | Uploaded data must be in CSV format. | Data is ready for machine learning analysis. | Wasif Nawaz | 1. Incomplete preprocessing due to corrupt data.<br>2. Incorrect encoding format. |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | categorical features.<br>3. Normalize numerical values.<br>4. Label data as normal or malicious. | network traffic data for machine learning analysis by cleaning, encoding, and normalizing it. | datasets that are already clean and formatted. | System must have preprocessing modules deployed. | | | 3. Timeout during large file processing. |
| Model Training | UC04 | 1. Preprocessed data is loaded.<br>2. Models (TabNet, CatBoost, LightGBM) are trained using labeled data.<br>3. Metrics (e.g., accuracy, precision, recall) are calculated and stored. | This use case handles training the machine learning models using preprocessed and labeled datasets. | Allow re-training specific models if others fail. | Preprocessed and labeled data must be available. Trained models must be saved for deployment. | Trained models with performance metrics are available for analysis. | Wasif Nawaz | 1. Insufficient labeled data.<br>2. Training failure due to incompatible data formats or model errors. |
| Analyze Data | UC05 | 1. Preprocess the uploaded data.<br>2. Classify the traffic using trained models.<br>3. Generate intrusion detection results. | This use case uses trained machine learning models (TabNet, CatBoost, and LightGBM) to classify the uploaded data and identify potential intrusions. | Use an alternate model if the primary model fails.<br>Skip classification for datasets with missing essential features. | Data must be preprocessed. Machine learning models must be trained and deployed. | Analysis results (normal or malicious traffic) are generated. | Wasif Nawaz | 1. Data preprocessing failure.<br>2. Model timeout or incorrect output.<br>3. Large dataset causes memory overflow. |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Display Results | UC06 | 1. User views results on the web app.<br>2. Key metrics and influencing factors are displayed.<br>3. User saves or exports the results (optional). | Displays the results of the intrusion detection analysis in an intuitive interface, highlighting key metrics and traffic features influencing the classification. | User chooses to view detailed logs or summary report only. Allow exporting data in various formats (CSV, PDF). | Analysis must be completed successfully. Web app must be functional. | Results are displayed on the interface with options for saving/exporting. | Wasif Nawaz | 1. Server error during result display.<br>2. Incomplete analysis results.<br>3. Export function failure. |
| Save Results | UC07 | 1. User clicks the "Save" or "Export" button.<br>2. System validates file format.<br>3. Saves results in the chosen format. | This optional use case allows users to save or export the analysis results for offline review or documentation purposes. | Skip the save/export step if not required. Default to CSV format if no format is selected. | Results must be ready and displayed in the interface. | Results are saved successfully in the chosen file format. | Wasif Nawaz | 1. File write permission denied.<br>2. Insufficient storage space.<br>3. Export format not supported. |

## Adopted Methodology:

For this project, the **VU Process Model** is adopted, combining the structured approach of the **Waterfall Model** with the iterative flexibility of the **Spiral Model**. This ensures a systematic development process while allowing refinements as needed.

## Phases of the Methodology

1. **Requirement Analysis (Waterfall Phase)**:
   - Gather and document project requirements, including functional and non-functional specifications.
   - Define the scope of work and deliverables.
2. **Design and Planning (Waterfall Phase)**:
   - Design the system architecture, including data capture, preprocessing pipeline, machine learning models, and web application.
   - Create a detailed project timeline using a Gantt chart.
3. **Development (Spiral Phase)**:
   - **First Iteration**: Implement Wireshark integration for traffic capture and CSV export.
   - **Second Iteration**: Develop and test preprocessing and machine learning modules.
   - **Third Iteration**: Build and refine the web application for real-time analysis.
   - Adjust based on iterative feedback after each phase.
4. **Testing (Spiral Phase)**:
   - Conduct unit testing for individual modules (e.g., data preprocessing, machine learning models).
   - Perform integration and system testing to ensure overall functionality.
5. **Deployment (Waterfall Phase)**:
   - Deploy the finalized system with a user-friendly interface and document instructions for usage.

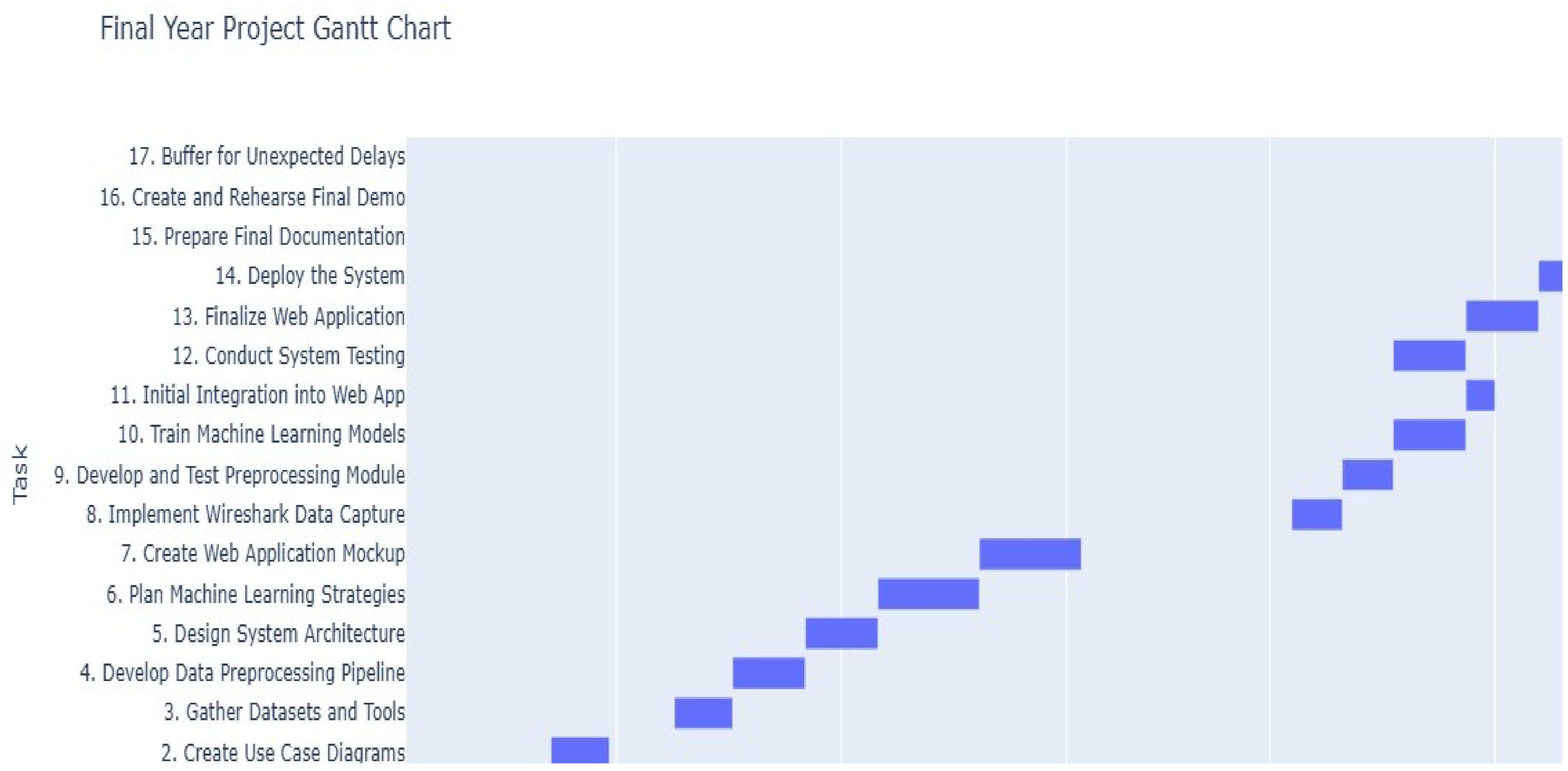
**Advantages of this Approach**

- **Structure**: The waterfall phase ensures clear requirements and structured planning.
- **Flexibility**: The spiral phase allows iterative development and adjustments based on testing feedback.

- **Risk Mitigation**: Continuous testing and refinement reduce the risk of major issues.

This methodology ensures the project is delivered on time, meets quality standards, and achieves the desired objectives.

**Work Plan (Use MS Project to create Schedule/Work Plan):**


Final Year Project Gantt Chart
Task
17. Buffer for Unexpected Delays
16. Create and Rehearse Final Demo
15. Prepare Final Documentation
14. Deploy the System
13. Finalize Web Application
12. Conduct System Testing
11. Initial Integration into Web App
10. Train Machine Learning Models
9. Develop and Test Preprocessing Module
8. Implement Wireshark Data Capture
7. Create Web Application Mockup
6. Plan Machine Learning Strategies
5. Design System Architecture
4. Develop Data Preprocessing Pipeline
3. Gather Datasets and Tools
2. Create Use Case Diagrams