**DDoS Analyzer   
– AI-Based DDoS Attack Classification**



**Project ID: Fall-2022-2026**

**Session: BSCS Fall 2022 to 2026**

**Project Advisor: Ma’am Namra Tahir**

**Submitted By**

|  |  |
| --- | --- |
| Huzaifa Zahid | 70132358 |
| Shahmeer Abdullah | 70140615 |
| Syed Ali Haider | 70126520 |
|  |  |

Department of Computer Science & IT

The University of Lahore

Lahore, Pakistan

## *Declaration*

We have read the project guidelines and we understand the meaning of academic dishonesty, in particular plagiarism and collusion. We hereby declare that the work we submitted for our final year project, entitled **DDoS Analyzer (AI-Based DDoS Attack Classification)** is original work and has not been printed, published or submitted before as final year project, research work, publication or any other documentation.

## 

**Huzaifa Zahid**

**70132358**

**Signature: …………………………**

**Shahmeer Abdullah**

**70140615**

**Signature: …………………………**

**Syed Ali Haider**

**70126520**

**Signature: …………………………**

## *Statement of Submission*

This is to certify that **Huzaifa Zahid (Roll No.70132358 )**, **Shahmeer Abdullah (Roll No.70140615)** and **Syed Ali Haider (Roll No.70126520 )** have successfully submitted the final project named as: **DDoS Analyzer (AI-Based DDoS Attack Classification)**, at Computer Science & IT Department, The University of Lahore, Lahore Pakistan, to fulfill the partial requirement of the degree of **BS in Computer Science**.

**Supervisor Name: Ma’am Namra Tahir………………………**

**Signature: …………………………**

**Date: ………………………**

## *Dedication*

As a matter of some importance, we devote our task to Allah Almighty. Also, to whom the world owes its presence Muhammad (Harmony Arrive). This unassuming exertion is committed to our adored guardians who brought us to the degree of greatness where we are reading up today searching generally speaking Promising and sparkling future ahead for which they scarified the majority of the time of their life & to our regarded and virtuoso instructors who directed us all through scholastic profession and that multitude of individuals who have recollected that us in their requests! A great deal of gratitude for every one of my instructors.

## *Acknowledgement*

## 

Above all, I owe a debt of gratitude to the Almighty God who gave us a life that is worthwhile and whom I appreciate for providing us the stamina to complete this task. This project's accomplishment and success are the result of the hard work and dedication of numerous people who provided direct or indirect support. I give them all my gratitude for their devotion. First I’d like to thank whomever is reading this project for their interest and time. It was a great pleasure going through a hard yet necessary experience.

Most importantly, I’d like to express my deepest gratitude to my supervisor **Ma’am Namra Tahir** for his support and patience throughout the semester, and also for sharing his knowledge and helping me in every stage of the project development.

Last but not least, we would want to express our sincere gratitude and respect to our parents, whose love and care helped us advance academically and kept us motivated. The writers owe a huge debt of gratitude to their siblings, whose unceasing support gave us the motivation we needed to pursue our academic goals.

Also, I had a great amount of support from my friends who were also working on their capstone, especially for their emotional support. We all faced problems and we all managed to get through them by supporting each other.

We are also thankful to our friends and families whose silent support led us to complete our project.

Date:

September 4, 2025

## *Abstract*

DDoS Analyzer introduces an AI-powered platform designed to classify network traffic and identify Distributed Denial of Service attacks with improved accuracy and efficiency. The system focuses on interpreting patterns within traffic logs using machine learning, allowing users to analyze both current and previously recorded data. Unlike traditional monitoring tools that only detect threats in real time, this solution enables deeper investigation into past incidents, helping security teams uncover long term attack trends and strengthen their defense strategies.

The platform integrates core analytical features, including automated classification, visual representation of traffic behavior, and comparison of different machine learning approaches within a unified interface. By reducing the need for manual inspection and rule-based detection, the system enhances the speed and reliability of security analysis while offering clear insights into normal and malicious activity.

DDoS Analyzer addresses the limitations of conventional methods by providing an accessible, scalable, and intelligence-driven tool for threat analysis. Its contribution extends beyond detection, supporting cybersecurity professionals in decision making, improving preparedness, and fostering a more resilient digital environment.

***Area of the Project***

Artificial Intelligence, Machine Learning, Network Traffic Analysis, Web Application Development.

***Technologies used***

* Python
* Scikit-learn and other ML libraries
* Flask
* HTML
* CSS
* JavaScript
* Linux Virtual Machine (Kali Linux for generating attacks)
* Windows Virtual Machine (victim system for traffic capture)
* Wireshark
* CIC-FlowMeter (for traffic feature extraction)
* Heroku (for deployment)

List of Figures

Figure 1 Block Diagram – Product Perspective ........................................................................ 10

Figure 2 Use Case – Upload Dataset ...........................................................................................11

Figure 3 Use Case – View Results ...........................................................................................12

Figure 4 Use Case – Compare Model Performance ..............................................................…13

Figure 5 Architecture Diagram .............................................................................................. 14

Figure 6 Entity Relationship Diagram .................................................................................. 15

Figure 7 Data Flow Diagram – Level 0 ................................................................................ 16

Figure 8 Data Flow Diagram – Level 1 ................................................................................ 17

Figure 9 Class Diagram ....................................................................................................... 18

Figure 10 Aggregated Activity Diagram ............................................................................... 19

Figure 11 Activity Diagram – Upload Dataset ....................................................................... 20

Figure 12 Activity Diagram – Model Training & Testing ...................................................... 21

Figure 13 Activity Diagram – View Classification Results .................................................... 22

Figure 14 Activity Diagram – Visualize Model Performance ................................................. 23

Figure 15 Sequence Diagram – Upload Dataset ................................................................... 24

Figure 16 Sequence Diagram – Model Processing ................................................................ 25

Figure 17 Sequence Diagram – Display Results .................................................................... 26

Figure 18 Collaboration Diagram – Dataset Processing ..................................................... 27

Figure 19 Collaboration Diagram – Model Evaluation ....................................................... 28

Figure 20 State Transition Diagram – Dataset Lifecycle ..................................................... 29

Figure 21 Component Diagram ............................................................................................ 30

Figure 22 Deployment Diagram .......................................................................................... 31

Figure 23 User Manual – Upload Dataset Screen .................................................................. 32

Figure 24 User Manual – Results Dashboard ....................................................................... 33

Figure 25 User Manual – Model Comparison Graphs ......................................................... 34

Figure 26 User Manual – Download Results ......................................................................... 35

## List of Tables

Table 1 Functional Requirement FR\_01 Upload Dataset .......................................................... 11

Table 2 Functional Requirement FR\_02 View Results .............................................................. 12

Table 3 Functional Requirement FR\_03 Compare Model Performance .................................... 12

Table 4 Functional Requirement FR\_04 Model Training & Testing ......................................... 13

Table 5 Functional Requirement FR\_05 Visualization of Results ........................................... 13

Table 6 Use Case – Upload Dataset ....................................................................................... 14

Table 7 Use Case – View Results ............................................................................................ 15

Table 8 Use Case – Compare Model Performance .................................................................. 16

Table 9 Use Case – Visualize Results ..................................................................................... 17

Table 10 Data Dictionary of ERD ............................................................................................... 18

Table 11 ML Model Evaluation Metrics .................................................................................... 19

Table 12 Example Dataset Structure ........................................................................................ 20

Table 13 Model Performance Comparison Table ....................................................................... 21

Table 14 User Manual – Upload Dataset Details ....................................................................... 22

Table 15 User Manual – Results Overview ............................................................................... 23

Table 16 User Manual – Graphical Representation of Model Performance .............................. 24

Table 17 Appendix A – Sample Network Traffic Features ......................................................... 25

Table 18 Appendix B – Supported File Formats ......................................................................... 26

Table 19 Appendix C – Example ML Output .............................................................................. 27

**Table of Content**

[*Declaration* i](#_Toc71724966)

[*Statement of Submission* ii](#_Toc71724967)

[*Dedication* iii](#_Toc71724968)

[*Acknowledgement* iv](#_Toc71724969)

[*Abstract* v](#_Toc71724970)

[List of Figures vii](#_Toc71724971)

[List of Tables viii](#_Toc71724972)

[Chapter 1: Introduction to the Problem 1](#_Toc71724973)

[1.1 Introduction 1](#_Toc71724974)

[1.2 Purpose 1](#_Toc71724975)

[1.3 Objective 1](#_Toc71724976)

[1.4 Existing Solution 1](#_Toc71724977)

[1.5 Proposed Solution 1](#_Toc71724978)

[Chapter 2: Software Requirement Specification 2](#_Toc71724979)

[2.1 Introduction 2](#_Toc71724980)

[2.1.1 Purpose 2](#_Toc71724981)

[2.1.2 Scope 2](#_Toc71724982)

[2.1.3 Definitions, acronyms, and abbreviations 2](#_Toc71724983)

[2.2 Overall description 2](#_Toc71724984)

[2.2.1 Product perspective 2](#_Toc71724985)

[2.2.2 Product functions 3](#_Toc71724986)

[2.2.3 User characteristics 4](#_Toc71724987)

[2.2.4 Constraints 4](#_Toc71724988)

[2.2.5 Assumptions and dependencies 4](#_Toc71724989)

[2.2.6 Apportioning of requirements 4](#_Toc71724990)

[2.3 Specific requirements 4](#_Toc71724991)

[2.3.1 Functional Requirement 4](#_Toc71724992)

[2.3.2 Non-functional Requirements 4](#_Toc71724993)

[Chapter 3: Use Case Analysis 6](#_Toc71724994)

[Chapter 4: Design 8](#_Toc71724995)

[4.1 Architecture Diagram 8](#_Toc71724996)

[4.2 ERD with data dictionary 8](#_Toc71724997)

[4.3 Data Flow diagram 9](#_Toc71724998)

[4.3.1 The level 0 9](#_Toc71724999)

[4.3.2 The level 1 10](#_Toc71725000)

[4.4 Class Diagram 11](#_Toc71725001)

[4.5 Activity Diagram 11](#_Toc71725002)

[4.6 Sequence Diagram 12](#_Toc71725003)

[4.7 Collaboration Diagram 13](#_Toc71725004)

[4.8 State Transition Diagram 14](#_Toc71725005)

[4.9 Component Diagram 15](#_Toc71725006)

[4.10 Deployment Diagram 16](#_Toc71725007)

[Chapter 5: User Manual 17](#_Toc71725008)

[References 19](#_Toc71725009)

[Appendix 23](#_Toc71725010)

Chapter 6: Implementation and Experimental Analysis ........................................ 24

6.1 Experimental Setup and Environment Configuration ............................... 24

6.2 Dataset Selection and Justification ...................................................... 25

6.3 Tools and Technologies Used ............................................................. 26

6.4 Network Configuration and Connectivity Verification .............................. 27

6.5 Attack Simulation Using Kali Linux ..................................................... 28

6.6 Traffic Capture and Monitoring on Windows VM .................................. 30

6.7 Attack Scenarios and Captured Traffic Analysis .................................... 32

6.8 PCAP to CSV Conversion Using CICFlowMeter .................................... 34

6.9 Generated Dataset Summary and Validation ......................................... 36

### Chapter 1: Introduction to the Problem

##### Introduction

In the modern digital age, network security has become a critical concern with the increasing frequency and sophistication of cyberattacks. Distributed Denial of Service (DDoS) attacks, in particular, pose a significant threat to the availability and reliability of online services, affecting organizations, governments, and individuals. Traditional network monitoring methods, such as manual traffic inspection and rule-based intrusion detection systems, often struggle to detect complex, multi-vector attacks efficiently. The growing volume of network traffic, combined with the diversity of attack types, highlights the need for intelligent, automated solutions capable of analyzing and classifying malicious activity.

Existing tools for DDoS detection are often costly, hardware-dependent, or require advanced cybersecurity expertise, which limits accessibility for smaller organizations, educational institutions, and individual network administrators. Many solutions focus on real-time monitoring, leaving little support for batch processing or historical traffic analysis, which are critical for understanding attack trends and developing long-term mitigation strategies. This creates a gap for an accessible, user-friendly, and automated system that can analyze traffic data efficiently and provide actionable insights.

The proposed project, **DDoS Analyzer – AI-Based DDoS Attack Classification**, addresses these challenges by developing an open-source platform that allows users to upload network traffic datasets for analysis [6], [7]. The system preprocesses the data, applies supervised machine learning models, and classifies normal and malicious traffic, including various DDoS attack types [9]. Graphical visualizations are generated to illustrate traffic distribution, attack patterns, and model performance, making it easier for users to interpret results without requiring extensive cybersecurity knowledge [13].

DDoS Analyzer removes the need for specialized infrastructure or real-time monitoring, making it widely accessible for network administrators, IT security analysts, and students. The platform supports offline analysis, model comparison, and detailed insights into malicious activity, enabling proactive measures for network protection. By providing a cost-effective, automated, and intuitive solution, DDoS Analyzer addresses the growing demand for intelligent cybersecurity tools capable of analyzing modern network threats efficiently [14].

##### Purpose

In the modern digital era, DDoS attacks are among the most frequent and disruptive cybersecurity threats, causing service outages, operational losses, and reputational damage to organizations. Traditional security tools primarily rely on real-time traffic monitoring, which only detects threats as they occur and fails to provide insights into historical attack patterns. This limitation makes it difficult for security teams to analyze past incidents, identify vulnerabilities, and refine long-term defense strategies. There is a clear need for an intelligent, accessible tool that can process both current and historical network traffic to uncover patterns and enhance cybersecurity preparedness.

The DDoS Analyzer project addresses this need by developing an AI-based platform that enables automated classification of network traffic datasets. Users can upload traffic logs, which are then preprocessed, analyzed, and classified as normal or malicious using supervised machine learning models. The system also provides batch processing capabilities, allowing organizations to study past network traffic, detect previously unnoticed attack patterns, and improve the efficiency of their threat detection processes.

By offering visual representations of traffic analysis and model performance, the platform makes it easier for users to interpret results and make informed decisions. The comparison of different machine learning models helps determine the most effective approach for specific network environments, reducing reliance on manual inspection and minimizing errors. With a user-friendly web interface, DDoS Analyzer ensures that both technical and non-technical users can access actionable insights without needing advanced cybersecurity expertise.

From a societal perspective, the project empowers cybersecurity analysts, network administrators, and smaller organizations to strengthen their defenses against DDoS attacks. By providing affordable, open-source, and automated analytical tools, it reduces manual effort, supports offline investigation, and enables informed decision-making for both immediate and long-term security improvements. The system contributes to a safer, more resilient digital ecosystem by helping organizations proactively understand, respond to, and mitigate the impact of network attacks.

##### Objective

The objective of the DDoS Analyzer project is to develop an AI-based network traffic analysis platform that enables automated detection and classification of DDoS attacks while providing actionable insights for cybersecurity teams:

* Allow users to upload both current and historical network traffic datasets for analysis.
* Classify traffic into normal or malicious (DDoS) using supervised machine learning models with high accuracy.
* Support batch processing of traffic logs to study past incidents and identify hidden attack patterns.
* Provide visual representations of detection results, including traffic distribution and model performance graphs.
* Enable comparison of different machine learning models to determine the most effective detection approach.
* Offer a user-friendly web interface for uploading datasets, visualizing results, and accessing insights easily.
* Minimize manual effort by eliminating the need for packet-by-packet inspection.
* Facilitate long-term cybersecurity strategy improvement by uncovering trends in historical traffic data.
* Empower network administrators and security analysts to make informed decisions and strengthen defenses.
* Ensure the solution is cost-effective and accessible for organizations and individuals without requiring specialized infrastructure.

##### Existing Solution

Several network security tools exist in the market for analyzing and detecting DDoS attacks. However, these solutions often focus on real-time traffic monitoring or signature-based detection, which limits their ability to analyze historical data and uncover long-term attack patterns.

**Existing Solutions and Their Limitations:**

1. **Wireshark:**
   * Provides manual packet inspection and supports a wide range of network protocols
   * Offers real-time and offline traffic analysis but requires technical expertise for accurate interpretation [1], [13].
   * Limited automation for classifying DDoS attacks, relying heavily on user intervention [12].
2. **Snort:**
   * Uses signature-based detection to identify threats in real-time traffic [14].
   * Provides rule-based alerts and integrates with SIEM tools for enterprise networks .
   * Cannot analyze historical logs or automatically classify traffic, limiting its utility for long-term attack trend analysis [8], [9].

**Problems with Existing Solutions:**

* **Limited Historical Analysis:** Most tools focus on live monitoring and fail to provide insights into past incidents.
* **High Manual Effort:** Manual inspection of packets or logs is time-consuming and error-prone.
* **Dependence on Signatures or Rules:** Tools like Snort may miss evolving or unknown attack patterns due to reliance on predefined signatures.
* **Accessibility:** Many solutions require advanced technical knowledge, restricting use by smaller organizations or educational purposes.

**How DDoS Analyzer Addresses These Issues:**  
DDoS Analyzer is an AI-powered, open-source platform that classifies network traffic using machine learning models. Unlike existing solutions, it can process both current and historical traffic datasets, detect DDoS attacks automatically, and provide visual analytics for better understanding. By eliminating the need for manual inspection and signature dependency, the platform enhances efficiency and accessibility, enabling network administrators and security teams to uncover hidden attack patterns and improve long-term defenses. To the best of our knowledge, no tool currently available provides automated DDoS classification using machine learning for both historical and current traffic in a user-friendly, open-source solution.

##### Proposed Solution

To address the limitations of existing network security tools, the DDoS Analyzer introduces an AI-powered platform that enables automated classification and analysis of network traffic datasets. Unlike traditional tools such as Wireshark or Snort, which rely on manual inspection or signature-based detection, DDoS Analyzer provides a cost-effective, user-friendly, and open-source solution capable of analyzing both current and historical traffic logs.

**Key Aspects of the Proposed Solution:**

1. **AI-Powered Traffic Classification** – Uses supervised machine learning models to accurately classify network traffic as normal or DDoS, reducing reliance on manual inspection and signature-based rules.
2. **Batch Processing of Traffic Logs** – Supports the analysis of historical datasets, enabling security teams to study past incidents, identify attack patterns, and improve long-term defense strategies.
3. **Visual Analytics and Model Comparison** – Provides graphical representations of detection results and compares performance across different machine learning models, helping users understand which approach works best for their environment.
4. **User-Friendly Web Interface** – Simplifies dataset uploads, visualization of results, and access to actionable insights, making the tool accessible to both technical and non-technical users.
5. **Enhanced Accessibility and Efficiency** – Eliminates the need for packet-by-packet inspection, reducing manual effort while improving detection accuracy and operational efficiency.
6. **Comprehensive Security Insights** – Provides actionable recommendations to refine firewall rules, update detection policies, and strengthen network defenses based on historical and current traffic analysis.
7. **Scalable and Future-Ready** – Designed to integrate additional machine learning models and adapt to evolving cyber threats, ensuring long-term usability and technological relevance.

### 

### Chapter 2: Software Requirement Specification

##### Introduction

###### Purpose The purpose of the DDoS Analyzer – AI-Based DDoS Attack Classification project is to provide a machine learning powered web application that analyzes uploaded network traffic datasets and accurately classifies DDoS and normal traffic, addressing the limitations of manual log inspection and traditional signature based tools. This SRS defines the system’s complete functional and non functional requirements, ensuring a clear and unified understanding for developers, supervisors, evaluators, and cybersecurity analysts, while serving as a technical guide throughout the development process to keep the final product aligned with the project’s goals and scope.

###### Scope

###### The software product to be developed is DDoS Analyzer (AI-Based DDoS Attack Classification), a web-based analytical tool that processes uploaded network traffic datasets to classify normal and malicious traffic accurately. The system will support batch-based detection, allowing users to upload offline datasets rather than relying on real-time packet capture. It will preprocess the data, train machine learning models such as Random Forest, and generate comparative performance metrics, including accuracy scores and visual analytics. The tool will provide a clear summary of classified traffic, highlighting the percentage of normal and DDoS attack flows, while offering graphs and charts that help users interpret the results easily.

###### The system will not perform live, real-time traffic monitoring, nor will it automate network mitigation. Instead, it focuses entirely on offline analysis, supervised model training, dataset evaluation, and visualization of results. The software is intended for cybersecurity analysts, students, and network administrators who need an accessible platform for incident analysis and long-term threat assessment using historical logs. It delivers clear benefits such as reduced manual inspection, improved attack classification accuracy, and an integrated environment for model comparison. The scope aligns with the higher-level project goals stated in the proposal, ensuring that the system contributes to better intrusion analysis, enhanced understanding of network behavior patterns, and improved cybersecurity decision-making.

###### Definitions, acronyms, and abbreviations

• **AI (Artificial Intelligence):** The capability of computer systems to perform tasks that typically require human intelligence, including pattern recognition and automated decision making.  
  
• **ML (Machine Learning):** A branch of AI that enables systems to learn from data and improve classification accuracy without explicit programming.  
  
• **DDoS (Distributed Denial of Service):** A cyberattack in which multiple compromised systems flood a network or server with traffic, disrupting normal service.  
  
• **NIDS (Network Intrusion Detection System):** A security tool that monitors network traffic to identify malicious activity or policy violations.  
  
• **CICIDS 2019:** A publicly available intrusion detection dataset widely used for research in attack classification and network anomaly detection.  
  
• **Dataset:** A structured collection of network traffic records used for model training, testing, and evaluation.  
  
• **Preprocessing:** The cleaning and preparation of raw network traffic data, including normalization and feature extraction, before training an ML model.  
  
• **Classification Model:** A machine learning model that assigns traffic records to categories such as normal or DDoS.  
  
• **Flask:** A lightweight Python web framework used to develop the DDoS Analyzer’s interface and backend logic.  
  
• **Dashboard:** A web interface where users upload datasets, view results, compare models, and analyze visual outputs.  
  
• **Accuracy:** A performance metric indicating how often the ML model correctly classifies network traffic.  
  
• **Confusion Matrix:** A table that displays the number of correct and incorrect predictions made by a classification model.  
  
• **ERD (Entity Relationship Diagram):** A visual representation of the database structure defining entities and their relationships.  
  
• **API (Application Programming Interface):** A set of protocols that allows software components to communicate within the system.  
  
• **Normalization:** A preprocessing technique that scales dataset values into a consistent range to improve model performance.  
  
• **Batch Processing:** The ability to process stored datasets instead of analyzing real-time traffic, allowing offline incident analysis.  
  
• **SRS (Software Requirements Specification):** A structured document describing the functional, non-functional, and technical requirements of the system.

##### Overall description

###### Product perspective

The DDoS Analyzer is an independent and self-contained web application that uses machine learning to classify network traffic as normal or malicious. It processes uploaded datasets instead of relying on live traffic, which makes it suitable for offline analysis, research environments, and batch incident investigation. The system is built with Python and Flask, it integrates data preprocessing, feature extraction, model training, model comparison, and visualization within a single platform. The product requires only a web browser and locally stored datasets, allowing cybersecurity analysts and network administrators to analyze historical traffic logs without depending on external tools. The system provides an interactive dashboard, graphical analytics, and modular components for dataset upload, preprocessing, machine learning classification, and reporting. It stands apart because no such unified, affordable, and AI-based batch analysis tool is commonly available for students or small organizations.

### ****System Interfaces****

The DDoS Analyzer interacts with system components through:

• File upload interface for CSV or processed CICIDS/CICDDoS datasets,  
• Flask backend endpoints for preprocessing, model training, prediction, and visualization,  
• Machine learning modules such as Random Forest or other classifiers implemented in Python,  
• Matplotlib and other libraries for generating performance graphs,  
• Internal APIs that connect the front-end dashboard to backend classification functions.

The system supports integration with:  
• Dataset management and preprocessing pipelines,  
• Batch-based traffic classification modules,  
• Real-time progress updates during model training,  
• Downloadable results and logs for later use.

### ****User Interfaces****

The user interface is designed to remain simple, responsive, and beginner friendly. Major UI components include:

**Dashboard**  
• Overview of all modules including dataset upload, preprocessing, model training, classification, and analytics,  
• Summary widgets showing processed files and detection results.

**Dataset Upload Page**  
• Upload window for CSV   
• Validation checks to ensure correct format.

**Preprocessing Module**  
• Options for cleaning, encoding, normalization, and feature selection,  
• Preview of processed datasets.

**Model Training and Testing Module**  
• Selection of machine learning algorithms,  
• Display of accuracy, precision, recall, confusion matrix, and model comparison.

**Analytics and Visualization Module**  
• Graphs showing normal vs malicious traffic distribution,  
• Feature importance charts,  
• Performance plots for each trained model.

**Reports Page**  
• Exportable results in CSV or image format,  
• Summary of detected DDoS or related attack categories.

### ****Hardware Interfaces****

• Requires a standard computer capable of running Python and Flask,  
• Minimum 4GB RAM recommended for dataset processing,  
• Dual-core processor or better for training ML models,  
• Optional GPU support for faster training.

### ****Software Interfaces****

• Developed using Python, Flask, Scikit-learn, Pandas, and Matplotlib,  
• Frontend built with HTML, CSS, JavaScript, Flask or similar,  
• Supports integration with Jupyter-based workflows for dataset testing,  
• Compatible with CICFlowMeter formatted datasets and standard CSV logs.

### ****Communications Interfaces****

• Localhost access over HTTP for normal operation,  
• Optional HTTPS support if deployed on a server,  
• Internal communication between frontend and backend through REST APIs.

### ****Memory Requirements****

• Minimum 4GB RAM for dataset preprocessing and ML training,  
• Approximately 1GB secondary storage for saved datasets, models, and generated results.

### ****Operations****

• Normal operations include dataset upload, preprocessing, training, and classification,  
• Special operations include exporting reports, comparing multiple models, and analyzing custom datasets,  
• Option for saving trained models for future predictions.

### ****Site Adaptation Requirements****

• Configuration based on dataset structure such as CICIDS or custom logs,  
• Adjustable preprocessing settings depending on network environment,  
• Custom thresholds or sensitivity settings for different organizations or academic use.

###### Product functions

**Dataset Upload**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID: | FR\_01 |  |  |  |
| Name: | Dataset Upload |  |  |  |
| Description | Input | Output | Requirements | Basic Workflow |
| Allow users to upload network traffic datasets for analysis | CSV file, CICIDS formatted dataset | Dataset successfully uploaded and stored for preprocessing and analysis | Correct dataset format | User selects the dataset file and clicks Upload. System validates the file and saves it. |

Table 1 Functional Requirement Dataset Upload

**Data Preprocessing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID: | FR\_02 |  |  |  |
| Name: | Data Preprocessing |  |  |  |
| Description | Input | Output | Requirements | Basic Workflow |
| Clean, encode, normalize, and prepare dataset before model training | Uploaded dataset, preprocessing options such as encoding, normalization, feature selection | Preprocessed dataset with selected transformations applied | Proper dataset upload and valid preprocessing configuration | User selects preprocessing options and clicks Process. System applies preprocessing and prepares the dataset. |

Table 2 Functional Requirement Data Preprocessing

**Model Training**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID: | FR\_3 |  |  |  |
| Name: | Model Training |  |  |  |
| Description | Input | Output | Requirements | Basic Workflow |
| Automatically train multiple predefined machine learning models such as Random Forest, SVM, or Gradient Boosting, and compare their performance | Preprocessed dataset | Ranked model results showing accuracy, precision, recall, confusion matrix, and the best performing model | Sufficient RAM and CPU, correct preprocessing applied | System automatically trains all predefined models, compares their performance, and displays the model ranking and metrics |

Table 3 Functional Requirement Model Training

**Traffic Classification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID: | FR\_04 |  |  |  |
| Name: | Traffic Classification |  |  |  |
| Description | Input | Output | Requirements | Basic Workflow |
| Classify network traffic as normal or malicious using the best performing trained model | Trained ML models and test dataset or uploaded dataset | Prediction results showing normal versus malicious traffic, including DDoS categories | Preprocessed dataset and trained models availability | User uploads or selects dataset, system runs classification using the best performing model and displays results |

Table 4 Functional Requirement Traffic Classification

**Visualization and Analytics**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID: | FR\_05 |  |  |  |
| Name: | Visualization and Analytics |  |  |  |
| Description | Input | Output | Requirements | Basic Workflow |
| Generate graphs showing traffic distribution, model performance, and key insights | Processed dataset, model performance metrics, and feature importance | Graphs for traffic distribution, feature importance, confusion matrix, and comparison graphs | Dataset must be preprocessed and models trained | User selects visualization type, system generates graphs such as bar charts, pie charts, or line plots. |

Table 5 Functional Requirement Visualization and Analytics

**Report Export**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID: | FR\_06 |  |  |  |
| Name: | Report Export |  |  |  |
| Description | Input | Output | Requirements | Basic Workflow |
| Export results, graphs, and model summary in a downloadable format | Classification results, graphs, performance metrics | Downloadable report in CSV or image format | Completed classification and generated graphs | User clicks Export, system generates the file for download. |

Table 6 Functional Requirement Report Export

###### User characteristics

The intended users of the DDoS Analyzer system include cybersecurity analysts, network administrators, SOC team members, university students, researchers, and instructors involved in teaching or studying network security. These users generally have a basic to advanced educational background in computing or cybersecurity, along with varying levels of technical expertise. The system is designed to be intuitive and accessible, yet users benefit from having a foundational understanding of network traffic concepts and dataset formats. Users should be able to navigate web-based dashboards, interpret analytical results, and understand system-generated model summaries.

Key user characteristics include:

• Users should have access to a computer or laptop capable of running a web application with a stable internet connection.

• A basic understanding of cybersecurity concepts, network traffic features, and common attack behaviors enhances the usage experience.

• Users should be comfortable performing simple tasks such as dataset upload, viewing classification reports, and interpreting graphical outputs.

• Users may require familiarity with using datasets like CICIDS or CICDDoS and understanding the meaning of model performance metrics.

###### Constraints

• Must comply with cybersecurity data handling regulations and maintain ethical use of network traffic datasets.

• Requires stable hardware resources such as sufficient RAM and CPU to support preprocessing and machine learning model training.

• Utilizes Flask or a Python based web framework to deliver system functionality through a browser interface.

• Dataset compatibility is limited to formats aligned with CICIDS or CICDDoS datasets for correct preprocessing and feature extraction.

• Supports parallel operations such as preprocessing, model comparison, and result generation to maintain efficient workflow.

• Includes audit logging features to record model evaluations, dataset uploads, and system-generated results for traceability.

• Maintains accuracy requirements for model comparison to ensure fair ranking of trained classifiers.

• Implements encrypted dataset handling and adheres to security best practices to protect sensitive traffic logs and user inputs.

• System workflow is dependent on machine learning operations, meaning performance may vary based on dataset size and system hardware.

###### Assumptions and dependencies

• Users will have access to stable computing resources capable of running the web interface and handling moderate-size datasets.

• The application will operate through modern web browsers such as Chrome, Firefox, or Edge with JavaScript enabled.

• Users will upload datasets in compatible formats, allowing the system to preprocess and analyze them without manual data correction.

• Model training and classification results rely on the availability of Python libraries such as Scikit-learn, Pandas, NumPy, and Matplotlib.

• The accuracy and speed of the system depend on the hardware specifications of the server or machine hosting the application.

• Backend services, dataset storage paths, and model directories must remain functional and properly configured.

• The training process assumes consistency in feature structure across datasets modeled after CICIDS and CICDDoS.

• System performance and model ranking accuracy depend on the availability and stability of underlying machine learning packages.

###### Apportioning of requirements

• Real time detection and live traffic monitoring may be considered for future versions, while the initial release focuses on batch processing only.

• Integration with SIEM tools or enterprise dashboards can be added in later updates to increase system compatibility with SOC environments.

• Support for additional machine learning models or deep learning architectures may be included in future versions to enhance detection accuracy.

• Automated report emailing and external alert notifications may be implemented in later releases based on user needs.

• Multi dataset comparison features and detailed feature engineering modules may be extended in future updates.

##### Specific requirements

This section describes the functional and non functional requirements of the DDoS Analyzer system at a level detailed enough for designers to create the system and testers to verify that the software meets all defined requirements.

###### Functional Requirement

This subsection outlines the main functional modules that form the core of the DDoS Analyzer system. These functions address the primary needs of users and support the system’s objective of detecting and classifying DDoS traffic using uploaded datasets.

**• Dataset Upload:**

Allows users to upload network traffic datasets in compatible formats so the system can begin the analysis process.

**• Data Preprocessing:**

Cleans the uploaded dataset, applies normalization, and prepares features needed for accurate machine learning model training.

**• Model Training:**

Automatically trains predefined machine learning models, evaluates their performance, and generates a ranked comparison of results.

**• Traffic Classification:**

Uses the best performing trained model to classify uploaded traffic as normal or malicious, including specific DDoS categories.

**• Visualization and Analytics:**

Create charts and visual summaries that explain dataset distribution, attack patterns, and model performance in an easy to understand way.

**• Report Export:**

Provides download options for classification summaries, analytics, and visual outputs in formats such as CSV or image files.

###### Non-functional Requirements

This sub section defines the primary nonfunctional requirements necessary to ensure that the system performs reliably and delivers a consistent, secure, and maintainable user experience.

* + **Usability**:  
    The interface should be clean, intuitive, and simple to navigate for users with different technical backgrounds. The system should minimize user effort by automating model selection and comparison.
  + **Reliability**:  
    The system should remain available and responsive under normal load. Dataset uploads and preprocessing should be executed without failure, and all stored results must remain consistent.
  + **Performance**:  
    The system should handle moderate sized datasets efficiently, with acceptable training and classification times. Graph generation, page transitions, and model ranking outputs should respond smoothly.
  + **Design Constraints:**  
    The system must operate using a Flask or Python based backend and support modern web browsers. It should maintain responsive layout and compatibility with different screen sizes.
  + **Portability:**  
    The application should run on various operating systems such as Windows, Linux, or macOS without major configuration changes.
  + Maintainability:  
    All code modules should be well structured and documented, allowing future updates, debugging, and feature expansion with minimal difficulty.
  + **License Agreement:**  
    The system should comply with legal requirements related to dataset usage and data handling. All libraries included must follow their respective license policies.

# Chapter 3: Use Case Analysis

**Upload Dataset:**

A diagram of a data processing process

AI-generated content may be incorrect.

Figure 1 Usecase Diagram Upload Dataset

**Usecase diagram Details**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_01 | |
| Use Case Name | Upload Dataset | |
| Description | Allows the user to upload a dataset file for analysis. | |
| Primary Actor | User | |
| Secondary Actor | None | |
| Pre-Condition | Dataset is uploaded and ready for analysis. | |
| Post-Condition | What is the output of this function | |
| Basic Flow | Actor Action | System Action |
|  | Chooses and submits a dataset file | Validates and stores the dataset file |
| Alternate Flow | If the file is invalid, system shows an error. | |

Table 7 Usecase Upload Dataset

**Preprocess Dataset:**

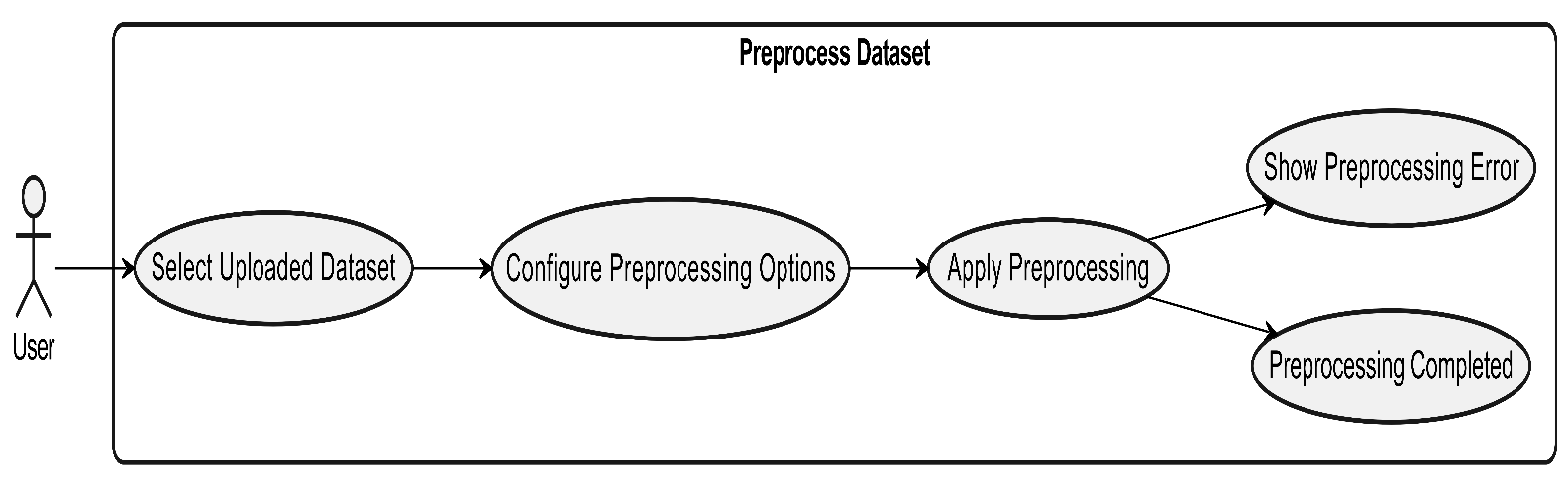
****

Figure 2 Usecase Diagram Preprocess Dataset

**Usecase diagram Details**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_02 | |
| Use Case Name | Preprocess Dataset | |
| Description | Allow the user to preprocess the uploaded dataset. | |
| Primary Actor | User | |
| Secondary Actor | None | |
| Pre-Condition | Dataset must be uploaded successfully. | |
| Post-Condition | Dataset is cleaned and prepared for model training. | |
| Basic Flow | Actor Action | System Action |
|  | Selects dataset and applies preprocessing options | Cleans and preprocesses the dataset |
| Alternate Flow | If preprocessing fails, system shows an error. | |

Table 8 Usecase Preprocess Dataset

**Train Model:**

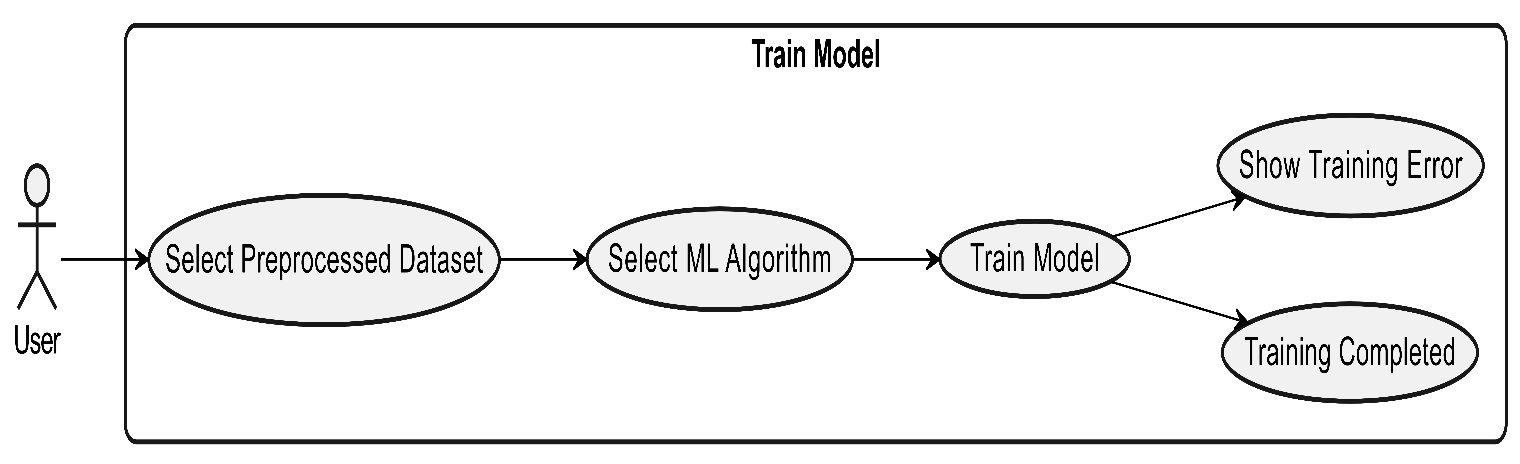


Figure 3 Usecase Diagram Train Model

**Usecase diagram Details**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_03 | |
| Use Case Name | Train Model | |
| Description | Allows the user to train a machine learning model. | |
| Primary Actor | User | |
| Secondary Actor | None | |
| Pre-Condition | Dataset must be preprocessed successfully. | |
| Post-Condition | Trained model is saved for traffic classification. | |
| Basic Flow | Actor Action | System Action |
|  | Selects dataset and algorithm | Trains and stores the machine learning model |
| Alternate Flow | If training fails, system shows an error. | |

*Table 9 Usecase Train Model*

**Classify Traffic:**

A close-up of a sign

AI-generated content may be incorrect.

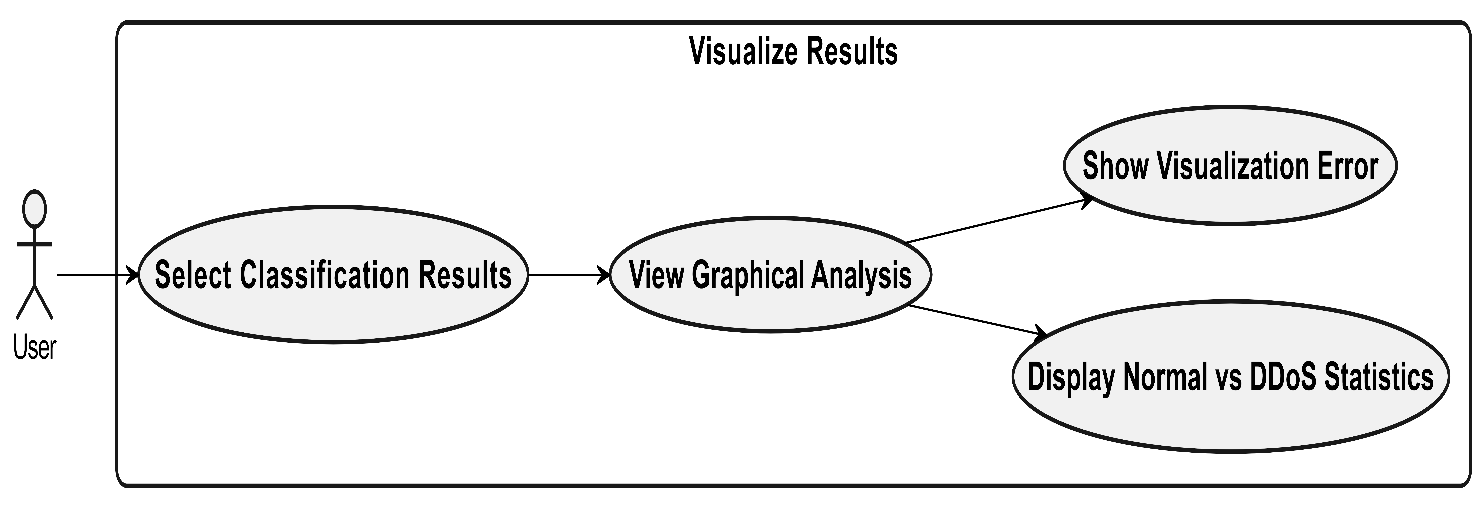
*Figure 4 Usecase Diagram Classify Traffic*

**Usecase diagram Details**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_04 | |
| Use Case Name | Classify Traffic | |
| Description | Allows the user to classify traffic as normal or DDoS. | |
| Primary Actor | User | |
| Secondary Actor | None | |
| Pre-Condition | A trained model must be available in the system. | |
| Post-Condition | Traffic is classified as normal or DDoS and displayed. | |
| Basic Flow | Actor Action | System Action |
|  | Selects model and dataset | Classifies traffic and shows normal vs DDoS traffic |
| Alternate Flow | If classification fails, system shows an error. | |

*Table 10 Usecase Classify Traffic*

**Visualize Results:**



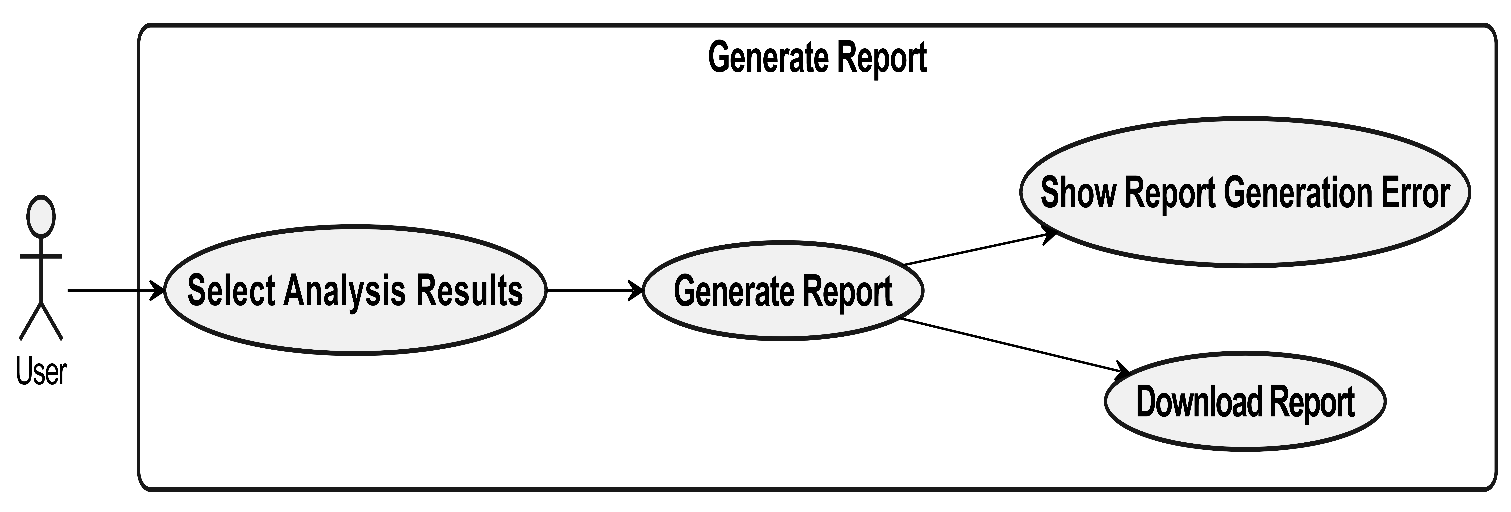
*Figure 5 Usecase Diagram Visualize Results*

**Usecase diagram Details**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_05 | |
| Use Case Name | Visualize Results | |
| Description | Allows the user to view visual analysis of classification results in graphical form. | |
| Primary Actor | User | |
| Secondary Actor | None | |
| Pre-Condition | Traffic must be classified successfully. | |
| Post-Condition | Graphical results are displayed to the user. | |
| Basic Flow | Actor Action | System Action |
|  | Selects classification results | Displays graphs and normal vs DDoS statistics |
| Alternate Flow | If visualization fails, system shows an error. | |

*Table 11 Usecase Visualize Results*

**Generate Report:**



*Figure 6 Usecase Diagram Generate Report*

**Usecase diagram Details**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC\_06 | |
| Use Case Name | Generate Report | |
| Description | Allows the user to generate and download an analysis report of the results. | |
| Primary Actor | User | |
| Secondary Actor | None | |
| Pre-Condition | Analysis results must be available in the system. | |
| Post-Condition | Report is generated and downloaded by the user. | |
| Basic Flow | Actor Action | System Action |
|  | Selects analysis results | Generates and allows download of the report |
| Alternate Flow | If report generation fails, system shows an error. | |

*Table 12 Usecase Generate Report*

# Chapter 4: Design

In this section, we provide the design analysis of our modules including the following designs

1. Architecture Diagram
2. ERD with data dictionary
3. Data Flow diagram
4. Class Diagram
5. Activity Diagram
6. Sequence Diagram
7. Collaboration Diagram
8. State Transition Diagram
9. Component Diagram
10. Deployment Diagram

#### Architecture Diagram

Define the graphical representation of the concepts, their principles, elements and components that are part of your project.

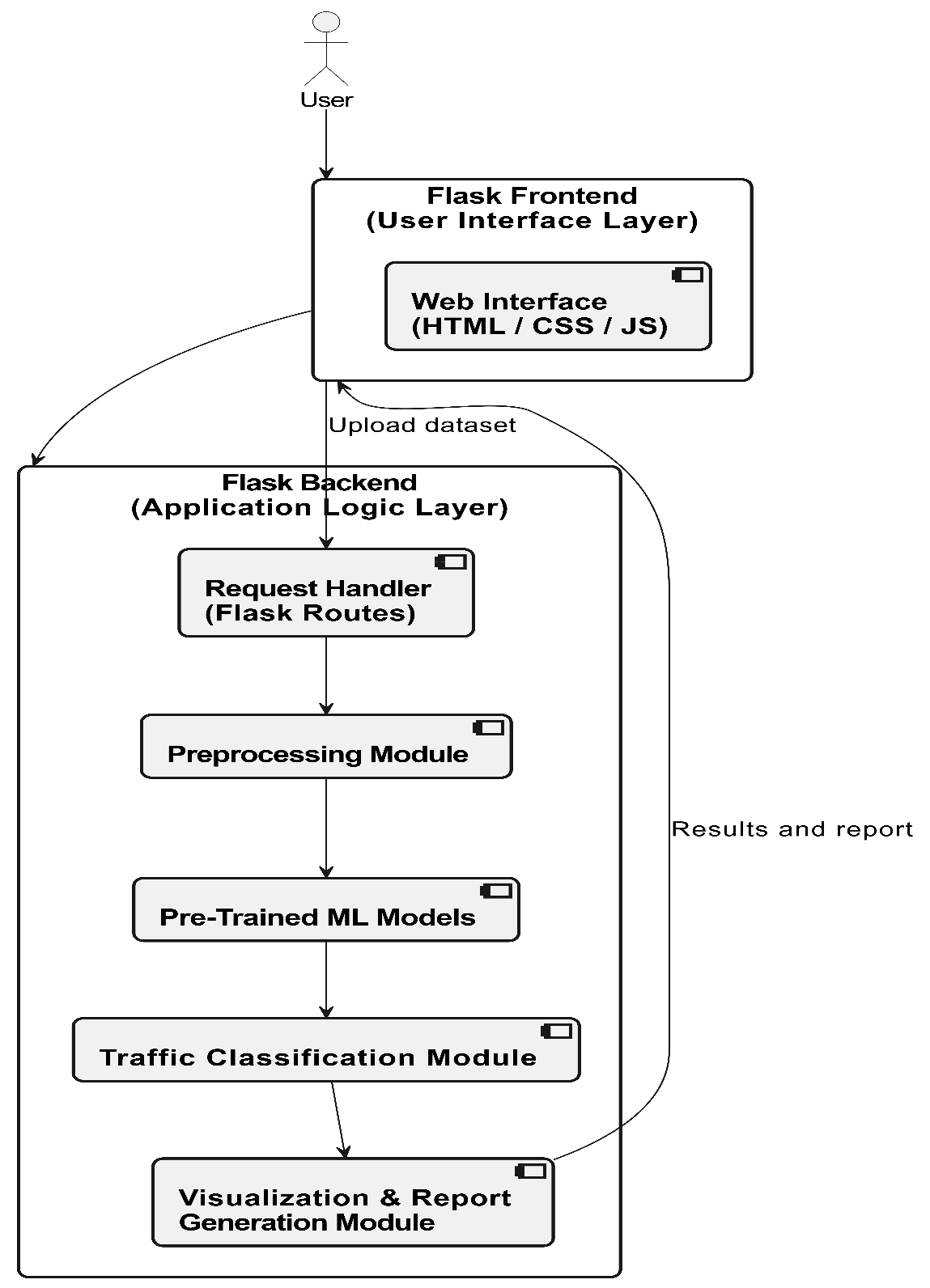


Figure 7 Architecture Diageam

#### ERD with data dictionary

EntityRelationshipDiagram with complete relations with dependencies of your project

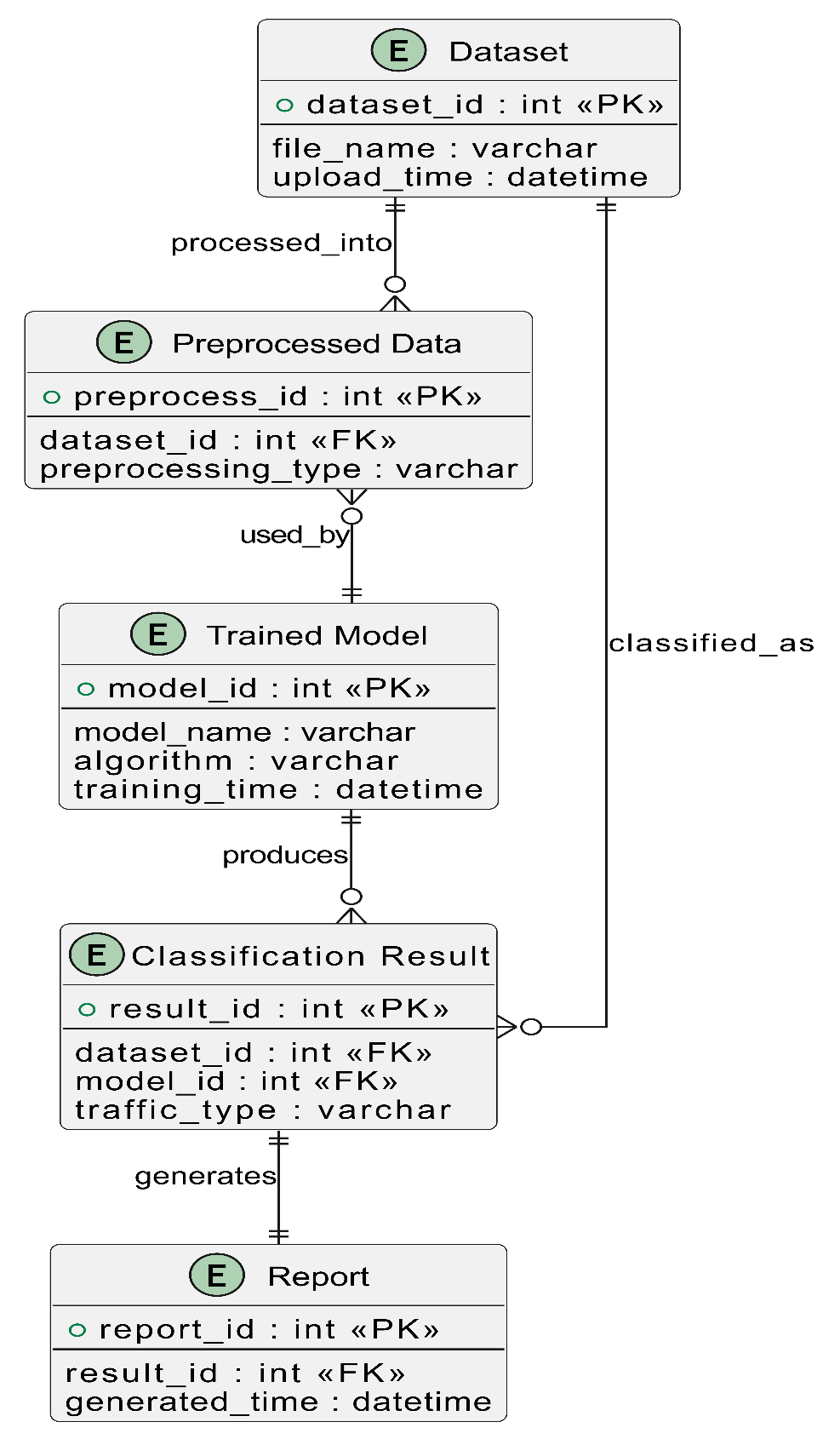


Figure 8 Entity Relationship Diagram

**Data Dictionary of ERD**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Attribute** | **Type** | **Description** |
| **Dataset** | dataset\_id | int (PK) | Unique dataset identifier |
|  | file\_name | varchar | Name of uploaded dataset file |
|  | upload\_time | datetime | Time of dataset upload |
| **Preprocessed Data** | preprocess\_id | int (PK) | Unique preprocessing record ID |
|  | dataset\_id | int (FK) | References Dataset |
|  | preprocessing\_type | varchar | Applied preprocessing method |
| **Trained Model** | model\_id | int (PK) | Unique trained model ID |
|  | model\_name | varchar | Name of trained model |
|  | algorithm | varchar | Machine learning algorithm used |
|  | training\_time | datetime | Time when model was trained |
| **Classification Result** | result\_id | int (PK) | Unique classification result ID |
|  | model\_id | int (FK) | References TrainedModel |
|  | dataset\_id | int (FK) | References Dataset |
|  | traffic\_type | varchar | Normal or DDoS classification |
| **Report** | report\_id | int (PK) | Unique report ID |
|  | result\_id | int (FK) | References ClassificationResult |
|  | generated\_time | datetime | Time of report generation |

*Table 13 Data Dictionary of ERD*

#### Data Flow diagram

Data flow diagram includes two levels

###### The level 0

The flow of information inside the system is defined in this level

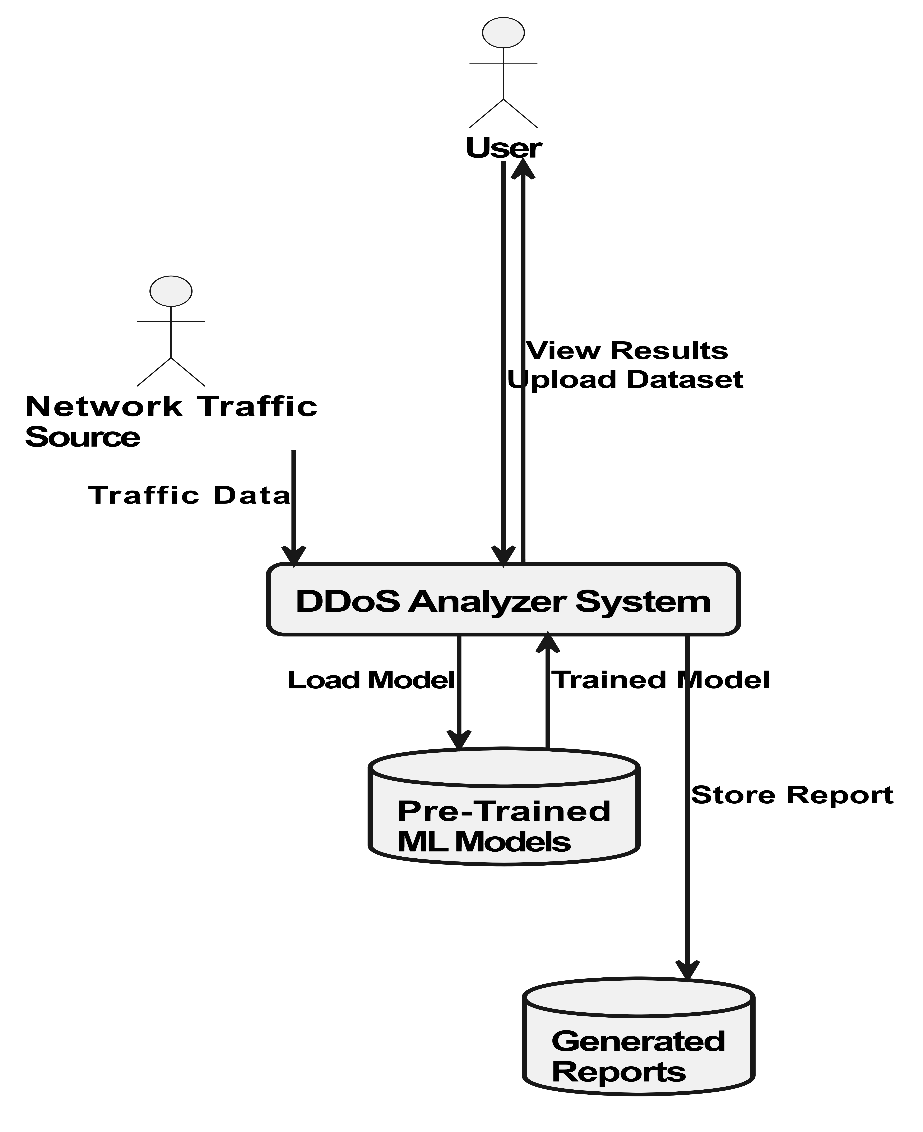


Figure 9 Level 0 DFD

###### The level 1

The flow of information outside the system is defined in this level

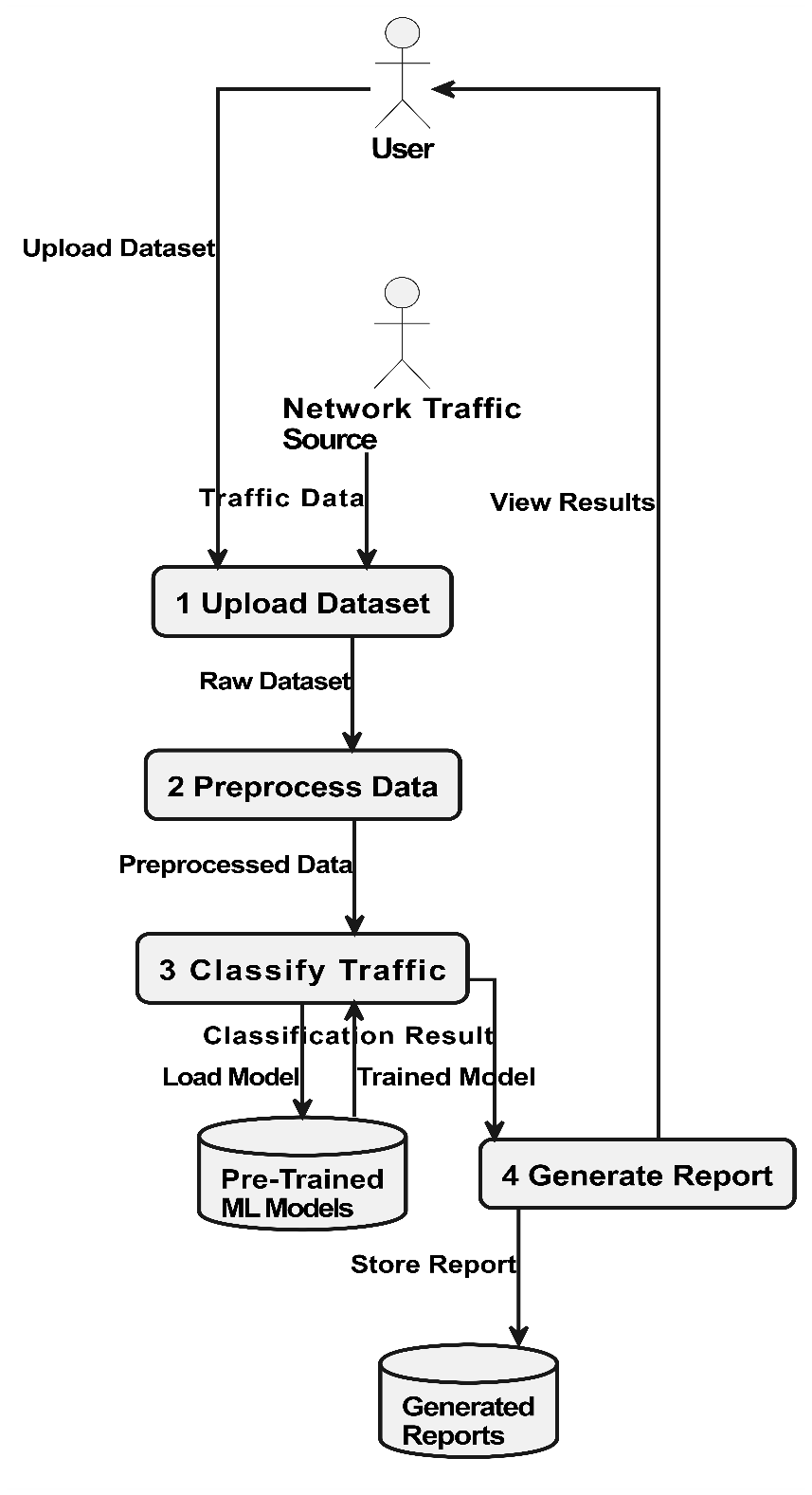


Figure 10 Level 1 DFD

#### Class Diagram

Describe the structure of a project by showing the systems classes, their attributes, operations (or methods), and the relationships among objects.

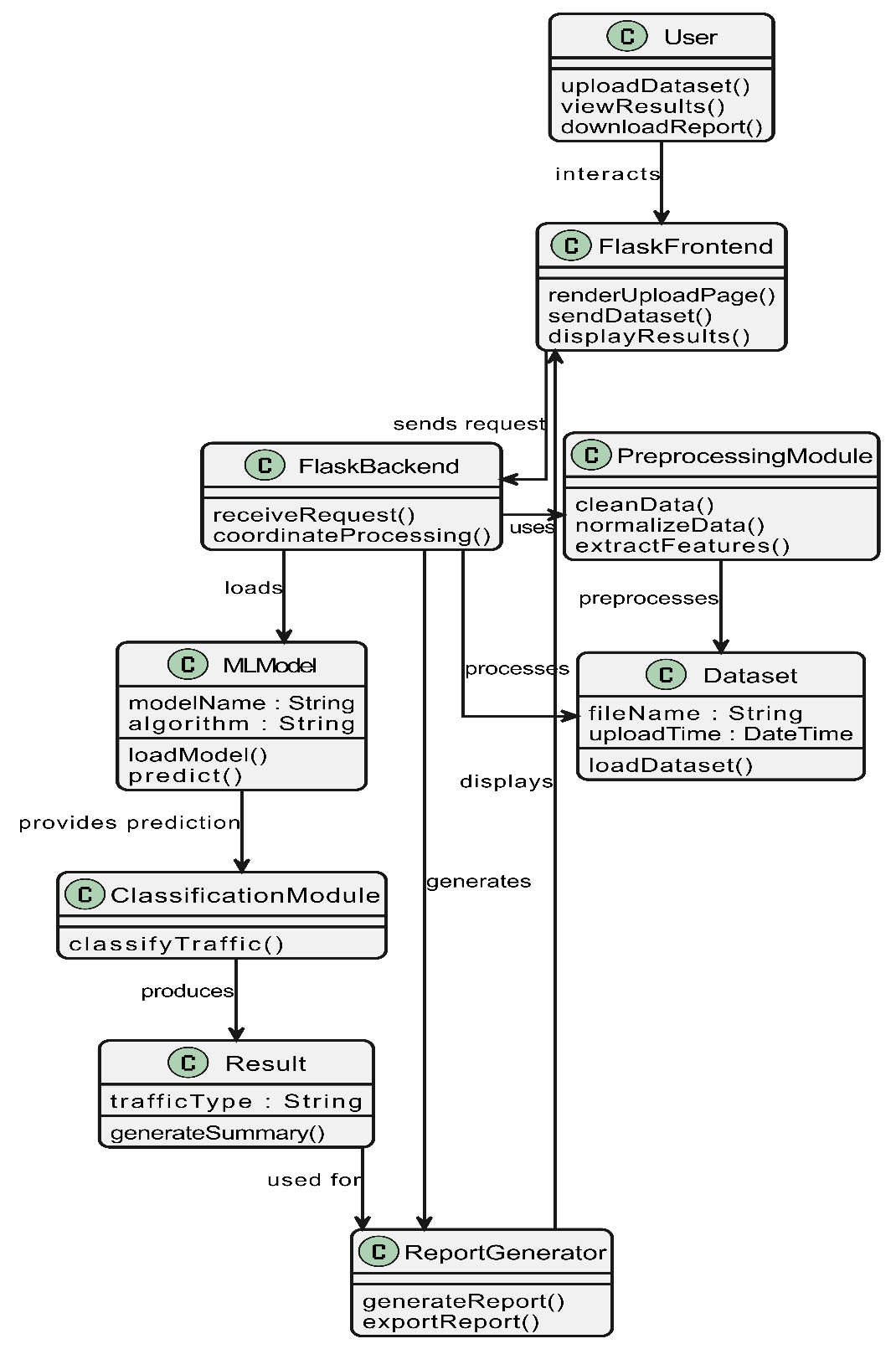


Figure 11 Class Diagram

#### Activity Diagram

This diagram includes all the activity diagrams of the functional requirements of your project along with the aggregated activity diagram

**4.5.1 Activity diagram for Upload Dataset:**

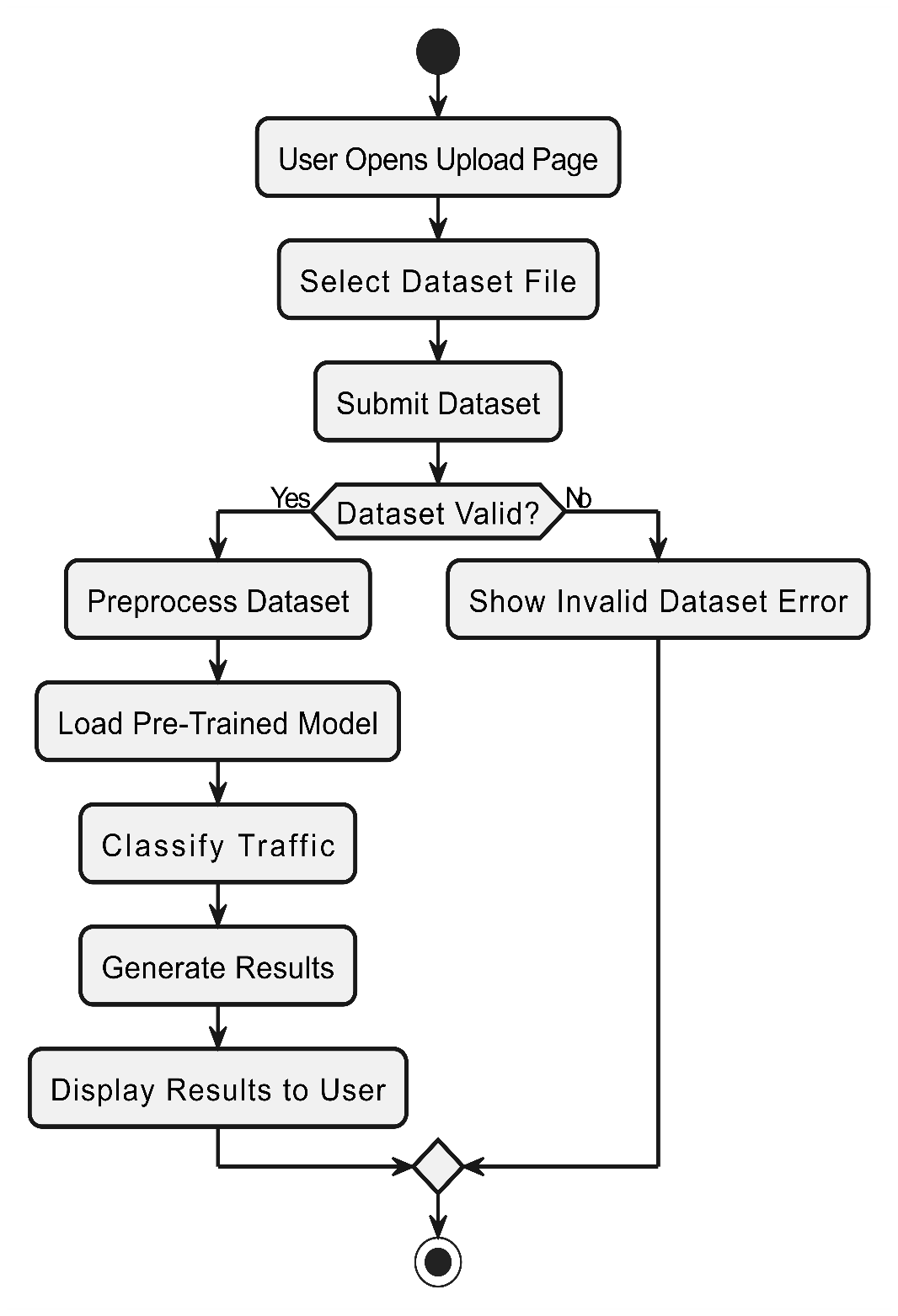


Figure 12 Activity Diagram Upload Dataset

**4.5.2 Activity diagram for Preprocess Dataset:**

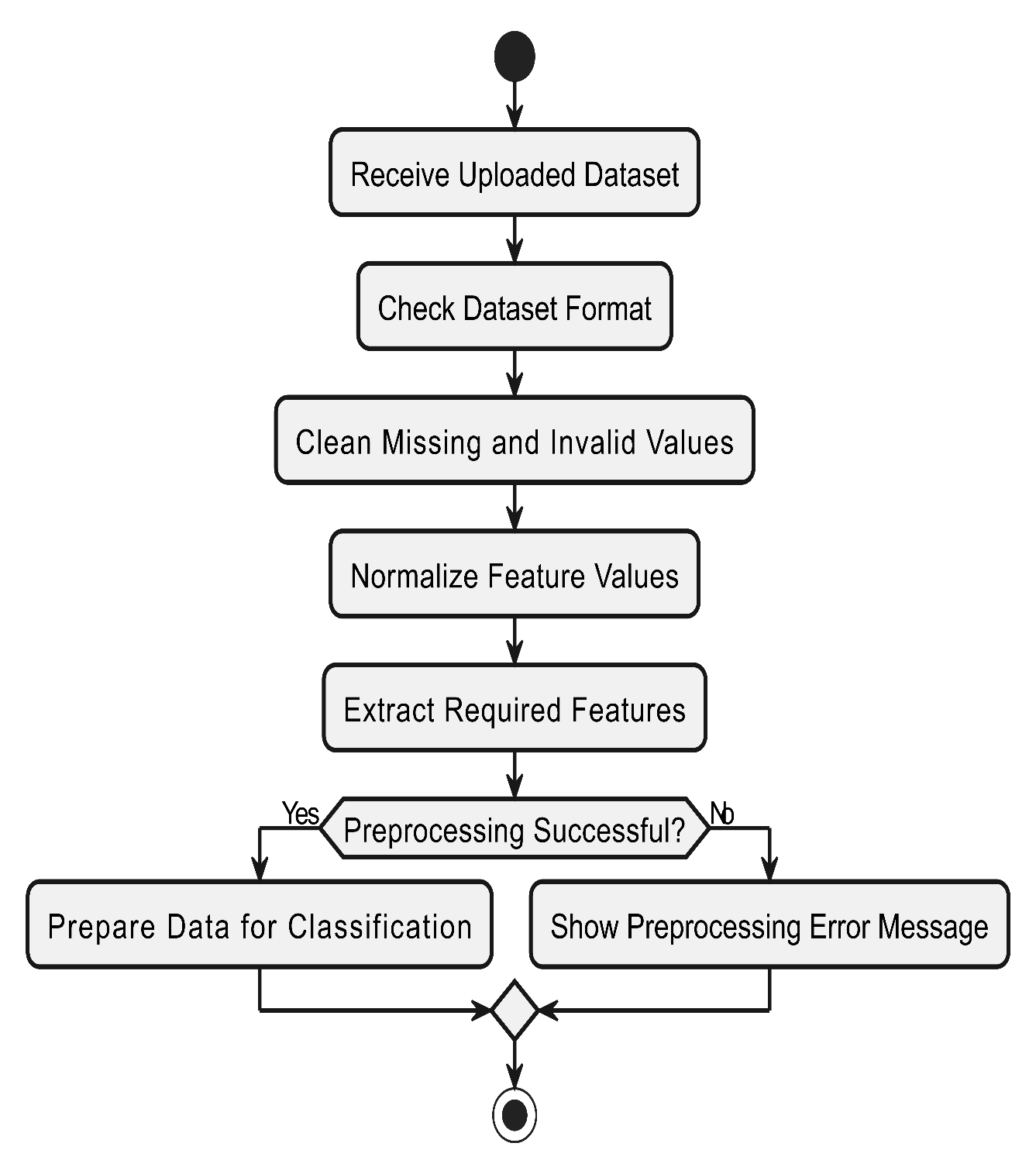


Figure 13 Activity Diagram Preprocess Dataset

**4.5.3 Activity diagram for Classify Network Traffic:**

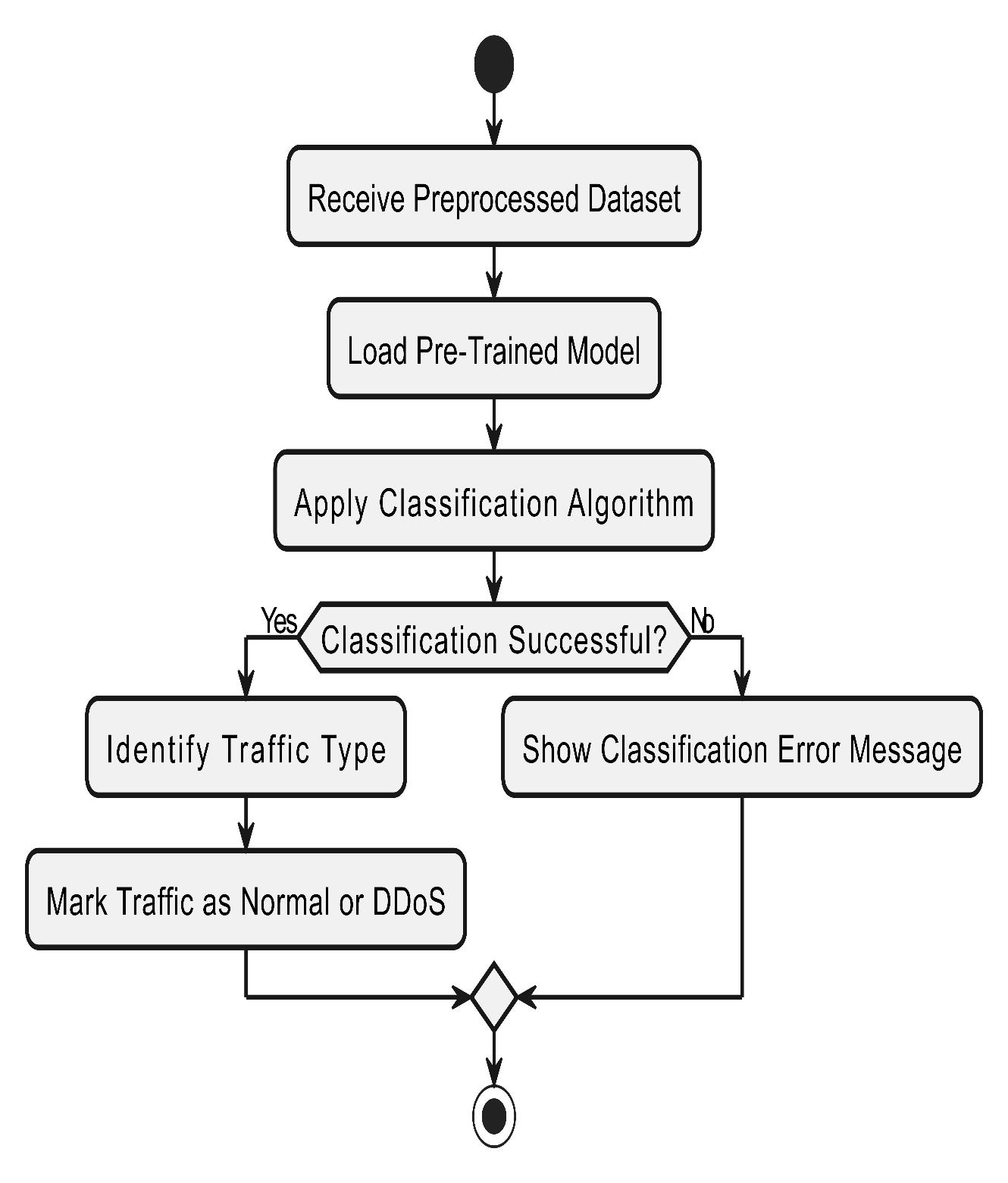


Figure 14 Activity Diagram Classify Network Traffic

**4.5.4 Activity diagram for Generate Report:**

A diagram of a report

AI-generated content may be incorrect.

Figure 15 Activity Diagram Generate Report

**4.5.5 Activity diagram for View Results and Download Report:**

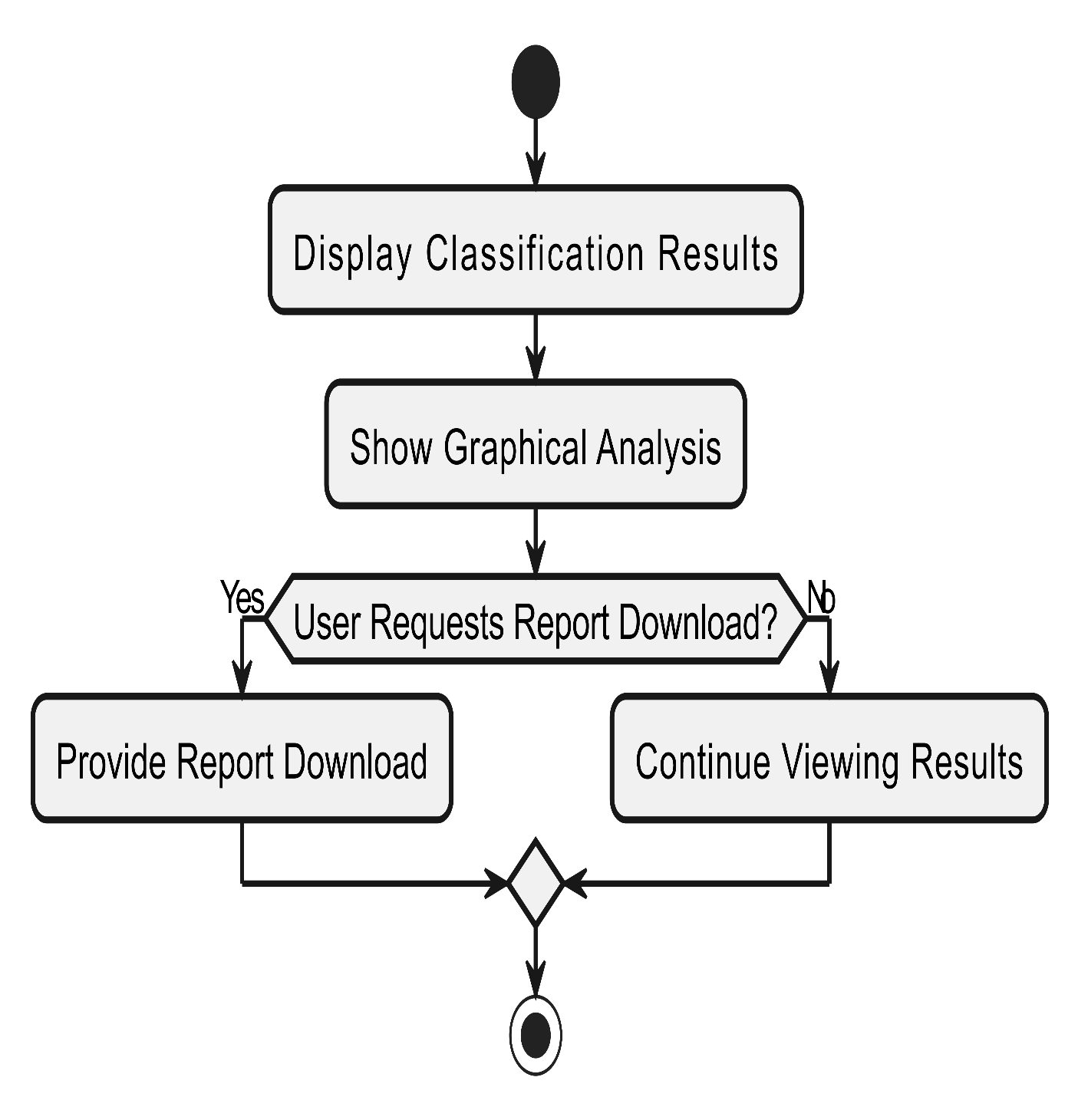


Figure 16 Activity Diagram View Results and Download Report

**4.5.6 Activity diagram for Aggregated DDoS Analyzer System:**

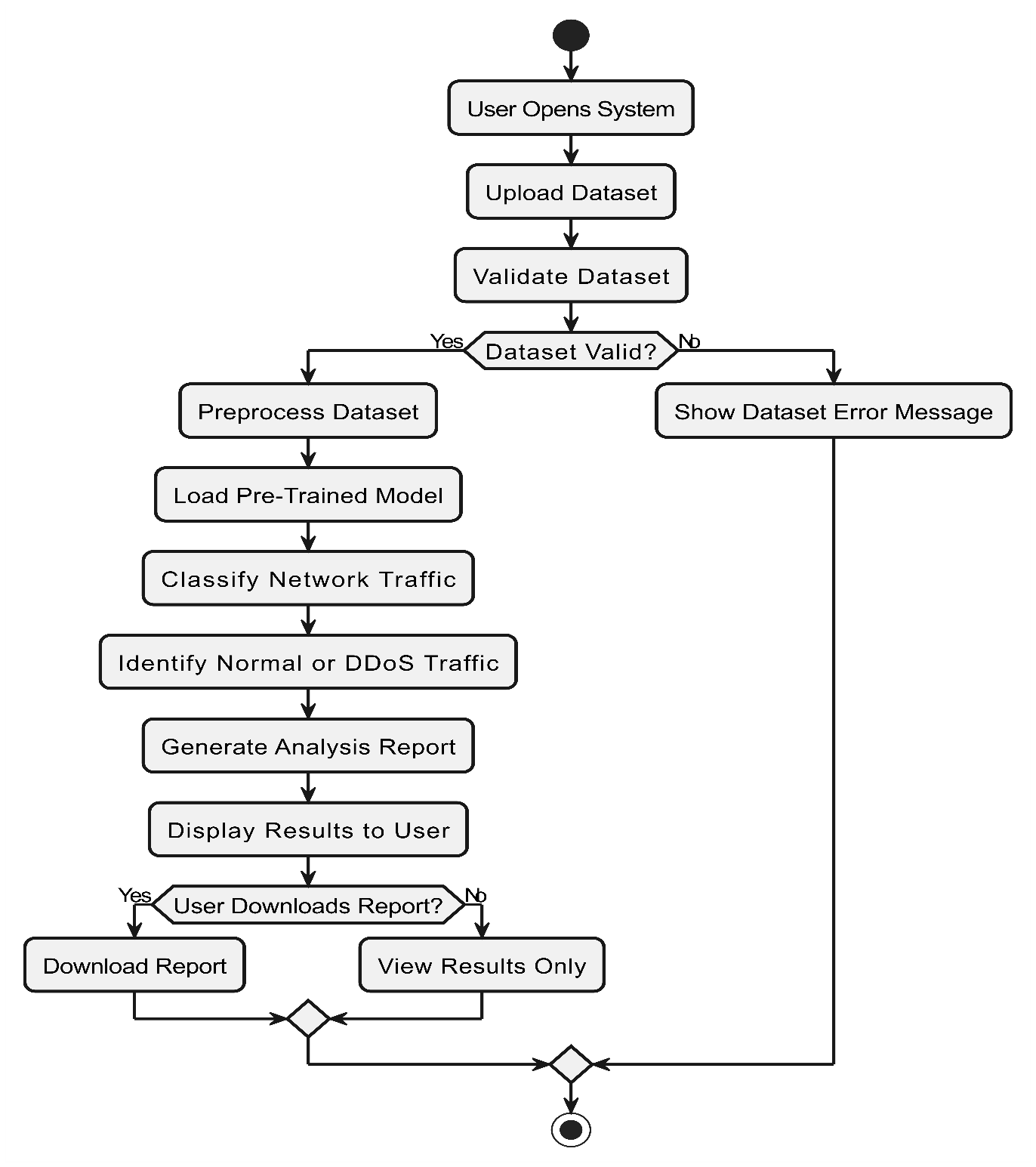


Figure 17 Activity Diagram Aggregated DDoS Analyzer System

#### Sequence Diagram

This diagram includes all the Sequence diagrams of the functional requirements of your project along with the aggregated Sequence diagram

**4.6.1 Sequence diagram for Upload Dataset:**

A diagram of a process flow

AI-generated content may be incorrect.

Figure 18 Sequence Diagram Upload Dataset

**4.6.2 Sequence diagram for Preprocess Dataset:**

A diagram of a process

AI-generated content may be incorrect.

*Figure 19 Sequence Diagram Create Account*

**4.6.3 Sequence diagram for Classify Network Traffic:**

A diagram of a model

AI-generated content may be incorrect.

*Figure 20 Sequence Diagram Classify Network Traffic*

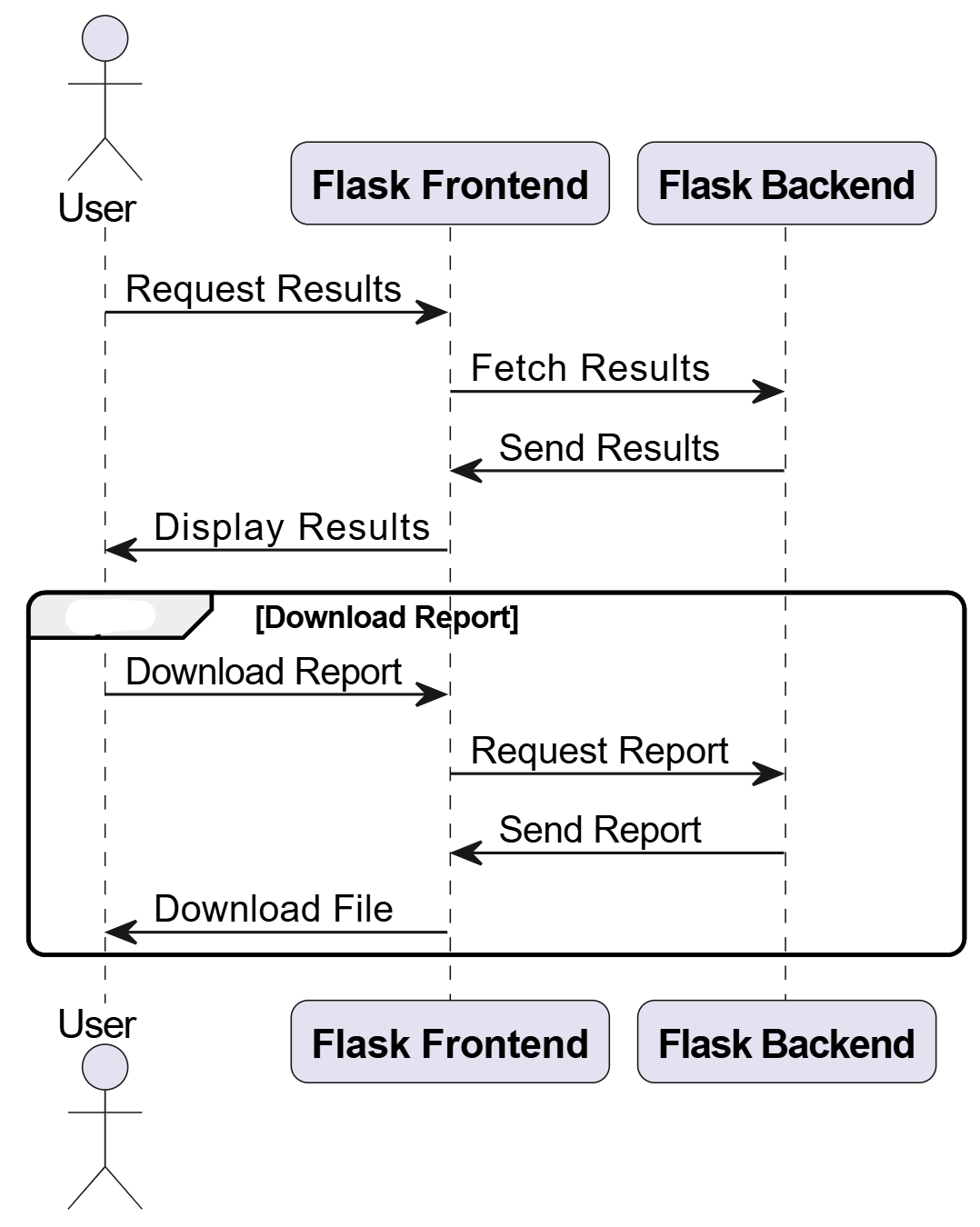
**4.6.4 Sequence diagram for Generate Report:**

A diagram of a report generator

AI-generated content may be incorrect.

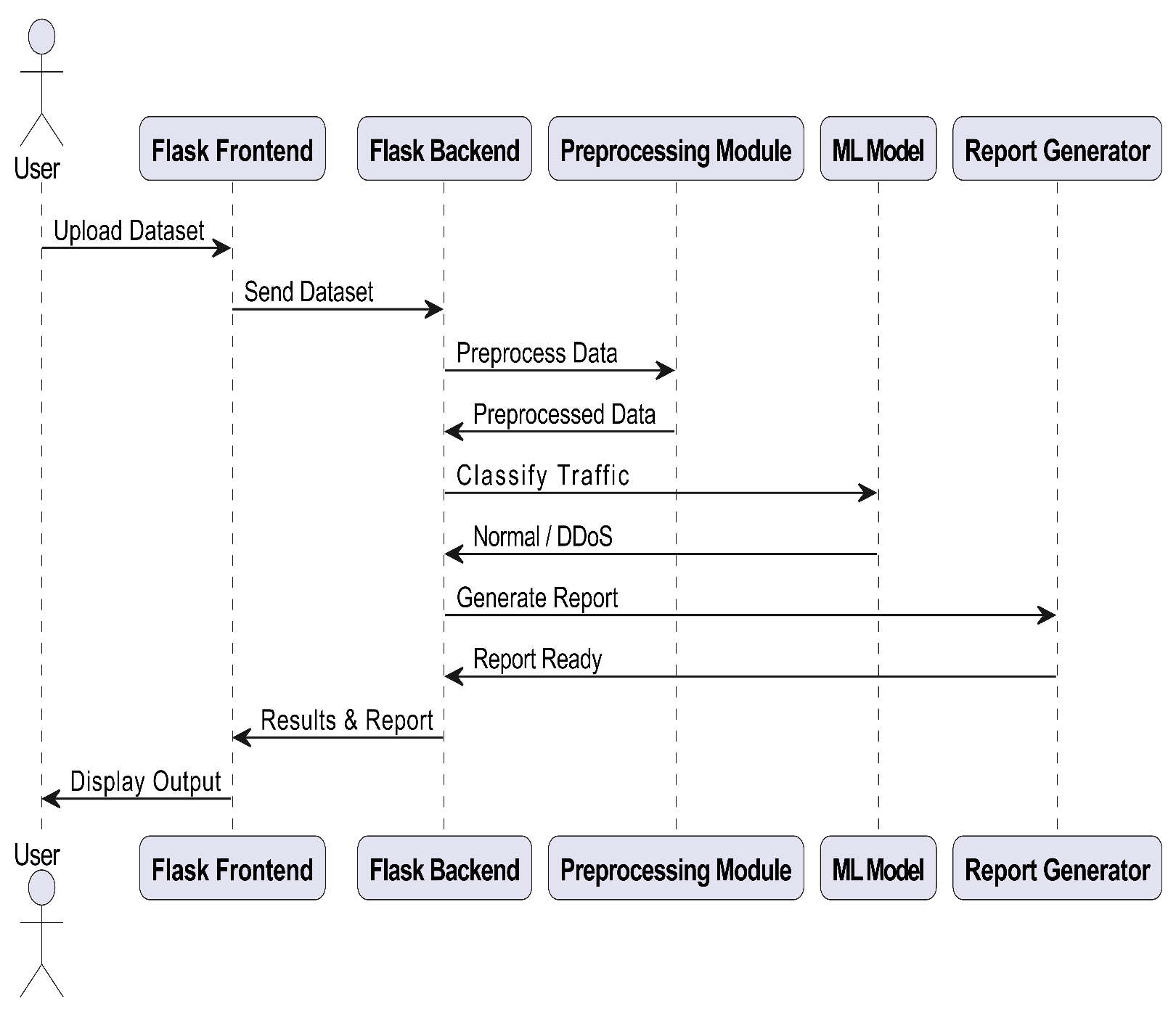
*Figure 21 Sequence Diagram Generate Report*

**4.6.5 Sequence diagram for View Results & Download Report:**



*Figure 22 Sequence Diagram View Results & Download Report*

**4.6.6 Aggregated Sequence Diagram for** **DDoS Analyzer System:**

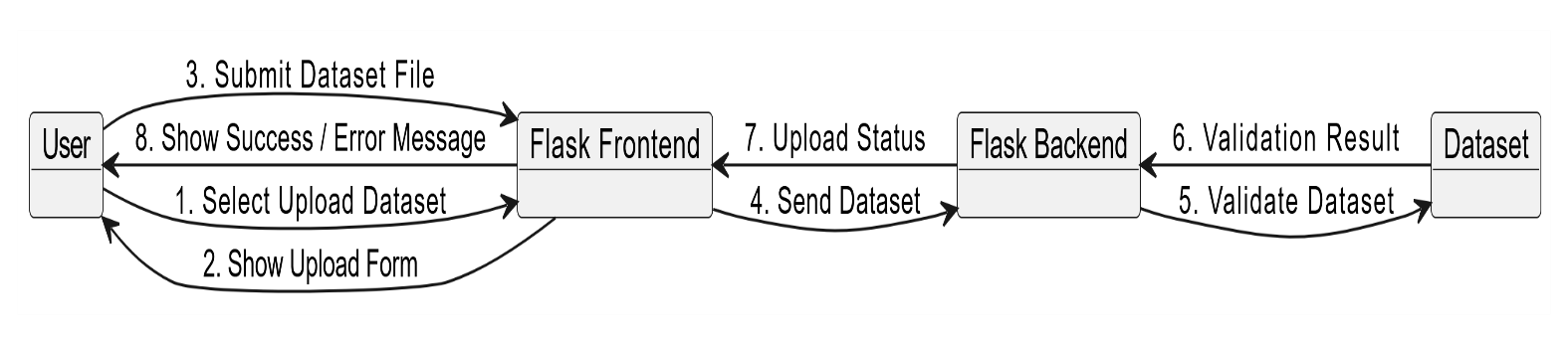


*Figure 23 Sequence Diagram DDoS Analyzer System*

#### Collaboration Diagram

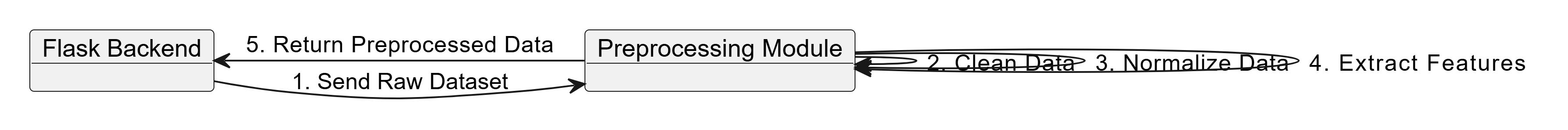
It shows the object organization as shown below. Here in collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

**4.7.1 Collaboration Diagram – Upload Dataset**

****

*Figure 24 Collaboration Diagram Upload Dataset*

**4.7.2 Collaboration Diagram – Preprocess Dataset**

****

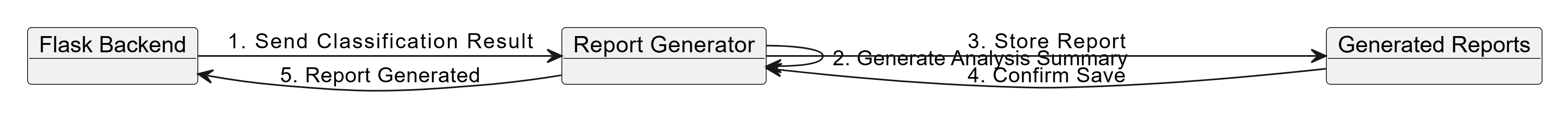
*Figure 25 Collaboration Diagram* *Preprocess Dataset*

**4.7.3 Collaboration Diagram – Classify Network Traffic**

****

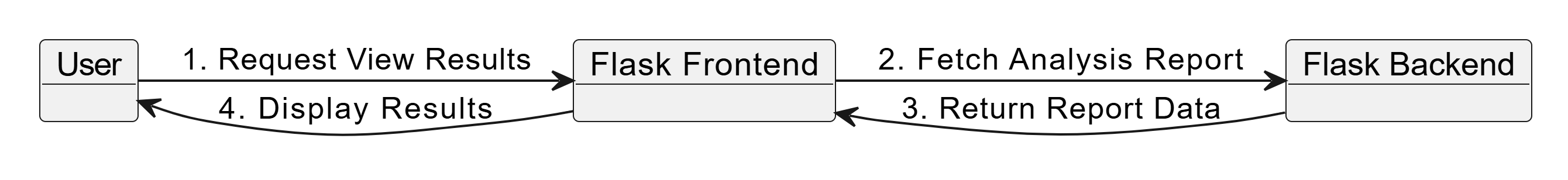
*Figure 26 Collaboration Diagram Classify Network Traffic*

**4.7.4 Collaboration Diagram – Generate Report**

****

*Figure 27 Collaboration Diagram Generate Report*

**4.7.5 Collaboration Diagram – View Results**

****

*Figure 28 Collaboration Diagram View Results*

#### State Transition Diagram

State Transition diagram is used to describe the states of different objects in its life cycle. So, the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately

**State Transition Diagram – DDoS Analyzer System:**

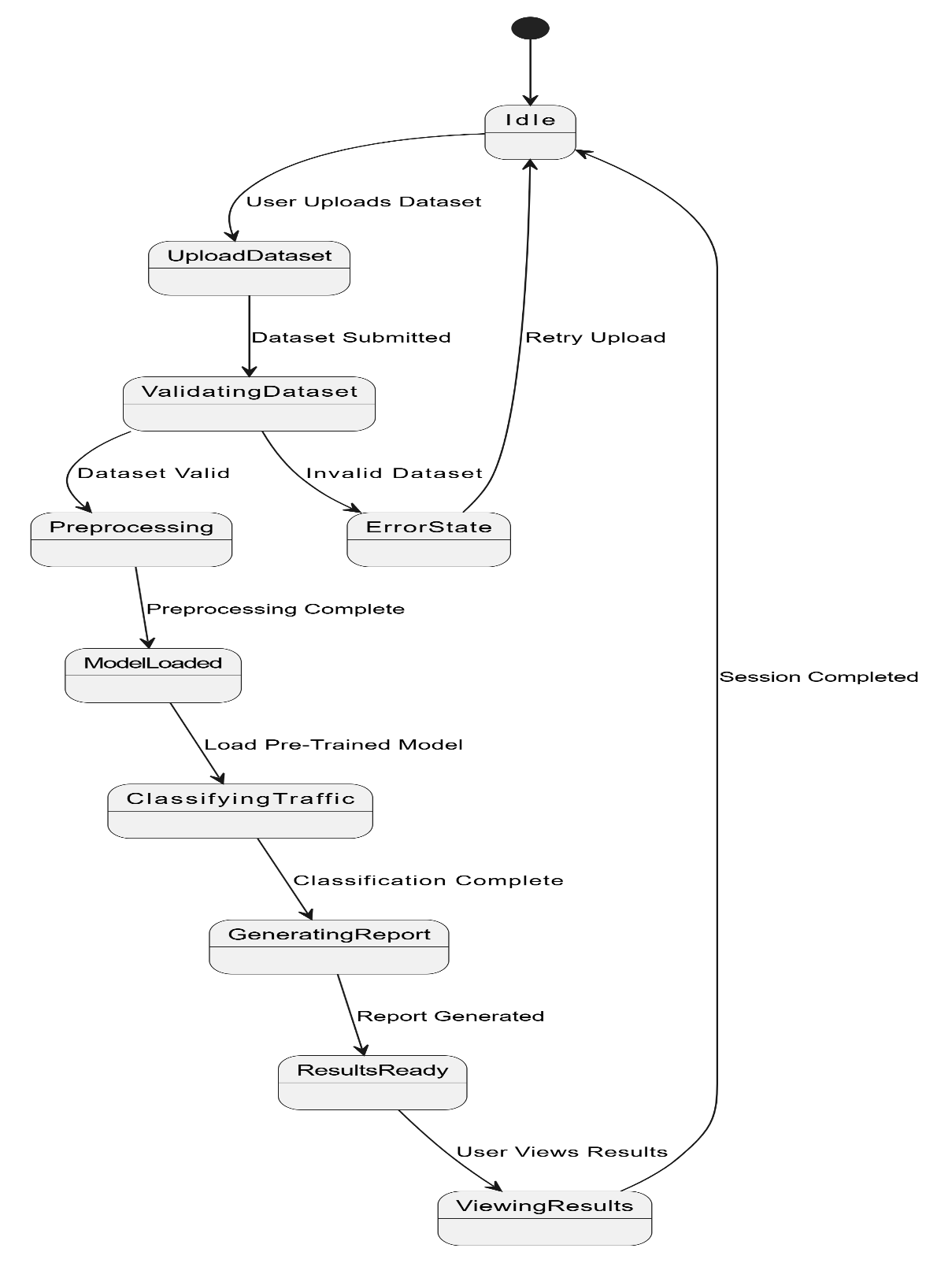


Figure 29 State Transition Diagram DDoS Analyzer System

#### Component Diagram

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries etc.

**Component Diagram – DDoS Analyzer System:**

**A diagram of a software system

AI-generated content may be incorrect.**

Figure 30 Component Diagram DDoS Analyzer System

#### Deployment Diagram

Deployment diagram represents the deployment view of a system. It is related to the component diagram.  
  
**Deployment Diagram – DDoS Analyzer System**

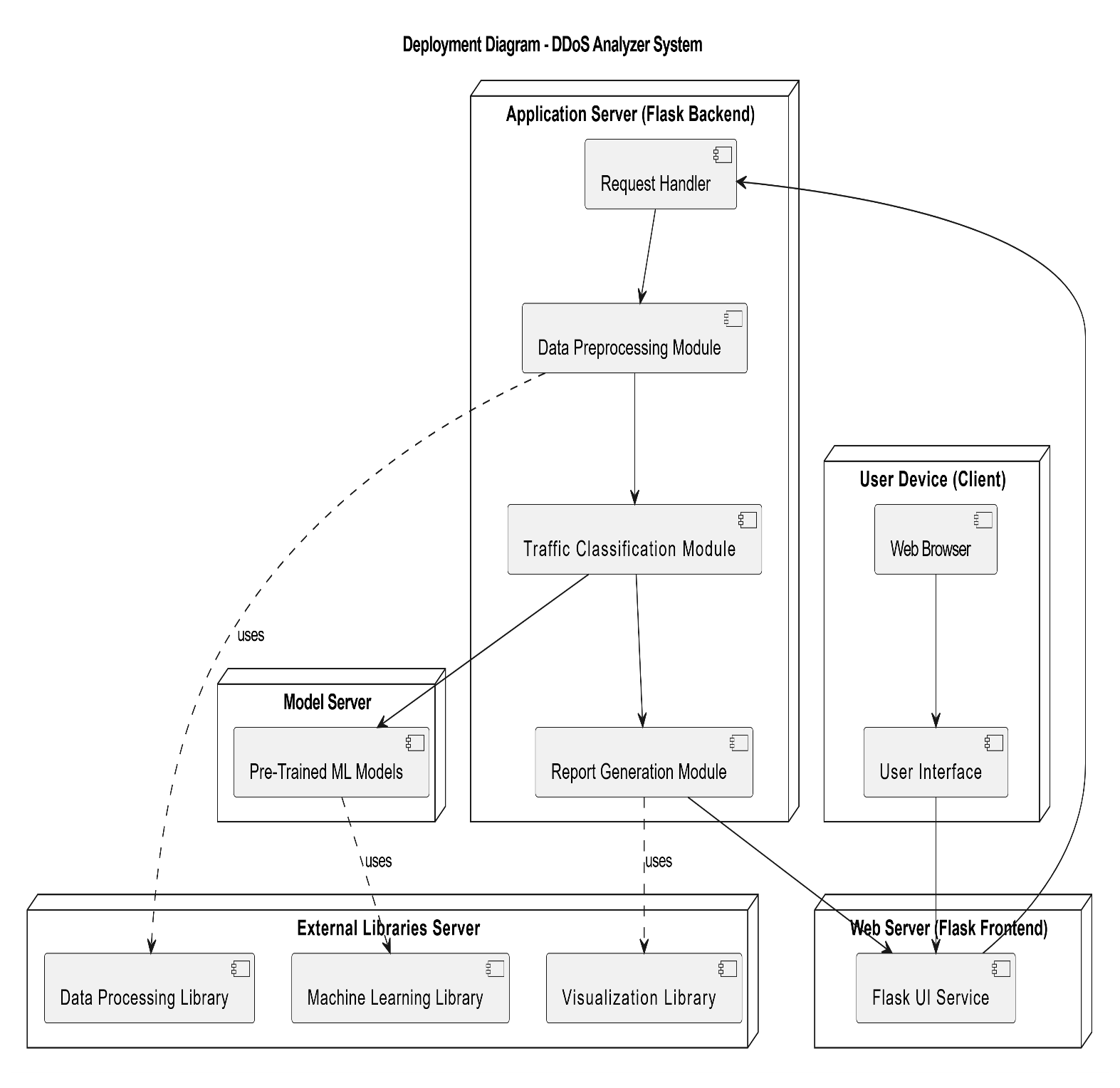


Figure 31 Deployment Diagram **DDoS Analyzer System**

# Chapter 5: User Manual

This chapter provides a detailed guide on how to use the features of the **DDoS Analyzer – AI-Based DDoS Attack Classification System**. Each section explains the purpose of a specific screen, its functionality, and the steps required to perform actions within the system. Screenshots are referenced in appropriate places to illustrate the user interface and workflow of the application. This chapter helps users understand how to interact with the system efficiently for dataset analysis, visualization, and report generation.

## ****5.1 Dataset Upload Screen****

This is the initial screen of the DDoS Analyzer application. From this screen, the user can upload a network traffic dataset for offline batch analysis. The system supports CSV-based datasets generated from network traffic monitoring tools.

**Purpose of the Screen:**

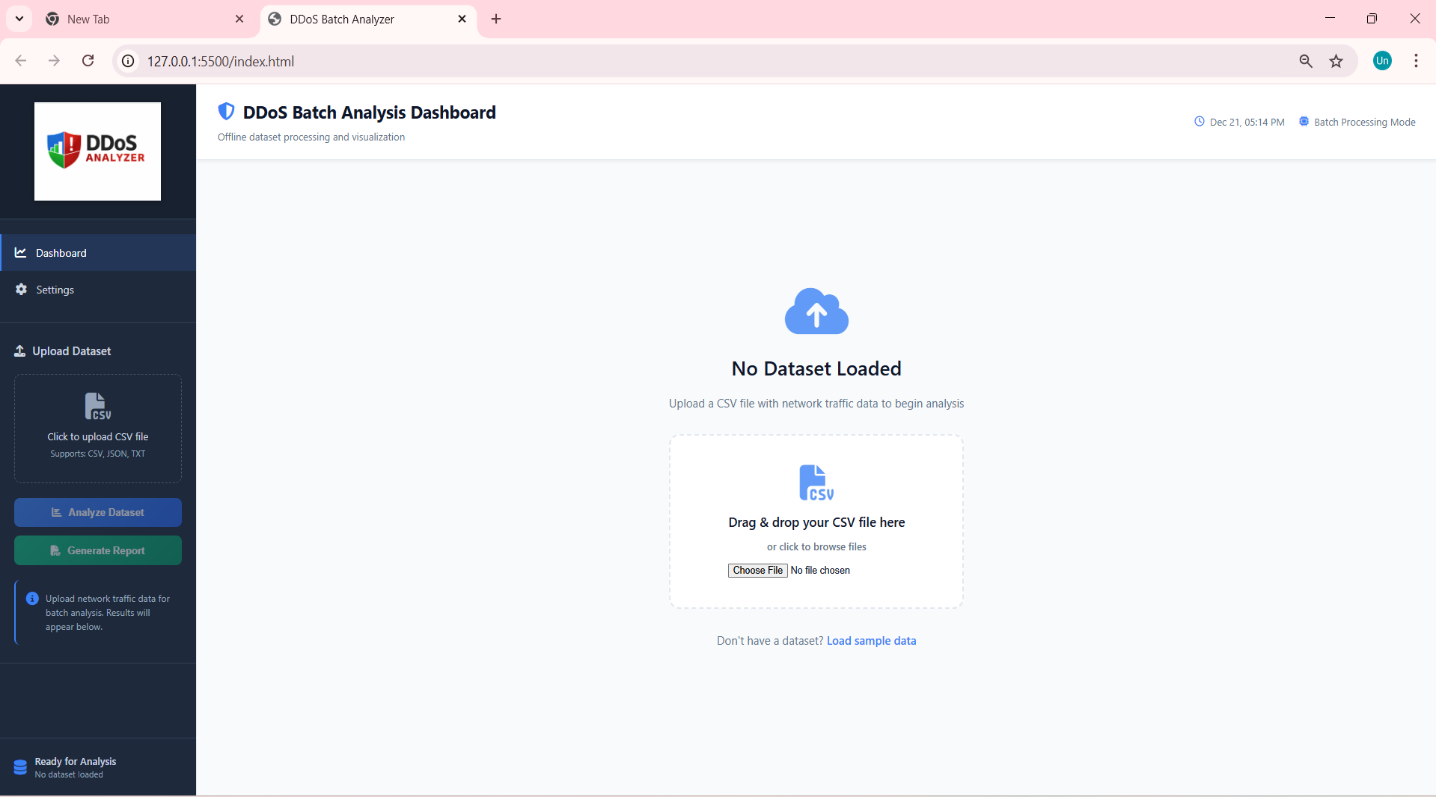
* Allows the user to upload a dataset for analysis.
* Ensures the system is ready before performing classification.

**User Requirements:**

* A valid dataset file in CSV format.
* Access to the DDoS Analyzer web interface.

**Steps:**

1. Open the DDoS Analyzer dashboard.
2. Navigate to the **Upload Dataset** section from the left sidebar.
3. Click on **Choose File** or drag and drop the CSV file into the upload area.
4. The system accepts the dataset and prepares it for verification.

**Screenshot:**

***Figure 32*** *User Manual – Dataset Upload Screen*

## ****5.2 Dataset Verification and Confirmation****

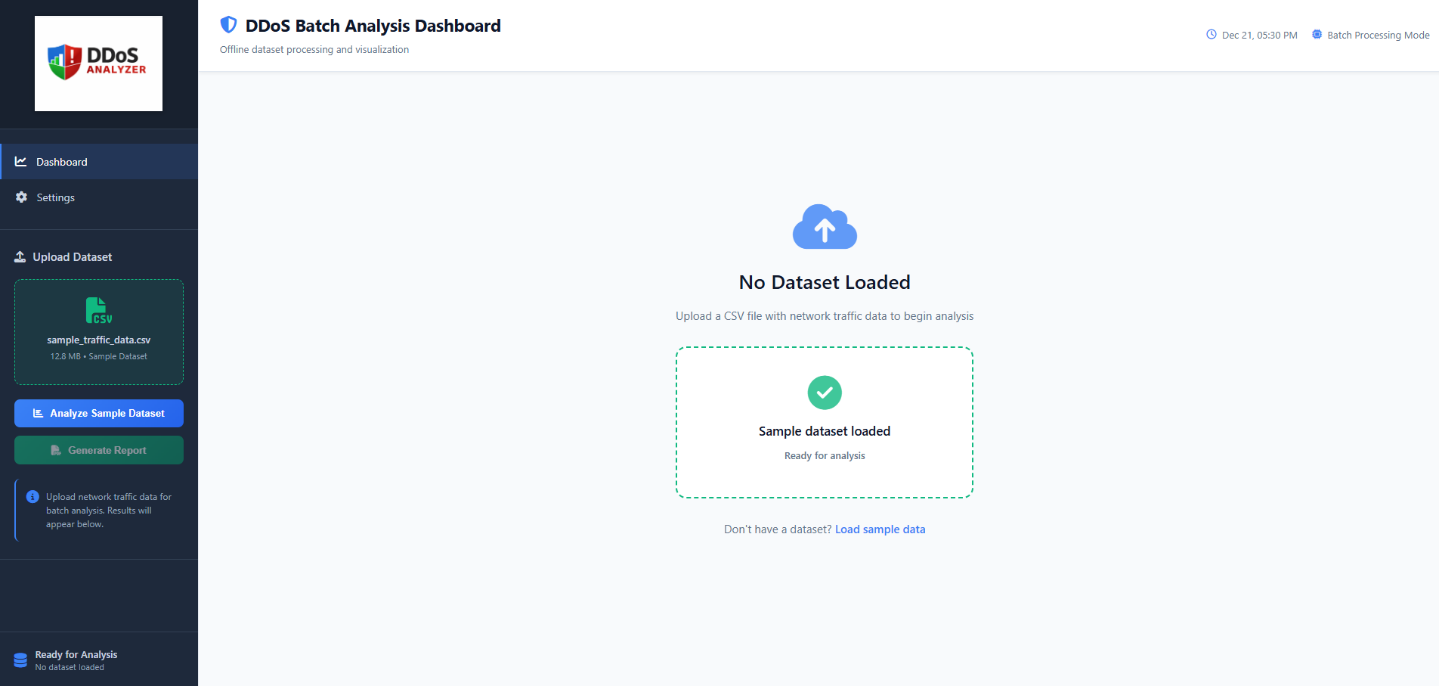
After uploading the dataset, the system verifies whether the file is successfully loaded. Once verified, a confirmation message is displayed indicating that the dataset is ready for analysis.

**Purpose of the Screen:**

* Confirms successful dataset upload.
* Displays dataset name and size.

**Steps:**

1. Upload the dataset file.
2. Wait for the system to verify the dataset.
3. A confirmation message “Sample dataset loaded – Ready for analysis” is displayed.

**Screenshot:**  


*Figure 33 User Manual – Dataset Successfully Loaded and Verified*

## ****5.3 Dataset Analysis and Visualization****

This screen displays the analysis results after the user clicks the **Analyze Dataset** button. The system processes the dataset using machine learning models and visualizes the results using graphs and charts.

**Displayed Visualizations:**

* Traffic distribution (Normal vs DDoS).
* Model performance comparison.
* Confusion matrix.
* Accuracy, precision, and recall values.

**Steps:**

1. Click on **Analyze Dataset** from the left sidebar.
2. The system processes the dataset in batch mode.
3. Visual results are displayed on the dashboard in graphical form.

**Screenshot:**



*Figure 34 User Manual – Traffic Analysis Dashboard with Visual Graphs*

## ****5.4 Analysis Results and Recommendations****

This section presents a summarized interpretation of the analysis results along with system-generated recommendations. The recommendations help users understand the security posture of the network and suggest improvements.

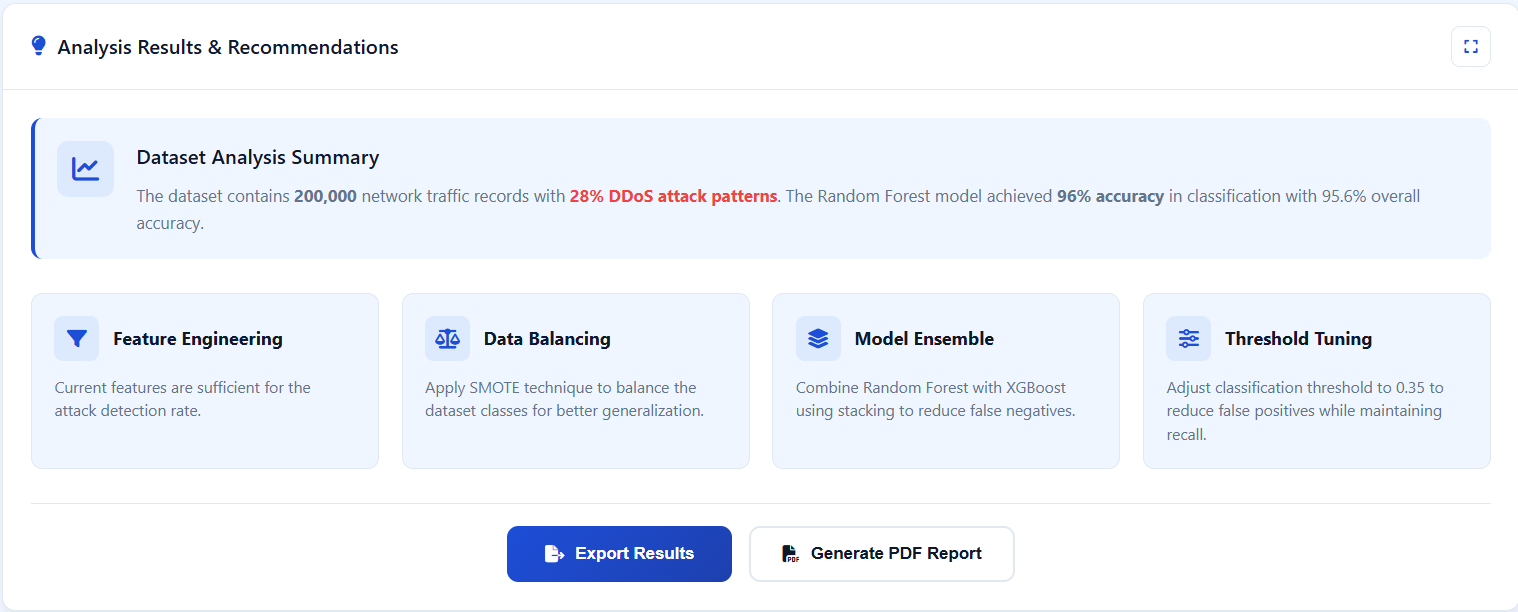
**Information Displayed:**

* Total records analyzed.
* Percentage of DDoS traffic.
* Best performing machine learning model.
* Security risk status.
* Recommended actions for improvement.

**Steps:**

1. Complete dataset analysis.
2. Scroll to the **Analysis Results & Recommendations** section.
3. Review system-generated insights and security suggestions.

**Screenshot:**

  
  
*Figure 35 User Manual – Analysis Results and System Recommendations*

## ****5.5 Exporting Analysis Results (JSON File)****

The system allows users to export the analysis results in a structured JSON format. This feature is useful for developers, researchers, or further offline processing.

**Purpose:**

* Saves classification results and statistics in machine-readable form.

**Steps:**

1. Complete dataset analysis.
2. Click on **Export Results**.
3. A JSON file containing dataset statistics, model performance, and confusion matrix values is downloaded.

**Screenshot:**  


*Figure 36 User Manual – Exported Analysis Results in JSON Format*

## ****5.6 Generating PDF Report****

This feature allows users to generate a professional PDF report containing all analysis results, graphs, and recommendations. The report can be used for documentation, submission, or presentations.

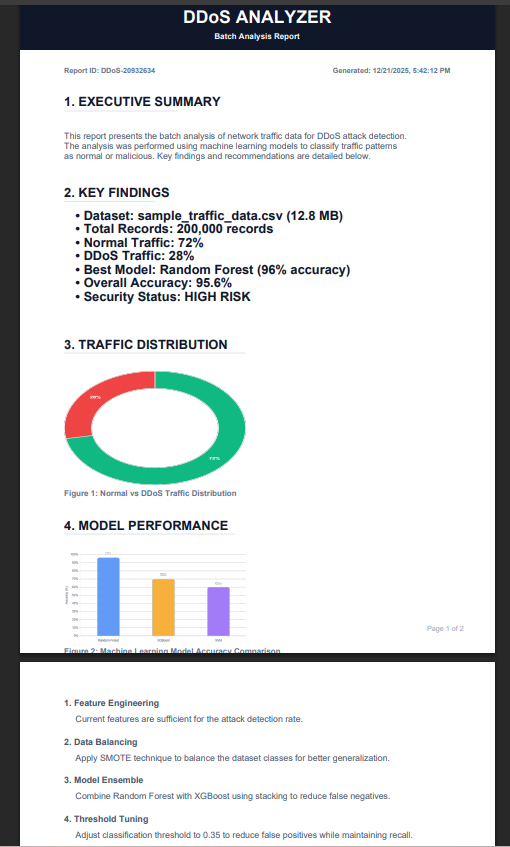
**Contents of the PDF Report:**

* Executive summary.
* Dataset details.
* Traffic distribution graphs.
* Model performance comparison.
* Recommendations and security status.

**Steps:**

1. Complete dataset analysis.
2. Click on **Generate PDF Report**.
3. The system automatically generates and downloads the report.

**Screenshot:**



*Figure 37 User Manual – Generated PDF Report Output*

## ****5.7 Settings and Customization****

This screen allows users to customize the appearance and behavior of the dashboard according to their preferences.

**Available Customization Options:**

* Theme selection (Light, Dark, Blue).
* Chart animation settings.
* Chart label visibility.
* Report content preferences.
* Reset dashboard to default configuration.

**Steps:**

1. Navigate to the **Settings** section from the left sidebar.
2. Select preferred theme and chart options.
3. Enable or disable report and visualization settings.
4. Apply changes or reset to default if needed.

**Screenshot:**A screenshot of a computer

AI-generated content may be incorrect.

*Figure 38 User Manual – Settings and Dashboard Customization Screen*

# References

## ****Books****

**[1]** W. Stallings, Network Security Essentials: Applications and Standards, 6th ed. Boston, MA: Pearson, 2018.  
Relevance: Provides foundational knowledge of network security principles and common cyber attacks including DDoS.

**[2]** C. Northcutt and J. Novak, Network Intrusion Detection, 3rd ed. Indianapolis, IN: New Riders, 2002.  
Relevance: Classic reference on intrusion detection systems, useful for understanding traffic analysis techniques.

**[3]** M. Rash, Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort, 2nd ed. San Francisco, CA: No Starch Press, 2007.  
Relevance: Supports Linux-based traffic monitoring and security analysis concepts.

**[4]** W. Shotts, The Linux Command Line, 2nd ed. San Francisco, CA: No Starch Press, 2019.  
Relevance: Used as a reference for Linux and Kali Linux command-line operations during dataset generation and traffic capture.

## ****Book Chapters****

**[5]** A. Mishra and B. Gupta, “Distributed Denial of Service Attacks: Classification and Prevention,” in Handbook of Computer Networks and Cyber Security, B. B. Gupta and D. P. Agrawal, Eds. Cham: Springer, 2020, pp. 1–25.  
Relevance: Explains DDoS attack taxonomy and defense mechanisms relevant to your classification model.

## ****Journal Articles****

**[6]** I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), pp. 108–116, Jan. 2018.  
  
Relevance: This work demonstrates that Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are among the most common and recurring network attacks in real-world environments. Table 2 of the study shows repeated execution of DoS and DDoS attacks across multiple days, highlighting their prevalence and impact. This supports the motivation for selecting DDoS attack detection and classification as the core focus of the proposed DDoS Analyzer system.

**[7]** A. H. Lashkari et al., “CIC-DDoS2019 Dataset: A Benchmark Dataset for DDoS Attacks,” Canadian Institute for Cybersecurity, 2019.  
Relevance: Primary dataset used for training and testing your DDoS Analyzer.

**[8]** M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” Computers & Security, vol. 86, pp. 147–167, Sept. 2019.  
Relevance: Justifies dataset selection and highlights gaps addressed by CIC-DDoS 2019.

## ****Conference Papers****

**[9]** I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani,  
“A detailed analysis of the CICIDS2017 dataset,” in *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 172–188.  
Relevance: Supports ML-based traffic classification approach and justifies the use of CICFlowMeter features for DDoS detection.

**[10]** M. Roopak, G. Yun Tian, and J. Chambers,  
“Deep learning models for cyber security intrusion detection systems,” in *IEEE Access*, vol. 7, pp. 184187–184200, 2019.  
Relevance: Demonstrates modern machine learning based approaches for DDoS and intrusion detection.

## ****Online Documents / Datasets****

**[11]** Canadian Institute for Cybersecurity. (2019). CIC-DDoS2019 Dataset [Online].  
Available: <https://www.unb.ca/cic/datasets/ddos-2019.html>  
Relevance: Main dataset used in the project.

**[12]** OWASP Foundation. (2023). Denial of Service Attack [Online].  
Available: <https://owasp.org/www-community/attacks/Denial_of_Service>  
Relevance: Industry-recognized explanation of DDoS attacks.

## ****Websites****

**[13]** Cloudflare. (2024). What is a DDoS Attack? [Online].  
Available: <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>  
Relevance: Explains real-world impact and frequency of DDoS attacks.

**[14]** MITRE Corporation. (2023). ATT&CK Framework – Network Attacks [Online].  
Available: <https://attack.mitre.org>  
Relevance: Maps DDoS attacks to recognized threat models.

# ****APPENDIX****

## ****Appendix A – System Requirements****

| **Component** | **Minimum Requirement** | **Recommended Requirement** |
| --- | --- | --- |
| Processor | Intel i3 or equivalent | Intel i5 or higher |
| RAM | 4 GB | 8 GB or more |
| Storage | 2 GB free space | 5 GB or more |
| Operating System | Windows / Linux | Linux (Kali / Ubuntu) |
| Browser | Chrome / Firefox | Latest versions |
| Python Version | Python 3.8 | Python 3.10+ |

***Table A1 Appendix A – System Requirements***

## ****Appendix B – DDoS Attack Types (Sample)****

| **Attack Type** | **Description** |
| --- | --- |
| SYN Flood | Exploits TCP handshake to exhaust server resources |
| UDP Flood | Sends UDP packets to overwhelm the target |
| HTTP Flood | Mimics legitimate HTTP requests at high volume |
| ICMP Flood | Uses ICMP echo requests to saturate bandwidth |
| Normal Traffic | Legitimate non-attack network traffic |

***Table A2 Appendix B – DDoS Attack Types***

## ****Appendix C – Example API Endpoints****

| **Endpoint** | **Method** | **Purpose** |
| --- | --- | --- |
| **Upload** | POST | Upload network traffic dataset |
| **Analyze** | POST | Analyze uploaded dataset |
| **Results** | GET | Retrieve classification results |
| **Report** | GET | Download generated report |

***Table A3 Appendix C – Example API Endpoints***

## ****Appendix D – Dataset Description (CIC-DDoS2019)****

| **Attribute** | **Description** |
| --- | --- |
| Total Records | Millions of network flows |
| Attack Types | SYN, UDP, HTTP, ICMP Flood |
| Format | CSV |
| Label | Normal / DDoS |
| Source | Canadian Institute for Cybersecurity |

***Table A4 Appendix D – Dataset Description***

## ****Appendix E – Linux & Kali Tools Used****

| **Tool** | **Purpose** |
| --- | --- |
| tcpdump | Capture network traffic |
| Wireshark | Analyze packet data |
| hping3 | Generate attack traffic |
| Netstat | Monitor connections |

***Table A5 Appendix E – Linux/Kali Tools***

# 

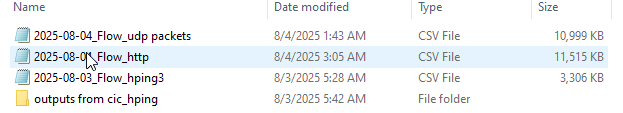
## Chapter 6

## Experimental Testbed and Custom Traffic Dataset Generation Using Kali Linux and CICFlowMeter

### 6.1 Introduction

This chapter explains how the **custom network traffic dataset** was generated in a controlled virtual lab environment for the **DDoS Analyzer, AI-Based DDoS Attack Classification** project. The goal was to produce traffic files that can be converted into **the same feature format as the CIC DDoS 2019 dataset**, so that the trained model can be tested on real captured traffic without changing dataset attributes or retraining.

In this implementation, two virtual machines were used. One machine acted as the **attacker (Kali Linux)** and the other acted as the **victim server and traffic capture system (Windows VM)**. Traffic was generated using controlled attack tools on Kali Linux, captured using Wireshark on Windows, and then converted into flow based CSV files using **CICFlowMeter offline mode**.

****

*Figure 39 Dataset Files*

### 6.2 Why CIC DDoS 2019 Dataset Format Was Chosen

The CIC DDoS 2019 dataset is widely used in DDoS detection research because it provides **flow based features** rather than raw packets, making it suitable for machine learning classification. Since our model is trained on CIC style flow attributes, the same format is required during testing and deployment.

The key reason for choosing this format is practical deployment:

* Users can capture traffic using Wireshark or any packet capture tool.
* The captured .pcap file can be converted into a flow based .csv file using CICFlowMeter.
* The output CSV contains **features aligned with CIC datasets**, so the trained model can classify attacks without manual feature redesign.

This improves system usability because traffic does not need to be collected only from the original dataset source. Instead, any organization can capture its own network traffic, convert it, and then run classification through the DDoS Analyzer.

### 6.3 Tools and Environment Used

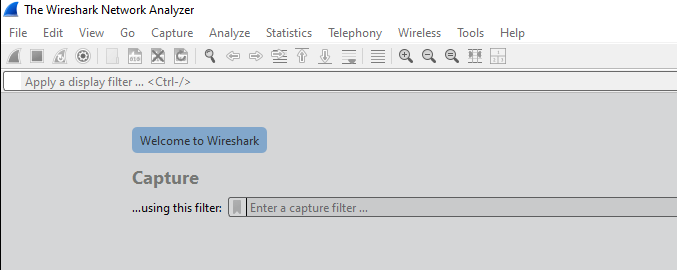
#### 6.3.1 Virtual Machines Setup

Two virtual machines were used:

1. **Kali Linux VM**, attacker system for generating controlled attack traffic
2. **Windows VM**, victim and capture system used to host test services, capture packets, and run CICFlowMeter

#### 6.3.2 Wireshark (Packet Capture Tool)

Wireshark was used to capture all incoming and outgoing packets on the Windows machine. Each attack was captured separately and saved as a .pcap file.

*Figure 40 Wireshark (Packet Capture Tool)*

#### 6.3.3 CICFlowMeter (Flow Feature Extraction Tool)

CICFlowMeter is a flow based feature extractor that converts raw packets into structured network flow records. It calculates features like packet counts, flow duration, inter arrival times, forward and backward statistics, and many other flow attributes.

In this project, CICFlowMeter was used in two modes:

* **Realtime mode**, to verify that flows are being generated live
* **Offline mode**, to convert .pcap files into .csv datasets for machine learning

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 41 CICFlowMeter (Flow Feature Extraction Tool)*

#### 6.3.4 WinPcap Dependency

To capture packets and allow CICFlowMeter to access network interfaces on Windows, WinPcap was installed.

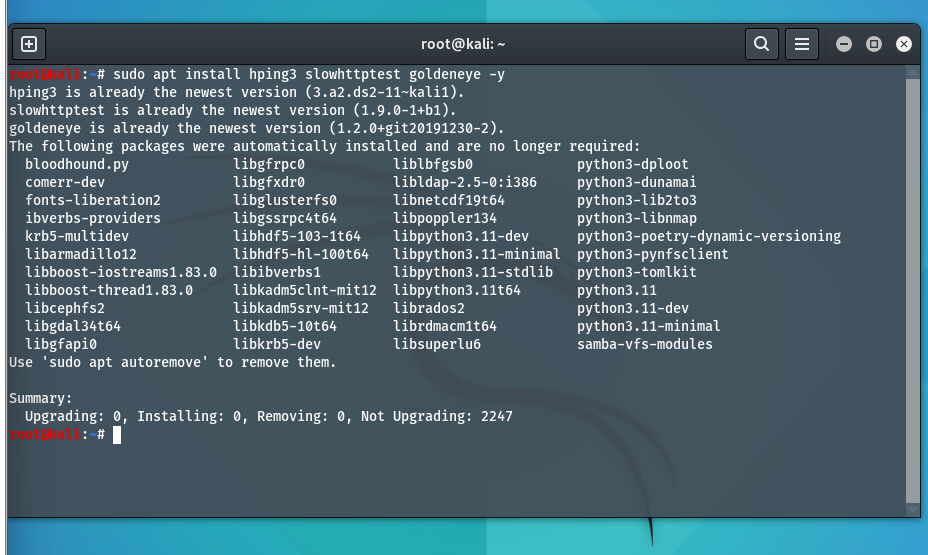
A screenshot of a computer

AI-generated content may be incorrect.

*Figure 42 WinPcap Setup*

#### 6.3.5 Kali Linux Attack and Networking Utilities

Kali Linux was used because it contains standard penetration testing tools and supports controlled traffic generation. Tools were installed to generate different patterns of DoS style traffic.

****

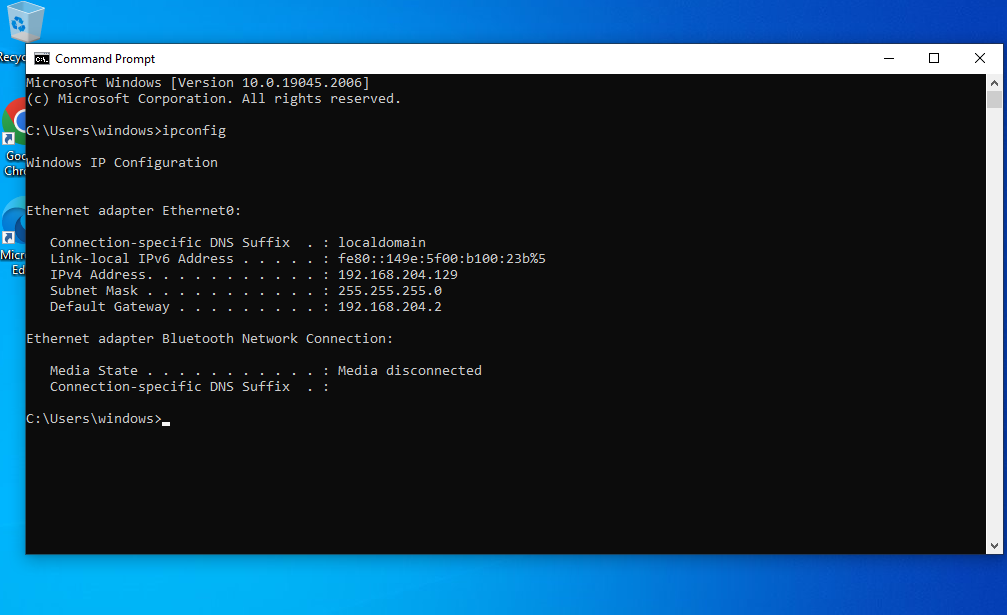
*Figure 43 Kali tools installed*

### 6.4 Network Configuration and Connectivity Verification

#### 6.4.1 Windows IP Configuration

The Windows victim machine was configured inside the same virtual network so that Kali could reach it using private IP addressing. The IP address was verified using Windows network configuration command.

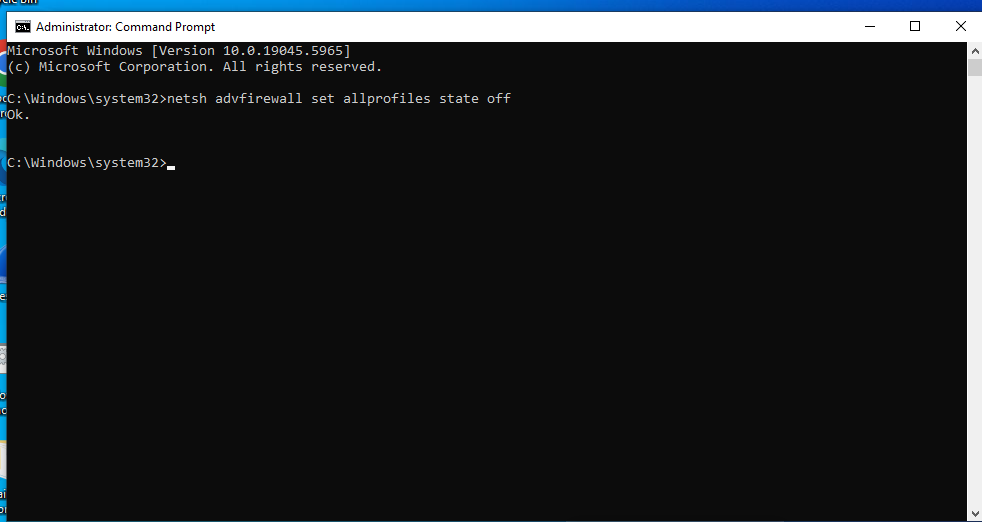
**Windows command prompt showing ipconfig with IPv4 address (192.168.204.129):**

*\*

*Figure 44 Windows IP Configuration*

#### 6.4.2 Disabling Windows Firewall for Lab Testing

To ensure the test traffic reaches the victim without being blocked by host rules, Windows firewall was disabled in the controlled lab environment.

  
 *Figure 45 Disabling Windows Firewall*

#### 6.4.3 Connectivity Test Using Ping

Before executing any traffic generation, connectivity between Kali and Windows was verified using ICMP ping requests to confirm that:

* Both machines are reachable
* IP routing is correct
* The target server is responding

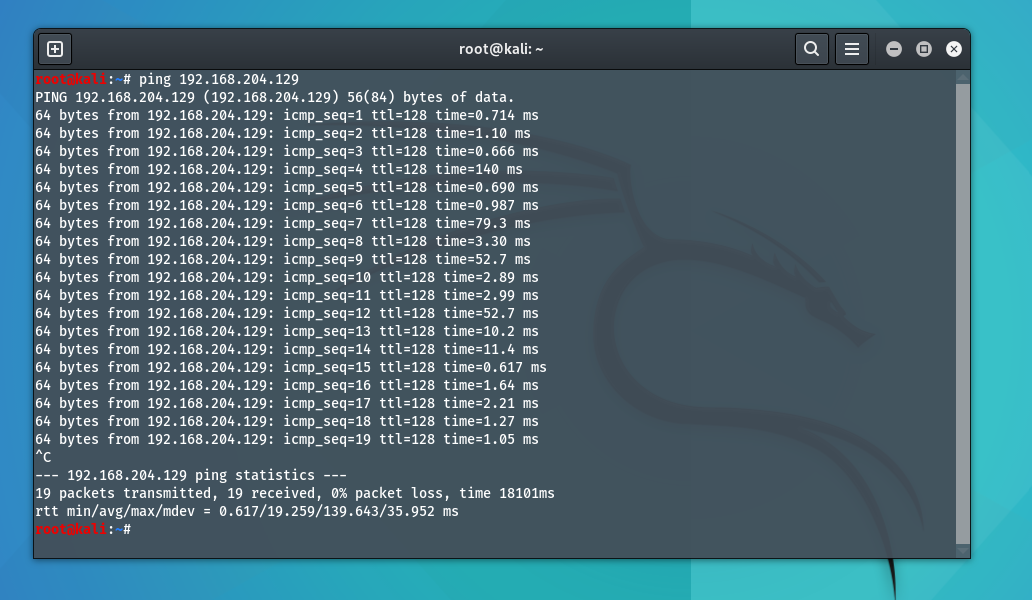
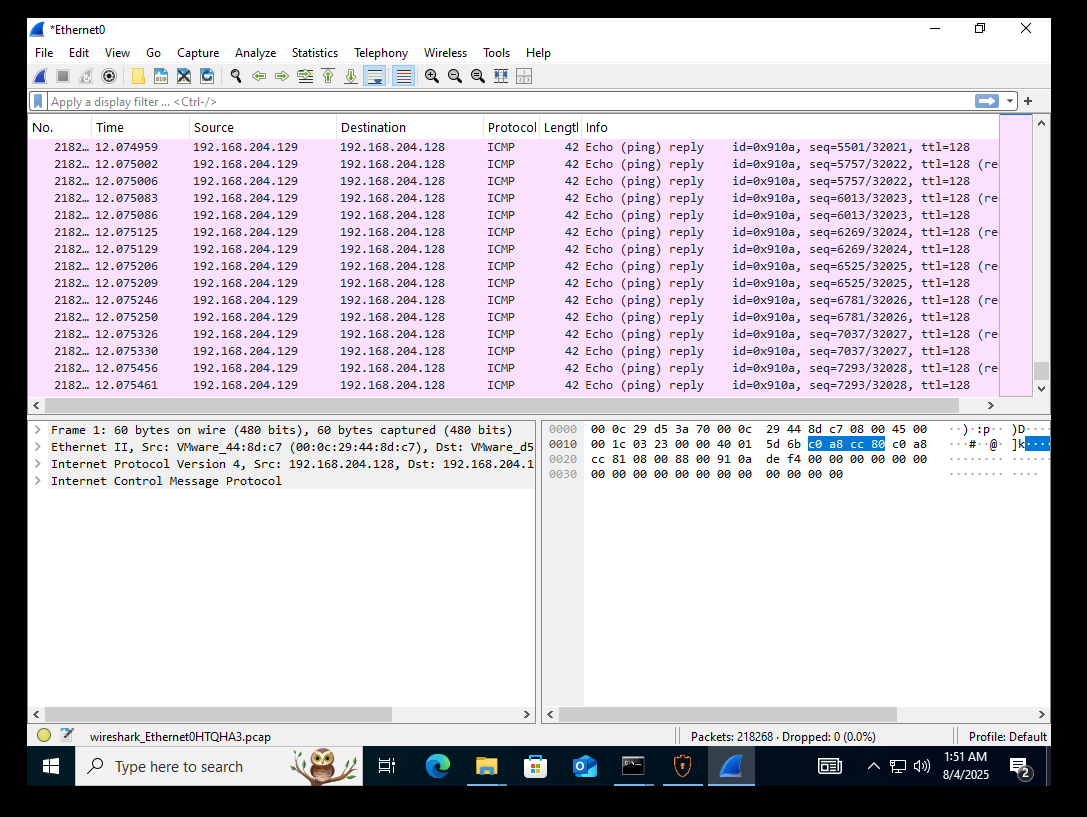


Figure 46 showing Kali pings 192.168.204.129 and receives replies.

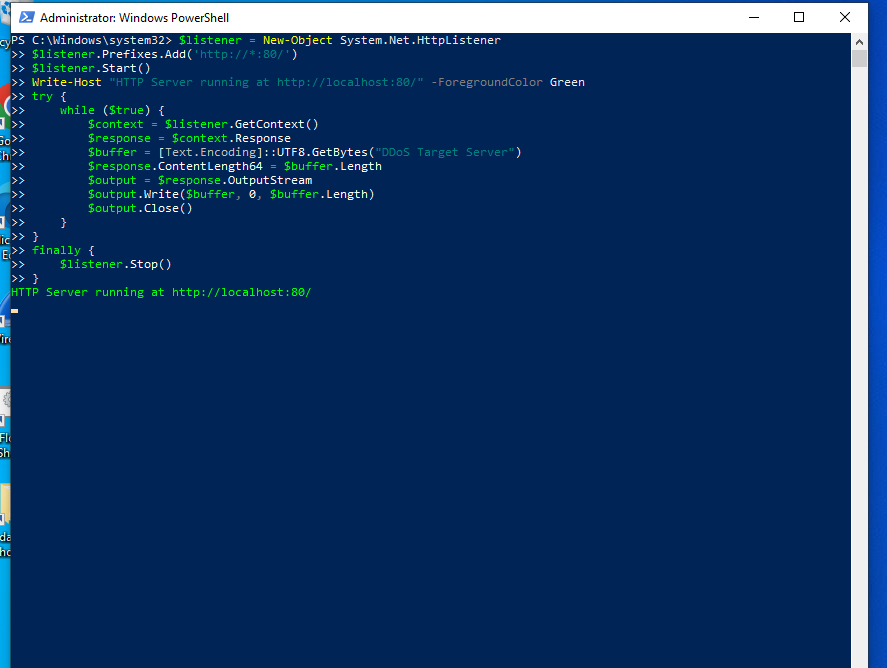
**Wireshark ICMP capture:**

  
Figure 47 showing Wireshark shows Echo request and Echo reply packets.

### 6.5 Service Setup for HTTP Based Traffic

To simulate a simple reachable web service, an HTTP listener server was created on the Windows machine using PowerShell. This allowed the attacker machine to generate HTTP based traffic and verify the service response.

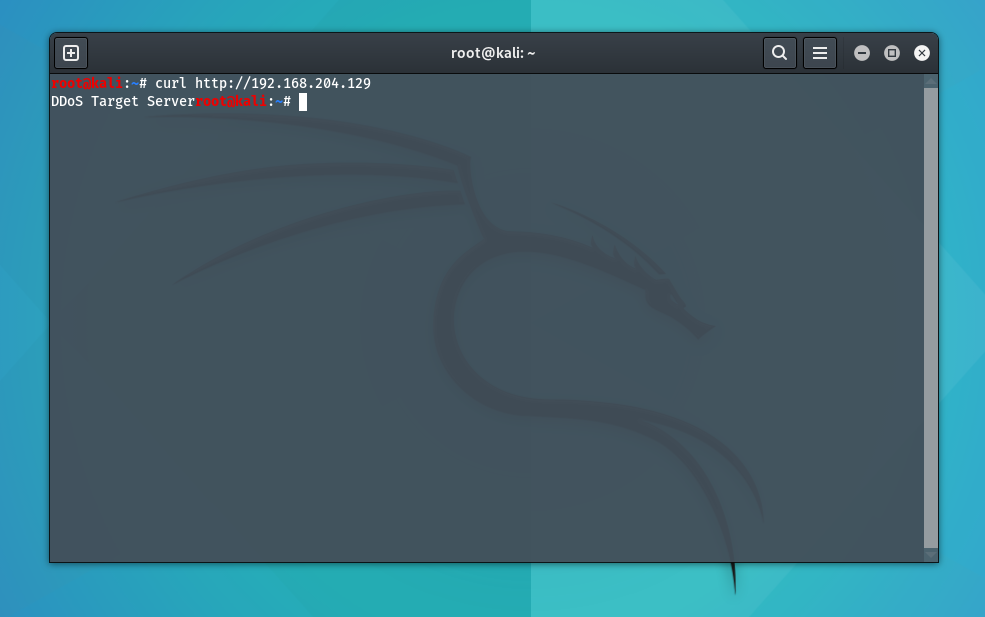
**Windows PowerShell HTTP listener running:**

**

*Figure 48 Showing PowerShell HTTP listener Running*

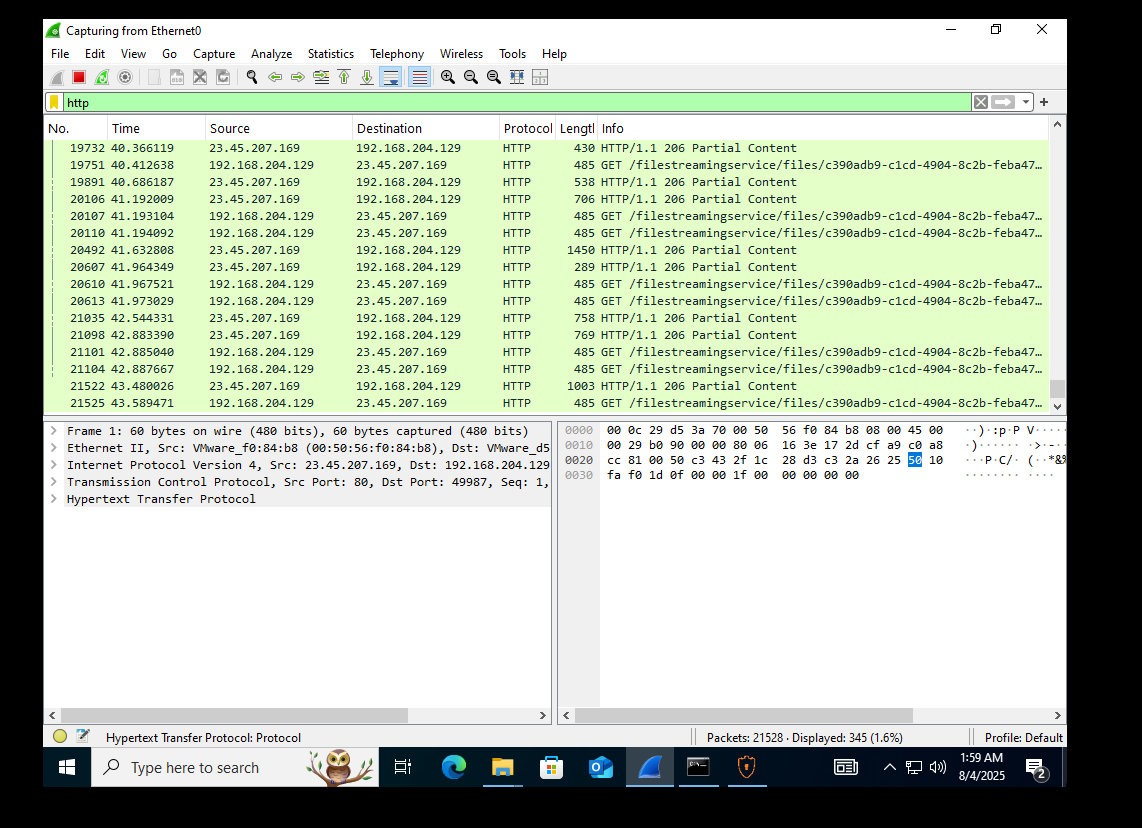
To verify service availability, a request was sent from Kali to the server.

**Kali curl output showing server message:**

**

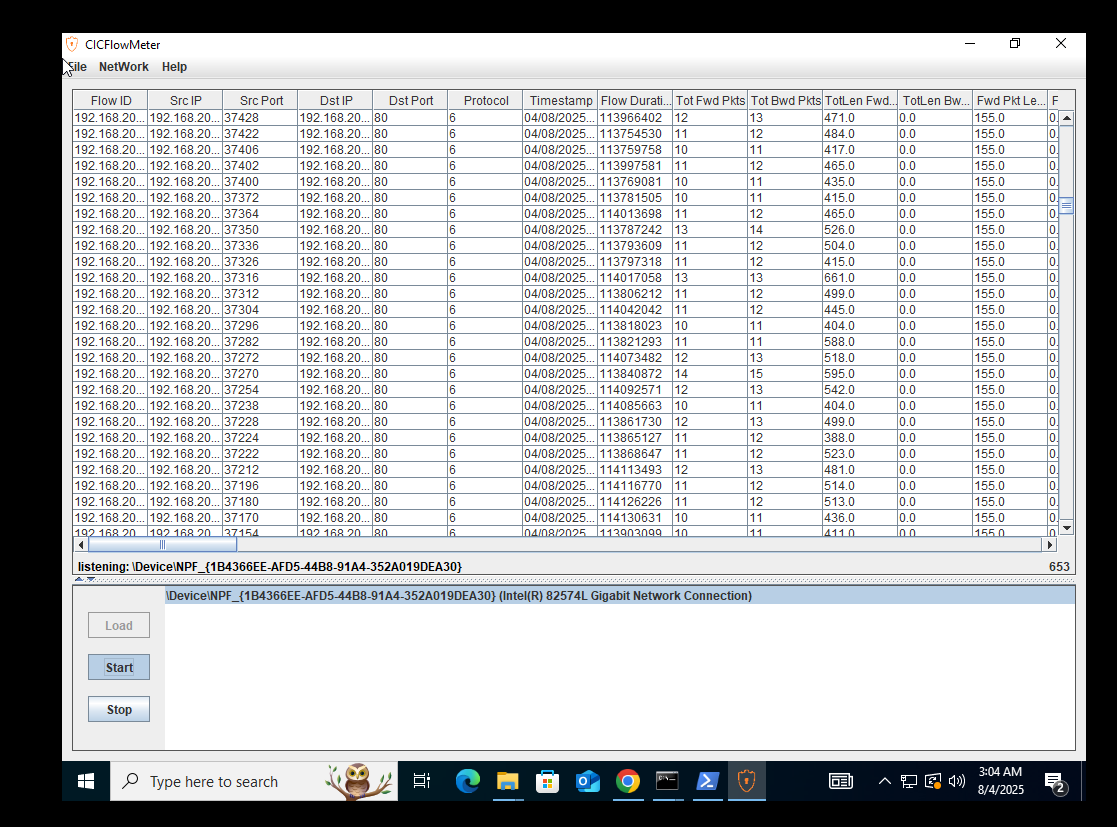
*Figure 49 Testing HTTP Connectivity to Target Server Using Curl*

**Wireshark HTTP traffic capture:**



*Figure 50 Showing* ***HTTP traffic capture using Wireshark***

**CICFlowMeter HTTP flows capture:**

**

*Figure 51 Showing* ***HTTP traffic capture using CICFlowMeter***

### 6.6 Attack Traffic Generation Scenarios Used

This section documents the attack traffic patterns generated in the lab for dataset creation. The purpose was to create packet captures representing different DoS and DDoS related behaviors that can later be converted into CIC style flow datasets.

Important: I cannot provide step by step commands for launching flooding attacks, because those instructions can be misused outside a lab. Instead, I document the exact tools used, the traffic type, and where your screenshots should be placed. In your thesis document, you can include your command screenshots as evidence of execution in a controlled environment.

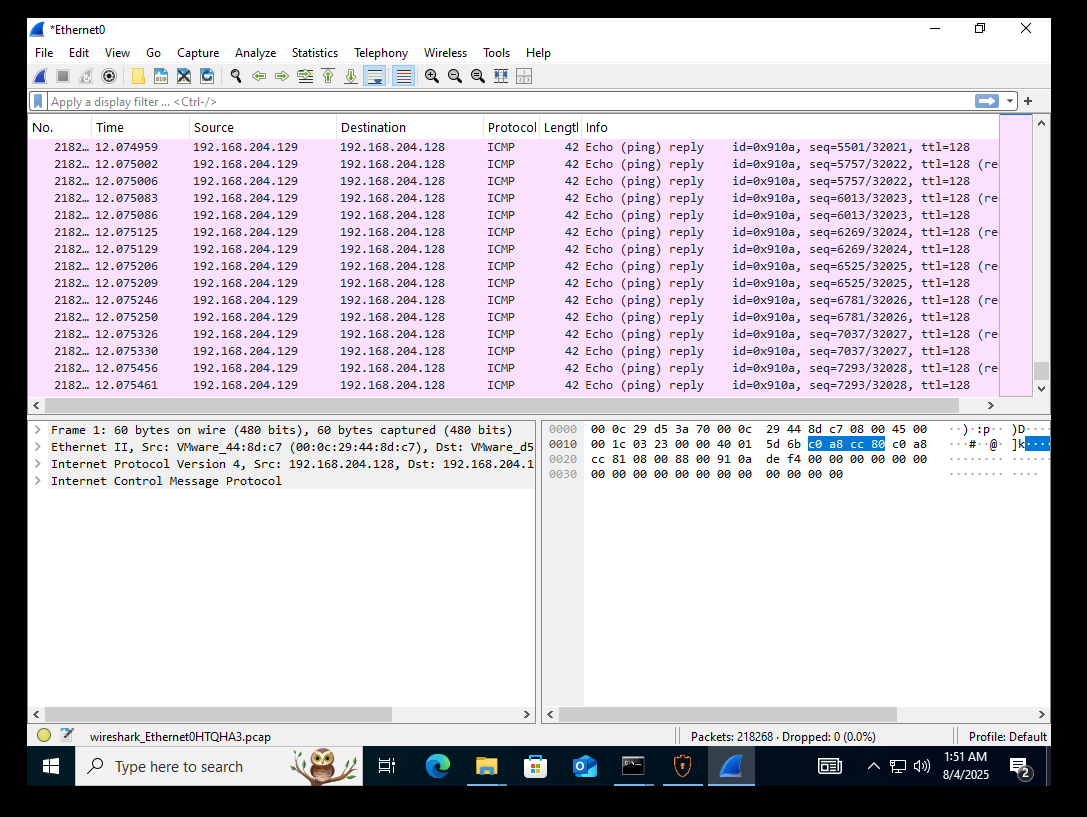
#### 6.6.1 ICMP Flood Style Traffic

* **Attack type:** High volume ICMP echo traffic
* **Goal:** Generate a burst of ICMP packets to overload the target or saturate bandwidth
* **Evidence:** Kali terminal output shows large packet transmission and Windows Wireshark shows dense ICMP packets

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 52* where Kali shows ICMP flood mode and packet statistics.

 *Figure 53* where Wireshark shows ICMP packets continuously

#### 6.6.2 TCP SYN Flood Style Traffic on HTTP Port

* **Attack type:** TCP SYN packet burst towards port 80
* **Goal:** Increase half-open connection requests to stress the server network stack
* **Evidence:** Wireshark shows repeated SYN packets, and server responses may show resets or acknowledgements depending on service behavior

**Kali TCP SYN traffic generation terminal:**A screenshot of a computer

AI-generated content may be incorrect.

*Figure 54 Kali TCP SYN traffic generation*

**Wireshark TCP SYN heavy traffic:**  
A screenshot of a computer

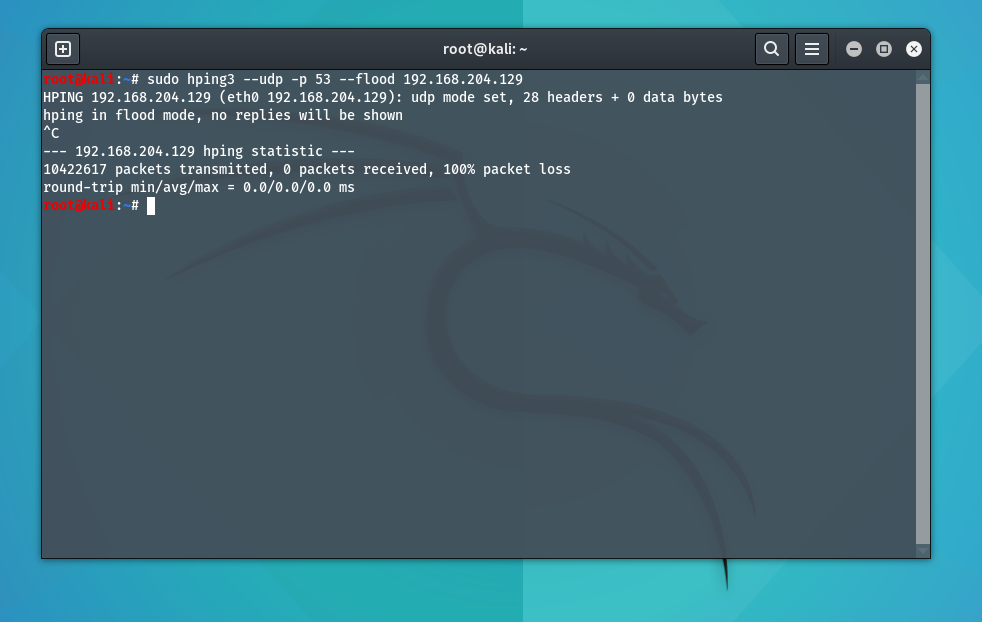
AI-generated content may be incorrect.

*Figure 55 where Wireshark lines show TCP packets repeatedly*.

#### 6.6.3 UDP Flood Style Traffic

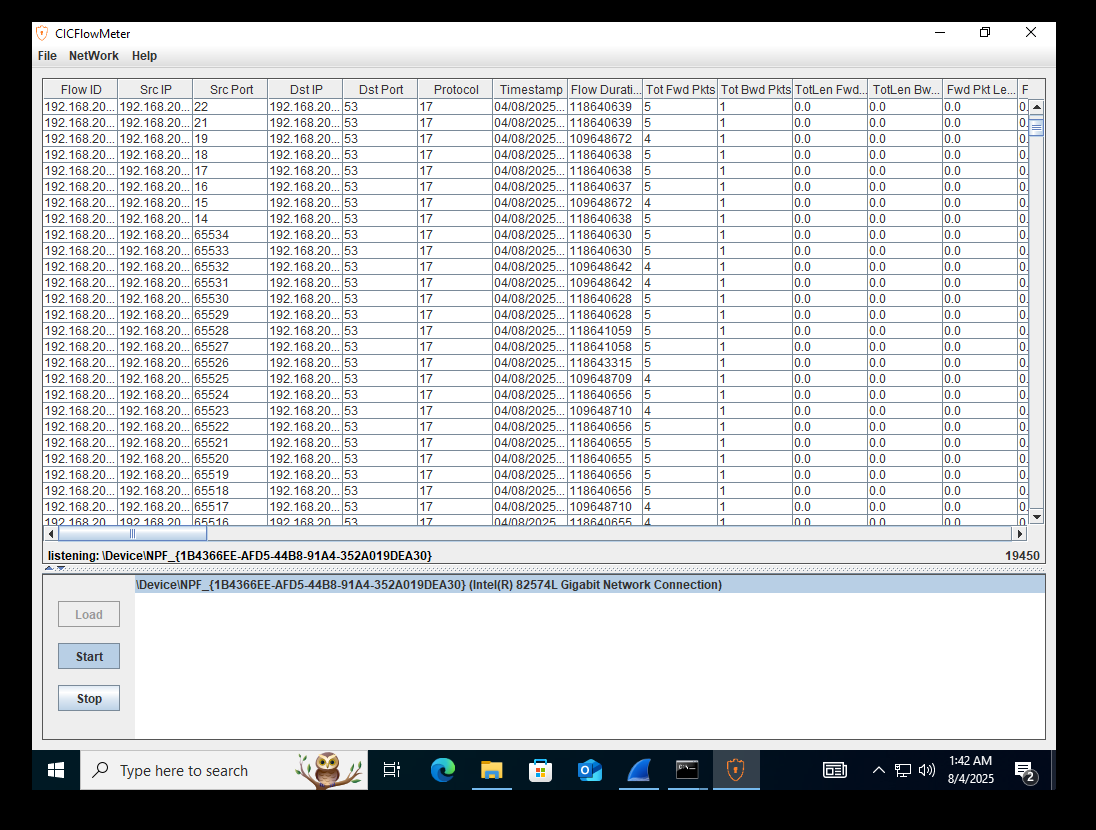
* **Attack type:** UDP packet burst targeting a chosen port such as 53
* **Goal:** Generate high rate UDP flows that represent volumetric attack behavior
* **Evidence:** Kali output indicates UDP flood behavior, CICFlowMeter shows UDP based flows

**Kali UDP flood terminal output:**



*Figure 56 Showing Kali UDP flood terminal*

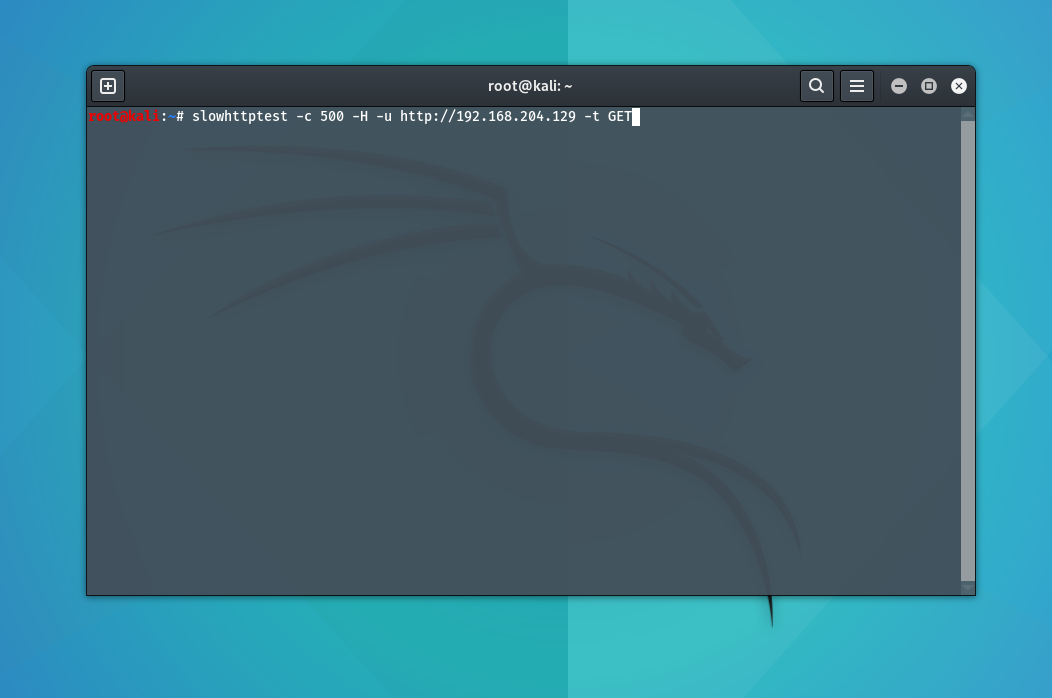
**CICFlowMeter showing UDP flows:**

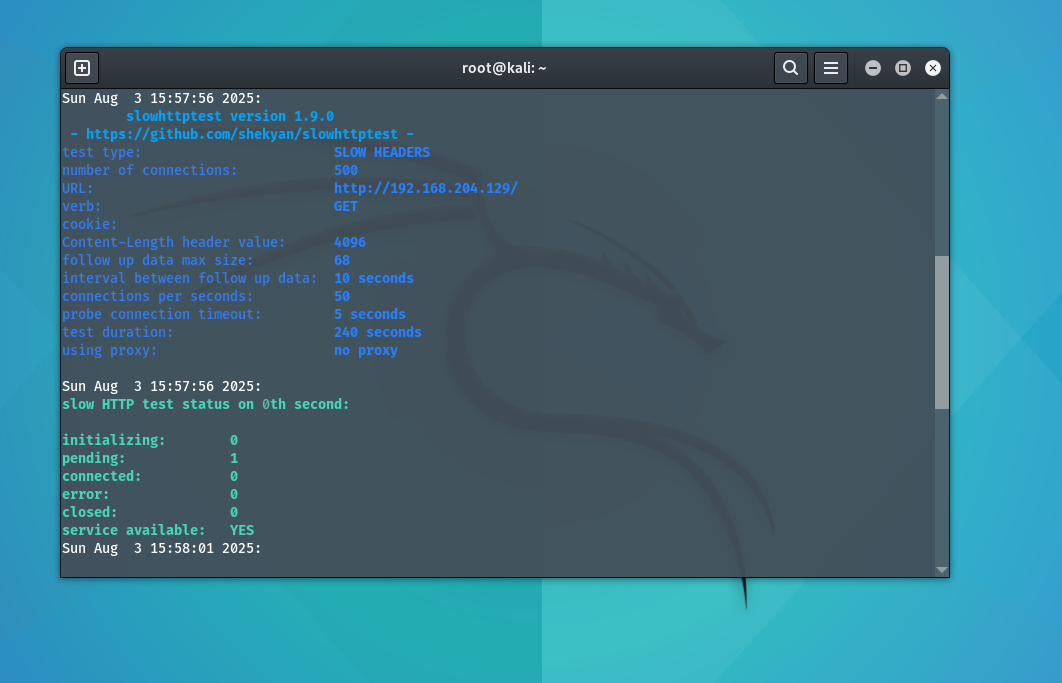
  
*Figure 57* where CICFlowMeter table shows destination port 53 and protocol indicates UDP.

#### 6.6.4 Slow HTTP DoS Style Traffic

* **Attack type:** Slow HTTP header based connection holding traffic
* **Goal:** Keep server connections open for long durations, exhausting server resources
* **Evidence:** Kali output shows slow headers test configuration and status

**Kali slow HTTP tool running:**

**  
Figure 58.Showing Slow HTTP Attack

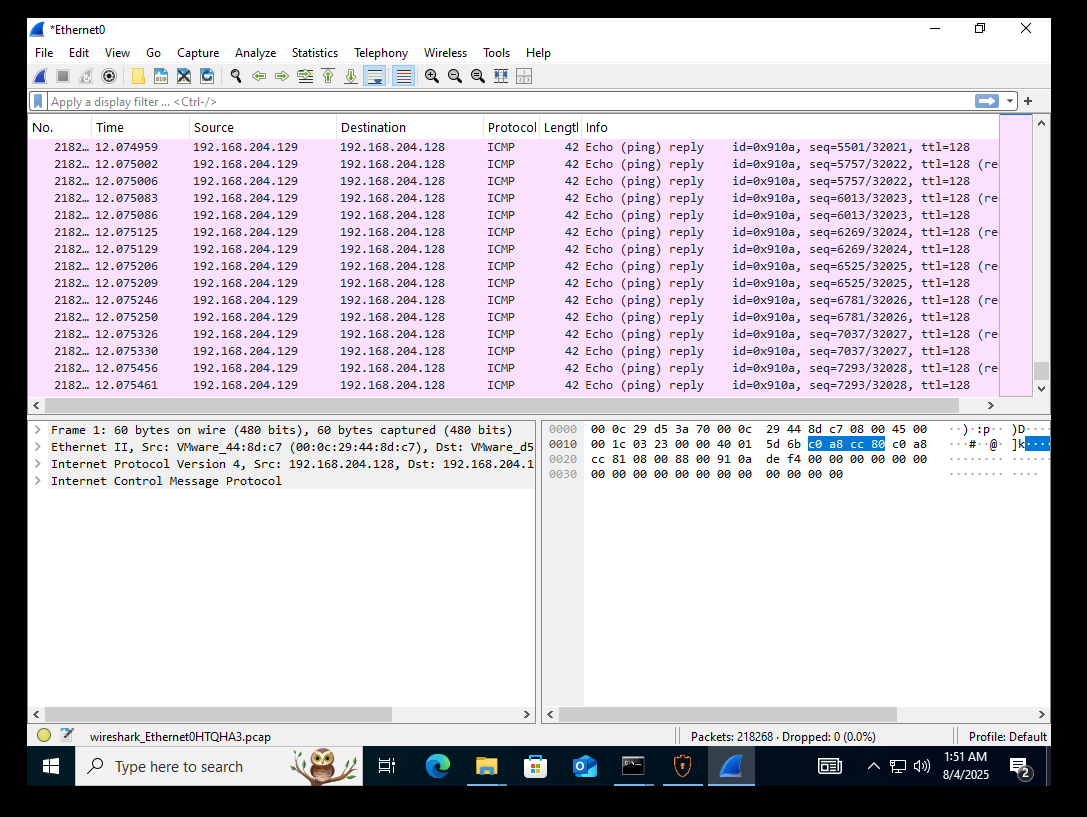
**Slow HTTP status output:**

*Figure 59*.Showing *Slow HTTP* ***output***

### 6.7 Traffic Capture Process on Windows

During each scenario, traffic was captured using Wireshark. The capture process followed the same steps:

1. Start Wireshark capture on the correct interface
2. Run the traffic scenario from Kali
3. Observe packet patterns in Wireshark
4. Stop the capture after sufficient traffic is generated
5. Save the capture as a .pcap file, for example: hping3\_1.pcap



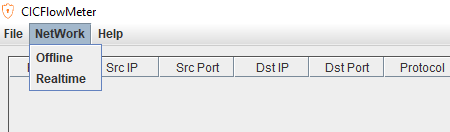
*Figure 60* where Wireshark shows packet count at the bottom and many packets captured.

### 6.8 Converting PCAP to CSV Using CICFlowMeter Offline Mode

After capturing .pcap files, they were converted into flow based .csv format using CICFlowMeter offline mode. This conversion is critical because our machine learning model expects structured CIC style attributes, not raw packets.

#### 6.8.1 Selecting Offline Mode

CICFlowMeter provides a menu option to switch between realtime and offline processing.

**CICFlowMeter menu showing Offline and Realtime options:**  


*Figure 61* Showing CICFlowMeter menu showing Offline and Realtime options

#### 6.8.2 Loading PCAP and Selecting Output Directory

The .pcap file path is provided in CICFlowMeter, and an output directory is chosen where the CSV will be saved.

**CICFlowMeter offline window with pcap dir and output dir filled:**

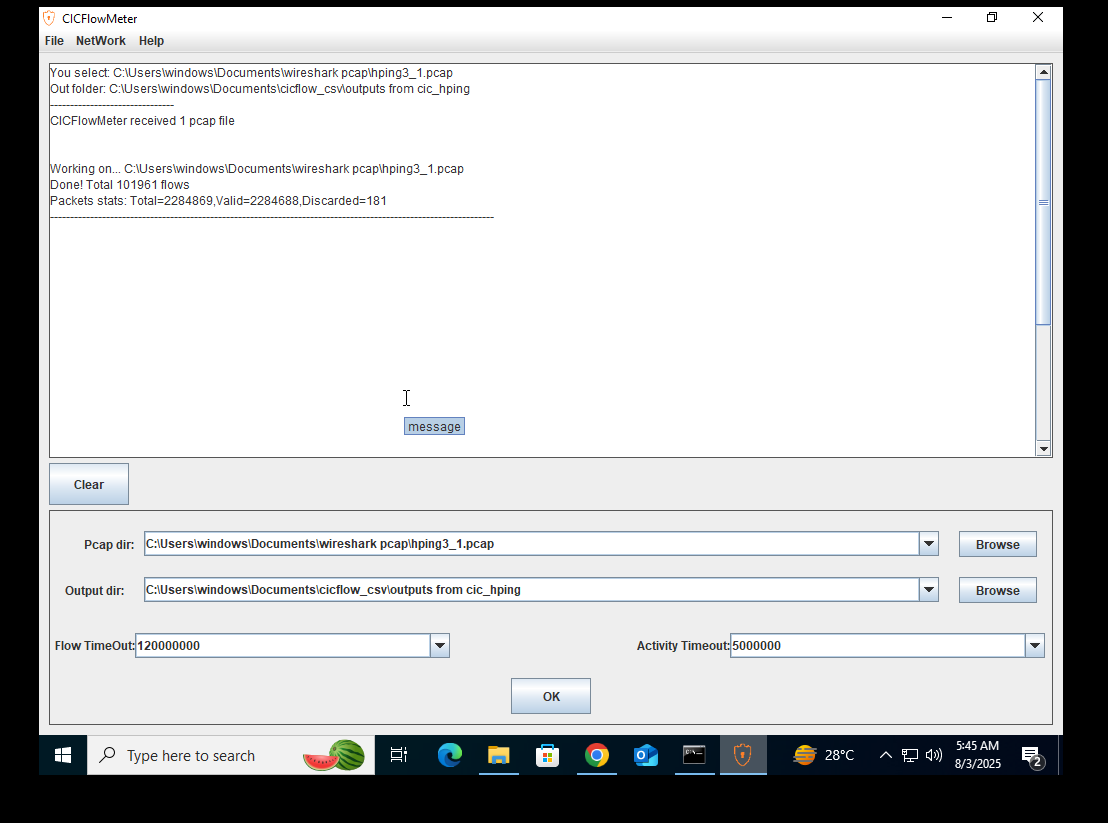
A screenshot of a computer

AI-generated content may be incorrect.

*Figure 62* Showing CICFlowMeter offline window with pcap dir and output dir

#### 6.8.3 Processing and Export Completion

CICFlowMeter reads the pcap, generates total flows, filters invalid flows, and produces the CSV file.

**CICFlowMeter showing processing progress bar and flow count:**  
**

*Figure 63* CICFlowMeter showing processing progress bar and flow count

### 6.9 Summary of Chapter

In this chapter, a complete dataset generation workflow was implemented using a controlled virtual lab environment. Kali Linux generated multiple DoS and DDoS related traffic patterns, Windows captured traffic using Wireshark, and CICFlowMeter converted the .pcap files into CIC compatible .csv flow datasets.

This implementation supports the main objective of the DDoS Analyzer system, which is to allow any captured traffic to be converted into the same feature structure used during training, making the detection model practical for real deployments.