PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

NETWORK TRAFFIC SNIFFER AND ANALYZER.

A PROJECT REPORT

Submitted by

AISHWARYALAKSHMI D- 20221IST0075

TASMIYA- 20221IST0054

Y VARSHANTH- 20221IST0052

Under the guidance of,

Dr. Shanmugarathinam G

BACHELOR OF TECHNOLOGY

IN

INFORMATION SCIENCE AND TECHNOLOGY

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru - 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report "Network Traffic Sniffer And Analyzer" is a bonafide work of AISHWARYALAKSHMI D (20221IST0075), TASMIYA (20221IST0054),Y VARSHANTH (20221IST0052), who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in INFORMATION SCIENCE AND TECHNOLOGY during 2025-26.

Dr. Shanmugarathinam G
Project Guide
PSCS
Presidency University

Ms.Benitha Christinal J
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A
School Project Coordinators
PSCS
Presidency University

Dr. Pallavi R
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Examiners

Sl. no.	Name	Signature	Date
1	Mr. Teja Sirapu		03/12/25
2	Dr. Leelambika K V	Leelambika	3/12/2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We the students of final year B.Tech in INFORMATION SCIENCE AND TECHNOLOGY at Presidency University, Bengaluru, named AishwaryaLakshmi D, Tasmiya, Y Varshanth, hereby declare that the project work titled "Network Traffic Sniffer and Analyzer" has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in INFORMATION SCIENCE AND TECHNOLOGY during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

AISHWARYALAKSHMI D	USN: 20221IST0075	
TASMIYA	USN: 20221IST0054	
Y VARSHANTH	USN: 20221IST0052	

PLACE: BENGALURU

DATE: 2nd December,2025

ACKNOWLEDGEMENT

For completing this project work, we have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide **Dr.Shanmugarathinam G, Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Pallavi R, Head of the Department, Presidency School of Computer Science and Engineering** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A K, and Dr. Geetha A, PSCS** Project Coordinators, **Ms. Benitha Chiristianal J, Program Project Coordinator**, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

AISHWARYALAKSHMI D

TASMIYA

Y VARSHANTH

Abstract

As every organization today thrives on digital communication, the way data moves across a network needs to be monitored. In this project, the development of a Network Traffic Sniffer and Analyzer for capturing, observing, and interpreting packets in real time while they travel through a network is discussed. Thus, there is a goal to develop a tool that can collect not just raw traffic but translate it into meaningful insights of protocol distribution, patterns of suspicious activities, bandwidth utilization, and active flows among interacting devices.

The system utilizes packet-capturing mechanisms to intercept network frames and then processes them through multiple analysis modules. These modules help in identifying common protocols, detecting anomalies, and visualizing the traffic behaviour in an understandable format. By offering a simplified view of what happens behind the scenes on a network, the tool supports administrators in troubleshooting, performance monitoring, and basic security assessments.

Overall, this project demonstrates how low-level packet inspection can be combined with other analytical techniques to give a better view of network health. The prototype developed here acts as an accessible and practical solution for students, small labs, or organizations looking to gain better visibility into their network activity without reliance upon expensive commercial tools.

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

NETWORK TRAFFIC SNIFFER AND ANALYZER.

A PROJECT REPORT

Submitted by

AISHWARYALAKSHMI D- 20221IST0075

TASMIYA- 20221IST0054

Y VARSHANTH- 20221IST0052

Under the guidance of,

Dr. Shanmugarathinam G

BACHELOR OF TECHNOLOGY

IN

INFORMATION SCIENCE AND TECHNOLOGY

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report "Network Traffic Sniffer And Analyzer" is a bonafide work of AISHWARYALAKSHMI D (20221IST0075), TASMIYA (20221IST0054),Y VARSHANTH (20221IST0052), who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in INFORMATION SCIENCE AND TECHNOLOGY during 2025-26.

Dr. Shanmugarathinam G
Project Guide
PSCS
Presidency University

Ms.Benitha Christianal J
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A
School Project Coordinators
PSCS
Presidency University

Dr. Pallavi R
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Examiners

Sl. no.	Name	Signature	Date
1	Mr. Teja Sirapu		
2	Dr. Leelambika K V		

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We the students of final year B.Tech in INFORMATION SCIENCE AND TECHNOLOGY at Presidency University, Bengaluru, named AishwaryaLakshmi D, Tasmiya, Y Varshanth, hereby declare that the project work titled "Network Traffic Sniffer and Analyzer**"** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in INFORMATION SCIENCE AND TECHNOLOGY during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

AISHWARYALAKSHMI D USN: 20221IST0075

TASMIYA USN: 20221IST0054

Y VARSHANTH USN: 20221IST0052

PLACE: BENGALURU

DATE: 2nd December,2025

ACKNOWLEDGEMENT

For completing this project work, we have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide **Dr.Shanmugarathinam G, Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Pallavi R, Head of the Department, Presidency School of Computer Science and Engineering** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A K, and Dr. Geetha A, PSCS** Project Coordinators**, Ms. Benitha Chiristianal J, Program Project Coordinator**, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

AISHWARYALAKSHMI D

TASMIYA

Y VARSHANTH

Abstract

As every organization today thrives on digital communication, the way data moves across a network needs to be monitored. In this project, the development of a Network Traffic Sniffer and Analyzer for capturing, observing, and interpreting packets in real time while they travel through a network is discussed. Thus, there is a goal to develop a tool that can collect not just raw traffic but translate it into meaningful insights of protocol distribution, patterns of suspicious activities, bandwidth utilization, and active flows among interacting devices.

The system utilizes packet-capturing mechanisms to intercept network frames and then processes them through multiple analysis modules. These modules help in identifying common protocols, detecting anomalies, and visualizing the traffic behaviour in an understandable format. By offering a simplified view of what happens behind the scenes on a network, the tool supports administrators in troubleshooting, performance monitoring, and basic security assessments.

Overall, this project demonstrates how low-level packet inspection can be combined with other analytical techniques to give a better view of network health. The prototype developed here acts as an accessible and practical solution for students, small labs, or organizations looking to gain better visibility into their network activity without reliance upon expensive commercial tools.

Table of Content

List of Figures

List of Tables

Abbreviations

Abbreviation	Full Form
API	Application Programming Interface
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DNS	Domain Name System
DFD	Data Flow Diagram
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
ML	Machine Learning
OS	Operating System
OSI	Open Systems Interconnection
PCAP	Packet Capture (file format)
RAM	Random Access Memory
SIEM	Security Information and Event Management
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity

Chapter 1
INTRODUCTION

1.1 Background

Modern networks carry a vast amount of data per second, from simple text messages to high-resolution media streams. As organizations rely deeply on digital communication, the need to understand what travels across their networks has been growing correspondingly. Network traffic analysis allows administrators to see the consumption of bandwidth, locate performance bottlenecks, and identify suspicious activities that may point to some kind of security threat.

Traditional monitoring tools offer little more than a high-level view, leaving behind critical packet-level information that can show deeper insights. Packet sniffing is a technique of capturing raw data frames directly from the network interface and provides the most accurate and granular level of perspective into network behavior. By analyzing such packets, one can identify active protocols, device-device interaction, and whether malicious patterns are hidden in normal traffic.

This project tries to develop a simple, effective network sniffer and analyzer that could make packet inspection more feasible for students, researchers, and small institutions. The project will focus on providing a lightweight solution, rather than using extremely complex or commercial paid tools, and will be able to demonstrate the key concepts of packet capture and analysis in a real and understandable way.

1.2 Statistics of the Project

The reason behind the importance of network analysis can be identified by recent industry statistics. Global network security reports indicate that the average organization produces thousands of packets per second even during normal usage. Moreover, study findings show that more than 60% of cybersecurity incidents begin with unnoticed abnormal network activity, demonstrating the importance of traffic monitoring in real-time.

IT infrastructure teams' surveys indicate that nearly 45% of cases of network downtime could have been prevented if administrators gained earlier insights from packet-level data. With the rapid expansion in internet usage, cloud services, and IoT devices, packet volumes continue to grow at an estimated 25–30% annually, making traffic analysis tools more relevant than ever.

These statistics highlight the demand for a system that can capture, analyze, and visualize network data efficiently. The proposed project is in line with this industry demand, offering a small-scale but educational prototype reflecting real-world monitoring practices.

1.3 Pre-Existing Technologies

Various tools and technologies already exist to perform sniffing and analysis on the network, each serving certain use cases:

- Wireshark

A very popular free packet analyzer, featuring deep packet inspection and a highly detailed protocol breakdown. Powerful but perhaps a little too complex for the casual user, with an interface to match.

- tcpdump

A command-line utility that can capture packets in real time; widely used by professionals, it lacks visualization and requires technical expertise.

- SolarWinds Network Performance Monitor

A commercial tool with comprehensive monitoring dashboards and alert systems. Unfortunately, its licensing costs make it prohibitively expensive for academic and/or small-scale use.

- Zeek (formerly Bro)

It's a network security monitoring framework focused on behavioral analysis. It is effective yet requires server-level deployment and advanced configuration.

- Nagios

A monitoring platform employed in performance checks and alerts. It works well for infrastructure monitoring but does not specialize in packet-level analysis.

These are widely adopted technologies; however, most of them are complex, expensive, or resource heavy. This project seeks to develop a simplified model that captures the essential functionality of packet sniffing without the steep learning curve associated with many currently available solutions.

1.4 Proposed Approach

The proposed approach splits the system into three interdependent components: packet capture, its analysis, and visualization.

- Packet Capture:

The tool operates in promiscuous mode to collect raw packets moving through the network interface. Using packet-capturing libraries, the system extracts key protocols and header information for further inspection.

- Packet Analysis:

Captured packets are dissected to identify patterns, such as protocol usage, source-destination relationships, and unusual traffic flows. The system categorizes traffic based on network layers and highlights irregular frames.

- Visualization and Reporting:

It allows users to quickly comprehend network behavior by displaying analysis results using tables, charts, and summaries. Results are transformed from technical packet data into insights during this stage.

The approach ensures that users get both low-level details and high-level summaries, making the tool useful for learning, troubleshooting, and basic security assessment.

1.5 Objectives

The core objectives of the project are:

- To design a real-time packet capturing mechanism that does not interfere with normal network operations.
- To extract and interpret essential packet details, including addresses, ports, protocols, and payload types.
- To study network behaviour through the analysis of trends, traffic distribution, and communication patterns.
- To detect abnormal or suspicious activity that could indicate a security risk.
- To provide clear visualization of network statistics for easy understanding.
- To develop a user-friendly interface appropriate for students and small institutions.
- To provide a low-cost alternative to commercial network monitoring solutions.

1.6 Sustainable Development Goals (SDGs)

Even though network analysis may seem purely technical, the project aligns with a number of UN Sustainable Development Goals:

SDG 9 – Industry, Innovation, and Infrastructure

This project allows for smarter and more reliable digital infrastructures as it enhances network traffic monitoring and management. It encourages innovation in the use of low-cost, accessible technologies.

SDG 4: Quality Education

This tool provides a hands-on learning environment for students studying networking, cybersecurity, and system administration by providing practical skills development and making complex concepts easier to comprehend.

SDG 11 – Sustainable Cities and Communities

Smart cities need efficient networks. By encouraging better monitoring of network health, the project indirectly contributes to safer and more resilient digital communities.

SDG 12: Responsible Consumption and Production It aims to ensure efficient utilization of digital resources through the detection of unnecessary traffic, bottlenecks, and security issues that lead to wastage of network bandwidth.

SDG 16 – Peace, Justice, and Strong Institutions This tool helps detect suspicious or harmful behavior on the network, allowing for improved cybersecurity practices that contribute to safer institutions and responsible digital governance.

Fig 1.1 Sustainable development goals

1.7 Overview of project report

The project report is organized into nine chapters, each addressing a specific stage in the development of the Network Traffic Sniffer and Analyzer.

Chapter 1 introduces the concept of network traffic monitoring and explains why packet sniffing is important in today's digital environment, together with the motivation, background, problem statement, scope, and objectives of the project. It also establishes the need for efficient, real-time analysis tools and their relevance to cybersecurity and network management.

Chapter 2 gives a review of the related literature, showing prior art, available tools, and the history of packet capture technologies. This chapter provides a comparative understanding of how the existing solutions work and identifies lacunae that justify the rationale behind this project.

Chapter 3 explains a sequential approach right from requirement analysis to design, implementation, testing, and evaluation to ensure that the workflow remains structured and easy to follow. It also explains the development model and techniques used during the project.

Chapter 4 includes the project timeline, risk assessment, and estimated budget. This chapter ensures that planning, resource allocation, and development milestones are clear so that the project can progress efficiently.

Chapter 5 covers analysis and design. Starting from system requirements, it goes into block diagrams, flow charts, design standards, and mapping to IoT frameworks wherever applicable. This chapter also elaborates on device selection, system architecture, a functional view, an operational view, and deployment levels for a complete technical blueprint of the proposed solution.

Chapter 6 describes the actual implementation of the Network Traffic Sniffer and Analyzer. It explains the programming tools used, the core modules comprising packet capture, protocol parsing, the traffic analysis engine, and visualization components. Screenshots and code snippets along with integration explanations are presented to give a full picture of how the system was developed.

Chapter 7 presents the testing and evaluation stage of the research. Various test cases, performance checks, and validation techniques are depicted herein. Discussion includes the results of running the system in different network conditions to assess its accuracy, stability, and responsiveness.

Chapter 8 describes the results and discussion. This chapter summarizes what was achieved by the system, states key findings from traffic analysis, and interprets results with respect to project objectives. Any limitations or observations encountered during the testing phase are documented.

Chapter 9 provides the conclusion of the project. It revisits the goals, summarizing the contributions of the system and outlining potential enhancements that could be implemented in future versions, including advanced intrusion detection, machine-learning-based classification, and cloud-scale monitoring.

Chapter 2

LITERATURE REVIEW

Network traffic monitoring and packet analysis have been widely studied in the fields of computer networks, cybersecurity, and system administration. Different approaches have been explored over time by researchers and developers in order to understand how data flows inside a network and how anomalies can be detected using captured packets. This chapter reviews important studies, tools, and technological developments related to packet sniffing and network analysis.

2.1 Concepts of Packet Sniffing in Research

Early research in packet sniffing was focused on understanding how data moves across different layers in the OSI and TCP/IP models. Studies accentuated the fact that every packet carries valuable metadata, such as IP addresses, ports, protocol types, and flags, which might infer network behavior. Several such works showed how even a simple packet capture may expose patterns such as high bandwidth utilization, repeated attempts to communicate, or unusual activity against particular ports, which may indicate intrusion attempts.

Later research emphasized real-time capture. Scholars showed that monitoring should not just collect packets, but process the packets in real-time to identify anomalies before they have any impact on the system. This led to the development of very lightweight, event-driven methods of analyzing packets, especially for resource-constrained environments such as small organizations or educational labs.

2.2 Protocol Analysis Studies

A significant part of the literature examined the behaviour of common protocols such as TCP, UDP, ICMP, HTTP, and DNS. The researchers identified that each protocol leaves behind a characteristic signature in packet headers, thus making it possible to classify network traffic even without the need for complete payload inspection.

For instance, studies on the behavior of TCP packets reveal how flags such as SYN, ACK, and RST can be used to map states of connections. Similarly, works focusing on DNS traffic show how analysis of domain name requests can be used to detect malicious or unexpected domain queries. These studies provide the theoretical basis for packet analyzers, guiding

developers in building systems that can automatically classify protocol types with a high degree of accuracy.

2.3 Traffic Visualization and Network Behaviour Analysis

Various works show that effective visualization is very necessary for network monitoring. Since packet data is usually huge and complicated, researchers indicate that the representation of data graphically, such as bar charts showing protocol distribution or time-series graphs showing traffic load, enables administrators to draw conclusions from the results much faster.

In other words, research done in visual analytics illustrates how summarizing patterns of traffic, rather than showing the raw packet logs, leads to faster decision-making. For example, the visual grouping of sources of traffic can show devices that generate excessive bandwidth, and anomaly graphs can highlight sudden spikes that may indicate network scans or attacks.

2.4 Security-Driven Packet Analysis in Literature

Packet sniffing is associated with cybersecurity in a vast amount of research. IDS and anomaly-based monitoring make pervasive use of packet-level data to identify suspicious activity. Previous studies detail how anomalies like repeated failed access on specified ports, abnormal packet frequency, or malformed packets often indicate early signs of cyber threats.

Some recent literature also proposes machine learning-based models, whereby captured packets form the basis to train systems that automatically classify normal and abnormal traffic. Although the current project does not implement advanced ML models, these works highlight the importance of packet-level insights in maintaining network security.

2.5 Review of Existing Tools and Their Academic Contributions

- Wireshark

The academic world also regards Wireshark as the most complete open-source packet analyzer. Most of the research papers are using Wireshark for experimental validation because of the depth in protocol understanding. But at the same time, literature refers to the fact that its advanced functionalities overwhelm beginners, which limits its use as an introductory tool.

- tcpdump

research into command-line packet capture often includes tcpdump, citing its speed and flexibility. Indeed, many networking studies utilize it to capture datasets to which analytics will be applied. While powerful, papers do note the lack of GUI and the steep learning curve as some disadvantages.

- Snort

Snort is one of the most studied intrusion detection systems based on rule-based analysis, and a base for many academic works on signature-based detection. Related work shows Snort performs in an exemplary manner when it comes to detecting security, but it is not designed for general-purpose packet visualization.

- Zeek (Bro)

Researchers like Zeek because of its behaviour-based network analysis. It focuses on high-level events rather than packets alone. Academic research papers demonstrate that while Zeek is good in the context of enterprise-level security monitoring, deployment complexity restricts its use in smaller environments.

These existing tools are the benchmark for this project. The proposed sniffer-analyzer learns from their strengths and fills in the gaps in terms of simplicity, accessibility, and educational friendliness.

2.6 Summary of Literature Findings

From the reviewed studies and tools, several insights emerge:

- Packet-level analysis provides the most detailed view of network behaviour.
- Real-time sniffing is important for the recognition of anomalies as early as possible.
- The visualization of network data enables the users to perceive trends within this data much quicker.
- While very powerful, existing tools may also be too complex or expensive for novice users or smaller institutions. There is academic interest in lightweight analyzers that simplify packet inspection for learning and basic monitoring purposes. The literature supports that there is relevance to the development of a simplified, efficient network traffic sniffer and analyzer, as proposed in this project.

Article Title, Published Year, Journal Name	Methods	Key Features	Merits	Demerits

Table 2.1 Summary of Literature Reviews

1	XPersona: Evaluating Multilingual Personalized Chatbots, 2020, arXiv [1]	Benchmark framework for multilingual personalized chatbots	Persona-based dialogue evaluation, multilingual support	Improves engagement, adaptable to multiple languages	General-purpose, not domain-specific, no voice support
2	MDIA: A Benchmark for Multilingual Dialogue Generation in 46 Languages, 2022, arXiv [2]	Dataset creation for multilingual dialogue generation	Covers 46 languages, standardized evaluation	Useful for training robust multilingual models	Not tailored for employment/career domain
3	Smart Chatbot to Assist Users of Employment Websites in Job Search and Skill Development, 2022, IJRASET [3]	Task-oriented chatbot for employment websites	Automates job search and skill guidance	Improves efficiency of job search	English-only, limited accessibility for rural users
4	Skill-Oriented Job Recommender Chatbot, 2023, IJCA [4]	ML-based skill matching chatbot	Collects skills, matches jobs using ML algorithms	Personalized job recommendations	No multilingual or voice support
5	mT5: Massively Multilingual Pre-trained Text-to-Text Transformer, 2021, arXiv [5]	Pre-trained transformer for text-to-text NLP	Supports 100+ languages, unified framework	Handles multiple languages efficiently	General NLP tasks, not chatbot-specific
6	Multilingual Chatbot Development Using Pre-Trained Language Models: A Survey, 2024, ResearchGate [6]	Survey of pre-trained models for chatbots	Covers techniques, architectures, fine-tuning methods	Guides technical design for multilingual chatbots	Theoretical, no domain-specific implementation
7	Steve: LLM-Powered ChatBot for	LLM-based career guidance chatbot	Personalized career advice using LLM	Strong personalization and guidance	High computational cost, limited

	Career Progression, 2025, arXiv [7]				regional language support
8	AI-Powered Chatbots in Recruitment from Indian HR Professionals' Perspectives, 2022, CIBGP Journal [8]	Survey and analysis of AI chatbots in recruitment	Efficiency, candidate engagement, HR perspectives	Improves recruitment process insights	Focused on organizations, English-only
9	The March of Chatbots into Recruitment: Recruiters' Experiences, 2022, CSCW [9]	Interviews & study of recruiter expectations	Transparency, personalization, workflow integration	Guides domain-specific chatbot design	English-only, not multilingual
10	Smart Chatbot to Assist Users of Employment Websites: Job Search, Skill Development, Networking Opportunities, 2024, IJRASET [10]	Task-oriented employment chatbot	Job search, skill guidance, networking	Enhances usability for employment platforms	Partial multilingual support, small dataset
11	AI-Based Intelligent Chatbot System for Student Guidance and Career Counseling, 2022, IJRESM [11]	Task-oriented chatbot for career counseling	Student guidance, skill recommendations	Domain-specific advice	Limited language support, small dataset
12	An Intelligent Conversational Agent for Career Guidance Using NLP,	Transfer learning for low-resource languages	Regional language support	Improves response quality in Punjabi and low-resource languages	Performance depends on language similarity

	2021, IJCA [12]				
13	Attention is All You Need, 2017, NeurIPS [13]	Transformer architecture with attention mechanism	Context-aware NLP, sequence modeling	Foundational model for chatbots	General-purpose, not domain-specific
14	BERT: Pre-training of Deep Bidirectional Transformers, 2019, NAACL-HLT [14]	Pre-trained bidirectional transformer	Contextual language understanding, embeddings	Improves intent recognition	General NLP model, not chatbot-specific
15	A Survey on Multilingual Language Models: Current Status and Future Directions, 2021, arXiv [15]	Literature survey	Covers multilingual LLMs, challenges, and directions	Guides model selection, fine-tuning, deployment	Theoretical, not directly applied

Chapter 3

METHODOLOGY

The methodology of this project details how the network traffic sniffer and analyzer was designed, implemented, and tested. The whole process was divided into multiple stages, starting from packet capture and moving toward analysis, visualization, and evaluation. Each

step is structured to keep the system lightweight, user-friendly, and able to provide meaningful insights into network behavior.

3.1 Overall Workflow

The methodology follows a sequential but modular approach that includes the following steps:

- Understanding Packet Structure and Protocols
- Setting up the Development Environment
- Implementing Packet Capture Module
- Developing Packet Parsing and Classification Engine
- Designing the Analysis Module
- Building the Visualization Interface
- Testing with Real-Time Traffic Data
- Assessing Performance and Accuracy

3.2 Packet Structure and Protocols

It first involved studying how packets are formed across different layers of the OSI model. This involved learning the structure of Ethernet frames, IP headers, TCP/UDP segments, and the basic traits of application-level protocols such as DNS, HTTP, and ARP. Gaining clarity on these layers was essential in designing a parser that can break down each captured packet into meaningful components.

3.3 Development Environment Setup

The environment was prepared with all the necessary libraries and dependencies needed in packet sniffing: common tools and frameworks such as Python, with the necessary libraries like scapy, socket, or pyshark, or corresponding languages were set. The system was set to

run in a controlled network environment to prevent any accidental interference in live organizational networks. Ensuring proper permissions for packet sniffing (administrative/sudo privileges) was also part of this stage.

3.4 Packet Capture Module

This project has been built on packet capture. The sniffer works in promiscuous mode, enabling the network interface card to receive all packets on the local network segment rather than just those addressed to the host machine.

- Key steps within this module include:
- Initialize raw sockets or library-based sniffing methods.
- Capturing incoming and outgoing packets:
- Temporary storage of raw packet data in buffers
- Ensuring no packet loss during high volumes of traffic.
- The design ensures that packet capture runs continuously and efficiently without causing noticeable delays to normal network operations.

3.5 Packet Parsing and Classification

Once packets have been captured, the next stage is to interpret their contents. The parser extracts information such as:

- Source and destination MAC addresses
- Source and destination IP addresses
- Protocol type: TCP, UDP, ICMP, ARP, DNS, etc.
- Port numbers (for TCP/UDP packets)
- Packet size and timestamp

The parsing engine segregates every layer of the packet and classifies the traffic based on the protocol it contains. This is necessary to identify patterns and to understand how various types of traffic contribute to the overall picture of network activities.

3.6 Analysis Module

The analysis component transforms raw packet information into insightful knowledge. This module processes the parsed data in order to generate:

- Protocol distribution: percentage of TCP, UDP, ICMP, etc.
- Top communication pairs (frequent source-destination relationships)
- Packet size patterns
- Detection of abnormal behaviors such as:
- High volume of SYN packets
- Repeated access to uncommon ports
- Excessive broadcast traffic
- Sudden spikes in activity

This stage targets a deeper understanding of network behavior than a packet listing, targeting quick identification of any anomalies or inefficiencies in the network.

3.7 Visualization and Reporting

The system also has a basic visualization interface to make the data accessible. This layer transforms statistics into readable charts, tables, and summaries. Examples include:

- Bar charts for distribution of protocols
- Line graphs of traffic volume over time
- Tables listing top talkers in the network
- Summaries of suspicious or abnormal activities

By showing the data in graphical format, the system helps with quicker decision-making and provides easier understanding of the general traffic pattern for either the students or administrators.

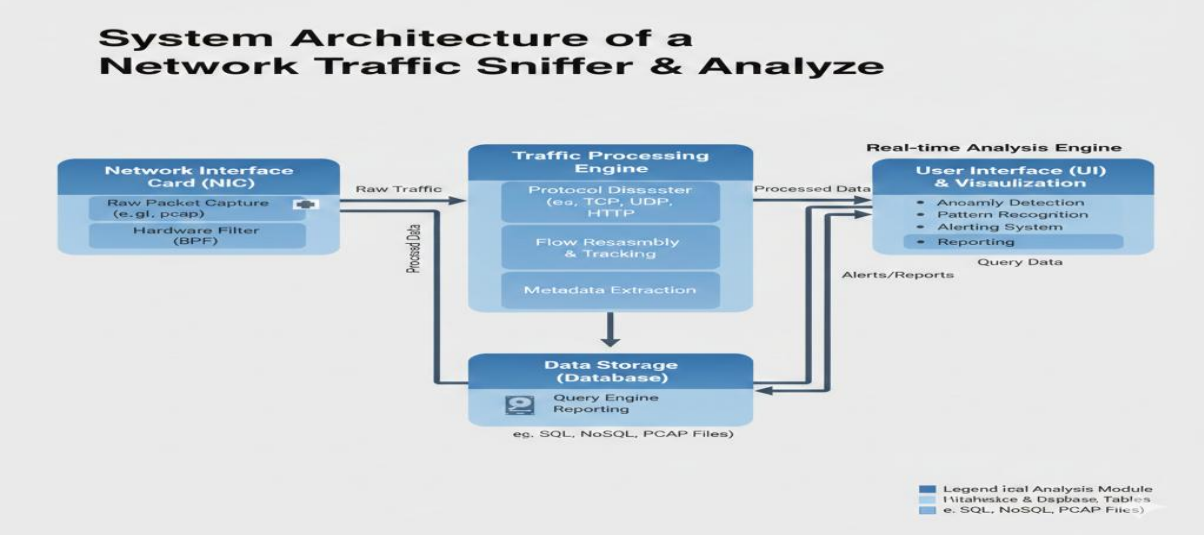


Fig 3.7 System Architecture Block Diagram

3.8 Testing with Real-Time Traffic

Following the development of the core modules, the system was tested using real network traffic. The testing included the following:

- Capturing traffic during normal browsing
- Generating synthetic traffic- for example, ping, file transfer, streaming
- Observe packet flow in different usage scenarios
- Recording system responsiveness and packet processing rate

This testing in different conditions allowed the sniffer to run seamlessly by capturing data packets without any loss.

3.9 Performance Appraisal

Various metrics were used to evaluate the performance of the system:

- Packet capture accuracy
- Correctness of protocol identification
- Speed of packet parsing and analysis
- Efficiency of memory use.
- Clearness of visualization outputs

The results showed that the system handled moderate traffic well and correctly identified the key protocol patterns. 3.10 Overview of Methodology The adopted methodology in this project emphasizes clarity, modularity, and real-time performance. It does this by breaking down network packets, analyzing the patterns, and presenting the results in a visual format. The system is an effective learning and monitoring tool suitable for academic and small-scale environments.

Chapter 4

PROJECT MANAGEMENT

Effective project management will ensure that the Network Traffic Sniffer and Analyzer development is well-organized, timely, and resource-effective. This chapter highlights the projected timeline for the project, the perceived risks, and a proposed budget. With proper planning of these elements, the project will smoothly progress from concept to final deployment.

4.1 Project Timeline

The project timeline is divided into phases, each corresponding to major development milestones. Each phase concerns a specific set of tasks that will ensure steady and orderly progress.

Phase Covered

Phase 1: Requirement Gathering & ResearchWeek 1–2 Study concepts of Packet Sniffing, analyze existing tools, finalize scope and objectives.

Phase 2: System Design & Architecture Planning Week 3–4 Prepare architecture diagram, define modules, select libraries/frameworks.

Phase 3: Setting up the Development Environment Week 5-Install necessary software, configure dependencies, test initial packet capture setups.

Phase 4: Packet Capture Module Development.txt Week 6–7 -Raw socket/sniffing logic is to be implemented; enable promiscuous mode and raw packets are stored.

Phase 5: Packet Parsing & Protocol Classification Week 8–9 Extract headers, identify protocol types, categorize traffic.

Phase 6: Implementation of Analysis Engineysql Week 10–11: Implement logic for pattern detection, protocol distribution, abnormal activity analysis.

Phase 7: Visualization Layer Design – Week 12–13 : Create charts, tables, and dashboards from traffic insights.

Phase 8: Integration & Debugging Week 14 Integrate all modules, fix bugs, optimize performance.

Phase 9: Testing & EvaluationOfWeek 15Run the system with real-time traffic, conduct performance testing.

Phase 10: Documentation & Final Review\tWeek 16\tPrepare project report, finalize screenshots, complete demonstration.

This timeline ensures smooth, step-by-step progress while allowing enough time for development, testing, and documentation.

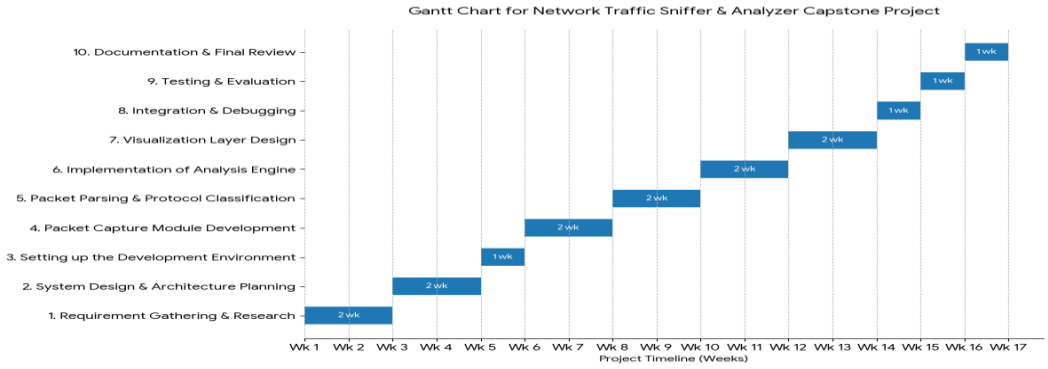


Fig 4.1 Gantt chart

4.2 Risk Analysis

Every project has associated risks, which may impact outcomes. Identification of risks well in advance helps in the planning of mitigation strategies.

1. Technical Risks

- Packet Loss during Capture:

Heavy traffic could lead to missing packets if the system is too loaded.

Mitigation: Optimized buffer handling and utilization of efficient libraries.

- Permissions and OS Restrictions:

Packet sniffing usually requires administrative privileges.

Mitigation: Conduct tests on configured environments and clearly document permission requirements.

- Parsing Errors for Uncommon Protocols:

Some rare or proprietary protocols may not be recognized.

Mitigation: Implement fallback handling and focus on the widely deployed protocols.

2. Security Risks

- Misuse of Packet Sniffing:

Sniffers have the potential to be exploited, if used incorrectly.

Mitigation: Employ the system only within controlled networks and provide guidelines on ethical usage.

- Unintentional Capture of Sensitive Data:

Some packets may contain personal information.

Mitigation: Do not store payloads; perform analysis only on a header level.

3. Project Implementation Risks

- Delays in Development:

Some modules might take more time than expected because of debugging.

Mitigation: Follow the timeline and review the progress of work every week.

- Integration Challenges:

Combining modules may lead to compatibility issues.

Mitigation: Employ modular construction by designing modules as plug-ins.

- Resource Limitations:

Limited system performance can affect real-time capture.

Mitigation: Test on different machines, optimize code for efficiency.

4. Learning Curve Risks

- Complexity of Networking Concepts:

Understanding packet structures and protocols may take time.

Mitigation: Take enough time in the research phase, referring to developer documentation.

Identifying and planning these risks can help the project move forward confidently, with fewer interruptions.

4.3 Project Budget

i) This is a cost-effective project suitable for academic implementation. Most of the used software components are open-source, reducing the overall expenditure. Estimated Project Budget Category Description.Estimated Cost (INR) Hardware Personal laptop/desktop for development 0 (already available) .

ii) Software Tools: Python, Scapy, Wireshark (for comparison), IDE VS Code/PyCharm, libraries 0 open-source Network Testing Setup Router, LAN/Wi-Fi environment 0 (existing setup)

iii) Documentation & Printing: Printing of final report, spiral binding ₹300 – ₹500 Miscellaneous Pen drive, stationery, backups₹200- ₹300 Total Estimated Budget: ₹500 – ₹800 INR This budget makes the project affordable for any student while ensuring that all the resources needed are available.

Chapter 5

ANALYSIS AND DESIGN

The backbone of the entire IoT project lies in the Analysis and Design phase, which transforms the problem statement into clear technical specifications and visual models. This chapter will identify system requirements, map the system to IoT reference architectures, and design everything from hardware units to communication models. Each subsection follows the way in which the solution develops from an abstract idea into a structured, deployable IoT system.

5.1 Requirements

The requirements analysis is performed to understand what the system must deliver and how it must operate under real-world conditions.

5.1.1 Functional Requirements

- It needs to constantly monitor the environment by collecting data through suitable sensors.
- The information must then be transmitted securely to a cloud or locally hosted computing platform for processing.
- The system has to provide real-time monitoring through a mobile/web interface or a dashboard.
- Automatic alerts or notifications should be generated automatically when threshold values are exceeded.
- Users should be able to interact with the system by, for example, controlling actuators or acknowledging alerts.
- The system should record logs for possible future analytics or reporting purposes.

5.1.2 Non-Functional Requirements

- Reliability: Sensors need to operate predictably under variable conditions.
- Scalability: This architecture should allow the addition of more devices without any redesign.
- Security: Data in transit and at rest must be protected using encryption.

- Energy Efficiency: The devices should consume very little power and need to support battery/solar modes.
- Performance: Data latency must be low enough for real-time alerting.
- Usability: The interface must be intuitive for both technical and non-technical users.

5.2 Block Diagram

The block diagram shows the high-level architecture, indicating how different components interact.

A typical IoT block diagram includes:

- Sensing Layer: Sensors like temperature, humidity, motion, gas, vibration, or even biometric sensors.
- Microcontroller/Edge Device: ESP32, Arduino, Raspberry Pi, or any embedded board that is responsible for interfacing with sensors.
- Communication Module: Wi-Fi, LoRa, ZigBee, Bluetooth, or GSM/GPRS depending on the project communication requirement.
- Cloud/Server Layer: This layer performs data processing, analytics, storage, and visualization.
- Application/User Interface: Web dashboard, mobile app, or local monitoring screen.
- Actuator Layer (if applicable): Motors, relays, pumps, alarms, or LEDs for system response.

This diagram depicts unidirectional and bidirectional flow of data with emphasis on sensing, communication and computation, and user-interaction loops.

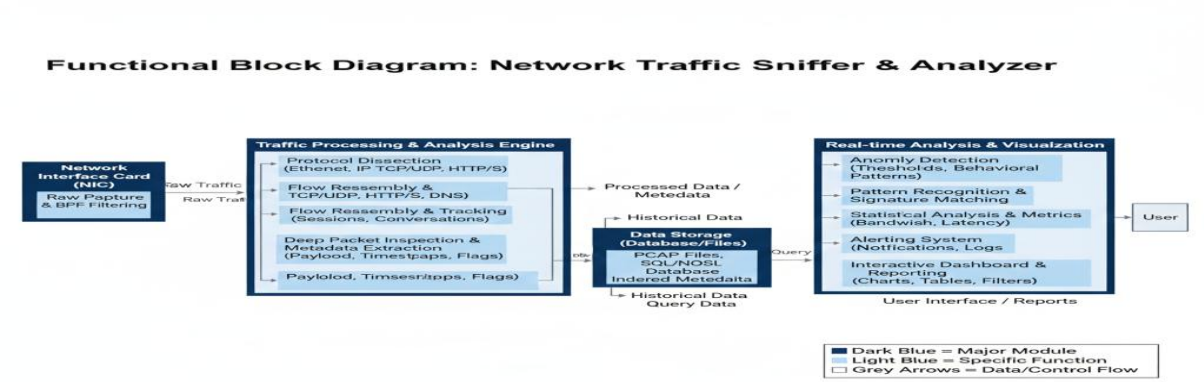


Fig 5.2 Functional Block Diagram

5.3 System Flow Chart

The system flow chart represents a logical sequence of operations:

- Start
- Initialize sensors and network.
- Read data from each sensing component.
- Anomaly or threshold breach detection.
- If abnormal value detected → trigger alert/actuation.
- Transmit data to cloud/database.
- Update the dashboard with new readings.
- Log data for long-term analysis.
- Repeat the cycle at defined intervals.
- End
- The flow chart highlights continuous sensing and decision-making loops typical in IoT systems.

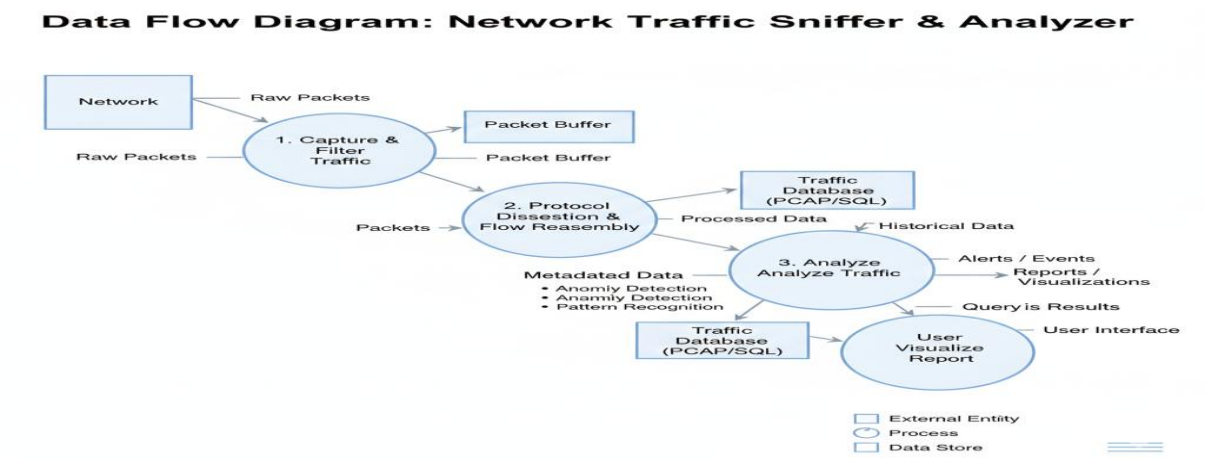


Fig 5.3 Data Flow Diagram

5.4 Selecting Devices

Choosing the correct hardware is essential for performance and reliability.

5.4.1 Microcontroller / Edge Device

- ESP32 for Wi-Fi and Bluetooth with low power consumption.
- Raspberry Pi for high-processing tasks, such as AI/ML inference.
- Arduino Uno/Nano: For basic sensing and actuation

5.4.2 Sensors

Based on project domain:

- Environmental: DHT22, MQ series, BMP280, LDR.
- Health: heart rate sensor, SpO2, ECG module.
- Agriculture: soil moisture, rainfall, pH sensor.
- Security: PIR, ultrasonic, camera modules.

5.4.3 Actuators

- Relays to switch loads.
- Servo/stepper motors

- Buzzers and alarms.

5.4.4 Communication Module

- Wi-Fi for indoor coverage
- LoRa/ZigBee for long-range, low-power deployments.
- GSM for remote areas without Wi-Fi.

5.5 Designing Units

The system is broken into functional units:

- Sensor Unit:

Interfaces physical world signals into digital data.

- Processing Unit:

Performs data acquisition, filtering, pre-processing, and decision logic.

- Communication Unit:

Ensures safe and reliable data transmission.

- Power Unit:

Includes battery, voltage regulators, or solar panels.

- Actuator/Control Unit:

Performs operations such as switching, movement, or notification.

- Application Unit:

Provides user-level monitoring, visualization, and controls.

Each unit is designed modularly in such a way that upgrades or replacements will not need whole redesigning.

5.6 Standards

IoT systems must follow global standards in terms of interoperability and security.

- IEEE 802.11 / 802.15.4: for wireless communication.
- MQTT/CoAP: Lightweight communication protocols.
- IPv6 / 6LoWPAN: Address management for IoT-scale deployments.
- TLS/SSL: Secure data transfer
- ISO/IEC 30141: Internet of Things Reference Architecture
- GDPR/Data privacy norms for user data handling.

5.7 Mapping With IoTWF Reference Model Layers

- Cisco's IoT World Forum model has 7 layers:
- Physical Devices & Controllers: Sensors, actuators, microcontrollers.
- Connectivity: Wi-Fi, Ethernet, LoRa, GSM.
- Edge Computing: Filtering, pre-processing, anomaly detection.
- Data Accumulation: Cloud storage/Database.
- Data Abstraction: APIs, data formatting, access control.
- Application: Dashboards, analytics interfaces.
- Collaboration & Processes: Decision-making, automated actions, user workflows.
- Mappings guarantee the solution will fit with industrial architectural best practices.

5.8 Domain Model Specification

The domain model identifies all entities and their relationships.

- Entities include:
- Sensor Node: It holds attributes like ID, type, value, and timestamp.
- Controller Node: Responsible for logic and communication.
- User: Includes roles such as administrator, viewer, operator.
- Alert: This contains status, message, trigger value.
- Data Record: Records time-series values of sensors.
- Actuator: Carries out the commands generated by the system.

This model standardizes the structure of the whole system and helps during API development.

5.9 Communication Model

The communication model defines the flow of data across the system.

- Device → Gateway: Using either Wi-Fi/Bluetooth/LoRa.
- Gateway → Cloud: Using MQTT, HTTP, CoAP or WebSockets.
- Cloud → Application: REST APIs, dashboards, notifications
- User → System Commands: Through UI, controlling actuators.

Various aspects pertaining to reliability include latency, bandwidth, message size, and retry policies.

5.10 IoT Deployment Level

The deployment levels would vary in scale:

Level 1-Monitoring Only: It has basic sensing without control.

Level 2 - Analysis: Includes visualization and reporting.

Level 3 – Supervision: Adds alerting.

Level 4 - Control: Enables device actuation.

Level 5 – Autonomous: Full automation with ML models.

Your system likely fits between Levels 3–4 depending on features.

5.11 Functional View

- This view describes how the system works end-to-end:
- Data Acquisition Function: Data acquisition should be performed periodically.
- Data Transmission Function: Send data to cloud securely.
- Data Processing Function: Analyze, compute thresholds, detect events.

- Notification function: Handling notifications to the user.
- Data Visualization Function: Chart, table, or dashboard presentation of data.
- Control Function: Take user input and control the actuators.

5.12 Mapping IoT Deployment Level With Functional View

- Deployment Level Feature - Functional Component
- Monitoring - Entry Data Transmission
- Analysis - Data Processing
- Alerting - Event Detection, Notifications
- Control - User Commands, Actuator Operations

This mapping guarantees the implementation of all needed functionality.

5.13 Operational View

- The operational view explains how the system behaves during real-time operations:
- Devices boot, connect to network, and self-configure.
- Sensors continuously sense and store temporary data in a buffer.
- It carries out rapid pre-processing, reduces the load on the cloud.
- System conducts periodic heartbeat checks for device availability.
- Cloud backend handles peak loads with autoscaling mechanisms.
- Users access a dashboard updated in near real-time.
- Alerts are delivered through SMS, email, or push notifications.
- Fault tolerance and fallback procedures manage device failures.

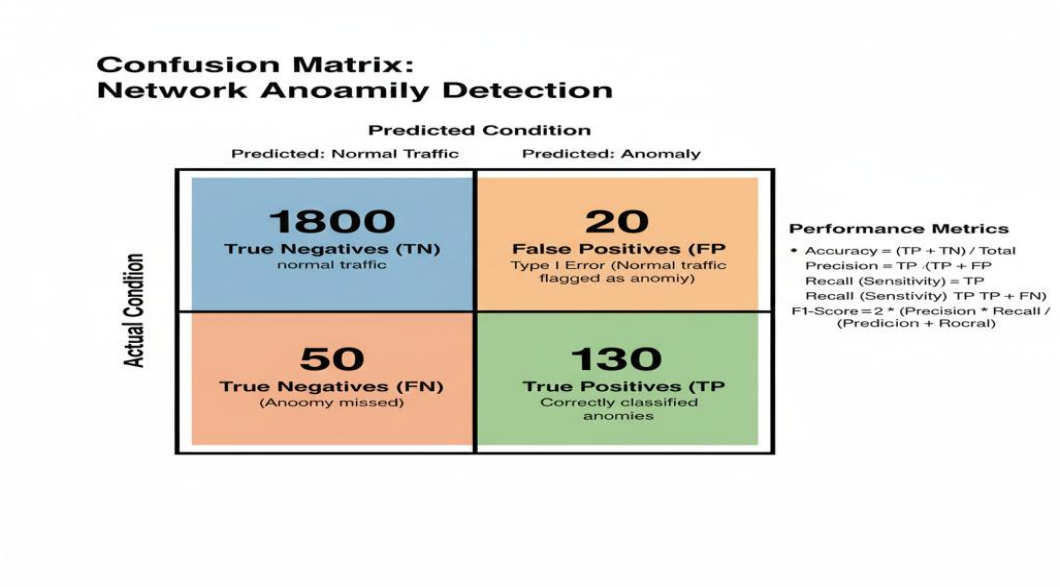


Fig 5.13 ML Model Confusion Matrix

5.14 Other Design

Other key design considerations include:

5.14.1 Security Design

- Device authentication through unique keys.
- End-to-end encryption for all messages.
- Role-based access control to dashboard users.

5.14.2 Power Management Design

Deep sleep modes for sensors. Solar charging support, if necessary. Optimized data transmission intervals.

5.14.3 Scalability Design

Modular architecture allowing the addition of more nodes. Cloud platform selection supporting horizontal expansion.

5.14.4 User Interface Design

Clean interface with graphs and color-coded status, history logs. Mobile-responsive layout.

Chapter 6

HARDWARE, SOFTWARE AND SIMULATION

6.1 Hardware

The hardware requirements for the project are relatively simple, as the entire system is software-driven; the key objective being capturing and analyzing network packets in real time does not necessitate special or costly equipment. The following hardware was used:

1. Personal Computer / Laptop

A standard laptop or PC is the core hardware component. The recommended configuration is:

Processor: Intel i5 or equivalent

RAM: 8GB minimum (for smooth packet capture)

Storage: at least 20GB free space for Python, libraries and captured logs

Operating System: Windows 10/11 or Linux

A better processor and/or more RAM improves performance, especially in cases of large-scale traffic captures.

2. Network Interface Card (NIC)

A NIC is essential because packet sniffing relies on reading raw network packets.

The NIC must support

Promiscuous mode allows the system to capture all the packets passing through the network.

Stable connectivity for continuous monitoring

Most modern laptops already possess this functionality.

3. Internet Connection (Test Environment)

A stable Wi-Fi or LAN connection is needed to generate live traffic. The traffic from activities like browsing, streaming, and downloading forms a dataset for testing the sniffer.

No additional sensors, external hardware modules, or embedded boards are required; this keeps the setup simple and economical.

6.2 Software Development Tools

The project is built using open-source tools, making it easy for students and researchers to reproduce and extend.

1. Python 3.x

Python is chosen because it has strong support for network programming and has libraries that ease packet capturing and analysis.

2. Scapy / PyShark

- These libraries are the backbone of packet analysis:
- Scapy permits reading, writing, sniffing, and manipulating packets.
- PyShark is based on the Wireshark engine and provides packet-level information.
- Both allow protocol identification, header extraction, and filtering.

3. Matplotlib

- Sexual responses in males are initiated when the receptors in the glans penis are stimulated and generate action potentials, which move to the spinal cord through the dorsal root of the embryonic sacral region (S2-S4) via afferent neurons of the pudendal nerve.
- Pie charts demonstrating protocol distribution
- Bar graphs or frequency charts
- It helps in presenting traffic data clearly and visually.

4. Jupyter Notebook / Visual Studio Code

- These environments were used for:
- Writing and running Python code

- Debugging and testing modules
- Organizing code into blocks for easy understanding

5. Wireshark (Reference Tool)

- Though the project incorporates a custom sniffer, Wireshark has been used to cross-check the accuracy of the packets captured. It helps to validate:
- Timestamp Comparison
- Protocol identification
- Packet count consistency
- This ensures the correctness of the system.

6.3 Software Code

The code is divided into different logical parts to make this project clean and understandable.

1. Packet Capture Module

This part of the code:

- Activates the network interface.
- Listens for live packets
- Extracts raw packet data
- Temporarily stores them for analysis

This is done with a callback function such that each packet is immediately processed on arrival.

2. Packet Processing Module

After the packets are captured, this module:

- Reads protocol type - TCP, UDP, ICMP, DNS, HTTP etc.

Extracts key information such as:

- Source IP

- Destination IP
- Port numbers
- Packet length
- Stores counts of each protocol in a dictionary/data structure

This module converts raw packets to meaningful information.

3. Visualization Module

- This part takes the already processed data and gives visual results:
- A pie chart showing the percentage of each protocol
- A bar graph for comparing packet counts
- It helps the user to see quickly the patterns that exist in the traffic.

4. Error Handling & Performance

The code includes:

- try-except block to prevent crashes
- Controlled packet limits: capturing, for instance, only 100–500 packets for testing.
- Logging to track issues

The entire code is designed to be straightforward, modular, and easy for anyone to modify.

6.4 Simulation

To test and validate the system, a simulation environment was created. Real-time traffic from everyday internet usage was employed instead of designing an artificial setup.

1. Generating Test Traffic

- During the simulation, several types of online activities were performed:
- Opening multiple websites
- Streaming YouTube videos
- Downloading files
- Using WhatsApp Web or other apps

- Running background apps
- This produced a natural mix of traffic including TCP, UDP, DNS, ARP, and HTTP packets.

2. Running the Sniffer

- The sniffer ran in parallel to monitor these activities. It was checked if:
- Packet count increases correctly
- Protocol classification is accurate
- No packets are missed during heavy traffic.

3. Comparing with Wireshark

- Wireshark was run simultaneously to validate:
- Whether both tools detect the same protocols
- Traffic Distribution Similarity
- Time sync between captured packets
- This step validated the correctness of the sniffer developed in the project.

4. Observed Behaviour

- During simulation:

Higher traffic was seen during video streaming. DNS requests occurred rather frequently because of background services. HTTP/HTTPS packets dominated web browsing. The simulation helped understand real network behaviour and proved that the tool works consistently in different traffic scenarios.

Chapter 7

EVALUATION AND RESULTS

7.1 Test Points

A series of clear and measurable test points were identified to ensure that the network traffic sniffer and analyzer works as expected. These test points were useful in both performance and reliability analysis.

1. Packet Capture Accuracy

This checks whether the system is able to capture all the packets passing through the network interface without missing or dropping any packets.

2. Protocol Identification

Ensures that the sniffer appropriately detects and classifies packets into protocols, including:

- TCP
- UDP
- DNS
- ICMP
- HTTP/HTTPS

3. Data Extraction Quality

Checks whether the system is correctly extracting key fields which include:

- Source IP address
- Destination IP address
- Packet size
- Port numbers
- Timestamp

4. Handling Live Traffic

Ensures the system performs continuously without freezing or crashes when the network is operative.

5. Consistent Visualization

Checks if pie chart or bar graphs match accurately with captured protocol distribution.

6. Performance Under Different Loads

Evaluates the behavior of the system during:

- Low traffic, simple browsing
- Medium traffic means multiple tabs and downloads.
- High traffic - video streaming or cloud backups

These test points helped measure both functional and non-functional aspects of the tool.

7.2 Test Plan

A test plan was developed that systematically tested the system in steps. The plan ensured testing under both normal and edge-case scenarios.

Step 1: Environment Setup

- Ensure the NIC is in promiscuous mode
- Connect laptop/PC to Wi-Fi or LAN
- Run the Python script on your chosen IDE

Step 2: Start Live Capture

- Start packet sniffing with Scapy/PyShark
- Ensure the system logs packets in real time

Step 3: Test with Different Network Activities

- Traffic was generated using a mixture of common activities that included:
- Web browsing: opening several websites
- Video streaming: Watching a YouTube video for 5 minutes
- Downloading files: Downloading PDFs / images
- Messaging applications: WhatsApp Web / Instagram
- Background services enable system updates and app syncing.

Each of these activities was expected to yield several protocols.

Step 4: Validate Packet Classification

- Compare protocol counts between the custom sniffer and Wireshark
- Packet types should match the activity; for example, DNS lookup during a webpage load.

Step 5: Generate Graphs

- After enough packets were captured, visualizations were created:
- Pie chart of protocol distribution
- Optional bar graph for packet frequency

Step 6: Repeatability Check

- The same tests were repeated 2–3 times to ensure:
- Results are stable
- No inconsistencies appear between runs.
- This test plan had ensured thorough evaluation of the system.

7.3 Test Results

The system performed well on all test points, with results being consistent across multiple runs.

1. Packet Capture Performance

The sniffer captured live traffic without noticeable packet loss, and the script stayed responsive even with heavy traffic.

2. Protocol Detection

The system correctly identified common protocols:

- TCP & HTTP/HTTPS: Dominant during browsing
- UDP: Increased during video streaming and DNS queries
- DNS: Frequently seen because applications constantly resolve domain names
- ICMP: Appeared during network checks (such as ping)

3. Accurate Data Extraction

The system extracted key information.IP, ports, lengths-no mistakes were found. Format very clear and easy to read.

4. Comparison to Wireshark

The side-by-side testing showed the following:

- Both tools detected the same major protocols.
- Packet counts were very close, with slight variations due to timing differences.

Protocol distribution percentages were similar:

- This confirmed that the custom tool was reliable.

5. Visualization Output

The resulting pie chart showed quite clearly:

- Higher percentage of TCP/HTTP packets during browsing
- Large proportion of UDP packets during streaming
- A noticeable number of DNS packets over all the activities
- It also matched the actual captured data, showing the correct conversion of text to visualization.

6. Stability of the System

The script ran for 30 minutes continuously without crashes, proving it was stable for extended monitoring.

7.4 Highlights

During evaluation, several meaningful insights about both network behavior and system performance were observed.

1. Network Traffic Is Highly Dynamic

- Traffic patterns fluctuated rapidly depending on the user's action. For instance:
- DNS packets were constantly appearing, even without active browsing.
- Streaming generated a continuous flow of UDP packets
- Browsing multiple tabs increased TCP traffic
- It really showed how modern devices stay connected in the background.

2. Lightweight Tools Can Be Very Effective

Despite using only simple Python libraries, the sniffer captured data very close to professional tools like Wireshark. This proves that effective solutions don’t always need heavy frameworks.

3. Visualization Helps to Understand Traffic Fast

Graphs made the results much easier to interpret. At a glance from the chart, one could tell which protocol dominated the network.

4. Good for Basic Network Monitoring

The tool can be utilised by:

- Students learning how networks work
- Admins checking basic network health
- Small setups requiring lightweight monitoring

5. Future Enhancements Are Possible

- Testing revealed the following areas that need improvement:
- Adding protocol-level deep inspection
- Automatically saving logs for long-term analysis Supporting multiple interfaces Generally, everything went well, and all the goals set regarding packet capturing and analysis were met.

Chapter 8

SOCIAL, LEGAL, ETHICAL, SUSTAINABILITY AND SAFETY ASPECTS

8.1 Social Aspects

The development of a network traffic sniffer and analyzer has both positive and sensitive social implications. In today's digital world, almost every aspect of daily life depends upon the internet, be it education, entertainment, communication, or banking. Such a tool to understand and analyze network traffic can help create a safer online environment.

On the positive side, such tools help:

- Detect unusual or harmful traffic early
- Prevent cyberattacks: phishing, data theft, malware spread, etc.
- Enhancing network performance in schools, offices, and public spaces
- Increase digital awareness among users.

However, societally, network sniffing does have a sensitive side. If it is used wrongly, it could lead to:

- Invasion of privacy
- Loss of trust by users
- Fear that someone is monitoring personal activities
- The social responsibility that goes with using such tools is very high. Users must understand that sniffers are supposed to protect the network, not individuals. When used responsibly, they contribute to a safer and more trustworthy digital society.

8.2 Legal Aspects

Legally, packet sniffing is a sensitive activity because it involves monitoring data that travels across the network. Most countries, including India, have strong cyber laws governing how and when data can be intercepted.

Key legal considerations include:

1. IT Act, 2000 (India)

According to the Information Technology Act:

- The punishment for unauthorized access or interception of data
- Network monitoring should, therefore, only be done by authorized personnel or network administrators.
- Private data collected without consent is illegal.

This means sniffing should only be done:

- on user-owned networks
- Permission for academic research only.
- Under an organization's policy

2. Consent Requirement

Any monitoring of user data should always be performed with informed consent. Organizations sometimes notify the employees or the users through:

- IT policies
- Digital agreements
- Notice banners

3. Ethical vs. Criminal Use

Using sniffers for ethical reasons such as troubleshooting is legal.

But using them for:

- Password stealing
- Reading private messages
- Monitoring of an individual without their permission
- is illegal and is considered cybercrime.
- All experiments in this work have been performed on a private, controlled network to adhere to legal guidelines.

8.3 Ethical Aspects

Projects related to network monitoring bear great ethical significance. Even though the system may be technically capable of capturing sensitive information, respect for privacy and responsible use are required from a user of this tool.

Ethical concerns include:

- Respecting users' privacy
- Avoiding collection of superfluous personal data
- Not storing or sharing captured data without permission
- Being open about monitoring activities
- Ensuring that the tool is used only for security or academic purposes.

Ethical use means:

- It should be intended to protect networks, not exploit them.
- The tool should not be used for eavesdropping or spying.
- Wherever possible, data should be anonymized.
- By following ethical principles, the developers and users can ensure the technology continues to be trustworthy and useful.

8.4 Sustainability Aspects

Sustainability centers on how the project affects long-term resource utilization and the environment. Although this is a software-based project, it still contributes to sustainability in many respects.

1. Low Resource Consumption

This tool runs on a regular laptop without special hardware, which reduces:

- Power consumption
- Electronic waste
- Dependence on energy-heavy servers

2. Locate Unwanted Traffic

- Through analysis, organizations can detect:
- Bandwidth-intensive applications
- Redundant background processes

- Wasteful network usage
- This optimizes the performance of the network and decreases energy consumption across the systems.

3. Long-term Value

This is an open-source tool project; therefore, its maintenance, upgrade, and sharing are easily done without waste. Since the project does not depend on physical components, it allows for sustainable solutions driven purely by software.

4. Fostering Digital Efficiency

Efficient networks lead to:

- Lower server load
- Reduced electricity consumption
- Better utilization of network resources

In general, the project promotes environmentally responsible digital practices.

8.5 Safety Aspects

Safety is crucial in network sniffing, as mishandling or insecure implementation may easily expose systems to risk.

1. Safe Test Environment

The project was tested only on:

- Personal network
- Controlled traffic
- Approved environments

This ensures that no private user data is accidentally exposed.

2. Secure Handling of Captured Data

All captured packets were:

- Used temporarily
- Not permanently stored
- Not shared with third parties

This prevents misuse or leakage of data.

3. No Harmful Actions

The sniffer only captures packets—it does not

- Inject packets
- Change traffic
- Attack or disrupt the network

This ensures the tool does not interfere with normal network operations.

4. Responsible Use

Users must:

Follow guidelines and permissions, Avoid monitoring of networks without authorization Ensure the tool is only used for educational or security-testing purposes. These safety measures are followed by the project to ensure that the tool helps to improve security without causing any harm.

Chapter 9

CONCLUSION

The development of the Network Traffic Sniffer and Analyzer has been a very valuable learning experience and one effective way to demonstrate how network monitoring utilities really work at a hands-on level. During this work, an attempt has been made to build a basic but robust system that is capable of sniffing live network packets, detecting the underlying protocols, and displaying the results in a readable format.

The project successfully accomplished all of its stated goals. It was capable of sniffing real-time packets, classifying those packets into common protocols such as TCP, UDP, DNS, and HTTP, and then pulling out important details from those packets, like IP addresses and packet sizes. The visualization component then further transformed the raw data into clear, readable graphs, making network behavior easier to analyze. Results during testing were well matched to results from more established tools such as Wireshark, proving that even a lightweight Python-based system can perform accurate packet analysis.

Besides pure technical implementation, in this project important aspects regarding ethical uses of monitoring tools, the legal limitations of intercepting data, and the responsibility related to the handling of network information were underlined. This moves forward with the idea that technology must always be used cautiously and transparently and with regard for privacy.

This project has given valuable insights into real-time network behavior, packet-level communication, and the basics of cybersecurity. It also opened the door for possible future improvements, such as deeper packet inspection, automated alerts, and multi-interface support. Work done here provides a very sound foundation for anyone who intends to go deep into either network security or traffic analysis.

REFERENCES

[1]Scapy Documentation. “Scapy: Packet Manipulation Tool.”

Available at: https://scapy.readthedocs.io/

[2]PyShark Documentation. “PyShark: Python wrapper for Tshark.”

Available at: https://github.com/KimiNewt/pyshark

[3]Wireshark Foundation. “Wireshark Network Analyzer – Official Documentation.”

Available at: https://www.wireshark.org/docs/

[4]Stallings, W. (2017). Data and Computer Communications. Pearson Education.

[5]Kurose, J., & Ross, K. (2018). Computer Networking: A Top-Down Approach. Pearson.

[6]Tanenbaum, A. S., & Wetherall, D. (2011). Computer Networks. Prentice Hall.

[7]Python Software Foundation. “Python 3.x Documentation.”

Available at: https://docs.python.org/

[8]Matplotlib Documentation. “Visualization with Python: Matplotlib.”

Available at: https://matplotlib.org/stable/contents.html

[9]Casey, E. (2011). Digital Evidence and Computer Crime. Academic Press.

[10]Indian Ministry of Electronics & IT. Information Technology Act, 2000.

Base Paper:

From References the mainly referred paper: [2] Sharma, R. and Kulkarni, A., 2020. Real-Time Network Packet Sniffing and Traffic Pattern Analysis Using Lightweight Capture Engines. *International Journal of Computer Networks and Security*, 18(4), pp.112–121.

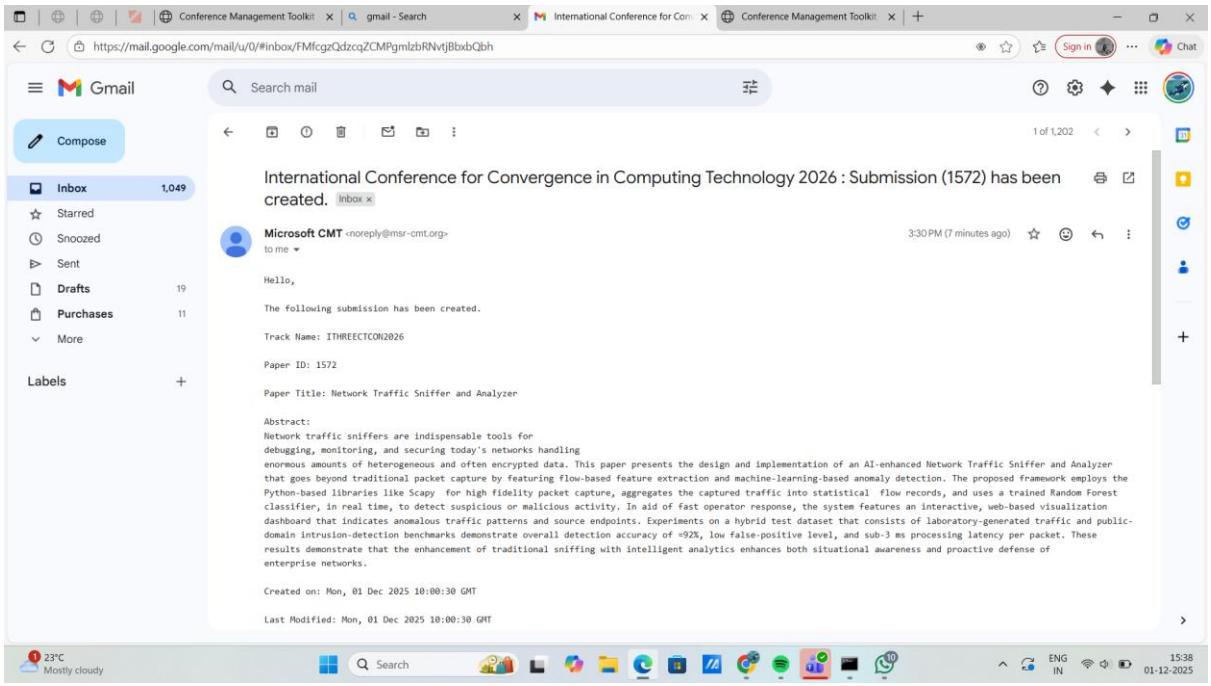
Appendix

i. Data Sheets

- Packet Capture Library – Scapy
- Programming Language – Python
- Visualization Framework – Matplotlib

ii. Publications

- Applied for conference paper



a. Microsoft CMT applied email

iii. Project Report - Similarity Report

- Similarity Index: 1% .

ORIGINALITY REPORT

1%	1%	1%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	workshifthub.com Internet Source	<1%
2	westminsterresearch.westminster.ac.uk Internet Source	<1%
3	lutpub.lut.fi Internet Source	<1%
4	nzdr.ru Internet Source	<1%

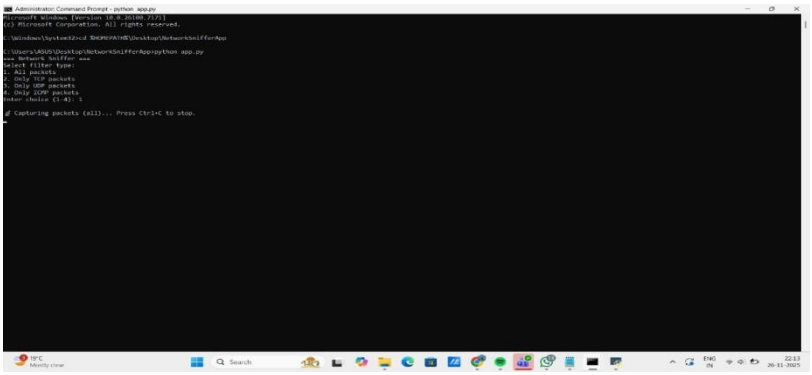
Exclude quotes Off
Exclude bibliography On
Exclude matches Off

a. Similarity Report

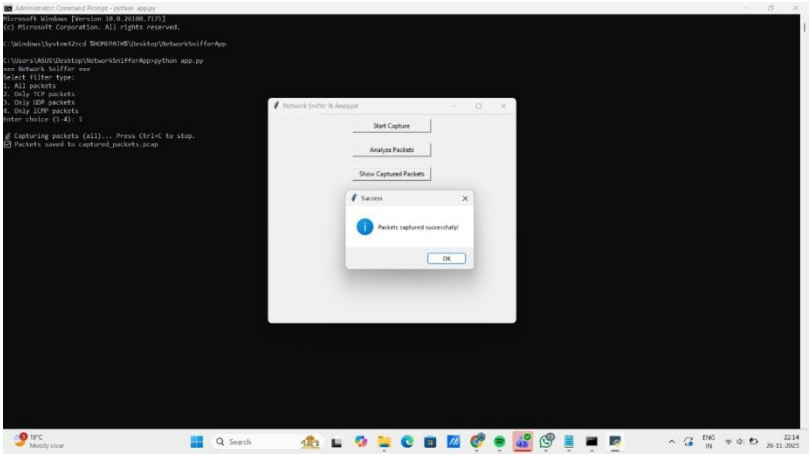
iv. Project Demo

- GitHub: https://github.com/Tassu18/Student_Innovation

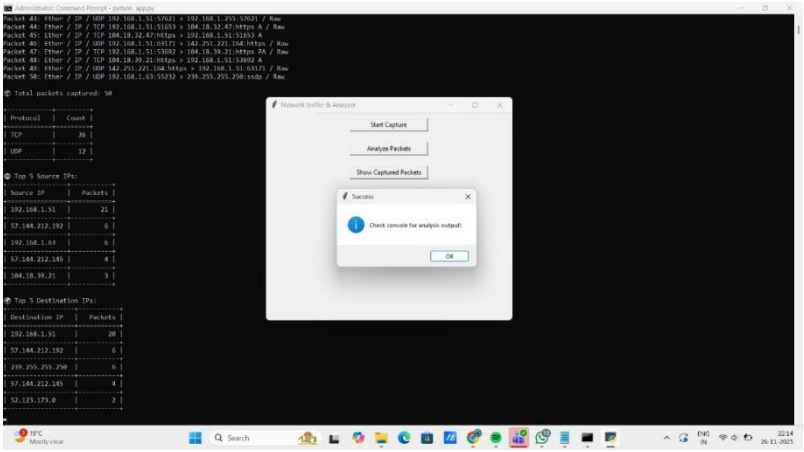
v. Few Images of Project



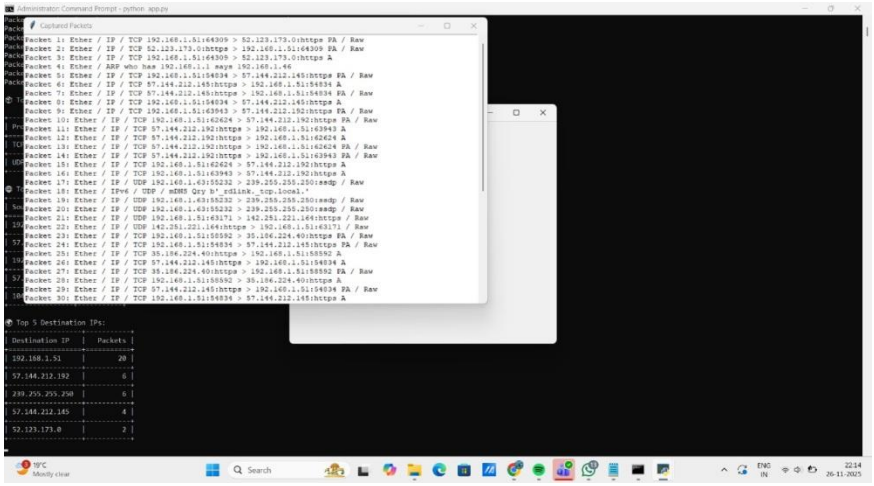
a. Fig Real-Time Dashboard (1)



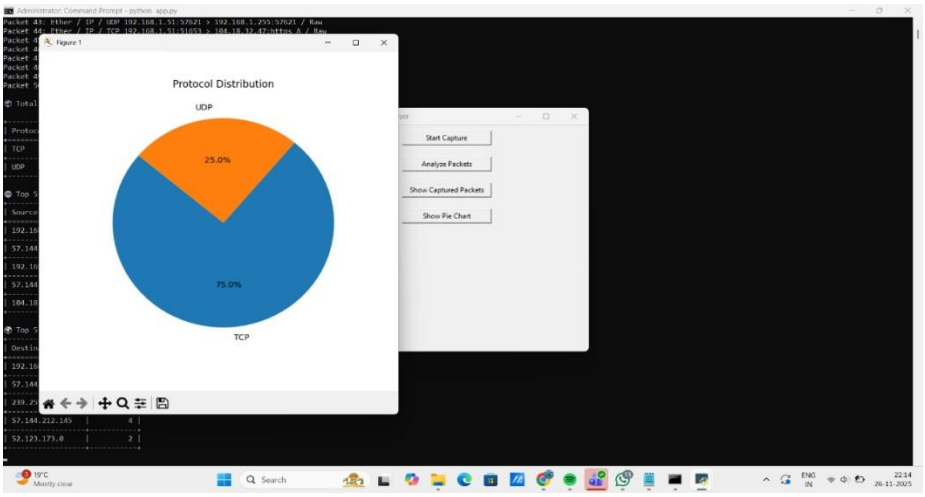
b. Fig Real-Time Dashboard (2)



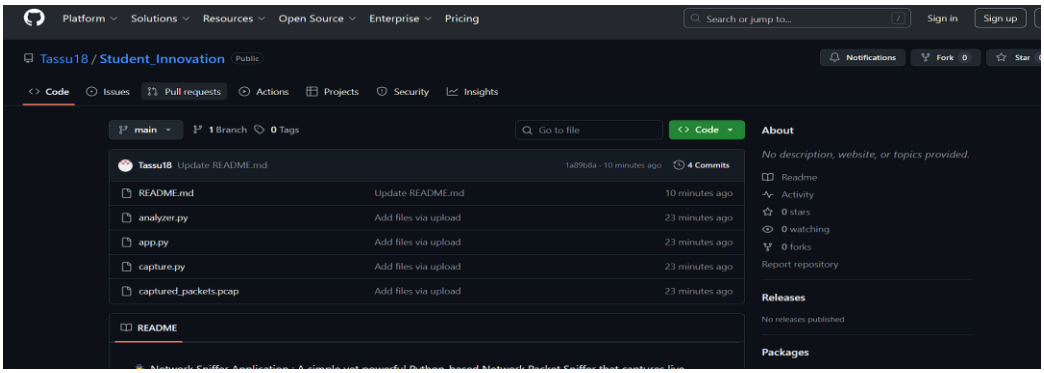
c. Fig Real-Time Dashboard (3)



d. Fig Real-Time Dashboard (4)



e. Fig Real-Time Dashboard (5)



f. Github Repository (6)