****

**PROJECT REPORT**

**ON**

**“ScapyUI – Network Packet Sniffer”**

**Submitted To**

**School of Cyber Security & Digital Forensics,**

**National Forensic Sciences University**

**For partial fulfilment for the award of degree**

**MASTER OF SCIENCE**

**In**

**DIGITAL FORENSIC AND INFORMATION SECURITY**

**Submitted By**

**Yash Ashwin Bhai Rana**

**(240347003003)**

**Under the Supervision of**

**Mr. Harsh Panchal (School of Cyber Security and Digital Forensics)**

**National Forensic Sciences University,**

**Goa Campus, Near Goa Dairy, Curti, Ponda, Goa - 403401.**

**March, 2025**

**DECLARATION**

I certify that

a. The work contained in the dissertation is original and has been done by myself under the supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

d. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

e. Whenever I have quoted written materials from other sources and due credit is given to the sources by citing them.

f. From the plagiarism test, it is found that the similarity index of whole dissertation within 25% and single paper is less than 10 % as per the university guidelines.

**Date:**

**Place:**

**Yash Rana**

**Enroll. No.: 240347003003**

**CERTIFICATE**

This is to certify that the work contained in the dissertation entitled **“ScapyUI – Network Packet Sniffer”**, submitted by **Yash Rana (Enroll. No.: 240347003003)** for the award of the degree of **Master of Science in Digital Forensic and Information Security** to the **National Forensic Sciences University**, **Goa Campus**, is a record of bonafide research works carried out by him under my direct supervision and guidance.

**Date:**

**Place:**

**Mr. Harsh Panchal**

**Designation,**

**School of Cyber Security & Digital Forensics,**

**National Forensic Sciences University,**

**Goa Campus, Ponda, Goa, India.**

**ACKNOWLEDGEMENTS**

First and foremost, I express my sincere gratitude to my supervisor, Mr. Harsh Panchal, for his invaluable guidance, constant support, and insightful feedback throughout the course of my dissertation. His expertise, patience, and encouragement have been instrumental in shaping the direction and successful completion of this work.

I would like to extend my heartfelt thanks to the Head of the Department, whose leadership and academic vision have fostered an environment conducive to learning and research. I also wish to acknowledge the faculty members of the department for imparting their knowledge and for their consistent encouragement during my academic journey.

My sincere thanks go to the technical staff and lab assistants for their timely help and support in facilitating various aspects of my practical work. I am also grateful to my lab mates and fellow classmates for their cooperation, suggestions, and the positive environment we shared during the research period.

Finally, I would like to acknowledge the support and motivation received from my friends and family, whose constant encouragement helped me stay focused and motivated.

This dissertation would not have been possible without the contributions of all the individuals mentioned above, and I remain deeply thankful to each of them.

With Sincere Regards,

**Yash Rana**

**Digital Forensics and Information Security**

**ABSTRACT**

This dissertation presents **ScapyUI**, a graphical user interface (GUI)-based network packet sniffer and analyser developed using **Python's Tkinter** and **Scapy** libraries. The primary objective of the project is to create an intuitive and effective tool for real-time packet monitoring, analysis, and forensic investigation. The tool is designed to aid cybersecurity professionals, digital forensic analysts, and network administrators by providing them with a compact yet capable solution for capturing and analysing network traffic.

The project encompasses a multi-panel interface that displays categorized packet data, including **basic headers**, **raw payloads**, **HTTP data**, and **DNS traffic**. It is equipped to support both **IPv4 and IPv6** protocols. An essential feature of the tool is the ability to read, interpret, and display **.pcap files** through a dedicated GUI window, allowing users to perform offline analysis of recorded network traffic. A functionality to **save captured data logs in PDF format** was integrated, enhancing reporting and documentation capabilities.

The proposed methodology includes live packet capture using Scapy’s sniff () function, parsing of layer-specific information, and real-time display within categorized panels using Tkinter’s ScrolledText widgets. A modular design ensures separation of concerns for improved maintainability and future scalability.

Findings indicate that ScapyUI performs efficiently in capturing and categorizing network traffic in real-time and presents an accessible interface for forensic interpretation. The project also demonstrates that open-source tools can be effectively leveraged to build customizable, platform-independent cybersecurity applications without reliance on expensive commercial software.

Future aspects include implementing packet filtering options, automatic threat detection based on packet patterns, and integration with cloud storage or centralized databases for collaborative analysis and long-term storage. The tool lays a strong foundation for further development in educational, research, and operational security environments.

**Keywords**: ScapyUI, Network Sniffer, Packet Analysis, PCAP Reader, Python, Tkinter, Scapy, Cyber Forensics, GUI Tool, Live Traffic Capture

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| GUI | Graphical User Interface |
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| ICMP | Internet Control Message Protocol |
| DNS | Domain Name System |
| PCAP | Packet Capture |
| IPv4 | Internet Protocol Version 4 |
| IPv6 | Internet Protocol Version 6 |
| OSI | Open Systems Interconnection |
| MAC | Media Access Control |
| UI | User Interface |
| SOC | Security Operations Center |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Description** | **Page No** |
| Table 2.4 | Tools/Technology Required |  |
| Table 3.3.2 | Data Dictionary |  |
| Table 4.3 | Laboratory Setup |  |
| Table 4.4 | Tools and Technology Used |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig No** | **Figure Description** | **Page No** |
| Figure 1 | Use Case Diagram |  |
| Figure 2 | Activity Diagram |  |
| Figure 3 | Sequence Diagram |  |
| Figure 4 | Entity Relationship Diagram |  |
| Figure 5 | Class Diagram |  |
| Figure 6 | Level 0 Data Flow Diagram |  |
| Figure 7 | Level 1 Data Flow Diagram |  |
| Figure 8 | Level 2 Data Flow Diagram |  |

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **Screenshot No** | **Screenshot Description** | **Page No** |
| Screenshot 1 | Main GUI Interface |  |
| Screenshot 2 | Live Packet Capture |  |
| Screenshot 3 | PCAP File Reader Window |  |
| Screenshot 4 | PDF Export Output |  |

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acknowledgement | | | | III |
| Abstract | | | | IV |
| Abbreviations | | | | V |
| List of Tables | | | | VI |
| List of Figures | | | | VII |
| List of Screenshots | | | | VIII |
| List of Symbols | | | | IX |
| **Chapter 1.** | **Introduction** | | | **1-X** |
|  | 1.1 | Introduction and Problem Summary | |  |
|  | 1.2 | Aim and Objectives of the Project | |  |
|  | 1.3 | Scope of the Project | |  |
| **Chapter 2.** | **Literature Survey** | | | **2-X** |
|  | 2.1 | Current/Existing System | |  |
|  |  | 2.1.1 | Study of Current System |  |
|  |  | 2.1.2 | Problem & Weakness of Current System |  |
|  | 2.2 | Requirements of New System | |  |
|  | 2.3 | Feasibility Study | |  |
|  |  | 2.3.1 | Technical Feasibility |  |
|  |  | 2.3.2 | Operational Feasibility |  |
|  | 2.4 | Tools/Technology Required | |  |
| **Chapter 3.** | **Design: Analysis, Design Methodology and Implementation Strategy** | | | **3-X** |
|  | 3.1 | Function of System | |  |
|  |  | 3.1.1 | Use Case Diagram |  |
|  |  | 3.1.2 | Activity Diagram |  |
|  |  | 3.1.3 | Sequence Diagram |  |
|  | 3.2 | Data Modelling | |  |
|  |  | 3.2.1 | Entity-Relationship Diagram |  |
|  |  | 3.2.2 | Class Diagram |  |
|  | 3.3 | Functional & Behavioural Modelling | |  |
|  |  | 3.3.1 | Data Flow Diagram |  |
|  |  | 3.3.2 | Data Dictionary |  |
| **Chapter 4.** | **Implementation** | | | **4-X** |
|  | 4.1 | Implementation Environment | |  |
|  |  | 4.1.1 | Model Used in Developing |  |
|  |  | 4.2.2 | Software Prototyping Types |  |
|  | 4.2 | Coding Standard | |  |
|  | 4.3 | Laboratory Setup | |  |
|  | 4.4 | Tools and Technology Used | |  |
|  | 4.5 | Screenshots/Snapshots | |  |
| **Chapter 5.** | **Summary of Results and Future Scope** | | | **5-X** |
|  | 5.1 | Advantages/Unique Features | |  |
|  | 5.2 | Results and Discussions | |  |
|  | 5.3 | Future Scope of Work | |  |
| **Chapter 6.** | **Conclusion** | | | **6-X** |
| **Bibliography- List of references** | | | |  |
| **Appendices (if Any)** | | | | I |
| **List of Publications/Online Reference/etc..** | | | | II |

**INTRODUCTION**

**1.1 Introduction and Problem Summary**

In the current digital era, where cyber threats and malicious network activities are continuously evolving, real-time network traffic monitoring has become essential for ensuring cybersecurity and network performance. Tools like Wireshark offer comprehensive capabilities for packet analysis but may present complexity and steep learning curves for beginners and professionals who seek simplicity and lightweight solutions.

There is a growing need for a user-friendly, GUI-based network sniffer tool that enables basic and advanced network traffic analysis with minimal configuration. Most traditional packet sniffers operate through command-line interfaces or require deep knowledge of network protocols, making them less approachable for learners, junior security analysts, or incident response teams seeking fast insights.

This project addresses this gap by introducing **ScapyUI**, a Python-based GUI tool that utilizes the power of the Scapy library for packet sniffing while offering a clean and interactive user interface built with Tkinter. It allows users to capture live network traffic, analyze protocol-specific data (like HTTP, DNS, TCP/UDP), read .pcap files, and store logs efficiently for documentation or forensic purposes.

**1.2 Aim and Objectives of the Project**

**Aim:**

To develop an interactive and user-friendly GUI-based packet sniffer using Scapy and Tkinter that enables effective network traffic monitoring and protocol-level packet analysis.

**Objectives:**

* To implement real-time packet sniffing with support for TCP, UDP, ICMP, DNS, and HTTP protocols.
* To create a GUI using Tkinter for displaying live and previously captured packet data.
* To enable users to open and analyze .pcap files in a new GUI window.
* To integrate the option to save logs and extracted data as PDF for reporting or auditing.
* To enhance accessibility for learners and professionals without requiring command-line expertise.

**1.3 Scope of the Project**

This project focuses on creating a lightweight desktop application capable of performing fundamental and advanced packet analysis with an easy-to-use GUI. It provides:

* Real-time network traffic capture and classification.
* Parsing and display of HTTP, DNS, TCP/UDP, and ICMP/ICMPv6 traffic.
* Compatibility with IPv4 and IPv6 protocols.
* PCAP file parsing and viewing capabilities.
* PDF log export feature for sharing and documentation.
* Open-source accessibility for further academic or commercial enhancement.

This project is ideal for educational institutions, network security learners, forensic investigators, and SOC analysts needing quick traffic insights without complex configurations or setups.

**LITERATURE SURVEY**

**2.1 Current/Existing System**

**2.1.1 Study of Current System**

Several network monitoring and packet analysis tools exist in the cybersecurity ecosystem, with **Wireshark**, **tcpdump**, and **Ettercap** being among the most widely used.

* **Wireshark** is a feature-rich GUI-based network protocol analyser used by professionals for deep packet inspection.
* **tcpdump** is a powerful command-line tool for network packet analysis and capturing, preferred by system administrators and developers for its lightweight performance.
* **Ettercap** is popular for MITM (Man-in-the-Middle) attacks and LAN sniffing.

These tools provide advanced capabilities for dissecting network packets and troubleshooting, but they often require users to possess significant domain knowledge, and some of them are resource-intensive for simple or targeted use cases.

**2.1.2 Problem & Weakness of Current System**

While existing systems like Wireshark are comprehensive, they come with the following challenges:

* **Complex UI and information overload**, making them less beginner-friendly.
* **Heavyweight applications** that may be unsuitable for low-resource systems.
* **Steep learning curve** for students or professionals new to packet analysis.
* **Limited customization** for adding specific protocol filters or lightweight features.
* **Lack of simplified logging or PDF report export** for basic audit or incident response use.

**2.2 Requirements of New System**

The proposed system, **ScapyUI**, aims to address the weaknesses of the current systems by offering:

* A **lightweight GUI** interface using Tkinter, easily deployable on any OS supporting Python.
* Real-time traffic sniffing and **filtering of important protocols** like TCP, UDP, HTTP, DNS, and ICMP.
* **Detailed packet view** in readable format without overwhelming the user.
* A **.pcap file reader**, allowing previously captured packets to be viewed in a user-friendly format.
* Ability to **export traffic logs as PDF** for documentation or forensics.
* Support for both **IPv4 and IPv6**, enhancing compatibility with modern networks.

**2.3 Feasibility Study**

**2.3.1 Technical Feasibility**

The system is technically feasible because:

* It is built with **Python**, an open-source and platform-independent language.
* The project uses **Scapy**, a powerful Python library for packet manipulation.
* The GUI is developed with **Tkinter**, a standard Python module with minimal dependencies.
* It does not require any commercial or licensed software.
* It can run on systems with modest hardware specifications.

**2.3.2 Operational Feasibility**

From an operational standpoint:

* The tool is user-friendly, making it **easy to adopt by students and security professionals**.
* It supports multiple use cases, including **live sniffing**, **pcap analysis**, and **report generation**.
* It reduces dependency on complex tools and supports **rapid learning and analysis**.
* The open-source nature allows users to customize or expand features as needed.

**2.4 Tools/Technology Required**

|  |  |
| --- | --- |
| **Tool/Technology** | **Purpose** |
| **Python 3.x** | Base programming language |
| **Scapy** | Packet sniffing and manipulation |
| **Tkinter** | GUI development |
| **PIL (Pillow)** | For image handling (if logos or images are used) |
| **reportlab** | PDF generation |
| **OS / Platform** | Windows/Linux (cross-platform compatibility) |
| **rdpcap (Scapy)** | To read .pcap files |
| **threading** | For handling live sniffing without freezing the UI |

**DESIGN: ANALYSIS, DESIGN METHODOLOGY AND IMPLEMENTATION STRATEGY**

**3.1 Function of System**

The ScapyUI Network Sniffer tool is designed to provide a simplified interface for network packet capturing, analysis, and visualization. The core functionalities include:

* Real-time packet sniffing
* Viewing packet details in a user-friendly GUI
* Reading and parsing .pcap files
* Exporting logs to PDF
* Viewing protocols like TCP, UDP, DNS, HTTP, and ICMP

**3.1.1 Use Case Diagram**

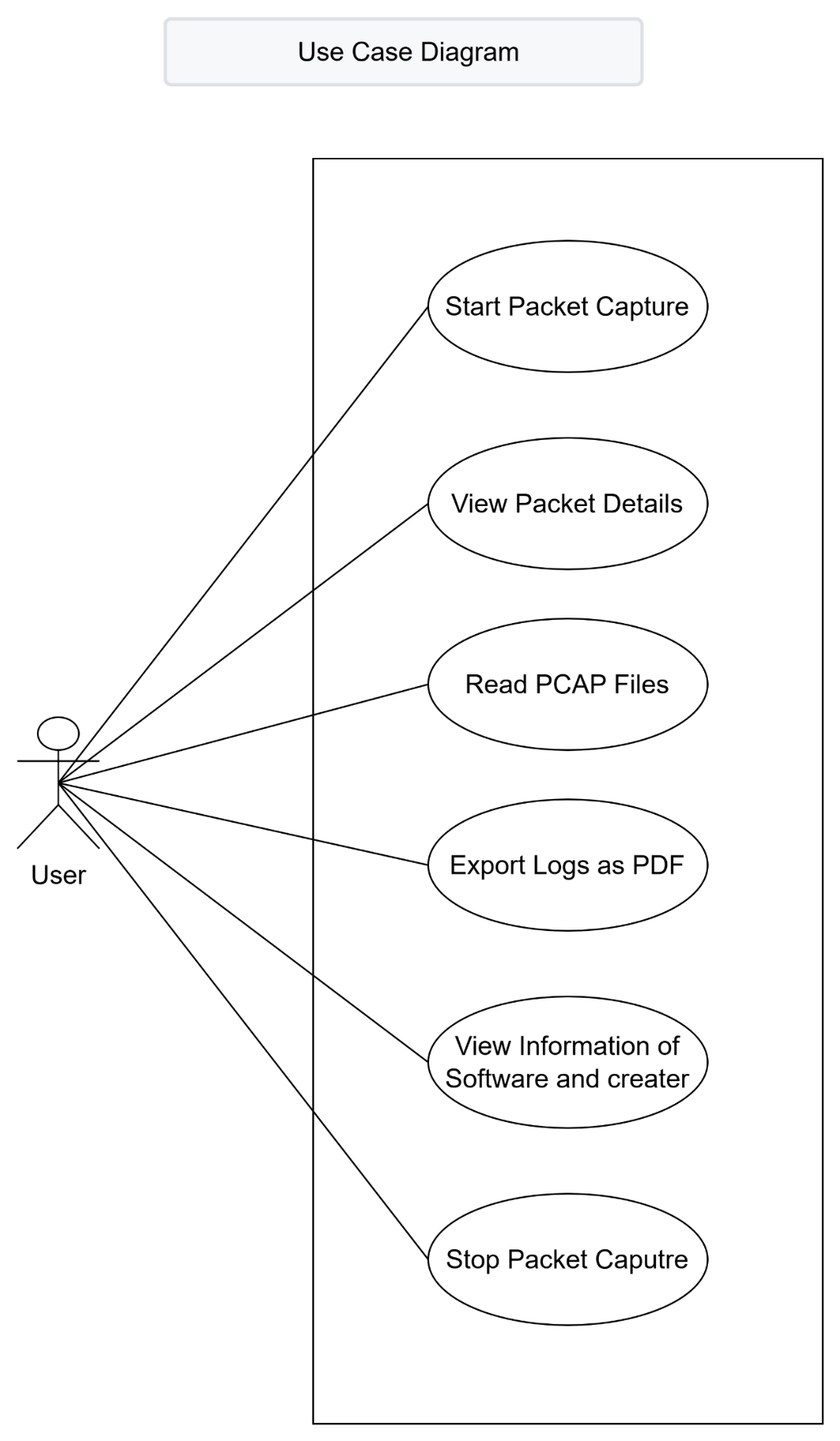
The Use Case Diagram outlines the high-level interactions between the user and the system. It identifies the main functionalities available to the user such as starting packet sniffing, stopping it, reading a .pcap file, and exporting the data to a PDF. This diagram helps in visualizing the system’s functionality from the end-user’s perspective and supports requirement gathering and validation.

**Actors:**

* User

**Use Cases:**

* Start Packet Capture
* View Packet Details
* Read PCAP File
* Export Logs to PDF
* View About Info
* Stop Packet Capture

Fig 1 Use Case Diagram

**3.1.2 Activity Diagram**

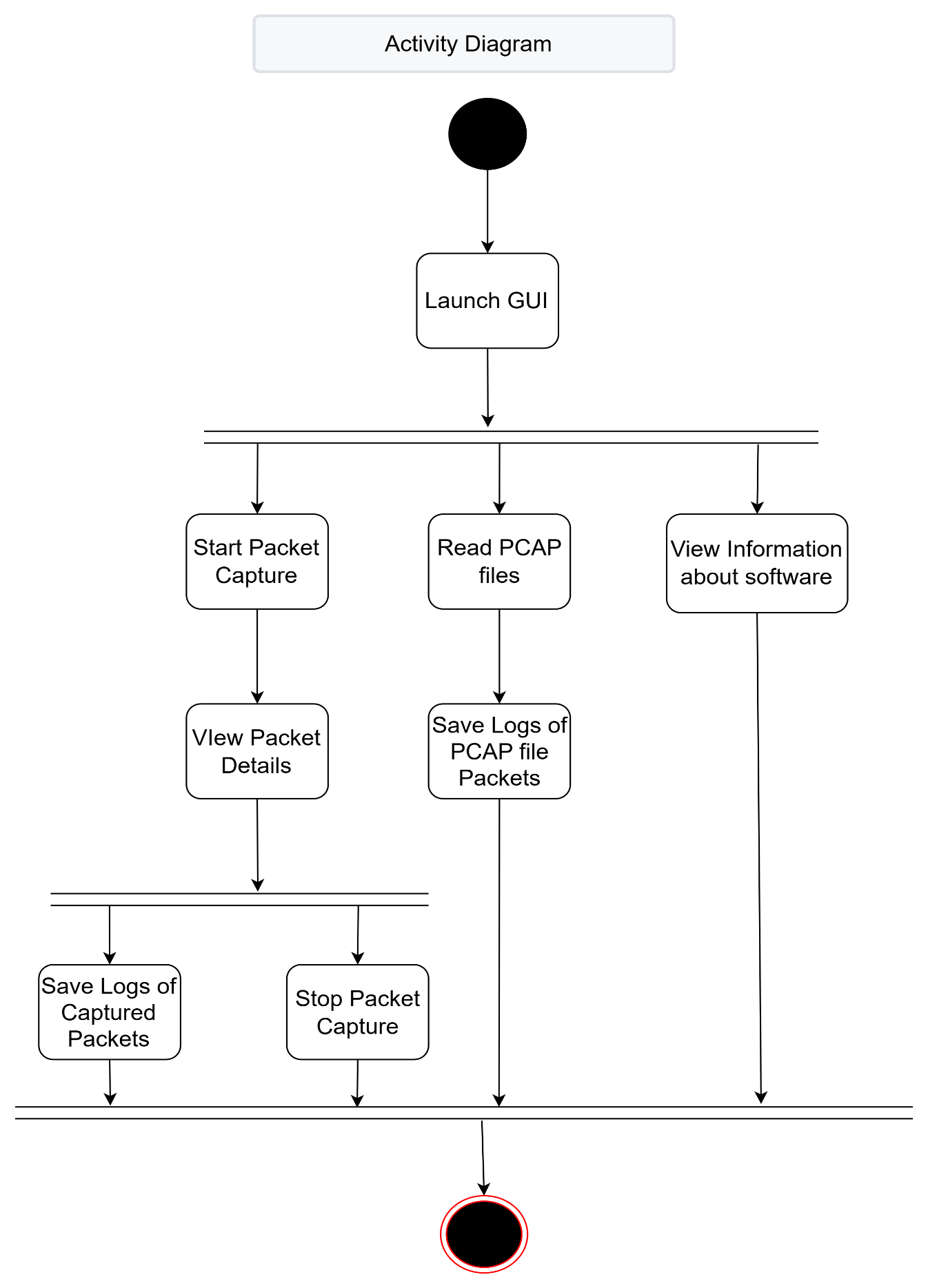
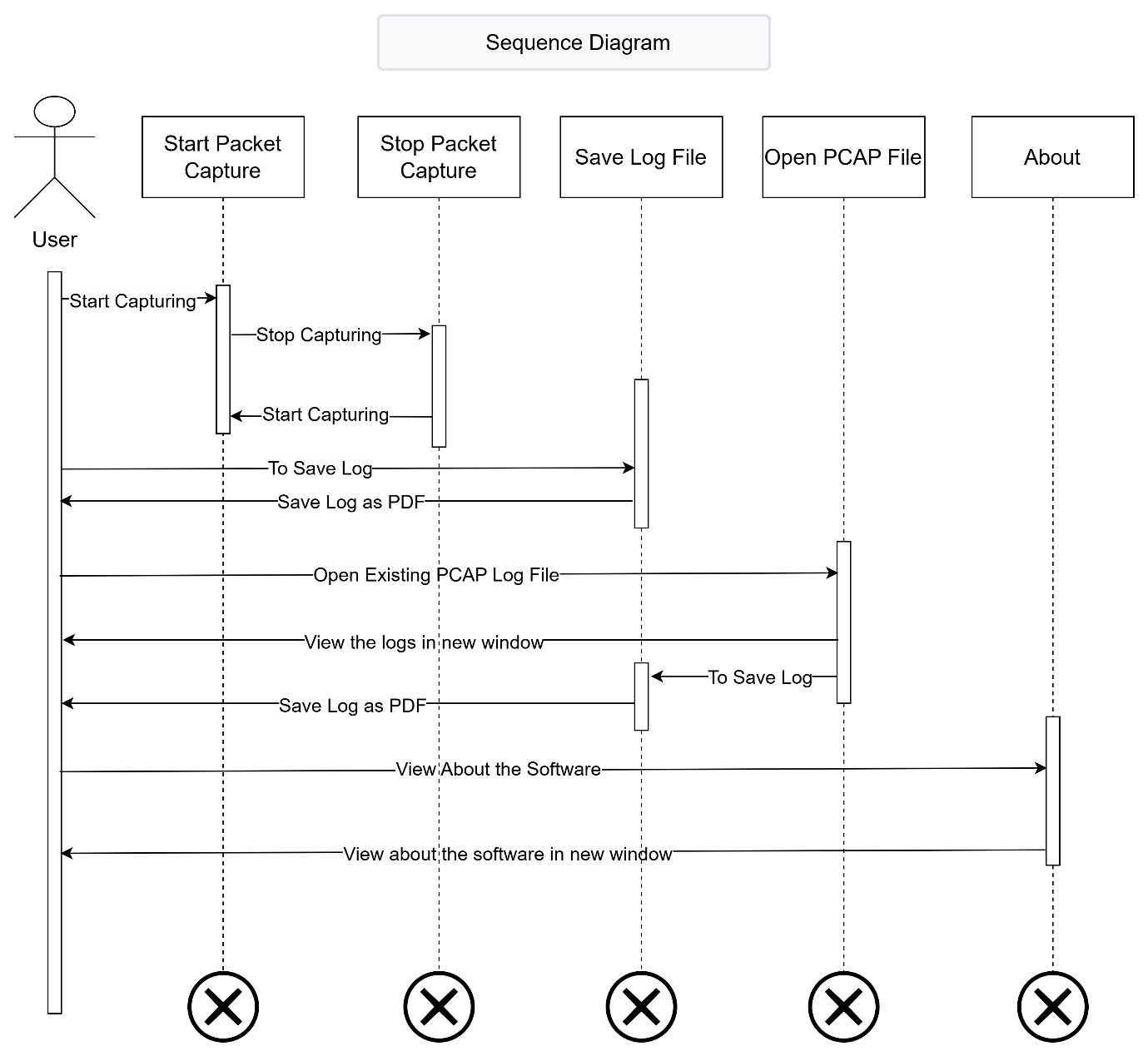
The Activity Diagram represents the flow of control within the system. It showcases the sequence of operations from launching the application, choosing an action (such as sniffing or opening a file), and viewing or exporting packet information. This diagram is useful for understanding the workflow and logic that the system follows in response to user actions.

Fig 2 Activity Diagram

**3.1.3 Sequence Diagram**

The Sequence Diagram demonstrates how different components of the system interact over time. It shows the interaction between the user interface, packet sniffer, .pcap reader, and PDF exporter classes. The diagram emphasizes the order of message exchanges, making it easier to understand object collaboration during runtime.

Fig 3 Sequence Diagram

**3.2 Data Modelling**

The tool works with structured packet data. While it does not connect to a database, internal object modelling helps with capturing, storing, and displaying packets.

**3.2.1 Entity-Relationship Diagram**

Although the system does not utilize a traditional database, the ER Diagram logically represents the relationship among key entities such as the User Action, Packet, Packet Details, and Export File. It helps to visualize how user interactions generate packets, which in turn contain details and can be exported.

Since there’s no database, showing logical entities like:

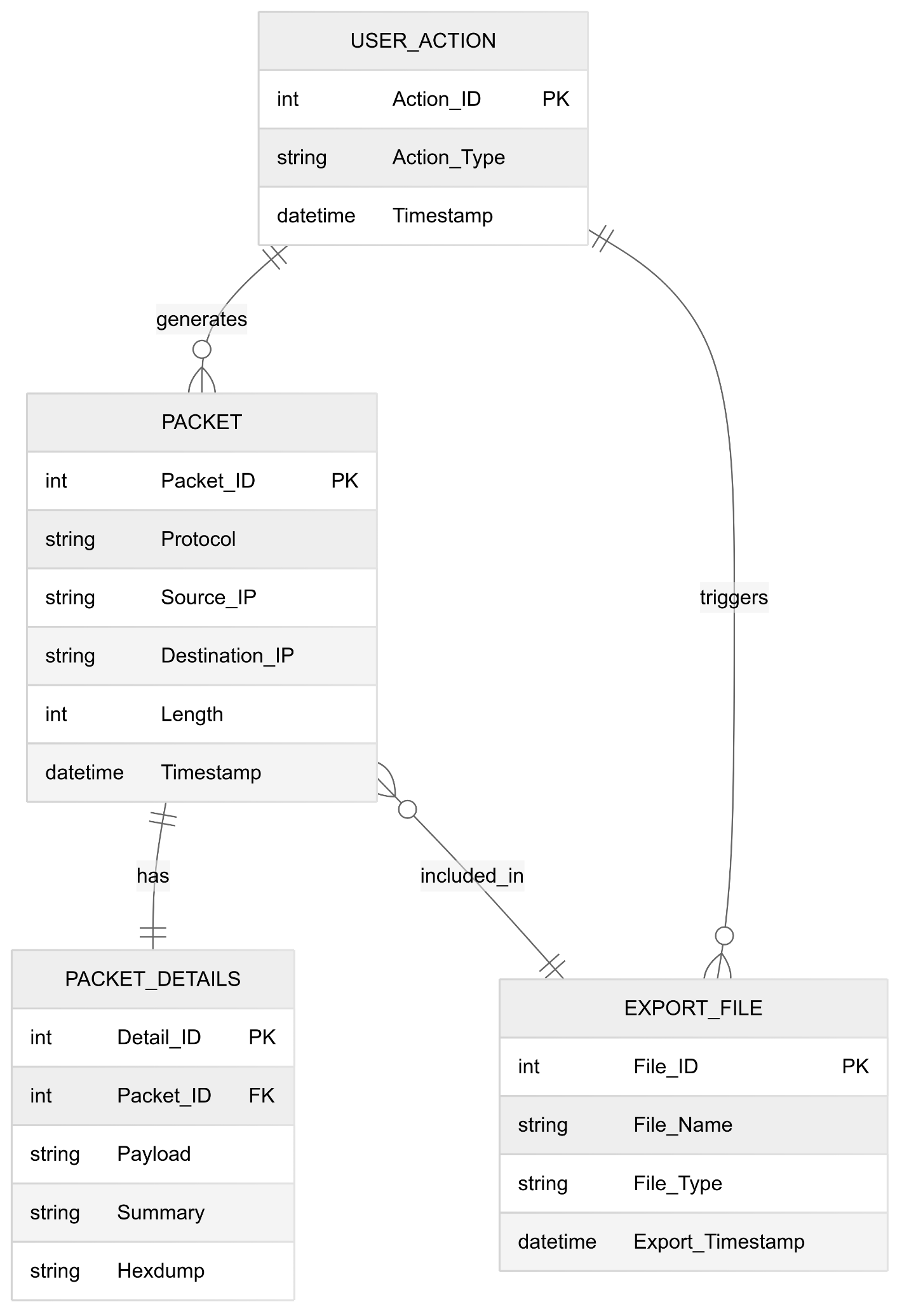
* **User Action**
* **Packet**
* **Packet Details**
* **Export File**

Fig 4 Entity Relationship Diagram

**3.2.2 Class Diagram**

The Class Diagram presents the structural design of the system in terms of classes and their relationships. Major classes include PacketSniffer, GUIHandler, PCAPReader, and PDFExporter, each encapsulating specific responsibilities like capturing, displaying, reading, and exporting packet data. This diagram provides insight into object-oriented architecture and code maintainability.

**Main Classes:**

* PacketSniffer
* GUIHandler
* PCAPReader
* PDFExporter

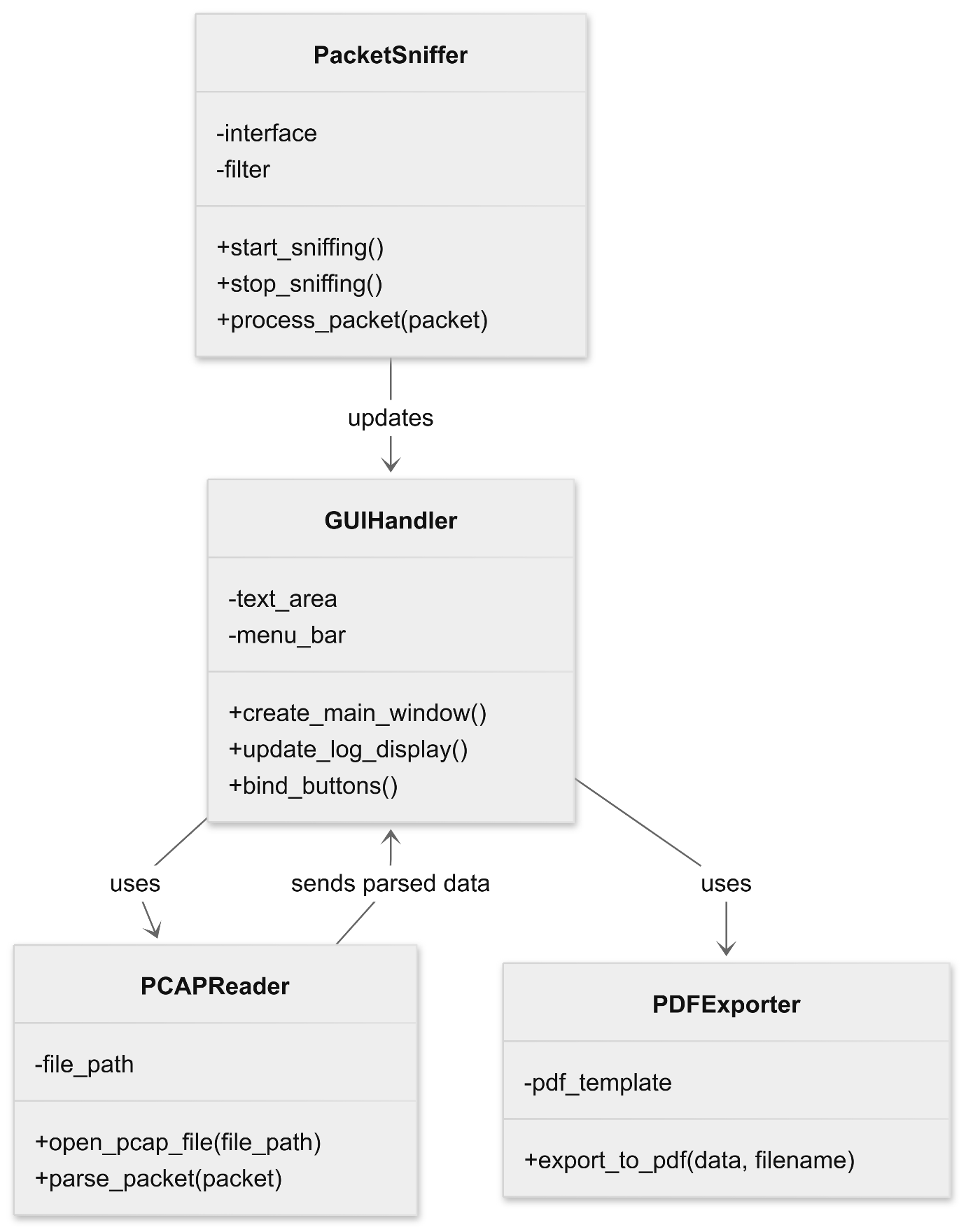
These classes encapsulate functionality for sniffing, GUI management, pcap reading, and export.

Fig 5 Class Diagram

**3.3 Functional & Behavioural Modelling**

**3.3.1 Data Flow Diagram**

**Level 0 DFD:**

The Level 0 DFD, also known as the context diagram, provides a top-level overview of the entire system. It identifies the system as a single process and shows the main external entities (like the user) and the data flowing between the system and those entities.

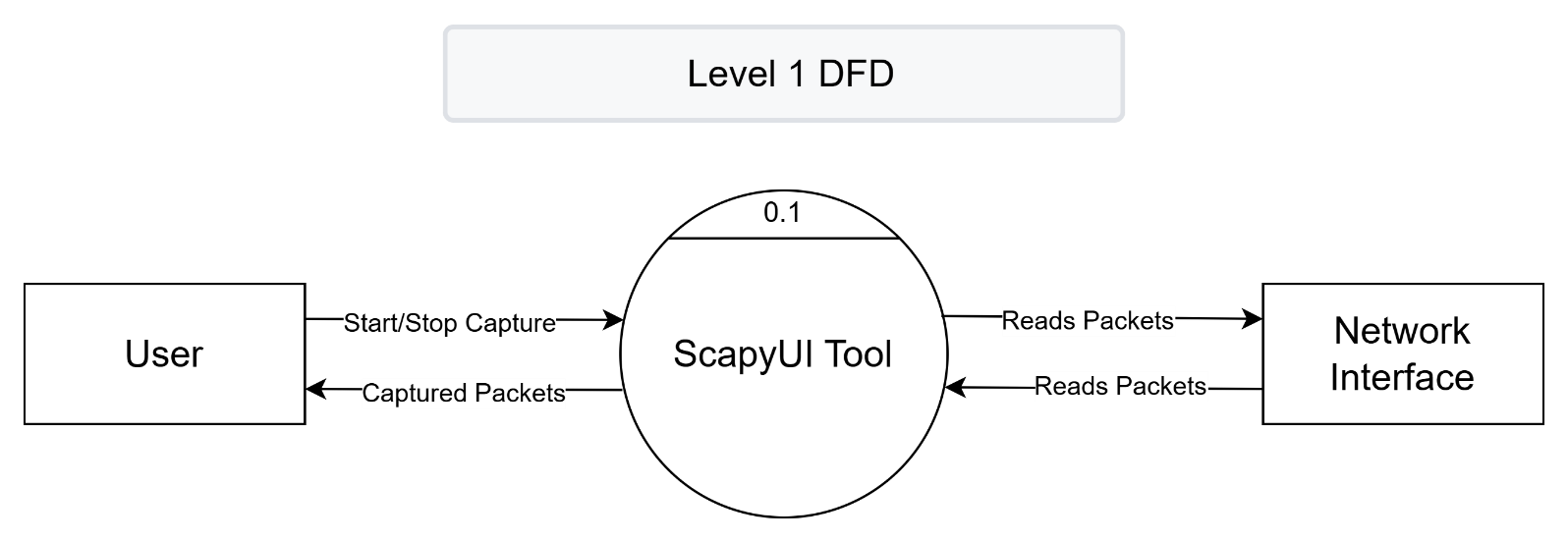
* User → GUI → Packet Sniffer → Display/Save/Export

Fig 6 Level 0 Data Flow Diagram

* GUI has multiple modules: Live Capture, PCAP Reader, Packet Detail Viewer, Export Handler

**Level 1 DFD:**

This diagram expands on the Level 0 DFD by breaking the main process into sub-processes such as packet capturing, packet display, and file exporting. It shows how data moves through different parts of the system and highlights intermediate data stores and transformations.

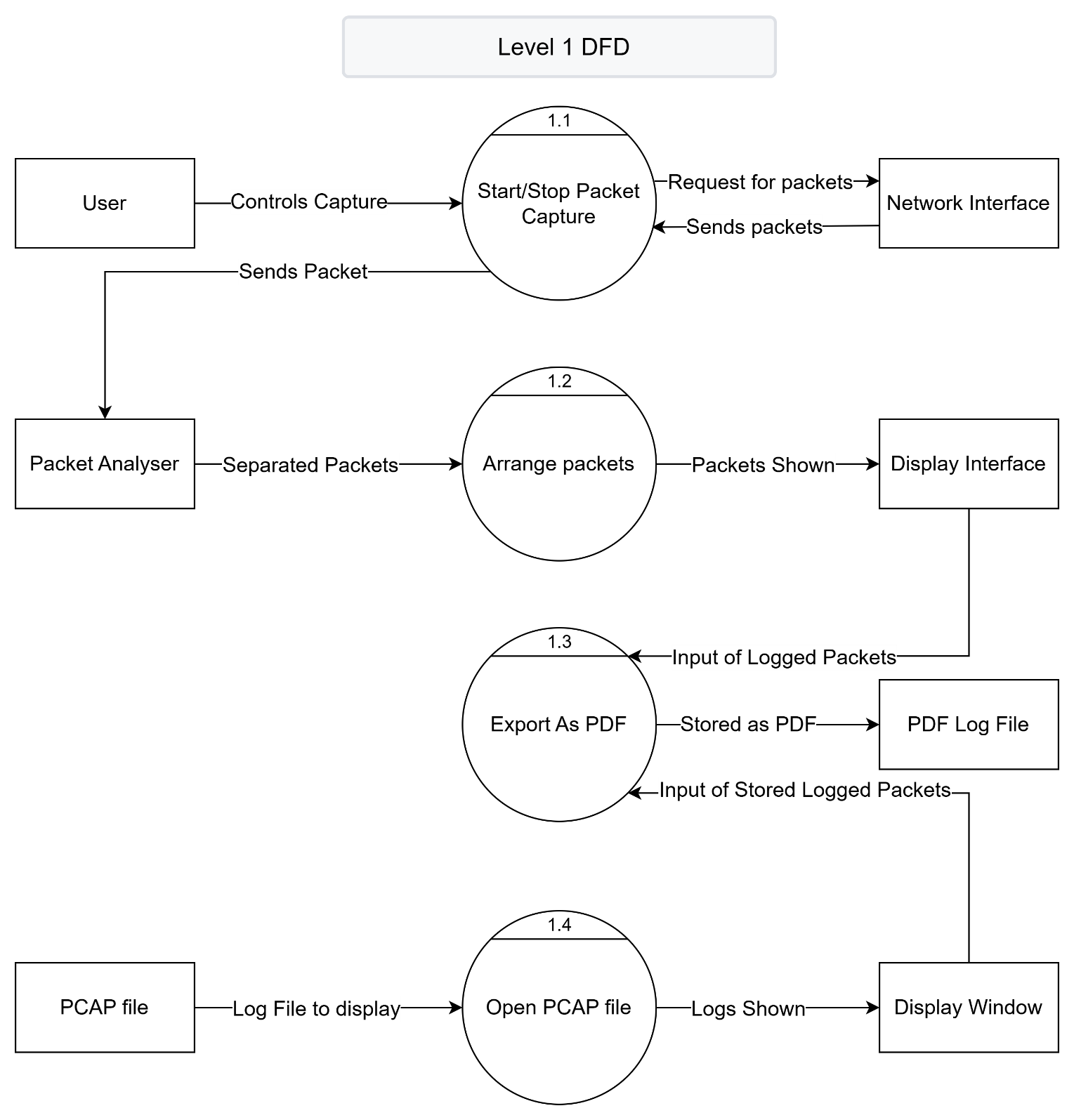
* GUI has multiple modules: Live Capture, PCAP Reader, Packet Detail Viewer, Export Handler

Fig 7 Level 1 Data Flow Diagram

**Level 2 DFD:**

The Level 2 DFD provides a more detailed decomposition of the sub-processes shown in Level 1. It delves into specific data operations such as filtering packets, extracting HTTP/DNS data, and formatting export files. This level of detail is particularly helpful for developers to understand internal data flows and processing logic.

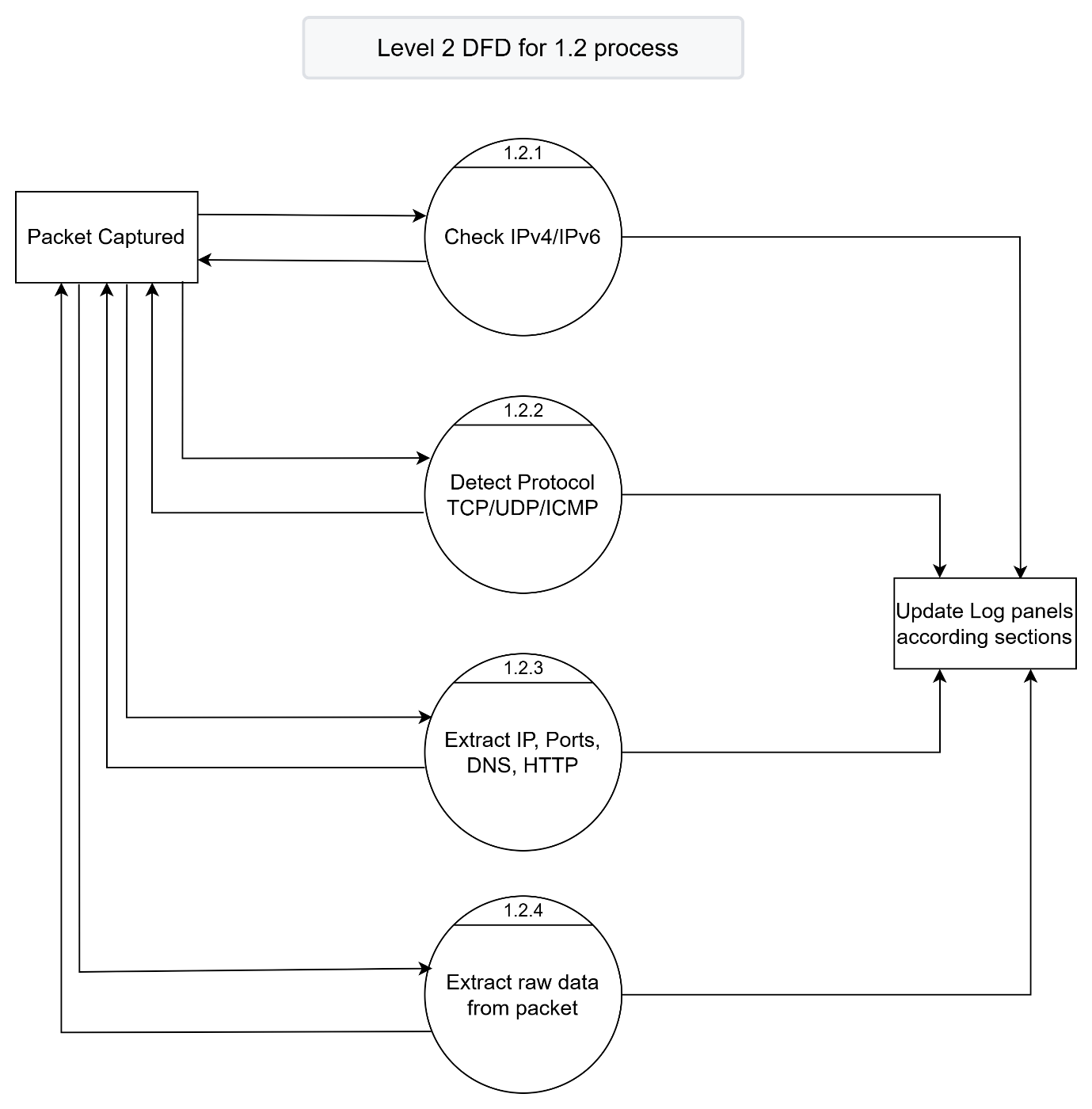


Fig 8 Level 2 Data Flow Diagram

* GUI has multiple modules: Live Capture, PCAP Reader, Packet Detail Viewer, Export Handler

**3.3.2 Data Dictionary**

| **Element** | **Description** | **Data Type** |
| --- | --- | --- |
| packet\_data | Captured packet info | Dictionary/List |
| protocol\_type | Type of protocol (TCP/UDP/etc.) | String |
| timestamp | Time of packet capture | String/Datetime |
| source\_ip | Sender’s IP address | String |
| dest\_ip | Receiver’s IP address | String |
| pcap\_file\_path | Filepath of loaded PCAP | String |
| log\_content | Text to be exported as PDF | String |

**IMPLEMENTATION**

**4.1 Implementation Environment**

The implementation of the ScapyUI – Network Sniffer was carried out using the Python programming language and the Tkinter library for the GUI. The tool was developed and tested on a Windows 10 operating system with administrative privileges to allow packet sniffing.

**4.1.1 Model Used in Developing**

The **Iterative Model** was used during development. This approach allowed the application to be built step-by-step with continuous feedback and testing at each phase. Each iteration added functionality such as:

* Live packet capture
* PCAP file reader
* GUI enhancements
* Export to PDF

This model enabled refining of requirements and allowed incremental upgrades, making it suitable for a student project.

**4.1.2 Software Prototyping Types**

**Evolutionary Prototyping** was followed. Initial prototypes focused on the basic GUI and packet capture features, which were gradually improved based on testing and feedback. This helped in enhancing usability, adding new features like:

* Real-time display
* Viewing packet structure
* PCAP reader in a new window
* PDF export with line formatting

**4.2 Coding Standard**

To ensure readability and maintainability, the following coding standards were followed:

* **Naming Convention**: CamelCase for class names, snake\_case for variables and functions.
* **Commenting**: Inline and block comments to describe code functionality.
* **Modular Design**: Code was broken into functions and classes to follow the DRY (Don’t Repeat Yourself) principle.
* **PEP8 Guidelines**: The Python code adheres to PEP8 standards for spacing, naming, and formatting.

**4.3 Laboratory Setup**

The project was implemented and tested in the following lab setup:

|  |  |
| --- | --- |
| **Component** | **Description** |
| Operating System | Windows 10 64-bit |
| RAM | 8 GB |
| Python Version | 3.10+ |
| Dependencies | scapy, tkinter, fpdf, os, sys |
| Permissions | Administrator access for packet sniffing |

**4.4 Tools and Technology Used**

|  |  |
| --- | --- |
| **Tool/Technology** | **Purpose** |
| Python | Core programming |
| Tkinter | GUI development |
| Scapy | Packet sniffing and decoding |
| FPDF | PDF export |
| Wireshark (optional) | PCAP verification |
| VS Code / PyCharm | Code writing and testing |

**4.5 Screenshots / Snapshots**

**1. Main GUI Interface**

Fig 9 Main GUI Interface

**2. Live Packet Capture**

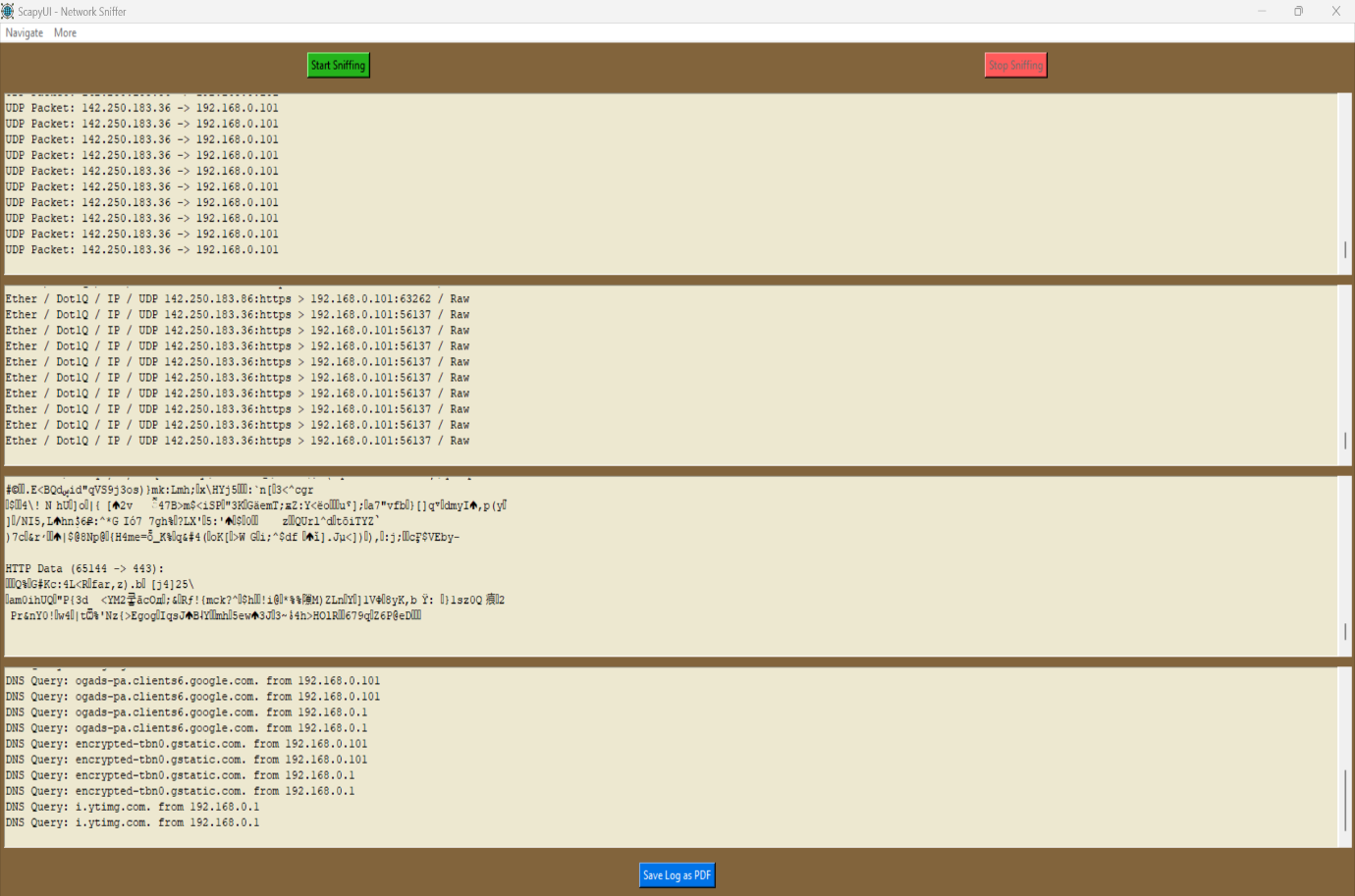


Fig 10 Live Packet Capture

**3. PCAP File Reader Window**



Fig 11 PCAP File Reader Window

**4. PDF Export Output**

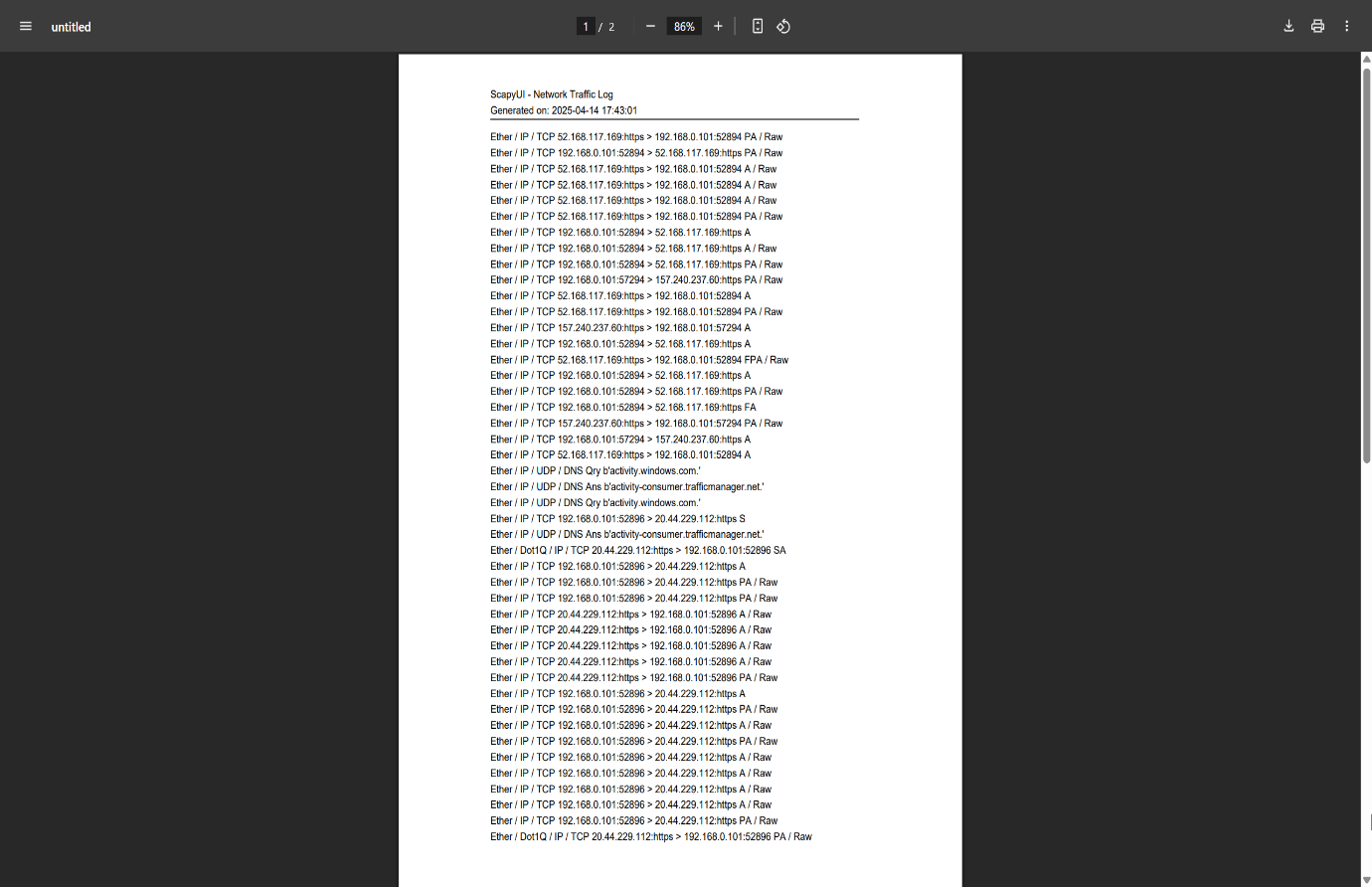
****

Fig 12 PDF Export Output

**SUMMARY OF RESULTS AND FUTURE SCOPE**

**5.1 Advantages / Unique Features**

The ScapyUI – Network Sniffer offers a user-friendly graphical interface to facilitate live packet capturing and analysis using Python. Unlike traditional command-line tools, this application provides accessibility and usability to students, educators, and entry-level professionals. Key advantages and features include:

* **GUI-Based Packet Sniffer**: Simplifies the packet analysis process with a clear and responsive interface.
* **Real-Time Capture Display**: Packets are captured and displayed in real-time with details on protocol, source, destination, and length.
* **PCAP File Reader**: Allows users to open and analyze .pcap files in a separate window.
* **PDF Export Feature**: Enables saving of log data as a PDF for reporting or documentation.
* **Modular and Scalable Design**: The architecture allows easy integration of advanced features such as filtering, protocol decoding, or statistics.
* **Cross-Platform Capability**: Can run on multiple operating systems with Python support.

**5.2 Results and Discussions**

The application was tested across various scenarios including:

* Live packet sniffing on local network traffic.
* Opening previously saved .pcap files for offline analysis.
* Exporting logs to a structured and formatted PDF file.

**Performance Observations:**

* The tool was able to capture and display packets efficiently with minimal delay.
* .pcap files were parsed accurately using Scapy, and the contents were correctly displayed.
* The log formatting for PDF export required handling of line breaks and dynamic content sizing, which was successfully implemented.
* No crashes or memory issues were observed during continuous captures under moderate traffic.

This demonstrates that the tool serves as a functional lightweight packet analyser suitable for academic and research use.

**5.3 Future Scope of Work**

While the current implementation meets the basic requirements of a GUI-based sniffer, the following enhancements can be considered for future versions:

* **Packet Filtering**: Add real-time protocol-based filtering (e.g., TCP, UDP, ICMP).
* **Search Functionality**: Enable keyword or IP-based search in captured data.
* **Protocol Decoding**: Enhance support for deep packet inspection and protocol decoding (HTTP, DNS, etc.).
* **Hex Dump View**: Display the raw hexadecimal view of packet data for forensic purposes.
* **Cloud Integration**: Store and analyze logs on cloud-based platforms for collaborative research or remote access.
* **Security Features**: Add alert-based mechanisms to detect suspicious or malicious packets in real-time.

These upgrades would significantly improve the tool’s capabilities, making it a more complete solution for educational and operational use in network security environments.

**CONCLUSION**

The development of *ScapyUI – Network Sniffer* has successfully addressed the goal of providing a lightweight, GUI-based network packet analyzer using Python and Scapy. The project aimed to bridge the gap between command-line-based packet analysis tools and the ease of graphical interfaces, especially for students, trainers, and entry-level cybersecurity professionals.

Throughout the project, we designed and implemented an interactive, modular, and efficient tool that allows real-time network monitoring, .pcap file analysis, and exporting of logs to a PDF format. The integration of Tkinter and Scapy provided a seamless environment for capturing and analyzing packets, while the implementation of PDF and GUI functionalities enhanced usability.

The tool meets the basic requirements of packet analysis and provides a strong foundation for future extensions, such as real-time filtering, protocol decoding, and threat detection. It demonstrates how Python can be effectively used to develop functional cybersecurity tools and contributes to the academic learning experience in digital forensics and network security.

By completing this project, we gained hands-on experience in software design, GUI development, network packet analysis, and tool prototyping. The project not only strengthens understanding of network protocols and packet structures but also builds practical skills in Python programming and GUI application development.

**BIBLIOGRAPHY**

1. Scapy Documentation – <https://scapy.readthedocs.io/>
2. Tkinter – Python Interface to Tcl/Tk – <https://docs.python.org/3/library/tkinter.html>
3. RFC 791 – Internet Protocol Specification.
4. RFC 793 – Transmission Control Protocol.
5. Core Python Programming, Wesley J. Chun, Pearson.
6. "Foundations of Python Network Programming" – John Goerzen, Brandon Rhodes.
7. PyPDF2 Documentation – <https://pypdf2.readthedocs.io/>
8. "Practical Packet Analysis" by Chris Sanders.
9. Stack Overflow – <https://stackoverflow.com/>
10. GeeksForGeeks – <https://www.geeksforgeeks.org/>

**Appendices (if any)**

**Appendix I – Screenshots of Application**  
(Include relevant screenshots of the GUI, packet logs, .pcap file window, PDF output, etc.)

**List of Publications / Online References / etc.**

**Appendix II**

1. GitHub (Project Repository): *If hosted*
2. Python.org – <https://www.python.org/>
3. Medium Articles on Scapy and Network Sniffers.
4. YouTube Tutorials on Packet Sniffing and Scapy.
5. Online Forums and Discussions (e.g., Reddit, Stack Overflow threads related to Tkinter & Scapy).

**GUIDELINES FOR DISSERTATION PREPARATION**

**1. INTRODUCTION**

**1.1 Purpose**

This document, herein after referred to as your Dissertation/Project Report Guide, lists the general and specific requirements governing dissertation preparation including guidelines for structuring the contents. For style, structure and presentation of the dissertation, students may refer to additional style manuals or reference guides (some of which are listed below) and to the published literature in their respective field of study.

**1.2 Dissertation Submission**

To have the dissertation examined, the number of dissertation copies to be submitted to the Dean of School of Cyber Security and Digital Forensics should correspond to

1. The number of dissertation need to be submitted is **3** (including dissertation of supervisor **{1 copy for student, 1 for supervisor and 1 for library}**) for an M.Tech/MSc Degree student.

**2. SPECIFICATIONS FOR DISSERTATION FORMAT**

**2.1 Preparation of Manuscript and Copies**

**2.1.1** The dissertation needs to be prepared using a standard text processing software and must be printed in black text (color for images, if necessary) using a laser printer or letter quality printer in standard typeface (Times New Roman).

**2.1.2** The dissertation must be printed or photocopied on single sides of good quality white paper. All copies of dissertation pages must be clear, sharp and even, with uniform size and uniformly spaced characters, lines and margins on every page of good quality white paper of 75 gsm or more.

**2.1.3** Dissertation should be free from typographical errors.

**2.1.4** The Chapter Header should be of size 18, Typeface: Times New Roman with Bold & Underline Tag, Header/Title should be of size 14, Typeface: Times New Roman and Bold, Sub-section/Text Content should be of size 12, Typeface: Times New Roman.

**2.2 Size and Margins**

**2.2.1** *Printing of Dissertation*

A4 (21 cm x 29.7 cm) is the recommended dissertation size. Dissertation should be printed SINGLE SIDE.

**2.2.2** The top and bottom margins should be 2.54 cm, whereas, the left side and right side margin should be 2.54 cm for both textual and non-textual (e.g., figures, tables) pages.

**2.2.3** Content should not extend beyond the bottom margin except for completing a footnote, last line of chapter/subdivision, or figure/table caption.

**2.2.4** A sub-head at the bottom of the page should have at least two full lines of content below it. If the sub-head is too short to allow this, it should begin on the next page.

**2.2.5** All tables and figures should conform to the same requirements as text. Color may be used for figures. If tables and figures are large, they may be reduced to the standard size (provided the reduced area is not less than 50% of the original) and/or folded just once to flush with the dissertation margin.

**2.2.6** Students should also submit the dissertation in soft form (both PDF and word) for storage and archival.

**2.3 Page Numbering**

**2.3.1** Beginning with the first page of the text in the dissertation (chapter 1), all pages should be numbered consecutively and consistently in Arabic numerals (1, 2, 3, …) through the appendices. *All these pages should be Single page printed.*

**2.3.2** Page numbers prior to Chapter 1 should be in lower case Roman numerals (i, ii, iii, …). The title page is considered to be page (i) but the number is not printed. *All these pages should be single page printed.*

**2.3.3** All page numbers should be placed without punctuation in the lower right hand corner, 1. 2 cm from the top edge and with the last digit even with the right hand margin.

**2.5 Line Spacing**

The general text of the manuscript should be in 1.5 spacing (3 lines per inch). Long tables, quotations, footnotes, multi-line captions and bibliographic entries (references) should be in single spacing (6 lines per inch), with a preferred text size in 12 points.

**2.6 Tables, Figures and Equations**

**2.6.1** All tables (tabulated data) and figures (charts, graphs, maps, images, diagrams, etc.) should be prepared, wherever possible, on the same paper used to type the text and conform to the specifications outlined earlier. They should be inserted as close to the textual reference as possible.

**2.6.2** Tables, figures and equations should be numbered sequentially either throughout the dissertation or chapter-wise using Arabic numerals. They are referred to in the body of the text capitalizing the first letter of the word and number, as for instance, Table 5.3, Figure 3.11, Equation (4.16), etc. For example the 1st figure of chapter 5 should be written as Figure 5.1 and 2nd table of chapter 4 is Table 4.2.

**2.6.3** If tables and figures are of only half a page or less, they may appear on the same page as text. Font size for text should be the same as for the general text.

**2.6.4** Good quality Line Drawings/figures must be drawn using standard software that provides vector rather than bit-map graphics. Figures must be scalable.

**2.6.5** Images, Photographs, etc. must be scanned in resolution exceeding 600 dpi with 256 grayscales for the monochrome images and 24 bit per pixel for the color images.

**2.7 Binding**

The student should submit the copies of the dissertation in fully bound form in hard cover. Once the dissertation is accepted, it is the student’s responsibility to get it properly bound before depositing the required number of copies with the Library and the Department concerned. The front cover of the bound copy should be the same as the title page of the dissertation. The front cover should have printing on the side to include the author’s name, abbreviated dissertation title (optional), degree, department, and the year.

**3. GUIDELINES FOR STRUCTURING CONTENTS**

**3.1 Sequence of Contents**

The sequence for the dissertation organization should be followed as mentioned in **Contents** page.

All the headings are centred (without punctuation) 2.54 cm down the top edge of the page. The subsequent type-setting begins four spaces below the heading.

**3.2 Preliminaries**

**3.2.1 Synopsis/Abstract**

**3.2.1.1** M.Tech/M.Sc. dissertation should contain an abstract not exceeding 300 words (about one page).

**3.2.1.2** Synopsis/Abstract should be self-complete and contain no citations for which the dissertation has to be referred.

**3.2.2 Table of Contents**

**3.2.1.1** The table of contents lists all material that follows it. No preceding material is listed. Chapter titles, sections, first and second order sub-divisions, etc. must be listed in it.

**3.2.2.2** Tables, figures, nomenclature, if used in the dissertation, are listed under separate headings.

**3.3 The Text of the Dissertation**

**3.3.1 Introduction**

Introduction may be the first chapter or its first major division. In either case, it should contain a brief statement of the problem investigated. It should outline the scope, aim, general character of the research and the reasons for the student’s interest in the problem.

**3.3.2 The body of Dissertation**

This is the substance of the dissertation inclusive of all divisions, subdivisions, tables, figures, etc.

**3.3.3 Summary and conclusions**

If required, these are given as the last major division (chapter) of the text. A further and final subdivision titled “Scope for Further Work” should follow.

**3.3.4 Reference material**

*The list of references should appear as a consolidated list with references listed either alphabetically (Harvard reference style) or sequentially (Oxford reference style) as they appear in the text of the dissertation*. A student can follow any standard reference style to prepare the dissertation. If pertinent works have been consulted but not specifically cited, they should be listed as Bibliography or General References. Spacing and font size should be consistent inside a single reference, and there should be 1.5 spacing between two different references.

**Reference Format**

For referencing an article in a scientific journal the suggested format should contain the following information: authors, year of publication, title, name of journal, volume number, issue number, and page numbers with range.

For referencing an article published in a book, the suggested format should contain, authors, year of publication, the title of the book, editors, publisher, and page number of the article in the book being referred to.

For referencing a dissertation, the suggested format should contain, author, year of publication, the title of dissertation, and name of institute/university/college where the dissertation was submitted or awarded.

A few examples of formats of references are given below and the student should be consistent in following the style.

The abbreviation of the journals can be referred to the Web of Science website or institute/university/college websites.

https://images.webofknowledge.com/images/help/WOS/A\_abrvjt.html

***Journals***

Abdullah, S.A.; Iqbal, A.; Frormann, L. (**2008**) Melt mixing of carbon fibers and carbon nanotubes incorporated polyurethanes, *J. Appl. Polym. Sci.*, **110** (1), *196–202*.

Swain, S. K.; Isayev, A.I. (**2007**) Effect of ultrasound on HDPE/clay nanocomposites: Rheology, structure and properties, *Polymer*, **48** (1), *281–289*.

*A DOI can be used to cite and link to electronic articles where an article is in press/early view.*

***Conference Proceedings***

Fischmeister, H.F. (**1982**) Development and present status of the science and technology of hard materials, Science of Hard Materials, Viswanadham, R.K.; Rowcliffe, D.J.; Gurland, J. (Eds.) Plenum Press, New York, USA, pp. 1–45.

Baek, W.H.; Hong, M.H.; Lee, S.; Chung, D.T. (**1995**) A study on the shear localization behavior of tungsten heavy alloy, Tungsten and Refractory Metals 2, Bose, A.; Dowding, R.J. (Eds.), Metal Powder Industries Federation, Princeton, New Jersey, USA, pp. 463–471.

***Books***

German, R.M. (**1990**) Powder Injection Molding, Metal Powder Industries Federation, Princeton, New Jersey, USA.

***Book Chapter***

Swain, S. K.; Sahoo, G.; Sarkar, N. **(2015)** “Manufacturing of Chemically Modified Date Palm Leaf Fibre-Reinforced Polymer Composites” In “Manufacturing of Natural Fibre Reinforced Polymer Composites”, Edited by Prof. M. Jawaid Springer International Publishing, pp. 291-308.

***Thesis***

Johnson, J.L. (**1994**) Densification, Microstructural Evolution, and Thermal Properties of Liquid Phase Sintered Composites, Ph.D. Thesis, The Pennsylvania State University, University Park, Pennsylvania, USA.

***Technical Reports***

Zukas, E.G.; Rogers, P.S.Z.; Rogers, R.S. (**1976**) Experimental evidence for spheroid growth mechanisms in the liquid phase sintered tungsten based composites, Informal Report: Los Alamos Scientific laboratory, USA, pp. 1-35.

***Patents:***

Oenning, V.; Clark, I.S.R. (**1991**) Title of the patents, U.S. Patent No.: 4988386.

***Journals in Non-English Language***

Weihong, L.; Xiuren, T. (**1988**) Tungsten Matrix in Cu-W Contact Materials by Impregnation Process, *Powder Metall. Technol.*, **6** (8), *1–4*. (in Chinese)

**3.3.5 Appendix or Appendices**

3.3.5.1 Supplementary illustrative material, original data, and quotations too lengthy for inclusion in the text or which is not immediately essential to an understanding of the subject can be presented in Appendix or Appendices (as Appendix A , Appendix B, etc.)

3.3.5.2 Each appendix with its title should be listed separately in the table of contents. Likewise, tables and figures contained in the Appendices are to be included in the lists of tables and figures, respectively.

**4. CONCLUDING REMARKS**

Conclusion part of the dissertation is most important part of the dissertation. This should include interesting findings of the work. This dissertation Guide lists only the basic requirements for preparing the dissertation. Over and above the aforementioned points, a dissertation should be reader-friendly in both its appearance and presentation. Several aspects of dissertation preparation, particularly style of writing and presentation, have not been discussed in great detail. The student should follow appropriate ideas from standard literature of his/her area of research, and adopt a uniform style and format throughout the dissertation, such as in the structural divisions/subdivisions of the dissertation, in the mode of citing references and footnotes in the text, in using dimensions, units and notations, and in preparing tables and figures, etc.

**FUTURE SCOPE OF THE WORKS**

Dissertation should have a future scope of the works.