Secure and Ethical Monitoring System for Keystroke and User Behaviour

A Dissertation submitted to the Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY
IN
CSE (CYBER SECURITY)

Submitted by

S PAVAN KUMAR	**21B81A6227**
G PRANAY	**21B81A6230**
R SAI UTTEJ	**21B81A6243**

Under the guidance of
Mrs. G. SUNITHA REKHA
Sr. Assistant Professor

DEPARTMENT OF CSE (CYBER SECURITY)

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

APRIL 2025

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

DEPARTMENT OF CSE (CYBER SECURITY)

CERTIFICATE

This is to certify that the project report entitled **"Secure and Ethical Monitoring System for Keystroke and User Behaviour"** bonafide record of work carried out by **S PAVAN KUMAR (21B81A6227), G PRANAY (21B81A6230)** and **R SAI UTTEJ (21B81A6243)** submitted to **Mrs. G Sunitha Rekha, Sr. Assistant Professor** for the requirement of the award of **Bachelor of Technology** in **CSE (Cyber Security)** to the CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025.

Project Guide	**Project Coordinator**
Mrs. G Sunitha Rekha	**Dr. B. Vikranth**
Sr. Assistant Professor	Professor
Department of CSE(CS)	Department of CSE(CS)

Head of the Department	**External Examiner**
Dr. M. Sunitha	
Professor	
HOD – CSE(CS)	

DECLARATION

We hereby declare that the project report entitled **“Secure and Ethical Monitoring System for Keystroke and User Behaviour”** is an original work done and submitted to CS Department, CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University Hyderabad in partial fulfilment for the requirement of the award of Bachelor of Technology in Computer Science and Information Technology and it is a record of bonafide project work carried out by us under the guidance of **Mrs. G Sunitha Rekha,** Sr. Assistant Professor, CSE(CS).

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

Signature of the Student

S PAVAN KUMAR

Signature of the Student

G PRANAY

Signature of the Student

R SAI UTTEJ

Date:

Place:

ACKNOWLEDGEMENT

We are deeply thankful and fortunate to have received constant encouragement, support, and guidance from all the teaching staff of the **Cyber Security Department**, which helped us successfully complete this project.

We sincerely thank **Dr. B. Vikranth,** Project Coordinator, and **Ms. K. Swathi,** Project Review Committee member, for their valuable guidance and support, which played a crucial role in the successful completion of our project.

We extend our heartfelt gratitude and respect to our internal guide, **Mrs. G Sunitha Rekha,** Sr. Assistant Professor, Department of CSE (CS), for her unwavering support and guidance, which enabled us to complete our project successfully.

We would also like to extend our sincere gratitude to **Dr. M. Sunitha,** Professor and Head of the Department of CSE (CS) for giving us the opportunity to undertake this project and for her unwavering support and guidance.

We are grateful to our **Vice-Principal, Prof. L. C. Siva Reddy,** for ensuring access to excellent computing facilities and fostering a disciplined atmosphere conducive to our work.

Finally, we extend our profound gratitude to **Dr. Rama Mohan Reddy, Principal,** and the **Management** for providing outstanding lab facilities and tools. We also thank everyone whose guidance and support have been invaluable to us throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

ABBREVIATIONS

Abbreviation	**Expansion**
SHA	Secure Hash Algorithm
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman (Encryption Algorithm)
NLP	Natural Language Processing
AI	Artificial Intelligence
ML	Machine Learning
SQL	Structured Query Language
JSON	JavaScript Object Notation
Pynput	Python Input Library (for capturing keystrokes)
MongoDB	NoSQL Database used for storing keylogs
UI	User Interface
GPU	Graphics Processing Unit
CPU	Central Processing Unit
SSD	Solid State Drive
RAM	Random Access Memory
QA	Question Answering
API	Application Programming Interface
Fuzzy Matching	Approximate String Matching Algorithm
SMTP	Simple Mail Transfer Protocol (for sending email alerts)

ABSTRACT

With the rise of digital communication, ensuring children's online safety has become a critical concern. This project presents a Secure and Ethical Monitoring System for Keystroke and User Behaviour Application, designed to detect and log potentially harmful online interactions. The system incorporates a keylogger module that captures keystrokes and processes them using a blacklist-based threat detection system. The blacklist, stored in a structured JSON format, categorizes words related to cyberbullying, hate speech, drugs, nudity, and other online threats.

Upon detecting flagged content, the system triggers a real-time monitoring mechanism, which includes automated screenshot capture using PyAutoGUI and logs the event in a secure database (SQL/MangoDB). The backend, developed using Flask, handles data processing and communicates with the React-based frontend dashboard, providing parents with real-time alerts, reports, and captured evidence.

The application ensures data security and privacy by implementing SHA256 encryption for stored logs and Flask-SSLify to enforce secure API interactions. The system is designed to run in the background with minimal performance overhead, making it an efficient solution for proactive parental control. This project delivers an industry-standard approach to child safety by integrating real-time monitoring, AI-driven word filtering, and automated evidence collection, providing parents with the tools to protect their children in an increasingly digital world.

CHAPTER 1

INTRODUCTION

The rise of digital communication has significantly impacted children's online interactions, exposing them to risks such as cyberbullying, harassment, explicit content, and online predators. Traditional parental control methods, such as website blocking and manual supervision, often fail to provide real-time insights into a child's digital activities. With the increasing adoption of social media, messaging platforms, and online gaming, ensuring online safety requires proactive monitoring and intelligent threat detection mechanisms.

This project introduces a Secure and Ethical Monitoring System for Keystroke and User Behaviour that utilizes blacklist-based threat detection with content filtering, real-time keylogging, and automated evidence collection to help parents safeguard their children from online threats. The system continuously monitors user interactions, detects flagged words or phrases based on a customizable blacklist, and captures relevant evidence through automated screenshot logging.

The solution is built using a Flask-based backend, a React-powered dashboard, and a secure SQL/MangoDB database for log storage. The application leverages Natural Language Processing (NLP) techniques to enhance blacklist filtering and ensures secure data transmission through SHA256 encryption and SSL protection. Unlike traditional parental control software, which relies on predefined website filtering, this system operates at the keystroke level, providing real-time, context-aware monitoring. By integrating automation, real-time detection, and secure data handling, this project offers a scalable and efficient solution for modern parental control needs.

1.1 MOTIVATION

With children spending increasing amounts of time online, the risk of exposure to cyber threats has grown exponentially. Studies indicate that cyberbullying, online harassment, and exposure to explicit content can have severe psychological effects on young users. Existing parental control solutions primarily focus on blocking access to websites, but they fail to monitor real-time conversations, gaming interactions, and social media engagements, where most online risks occur.

Keyword-based detection systems often lack contextual understanding, leading to false positives or missing threats entirely. Additionally, many parental monitoring solutions are cloud-based, raising concerns about data privacy, latency, and third-party access. To address these challenges, this project integrates a locally hosted, enhanced monitoring system that operates securely and efficiently without external dependencies.

The system is motivated by the need to provide real-time, privacy-conscious, and intelligent monitoring, leveraging:

- **Advanced keylogging** to track user interactions seamlessly.
- **Blacklist-based word** filtering with fuzzy word enhancements accurate threat detection.
- **Automated screenshot capture** to provide concrete evidence of flagged interactions.
- **End-to-end encryption** to ensure secure storage and retrieval of logs.

Unlike existing solutions, this system prioritizes local processing to minimize privacy risks and operational costs while ensuring parents receive timely alerts and actionable insights. By offering a scalable parental monitoring system, this project aims to set a new industry standard for child online safety, making digital environments safer and more transparent for families.

1.2 PROBLEM STATEMENT

In today's digital landscape, monitoring user activity has become a crucial aspect of cybersecurity, corporate oversight, and parental control. However, traditional keylogging systems often pose significant privacy and security risks, as they can be exploited for unethical surveillance or data breaches. The challenge lies in developing a monitoring solution that ensures security without violating ethical boundaries. Many existing systems lack proper encryption, making sensitive user data vulnerable to unauthorized access. The need for a secure, efficient, and privacy-compliant monitoring system is greater than ever.

This project addresses these concerns by developing a Secure and Ethical Monitoring System that captures keystrokes, clipboard activity, and screenshots while maintaining strict privacy protections. A major challenge is ensuring that the captured data remains tamper-proof, preventing attackers from altering logs to erase evidence of malicious activity. To combat this, our system employs strong encryption techniques to protect all stored logs from unauthorized access. Another critical issue is the real-time detection of potential threats without degrading system performance. To achieve this, we integrate an optimized fuzzy logic-based approach that analyses keystrokes and triggers alerts when blacklisted terms are detected. Additionally, whenever suspicious words appear, the system captures screenshots to provide forensic evidence.

The monitoring process is further enhanced by an automated alert system that notifies administrators through email whenever a security violation is detected, ensuring timely intervention.

By combining encryption, automated threat detection, and a privacy-preserving design, this project offers a robust and responsible monitoring solution. Unlike conventional keyloggers, which often compromise privacy, this system upholds ethical standards while providing real-time security insights. Its applications span across corporate cybersecurity, parental control, and insider threat detection, setting a new benchmark for intelligent and privacy-conscious monitoring technologies.

1.3 PROJECT OBJECTIVES

The main objective of this project is to develop a Secure and Ethical Monitoring System that captures keystrokes, clipboard activity, and screenshots while maintaining privacy, security, and compliance with ethical standards. It integrates encryption, real-time threat detection, and automated alerts to ensure responsible monitoring without violating user rights.

Below are the detailed objectives:

1. **Develop a Privacy-Compliant Keylogging System**
    - Capture keystrokes and user activity while adhering to ethical guidelines.
2. **Implement Secure Data Encryption**
    - Use SHA256/RSA encryption to protect logs from unauthorized access.
3. **Enable Automated Threat Detection & Alerts**
    - Detect blacklisted words in real time and trigger security alerts.
4. **Capture Screenshots for Forensic Evidence**
    - Take automatic snapshots upon detecting suspicious activity.
5. **Provide an Interactive Monitoring Dashboard**
    - Offer administrators a user-friendly interface for reviewing logs and alerts.
6. **Optimize System Performance & Scalability**

- Ensure efficient monitoring with minimal computational overhead..

In summary, this project delivers a privacy-conscious, security-focused monitoring solution for corporate security, parental control, and insider threat prevention by integrating encryption, real-time detection, and forensic capabilities.

1.4 PROJECT REPORT ORGANIZATION

This report is organized into six sections, each covering a key aspect of the project's development and implementation.

Below is an overview of the sections included in this report:

Introduction

- Overview of keylogging and user activity monitoring challenges.
- Motivation behind the project, problem statement, and objectives**.**

Literature Survey

- Review of existing keylogging and monitoring techniques.
- Limitations of traditional monitoring systems in terms of security and privacy.

Software and Hardware Specifications

- List of required software tools (Flask, MongoDB, Pynput, React).
- System architecture and key dependencies.
- Hardware specifications for optimal system performance.

Proposed System Design

- Explanation of secure keylogging methodology.
- Encryption mechanisms for secure data storage and transmission.
- System architecture, including key components and data flow.

Implementation and Testing

- Screenshots of the system interface and workflow.
- Test cases

Conclusion and Future Scope

- Summary of key findings and project impact.
- Future enhancements, such as mobile support, NLP-based threat detection, and cloud integration.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING WORK

Existing user activity monitoring and keylogging systems rely on various approaches, including hardware-based keyloggers, software keyloggers, and remote monitoring tools. While these methods provide basic surveillance capabilities, they often lack security, ethical considerations, and advanced threat detection mechanisms.

Below is an overview of key existing methods:

1. Hardware Keyloggers

Overview: Hardware keyloggers are physical devices placed between a keyboard and a computer to capture keystrokes before they reach the system. They are commonly used for surveillance and forensic investigations but require physical access to install and retrieve data.

Working:

1. The device intercepts signals from the keyboard and logs each keystroke before it is processed by the operating system.
2. Logs are stored in the internal memory of the keylogger.
3. Data can be retrieved manually or, in advanced versions, transmitted wirelessly to a remote system.

Advantages	**Disadvantages**
Difficult to detect as they operate outside the software environment.	Require physical access to install and retrieve data.
Bypasses anti-malware and software security mechanisms.	Cannot capture clipboard data, screenshots, or application activity.

Works independently of the operating system, ensuring continuous logging.	Expensive and impractical for large-scale monitoring.

Table 2.1: Advantages and Disadvantages of Hardware Keyloggers

2. Software Keyloggers

Overview: Software keyloggers are applications that run in the background of an operating system, recording keystrokes, clipboard content, and in some cases, screenshots. They can be installed with or without user consent, making them widely used for both ethical and malicious purposes.

Working:

- The keylogger runs as a background process, intercepting keystrokes before they reach the application.
- Captured keystrokes are logged into a file or database.
- Some software keyloggers also capture screenshots and monitor clipboard activity.
- Logs are stored locally or transmitted to an external server.

Advantages	**Disadvantages**
Easy to deploy and configure on multiple systems.	Often detected by modern antivirus and security tools.
Can capture additional data, such as clipboard activity and screenshots.	Poses significant privacy risks if misused for unethical surveillance.
Allows real-time monitoring and integration with alert mechanisms.	Requires system permissions, making installation complex on secure systems.

Table 2.2: Advantages and Disadvantages of Software Keyloggers

3. Remote Keyloggers

Overview: Remote keyloggers are a type of software-based keylogger that transmits recorded keystrokes and user activity data to a remote server, allowing real-time monitoring from a different location. They are often used in corporate environments for employee monitoring and parental control applications.

Working:

- The software keylogger runs on the target device and captures keystrokes.
- Data is encrypted and transmitted to a remote server for secure access.
- Administrators can review logs and set up automated alerts for suspicious activity.

Advantages	**Disadvantages**
Enables real-time monitoring from remote locations.	Poses high security risks due to potential misuse by cybercriminals.
Useful for corporate security, parental control, and insider threat detection.	Requires strong encryption to prevent unauthorized data interception.
Allows centralized access to keystroke logs for multiple devices.	May be illegal or require user consent in certain regions due to privacy concerns.

Table 2.3: Advantages and Disadvantages of Remote keyloggers

2.2 LIMITATIONS OF EXISTING WORK

While existing document retrieval methods have been widely used, each approach has notable limitations that hinder their effectiveness. Keyword-based searches lack contextual understanding, rule-based systems are rigid, traditional machine learning retrieval struggles with complex queries, and deep learning-based retrieval requires extensive resources.

Below are the specific limitations of each method:

1. Limitations of Hardware Keyloggers

Description: Hardware keyloggers are physical devices installed between a keyboard and a computer to capture keystrokes before they reach the operating system. These are often used in surveillance, forensic analysis, and cyberattacks.

- **Requires Physical Access:** The device must be manually installed and retrieved, making remote deployment impossible and limiting large-scale usage.
- **Limited Functionality:** Hardware keyloggers only capture keystrokes; they cannot monitor clipboard activity, screenshots, or application usage, reducing their effectiveness in advanced monitoring.
- **Easy to Detect Through Inspection:** A simple physical check of the keyboard connection can reveal the presence of a keylogger, making them unreliable for covert monitoring.
- **Expensive for Advanced Models:** Wireless or encrypted hardware keyloggers are costly, making them impractical for widespread use.

2. Limitations of Software Keyloggers

Description: Software keyloggers operate at the system level, logging keystrokes, clipboard data, and screenshots. They are widely used for employee monitoring, parental control, and cybercrime.

- **Easily Detectable by Security Software:** Most modern antivirus programs and endpoint security solutions can detect and block software keyloggers, making them ineffective for stealth monitoring.

- **Vulnerable to Malware and Exploits:** Many software keyloggers do not encrypt stored logs, making them a potential target for hackers who can steal sensitive user data.
- **High Risk of Misuse and Legal Issues:** Unauthorized installation of software keyloggers can lead to ethical and legal violations, as many jurisdictions prohibit the use of keyloggers without user consent.
- **Consumes System Resources:** Some poorly optimized keyloggers can cause lag, slow down applications, or create security vulnerabilities in the system.

3. Limitations of Remote Keyloggers

Description: Remote keyloggers send recorded keystrokes and user activity logs to a remote server for real-time monitoring. They are used in corporate surveillance, parental control, and malicious spying activities.

- **High Risk of Unauthorized Access:** If the keylogger transmits data without encryption, attackers can intercept and steal sensitive information, leading to serious security breaches.
- **Legal and Privacy Concerns:** Many countries have strict regulations against unauthorized surveillance, making remote keyloggers legally questionable in many scenarios.
- **Computational Overhead and Network Congestion:** Since remote keyloggers continuously send data to a server, they can consume significant bandwidth and processing power, slowing down system performance.

- **Difficult to Maintain Anonymity:** Internet service providers (ISPs) and network administrators can often detect and block remote keylogger traffic, reducing their effectiveness.

How This Project Differs from Existing Approaches

This project addresses the weaknesses of traditional keyloggers by integrating security, ethical compliance, and advanced threat detection mechanisms:

- **Data Encryption:** Unlike existing solutions that store logs in plain text, this system encrypts all keystroke data using SHA256 and RSA encryption, ensuring secure storage and transmission.
- **Privacy-Compliant Monitoring:** Instead of tracking all user activity, this project monitors only predefined suspicious actions, reducing privacy violations.
- **Automated Threat Detection & Alerts:** Unlike passive keyloggers that only record data, this system detects blacklisted words and sends real-time alerts, enhancing security.
- **Comprehensive Monitoring Features:** While traditional keyloggers only capture keystrokes, this system also monitors clipboard activity, takes screenshots upon detecting suspicious behaviour, and provides a graphical dashboard for analysis.
- **Optimized Performance:** Unlike remote keyloggers that cause system slowdowns, this system ensures minimal computational overhead through efficient logging and alert mechanisms.

By overcoming the limitations of existing keylogging solutions, this project delivers a secure, ethical, and intelligent monitoring system that balances cybersecurity needs with user privacy and compliance standards.

CHAPTER 3

SOFTWARE AND HARDWARE SPECIFICATIONS

3.1 SOFTWARE REQUIREMENTS

1. Operating System

- Microsoft Windows 11 Home Single Language (Build 22631)

2. Programming Languages

- Python 3.8 or later – Backend development, keystroke capture, and encryption processing.

3. Keylogging and Security Libraries

- Pynput – Captures keystrokes and input events.
- PyAutoGUI – Takes automated screenshots based on triggers.
- Cryptography (SHA256, RSA) – Encrypts captured keystrokes and logs.
- FuzzyWuzzy – Implements fuzzy word detection for blacklisted terms.

4. Development Tools

- VS Code / PyCharm – Code writing and debugging.
- Postman – API testing for log transmission.
- Git – Version control and collaboration.

5. User Interface and Dashboard

- React.js – Develops the interactive web-based monitoring dashboard.
- Bootstrap – Enhances UI responsiveness and design.

6. Database and Storage

- MongoDB – Stores encrypted keystroke logs securely.

7. Dependency and Package Management

- Pip – Handles Python library installations.
- NPM – Manages frontend dependencies.

3.2 HARDWARE REQUIREMENTS

1. Processor (CPU)

- 12th Gen Intel® Core™ i5-1235U (10 Cores, 12 Threads, 1300 MHz)

2. Memory (RAM)

- 8 GB RAM

3. Storage

- 256 GB SSD

4. Graphics Processing Unit (GPU)

- Integrated Intel UHD Graphics

5. Display

- 1920 x 1080 resolution (Full HD) – HP Pavilion x360 2-in-1 Laptop

6. Network Requirements

- Basic internet for email alert transmission and data sync

CHAPTER 4

PROPOSED SYSTEM DESIGN

4.1 PROPOSED METHOD

To ensure efficient keylogging, secure data storage, and real-time monitoring, this project introduces an advanced keylogging system with SHA256 encryption, fuzzy matching for threat detection, and automated alerts. Unlike traditional keyloggers that simply log keystrokes, this system integrates real-time event processing, screenshot capture, and email notifications to enhance security and usability.

Overview of the Proposed System

1. Keystroke Logging & Secure Storage

- Captures keystrokes using Pynput and stores them in a queue before processing.
- Encrypts keystroke data with SHA256 encryption before saving to MongoDB.
- Batches keystroke logs for efficient database insertion, reducing performance overhead.

2. Real-Time Threat Detection & Alerts

- Uses fuzzy matching to detect blacklisted words, even with minor variations.
- Automatically captures a screenshot when a flagged word is detected.
- Stores screenshots securely in GridFS (MongoDB) for later review.
- Sends real-time alerts to the administrator via API requests.

3. Email & Admin Notification System

- Emails the admin whenever a flagged word is detected, including details of the violation.
- Pushes notifications to an API endpoint, allowing external monitoring.

- Logs all flagged activities along with associated screenshots.

Method	Limitations	How This Project Improves
Basic Keystroke Logging	Only records raw keystrokes without context	Implements blacklisted word detection with fuzzy matching.
Traditional Keyloggers	Lack encryption, making logs vulnerable	Uses SHA256 encryption to ensure secure data storage.
Rule-Based Detection	Requires manual rule updates for new threats	Dynamically adapts by analyzing flagged words with AI.
Simple Alert Systems	Sends alerts without detailed threat analysis	Provides screenshots and categorized alerts for review.
Cloud-Based Monitoring	Poses privacy risks by storing data online	Uses local storage and MongoDB for enhanced security.

Table 4.2: Comparison with Existing Methods

Key Steps in the Proposed Method

Step 1: Keystroke Logging and Data Collection: The system captures keystrokes in real-time using Pynput and stores them temporarily in a queue. Each keystroke event includes metadata such as timestamp and user ID before being batched for efficient storage in MongoDB.

Importance

- Ensures real-time keystroke tracking without affecting system performance.
- Enables batch processing, reducing database overhead.

Step 2: Data Encryption and Secure Storage: Keystrokes are securely encrypted using SHA256 encryption before storage, ensuring data privacy and protection. The encrypted logs are then safely stored in MongoDB, allowing for secure retrieval and analysis when needed.

Importance

- Protects keystroke data from unauthorized access.
- Ensures compliance with security best practices.

Step 3: Blacklisted Word Detection (Fuzzy Matching): Each keystroke is analysed using fuzzy matching to detect blacklisted or suspicious words. The system compares the typed input against a predefined blacklist, identifying threats even if words contain minor modifications or obfuscations.

Importance

- Detects suspicious words even with misspellings, substitutions, or typos.
- Enhances detection accuracy by reducing false-negative matches effectively.

Step 4: Trigger Based Screenshot Capture: When a blacklisted word is detected, the system automatically captures a screenshot of the user's screen. These images are securely stored in GridFS (MongoDB) for further review by administrators.

Importance

- Provides strong visual proof for all detected flagged activities.
- Helps administrators analyse potential threats with reliable evidence.

Step 5: Alerting and Notifications: After detecting a flagged word, the system immediately sends an alert via API and emails the administrator. The notification contains detailed incident information and a screenshot link for review.

Importance

- Allows administrators to take instant action on security threats.
- Provides complete details for comprehensive monitoring and reporting.

Step 6: Admin Dashboard for Monitoring:

The system offers an interactive dashboard where administrators can:

- View recorded keystrokes and analyse user activity logs.
- Monitor flagged words, threats, and detected security risks.
- Access stored screenshots to review potential violations.

Importance

- Provides a centralized platform for efficient keylogging monitoring.
- Enables quick searching and filtering for enhanced investigation.

Feature	**Benefit**
Real-time Keystroke Logging	Captures and stores keystrokes without performance lag.
SHA256 Encryption	Ensures secure storage by encrypting all keystroke data.
Fuzzy Matching Detection	Identifies threats even with misspellings or modifications.
Automated Screenshot Capture	Provides visual proof when blacklisted words are detected.
Instant Alert System	Notifies administrators immediately upon flagged activity.
Secure MongoDB Storage	Logs and screenshots are stored safely for future review.
Admin Monitoring Dashboard	Allows easy tracking, filtering, and reviewing of logs.
Continuous System Optimization	Enhances detection accuracy by adapting to new threats.

Table 4.1: Advantages of the Proposed System

4.2 CLASS DIAGRAM

The class diagram represents the architecture of the Keystroke Monitoring System, illustrating key components and their interactions. This system securely captures, encrypts, stores, analyses,and monitors user keystrokes to detect blacklisted words and trigger alerts when necessary.

The diagram consists of seven primary classes, each fulfilling a distinct role:

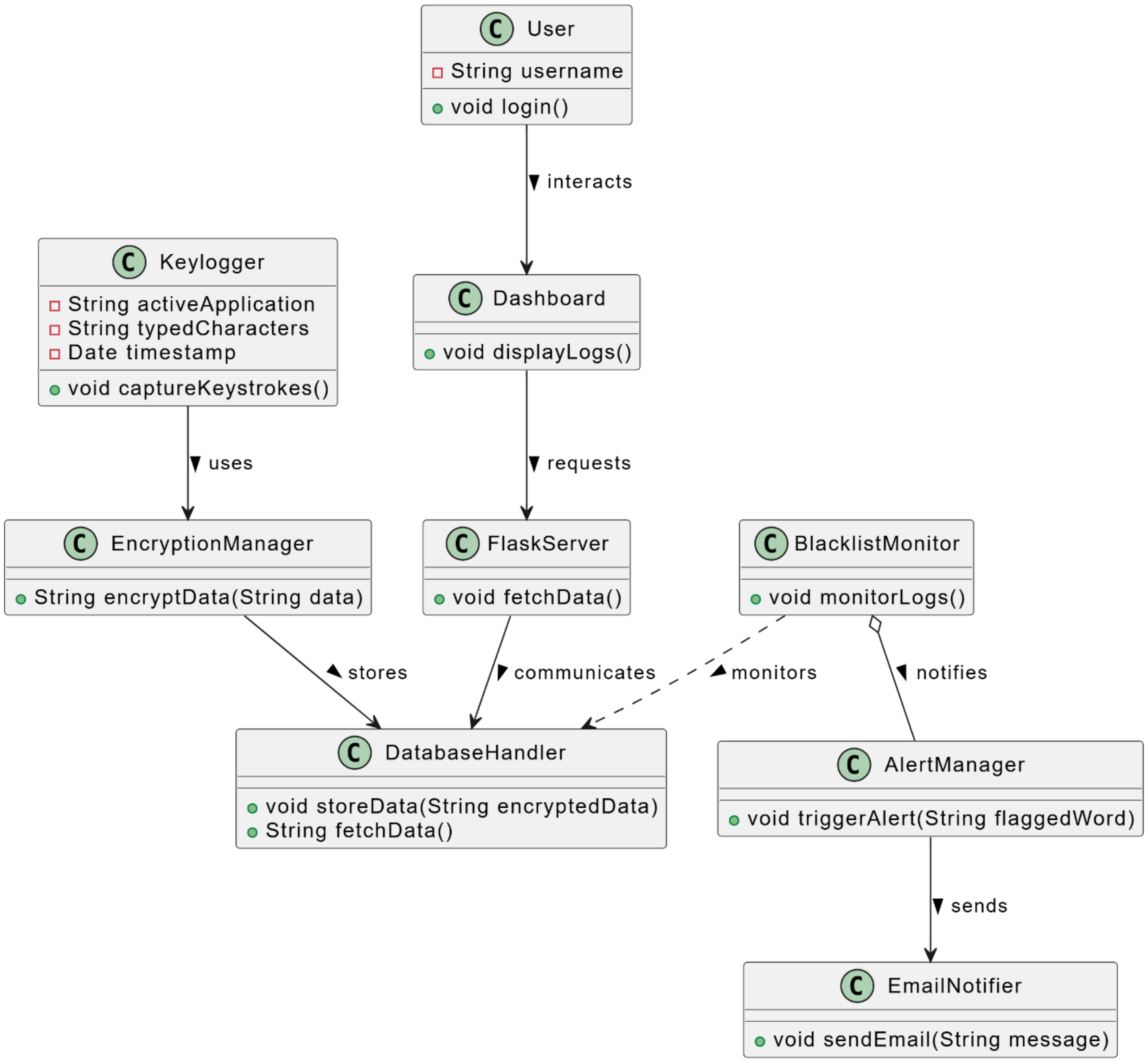


Figure 4.1: Class Diagram

1. User

- **Description:** Represents the system user who logs in and interacts with the monitoring dashboard.
- **Attributes:**
    - username: String – Stores the user's login credentials.
- **Methods:**
    - void login() – Handles user authentication.
- **Role in the System:** Enables user authentication and access to the monitoring dashboard.

2. Keylogger

- **Description:** Captures keystrokes, including typed characters, timestamps, and active applications.
- **Attributes:**
    - Active Application: String – Tracks the current application in use.
    - Typed Characters: String – Stores the captured keystrokes.
    - timestamp: Date – Records the time of each keystroke.
- **Methods:**
    - void capture Keystrokes() – Monitors and records user keystrokes.
- **Role in the System:** Acts as the core logging component, capturing user keystrokes in real time.

3. Encryption Manager

- **Description:** Ensures secure storage of keystrokes by encrypting the captured data.
- **Attributes:**
    - Encrypt Data(String data): String – Encrypts keystroke data using SHA256 encryption.
- **Role in the System:** Protects sensitive keystroke logs by encrypting data before storage.

4. Database Handler

- **Description:** Manages data storage and retrieval operations.
- **Methods:**

    - void store Data(String encrypted Data) – Stores encrypted keystroke logs in MongoDB.
    - String fetch Data() – Retrieves stored logs for analysis.
- **Role in the System:** Ensures secure and efficient log management.

5. Flask Server

- **Description:** Facilitates communication between the front-end dashboard and the backend storage.
- **Methods:**
    - void fetch Data() – Fetches logs from the database upon request.
- **Role in the System:** Acts as the bridge between the data storage layer and the user interface.

6. Blacklist Monitor

- **Description:** Analyses stored keystrokes and detects blacklisted words using fuzzy matching.
- **Methods:**
    - void monitor Logs() – Scans keystrokes for flagged words and suspicious activity.
- **Role in the System:** Enhances threat detection by identifying potentially harmful keystrokes.

7. Alert Manager

- **Description:** Triggers alerts when blacklisted words are detected.
- **Methods:**
    - void trigger Alert(String flagged Word) – Notifies administrators when a suspicious word is found.
- **Role in the System:** Ensures real-time alerts for quick response to threats.

8. Email Notifier

- **Description:** Sends email notifications to administrators upon detecting suspicious activity.
- **Methods:**
    - void send Email(String message) – Sends an alert email with relevant details.
- **Role in the System:** Ensures administrators receive real-time notifications of flagged activity.

4.3 USE CASE DIAGRAM

The Use Case Diagram highlights the interactions between users and administrators within the Keylogger Monitoring System. It illustrates the core functionalities of keystroke logging, monitoring, and alerting mechanisms.

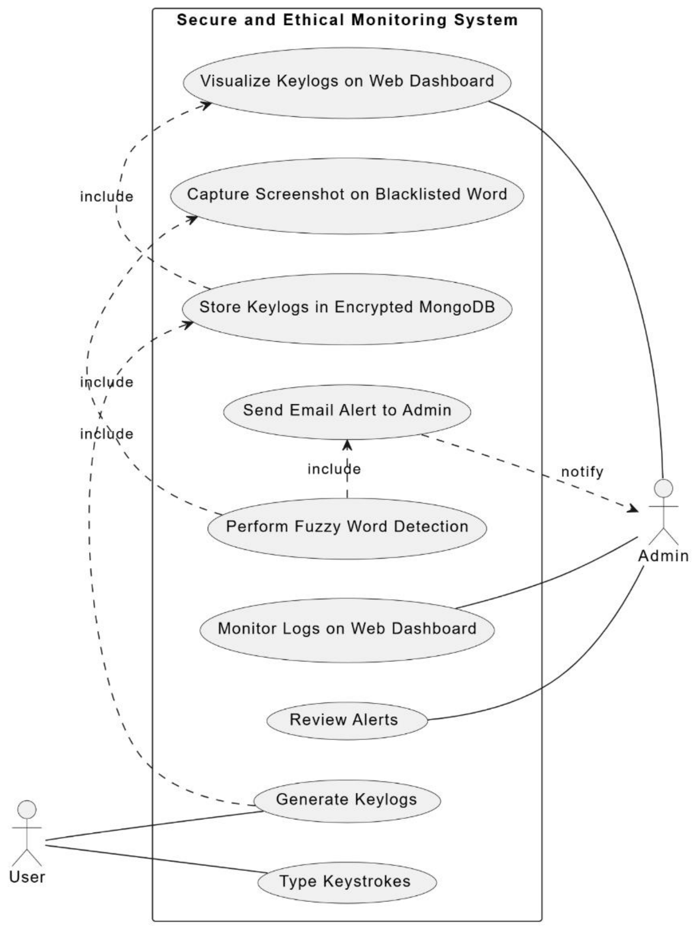

Secure and Ethical Monitoring System
Visualize Keylogs on Web Dashboard
Capture Screenshot on Blacklisted Word
Store Keylogs in Encrypted MongoDB
Send Email Alert to Admin
Perform Fuzzy Word Detection
Monitor Logs on Web Dashboard
Review Alerts
Generate Keylogs
Type Keystrokes
include
include
include
include
notify
Admin
User

Figure 4.2: Use Case Diagram

Actors in the System:

1. **User**

   Role: The individual typing on the system, whose keystrokes are logged and analysed.

2. **Admin**

   Role: Monitors and manages the logged data, reviewing alerts and taking necessary actions.

Use Cases:

1. **Type Keystrokes**
   - **Description:** The user types on the keyboard.
   - **System Response:** Captures and logs each keystroke.30
   - **Trigger:** Every keystroke entry by the user.
2. **Generate Keylogs**
   - **Description:** The system continuously logs keystrokes in the background.
   - **Dependency:** Requires the "Type Keystrokes" action.
   - **Impact:** Enables further analysis and monitoring.
3. **Store Keylogs in Encrypted MongoDB**
   - **Description:** Stores recorded keystrokes securely in an encrypted database.
   - **Impact:** Ensures data privacy and integrity.
   - **Inclusion:** Included in "Generate Keylogs."
4. **Perform Fuzzy Word Detection**
   - **Description:** Detects and analyses words for potential security threats based on a predefined blacklist.

- **Dependency:** Requires stored Keylogs for analysis.
- **Impact:** Helps identify suspicious activity.

5. **Capture Screenshot on Blacklisted Word**
   - **Description:** If a blacklisted word is detected, the system captures a screenshot of the user's activity.
   - **Dependency:** Triggered by "Perform Fuzzy Word Detection."
   - **Inclusion:** Included in "Perform Fuzzy Word Detection."
6. **Send Email Alert to Admin**
   - **Description:** Notifies the administrator when a blacklisted word is detected.
   - **Trigger:** Activated after fuzzy detection identifies a threat.
   - **Inclusion:** Included in "Perform Fuzzy Word Detection."
7. **Monitor Logs on Web Dashboard**
   - **Description:** The admin can view keylog records and potential alerts on a web-based dashboard.
   - **Impact:** Enables real-time monitoring.
8. **Visualize Keylogs on Web Dashboard**
   - **Description:** Displays keystroke data in an organized format for easier analysis.
   - **Dependency:** Requires keylog storage.
9. **Review Alerts**
   - **Description:** The admin reviews flagged alerts and takes action if needed.
   - **Dependency:** Alerts generated from "Send Email Alert to Admin."

System Interaction and Flow:

1. **User Types Keystrokes → System Logs Keystrokes**
   - Keystrokes are continuously recorded.
2. **Keylogs Stored in Encrypted MongoDB**
   - Secure storage ensures data integrity.
3. **Perform Fuzzy Word Detection → Capture Screenshot if Blacklisted Word Detected**
   - Identifies potential threats and logs evidence.
4. **Send Email Alert to Admin → Admin Reviews Alerts**
   - Notifies admin about suspicious activity.
5. **Admin Monitors Logs on Web Dashboard**
   - The admin can view, analyse, and take necessary actions based on recorded data.

4.4 ACTIVITY DIAGRAM

The Activity Diagram illustrates the complete workflow of the Keylogger Monitoring System, showcasing how it captures user activity, detects potential threats, and alerts administrators in case of suspicious behaviour. It highlights the interaction between system processes and administrators, ensuring continuous monitoring and reporting.

This diagram consists of two primary roles:

1. **System Process (Keylogger)** :

   Handles logging, storage, monitoring, detection, and alerts.
2. **Administrator Actions** :

   Receives alerts and reports for further action.

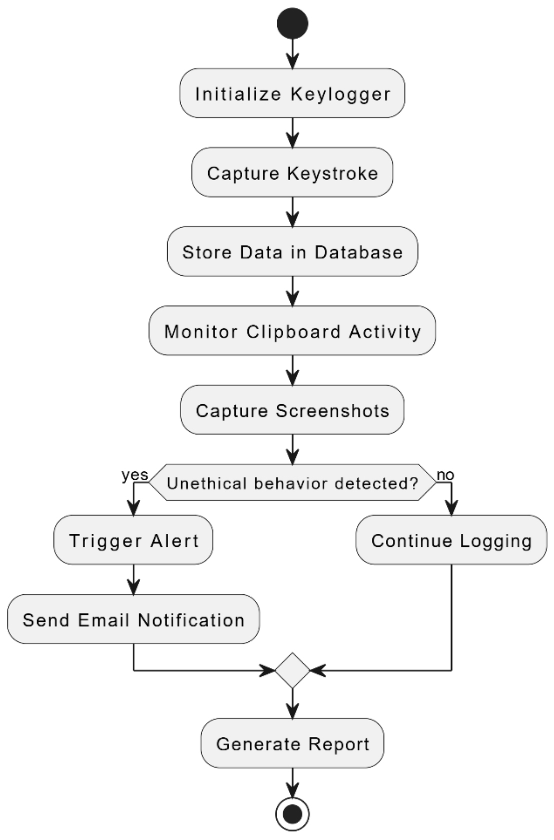


Figure 4.4: Activity Diagram

1. Initialize Keylogger

- The keylogger system starts running as a background process.
- It may launch automatically at system startup or be manually activated by an administrator.
- The keylogger ensures it operates in stealth mode, remaining undetectable to the user.

2. Capture Keystroke

- Every keystroke entered by the user is captured in real-time.
- The system records each keypress, including special characters, numbers, and function keys.

- Data is logged in chronological order to maintain the sequence of typed words and sentences.
- The system can identify specific key combinations, such as passwords or commands.

3. Store Data in Database

- The captured keystrokes are securely stored in a database.
- Data encryption ensures that only authorized administrators can access or decrypt the logs.
- Each keystroke entry is stored with timestamps, user information, and application details (e.g., which software or browser was used).
- The system categorizes stored data to differentiate between normal text, passwords, and command inputs.

4. Monitor Clipboard Activity

- The system tracks copy-paste actions, monitoring clipboard usage.
- If a user copies sensitive information (e.g., credit card numbers, passwords, or confidential data), the keylogger logs it.
- Clipboard history is checked against a predefined blacklist of sensitive keywords.
- It ensures that sensitive or restricted content is flagged before unauthorized sharing occurs.

5. Capture Screenshots

- The system takes periodic screenshots of the user's screen at predefined time intervals.
- Screenshots can also be triggered when certain keywords are detected in typed text or clipboard activity.
- The images are stored securely, with timestamps, for review by the administrator.

- This feature helps track visual evidence of unethical behaviour, such as accessing restricted websites or applications.

6. Detect Unethical Behaviour

- The system analyses logged data (keystrokes, clipboard history, and screenshots) for potential threats.
- It matches user activity against a blacklist of suspicious words, websites, or actions (e.g., hacking tools, data leaks).
- Decision Point:
    - Yes → If suspicious behaviour is detected, proceed to Trigger Alert.
    - No → If no unethical activity is found, the system continues logging normally.

7. Trigger Alert (If Unethical Behaviour is Detected)

- When suspicious activity is detected, an alert is automatically generated.
- The system creates a detailed incident log containing:
    - The detected keyword or activity.
    - The exact timestamp and application used.
    - A corresponding screenshot for verification.
- This step ensures that any potential security risks are flagged immediately.

8. Send Email Notification

- The system sends an automated email notification to the administrator.
- The email contains:
    - A brief summary of the detected threat.
    - Logged keystrokes and clipboard data relevant to the alert.
    - A screenshot attachment for additional context.

    - Date, time, and system details for investigation.
- The admin receives real-time updates, allowing for immediate action against any potential breaches.

9. Generate Report

- The system generates a detailed activity report summarizing:
    - All logged keystrokes.
    - Clipboard activity history.
    - Captured screenshots.
    - Any flagged suspicious behaviour.
- The report is formatted for easy review, including filtering options for faster analysis.
- The administrator can access and export the report for record-keeping or further action.
- This report helps with investigations, compliance checks, and security audits.

4.5 SYSTEM ARCHITECTURE

This system architecture diagram visually represents how the child monitoring parental app works. The system is designed to capture, analyse, and securely store keystrokes while also detecting blacklisted words and alerting parents through an admin dashboard.

It consists of four major components:

1. **User Interaction** – Child interacts with various applications.
2. **Backend Processing** – Logs keystrokes, detects blacklisted words, encrypts data, and triggers alerts.
3. **Database & Security** – Securely stores encrypted keystrokes.
4. **Admin Dashboard** – Allows monitoring and access control.

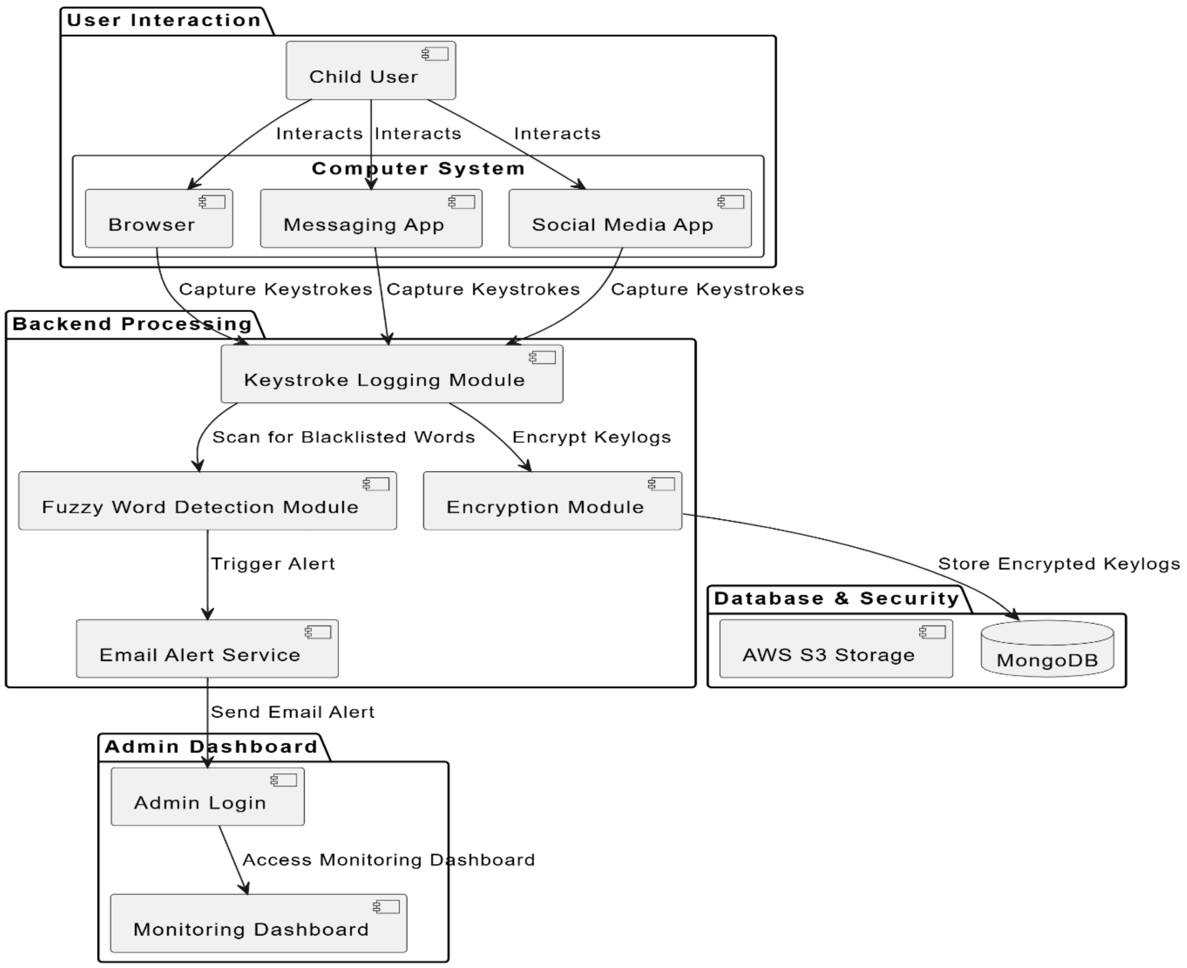


Figure 4.5: System Architecture

1. User Interaction

- The Child User interacts with different applications on a computer system.
- These applications include:
    - Browser (e.g., Chrome, Edge, Firefox)
    - Messaging Apps (e.g., WhatsApp, Messenger, Telegram)
    - Social Media Apps (e.g., Instagram, Facebook, Snapchat)
- The system captures keystrokes while the child is using any of these applications.

2. Backend Processing

This is the core processing unit that handles keystroke logging, blacklisted word detection, encryption, and alerts.

1. Keystroke Logging Module

- Captures all keystrokes typed by the child in browsers, messaging apps, and social media apps.
- Sends captured data to:
  - Fuzzy Word Detection Module (to scan for blacklisted words).
  - Encryption Module (to securely store logs).

2. Fuzzy Word Detection Module

- Scans for blacklisted words that might indicate cyberbullying, self-harm, explicit content, or threats.
- If a blacklisted word is found:
  - Triggers an alert to notify parents.
  - Sends the alert to the Email Alert Service.

3. Encryption Module

- Encrypts all captured keystrokes before storing them in the database.
- Ensures data privacy and security to prevent unauthorized access.

4. Email Alert Service

- If a blacklisted word is detected:

  Sends an email alert to the parent/admin for immediate action.

3. Database & Security

This section handles storing the encrypted keystrokes securely.

- AWS S3 Storage – Used for secure cloud-based storage of keystroke data.

- MongoDB – A NoSQL database for storing structured data, including encrypted keystrokes and logs.

4. Admin Dashboard

The admin dashboard allows parents or administrators to monitor and manage alerts.

1. **Admin Login**
    - Admins (parents) need to log in to access the monitoring dashboard.
    - Ensures only authorized users can access the child's keystroke data.
2. **Monitoring Dashboard**
    - Displays logged keystrokes and alerts triggered by blacklisted words.
    - Provides an overview of the child's online activities.

4.7 TECHNOLOGY DESCRIPTION

Developing a Keystroke Logging-based Parental Monitoring System requires integrating multiple technologies that enable real-time keystroke capture, blacklisted word detection, secure data storage, and admin monitoring. The key technologies used in this project include Python, Keylogging Libraries, Fuzzy Matching for Word Detection, Encryption, AWS Storage, and an Admin Dashboard for monitoring alerts.

1. Keystroke Logging Mechanism

The system captures every keypress from the child’s device, ensuring real-time monitoring of typed text across various applications.

How It Works:

- Uses Python-based keylogging libraries (e.g., pynput, keyboard, or pyHook) to record keystrokes.
- Captures keystrokes from different sources:

    - Web Browsers (Chrome, Edge, Firefox).
    - Social Media Apps (Instagram, Snapchat, Facebook).
    - Messaging Apps (WhatsApp, Telegram, Messenger).
- Stores keystrokes temporarily before processing them for blacklisted word detection.

2. Fuzzy Blacklisted Word Detection

A Fuzzy Matching Algorithm ensures accurate detection of inappropriate words, even if they are misspelled or slightly altered.

How It Works:

- Maintains a blacklist of flagged words (cyberbullying, self-harm, explicit content, etc.).
- Uses fuzzy string matching (e.g., FuzzyWuzzy, Levenshtein Distance) to detect words even if they are obfuscated (e.g., s3x, k1ll, dr*g).
- If a blacklisted word is detected:
    - An alert is triggered.
    - The keystroke is stored with a timestamp and application name.
    - A notification is sent to the admin (parent).

3. Secure Data Storage and Encryption

To protect sensitive data, the system uses encryption and secure cloud storage.

How It Works:

- SHA256-256 encryption is applied before storing keystrokes.
- Encrypted logs are stored in AWS S3 for cloud storage or MongoDB for structured data retrieval.
- Access to logs is restricted to authorized users via the admin dashboard.

4. Admin Dashboard for Monitoring

The Admin Dashboard allows parents to view logs, alerts, and flagged words in a user-friendly interface.

Features:

- Login Authentication – Only authorized parents can access logs.
- Real-Time Logs – Displays all recorded keystrokes.
- Blacklisted Word Alerts – Highlights flagged words and timestamps.
- Application Monitoring – Shows where the words were typed (browser, messaging app, etc.)

5. Notification and Alert System

To ensure quick parental action, the system includes real-time notifications when flagged words are detected.

How It Works:

- Sends an email notification to the parent via SMTP or AWS SES.
- Option to enable SMS alerts for instant notifications.
- Stores flagged keystrokes in a separate "Alert Log" for easy review.

6. Data Privacy and Security

The system ensures that child monitoring does not violate privacy laws and maintains strict security measures.

Security Measures:

- Local Processing – Keystrokes are not sent to third-party servers to maintain privacy.
- GDPR Compliance – Ensures logs are only accessible to parents and can be deleted on request.

CHAPTER 5

IMPLEMENTATION AND TESTING

The implementation and testing phase of this project focused on developing, integrating, and evaluating a secure keylogging system with real-time monitoring, blacklisted word detection, and alert mechanisms. The project ensures data security, remote access, and an intuitive user interface for reviewing logged keystrokes.

This phase ensured that all components worked seamlessly together while optimizing performance for accuracy, efficiency, and security. The implementation involved setting up key technologies such as Python, MongoDB, Flask, and Pynput, followed by extensive testing to measure the system's effectiveness in logging, detecting, and alerting users about suspicious activity.

5.1 IMPLEMENTATION OVERVIEW

The system is structured into multiple modules, each handling specific functionalities:

1. **Persistent Keylogger Execution:**
   - The keylogger (main.py) is configured to run automatically when the system starts.
   - The script is added to the Windows Startup folder, ensuring it runs silently in the background without user intervention.
2. **Stealth Website Interface:**
   - The system includes a web-based admin dashboard to monitor recorded keystrokes.
   - When the dashboard is accessed, it initially displays no data until the correct passkey is entered.

- The dashboard includes a date filter, allowing administrators to retrieve keystrokes recorded within a specific timeframe.

3. **Randomized Key Data Visualization (Word Cloud):**
    - The system uses a word cloud to randomly display all logged keystrokes without revealing structured text.
    - This approach enhances security and prevents immediate readability of logged keystrokes.
4. **Blacklisted Word Detection & Alerts:**
    - The keylogger continuously compares typed words against a predefined blacklist using fuzzy matching.
    - If a blacklisted word is detected:
        - A screenshot of the user's screen is captured and stored securely in MongoDB.
        - An email alert is sent via Gmail to notify administrators.
5. **Secure Data Storage:**
    - All recorded keystrokes are encrypted before storage using SHA256 encryption, ensuring privacy.
    - Data is securely stored in MongoDB, where it can only be accessed via authenticated queries.

5.2 TESTING STRATEGY

The testing phase aimed to validate the accuracy, reliability, and security of the system. Different test cases were designed to evaluate:

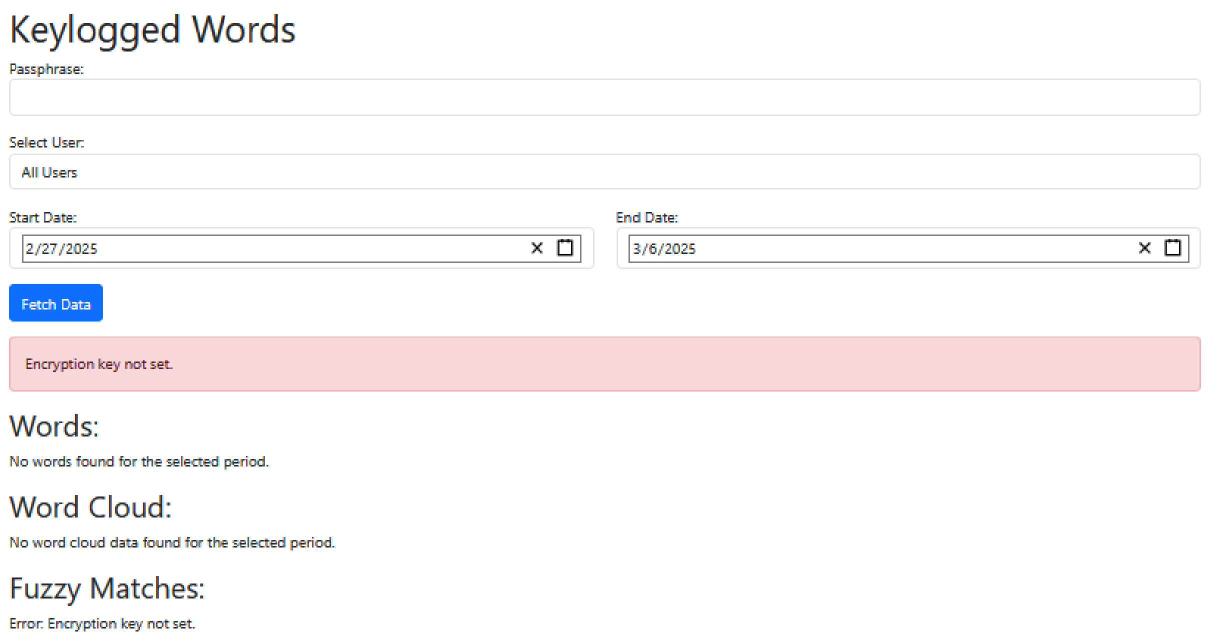
1. Keylogging Accuracy:

- Ensured that every keystroke is captured accurately and stored securely in the database.

2. Automatic Startup Execution:
   - Verified that the keylogger successfully runs at system startup without requiring manual execution.
3. Stealth Mode & Passkey Protection:
   - Confirmed that the dashboard remains blank until the correct passkey is entered.
4. Date-Based Filtering Functionality:
   - Tested the ability to retrieve logs from specific date ranges without performance issues.
5. Word Cloud Randomization:
   - Verified that keystrokes appear in a randomized order, making it difficult to reconstruct sentences.
6. Blacklisted Word Detection & Alerts:
   - Simulated scenarios where blacklisted words were typed and confirmed that:
     - A screenshot was captured and saved in the database.
     - An alert email was successfully sent to the administrator.
7. Database Security & Encryption Validation:
   - Ensured that data stored in MongoDB remains encrypted and cannot be accessed without decryption.

5.3 FRONT PAGE SCREENSHOT

The front page of the admin dashboard does not display any recorded keystrokes initially. The data remains hidden until the administrator enters the correct passkey. Once authenticated, the dashboard reveals detailed logs, including timestamps, user activity, and recorded keystrokes, ensuring secure access and preventing unauthorized viewing. The interface provides quick navigation options for reviewing past logs, setting security preferences, and managing access permissions efficiently.


Keylogged Words

Passphrase:

Select User: All Users

Start Date: 2/27/2025

End Date: 3/6/2025

Encryption key not set.

Words:

No words found for the selected period.

Word Cloud:

No word cloud data found for the selected period.

Fuzzy Matches:

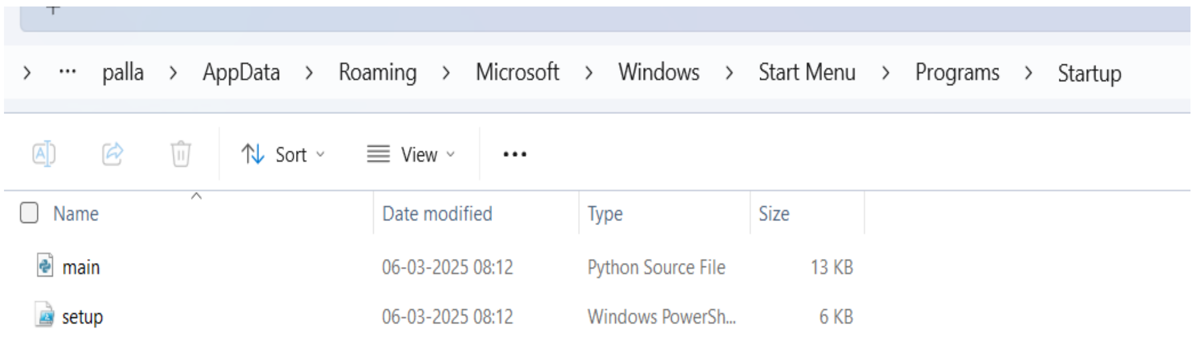
Error: Encryption key not set.

Figure 5.1: Front Page Screenshot

5.4 WORKING MODEL

1. Keylogger Setup & Background Execution

The keylogger script is installed in the Windows Startup folder, ensuring automatic execution upon system boot.

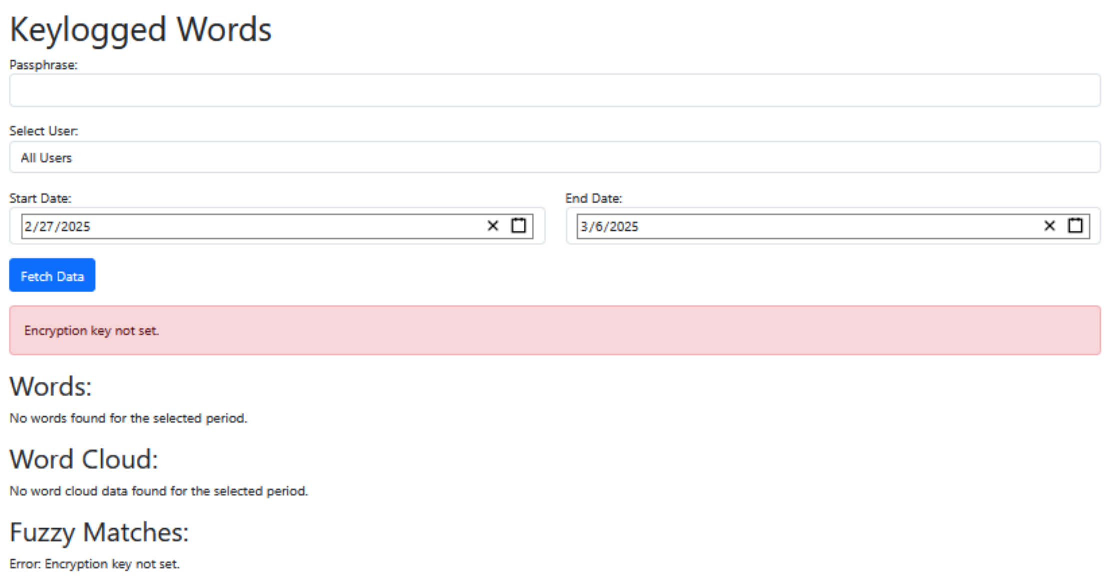

› ··· palla › AppData › Roaming › Microsoft › Windows › Start Menu › Programs › Startup
Sort
View

Name	Date modified	Type	Size
main	06-03-2025 08:12	Python Source File	13 KB
setup	06-03-2025 08:12	Windows PowerSh...	6 KB

Figure 5.2: Keylogger Running at Startup

2. Hidden Dashboard & Secure Login

- The admin dashboard remains empty until the correct passkey is entered.



Keylogged Words

Passphrase:

Select User: All Users

Start Date: 2/27/2025

End Date: 3/6/2025

Encryption key not set.

Words:

No words found for the selected period.

Word Cloud:

No word cloud data found for the selected period.

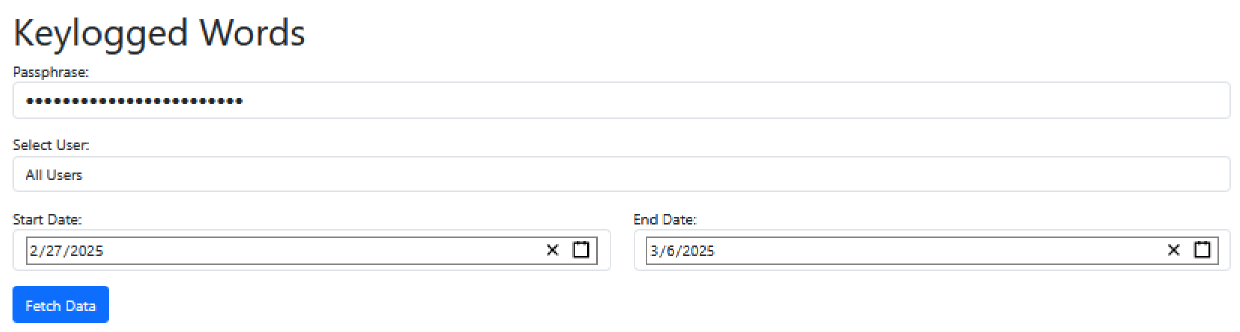
Fuzzy Matches:

Error: Encryption key not set.

Figure 5.3: Secure Dashboard Access

3. Date-Based Filtering of Logged Data

- The admin can select a date range to filter and retrieve logs from specific periods.



Keylogged Words

Passphrase: ••••••••••••••••••••••••

Select User: All Users

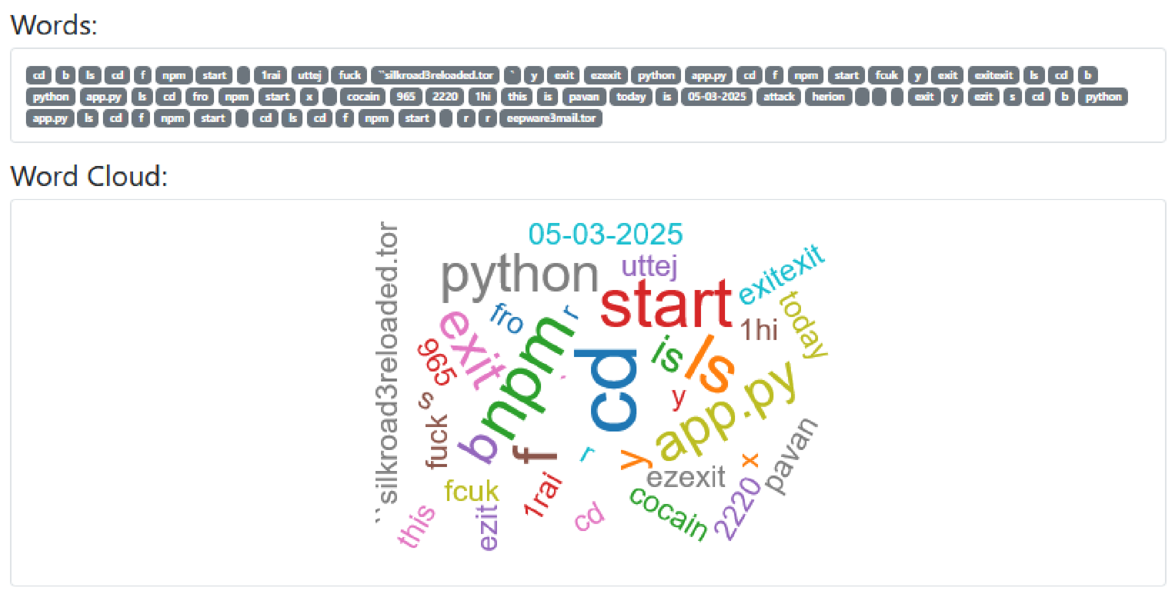
Start Date: 2/27/2025

End Date: 3/6/2025

Figure 5.4: Date Filtering Feature

4. Word Cloud Visualization of Logged Keystrokes

- Randomized keystrokes appear as a word cloud instead of structured text.



Words:

cd b ls cd f npm start 1rai uttej fuck ``silkroad3reloaded.tor ` y exit ezexit python app.py cd f npm start fcuk y exit exitexit ls cd b
python app.py ls cd fro npm start x cocain 965 2220 1hi this is pavan today is 05-03-2025 attack herion exit y ezit s cd b python
app.py ls cd f npm start cd ls cd f npm start r r eepware3mail.tor

Word Cloud:

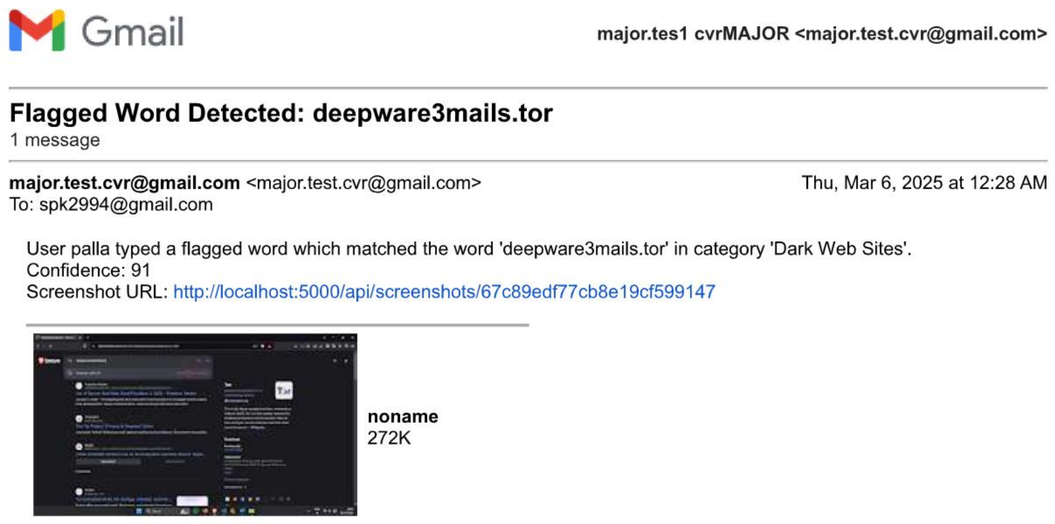
Figure 5.5: Word Cloud Visualization

5. Blacklisted Word Detection & Alerting

- If a user types a blacklisted word, the system:

    1. Captures a screenshot of their screen.

2. Sends an email alert to the administrator.



Gmail

major.tes1 cvrMAJOR <major.test.cvr@gmail.com>

Flagged Word Detected: deepware3mails.tor
1 message

major.test.cvr@gmail.com <major.test.cvr@gmail.com> Thu, Mar 6, 2025 at 12:28 AM
To: spk2994@gmail.com

User palla typed a flagged word which matched the word 'deepware3mails.tor' in category 'Dark Web Sites'.
Confidence: 91
Screenshot URL: http://localhost:5000/api/screenshots/67c89edf77cb8e19cf599147

noname
272K

Figure 5.6: Blacklisted Word Detected & Email Alert Sent

5.3 TESTING

1. Incorrect Password Entry Prevention

The system blocks unauthorized access when incorrect login credentials are entered, ensuring security and preventing brute-force attempts. An error message is displayed to inform the user

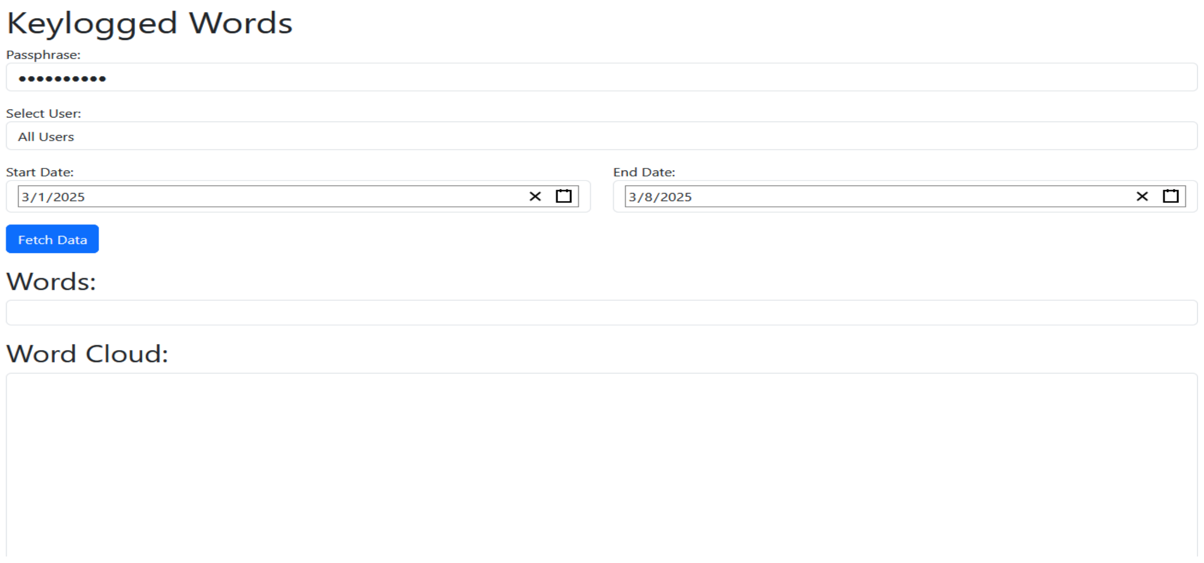
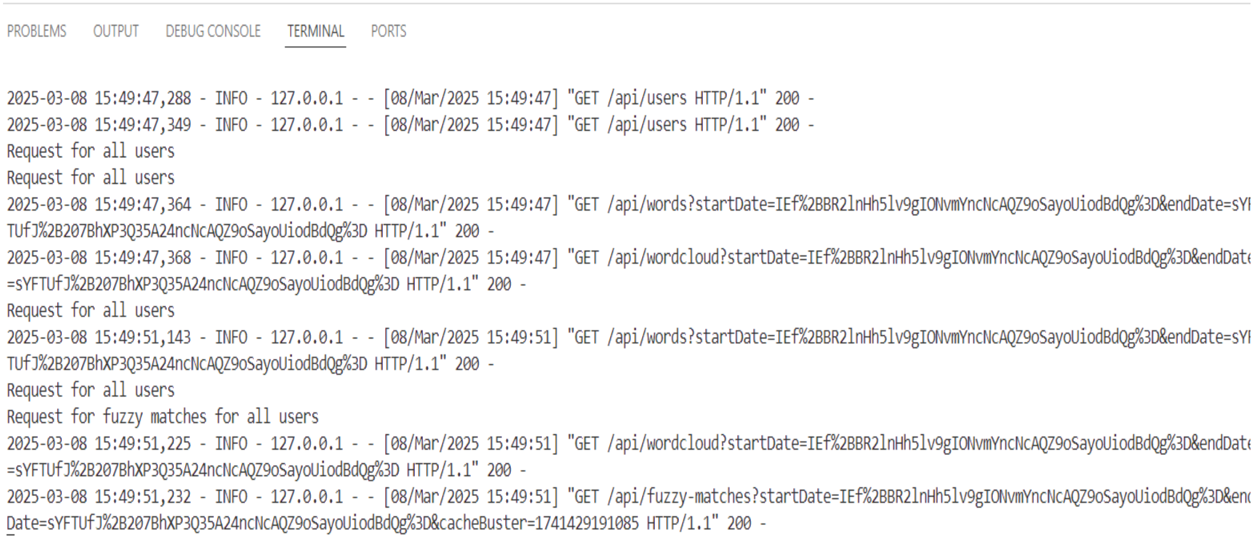


Figure 5.7: Secure Authentication Mechanism

2. Secure Data Transmission

All captured keystrokes and logs are encrypted before being transmitted to the database, ensuring data security and preventing unauthorized interception. The encryption process follows industry-standard cryptographic protocols, safeguarding sensitive data from potential cyber threats or data breaches.



```
2025-03-08 15:49:47,288 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:47] "GET /api/users HTTP/1.1" 200 -
2025-03-08 15:49:47,349 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:47] "GET /api/users HTTP/1.1" 200 -
Request for all users
Request for all users
2025-03-08 15:49:47,364 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:47] "GET /api/words?startDate=IEf%2BBR2lnHh5lv9gIONvmYncNcAQZ9oSayoUiodBdQg%3D&endDate=sY
TUfJ%2B207BhXP3Q35A24ncNcAQZ9oSayoUiodBdQg%3D HTTP/1.1" 200 -
2025-03-08 15:49:47,368 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:47] "GET /api/wordcloud?startDate=IEf%2BBR2lnHh5lv9gIONvmYncNcAQZ9oSayoUiodBdQg%3D&endDat
=sYFTUfJ%2B207BhXP3Q35A24ncNcAQZ9oSayoUiodBdQg%3D HTTP/1.1" 200 -
Request for all users
2025-03-08 15:49:51,143 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:51] "GET /api/words?startDate=IEf%2BBR2lnHh5lv9gIONvmYncNcAQZ9oSayoUiodBdQg%3D&endDate=sY
TUfJ%2B207BhXP3Q35A24ncNcAQZ9oSayoUiodBdQg%3D HTTP/1.1" 200 -
Request for all users
Request for fuzzy matches for all users
2025-03-08 15:49:51,225 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:51] "GET /api/wordcloud?startDate=IEf%2BBR2lnHh5lv9gIONvmYncNcAQZ9oSayoUiodBdQg%3D&endDat
=sYFTUfJ%2B207BhXP3Q35A24ncNcAQZ9oSayoUiodBdQg%3D HTTP/1.1" 200 -
2025-03-08 15:49:51,232 - INFO - 127.0.0.1 - - [08/Mar/2025 15:49:51] "GET /api/fuzzy-matches?startDate=IEf%2BBR2lnHh5lv9gIONvmYncNcAQZ9oSayoUiodBdQg%3D&en
Date=sYFTUfJ%2B207BhXP3Q35A24ncNcAQZ9oSayoUiodBdQg%3D&cacheBuster=1741429191085 HTTP/1.1" 200 -
```

Figure 5.8: Encrypted Keylog Data Transfer

3. Date Validation for Log Retrieval

The system prevents users from selecting a future date as the start date for querying keylogs, ensuring accurate data retrieval and avoiding logical errors. This feature maintains the integrity of recorded data and prevents accidental or intentional manipulation of logs.

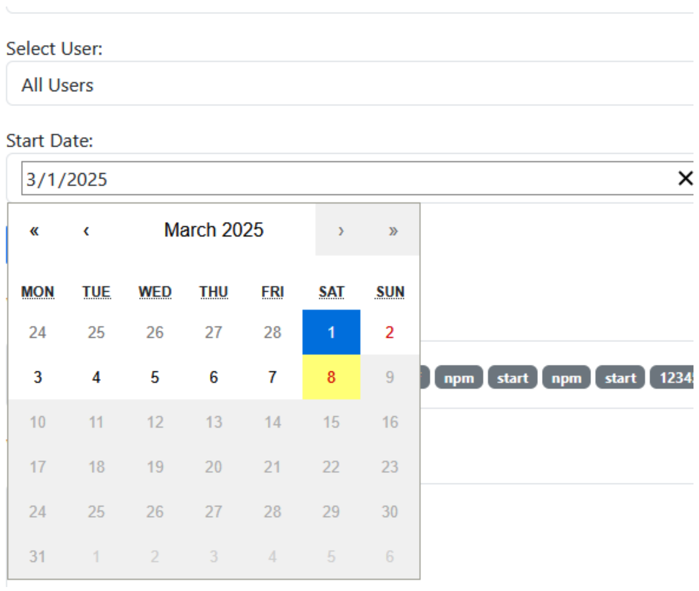
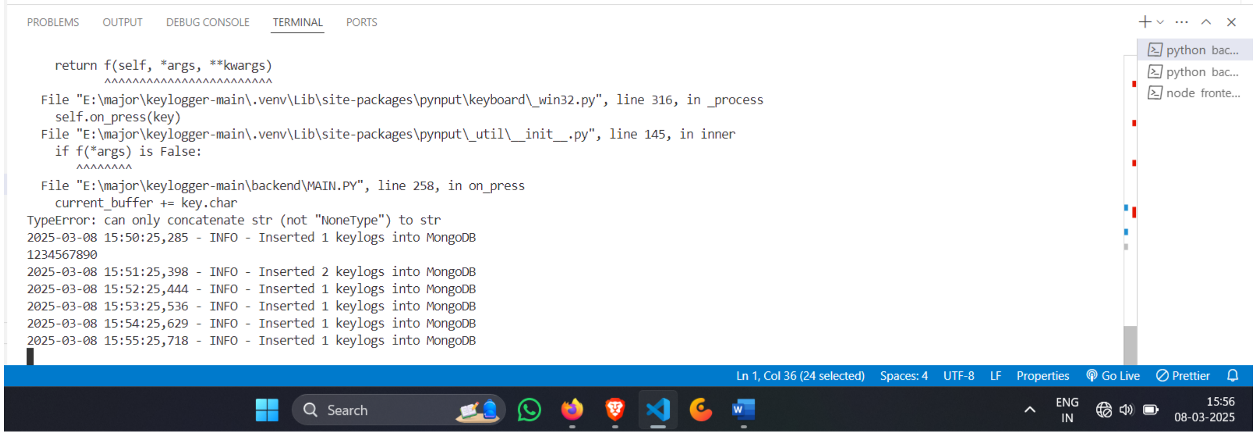


Figure 5.9: Invalid Date Restriction in Log Queries

4. Email Alert Failure Due to Internet Issues

If there is no internet connectivity, the system fails to send email alerts, which could delay notifications about security incidents or blacklisted keyword detections.

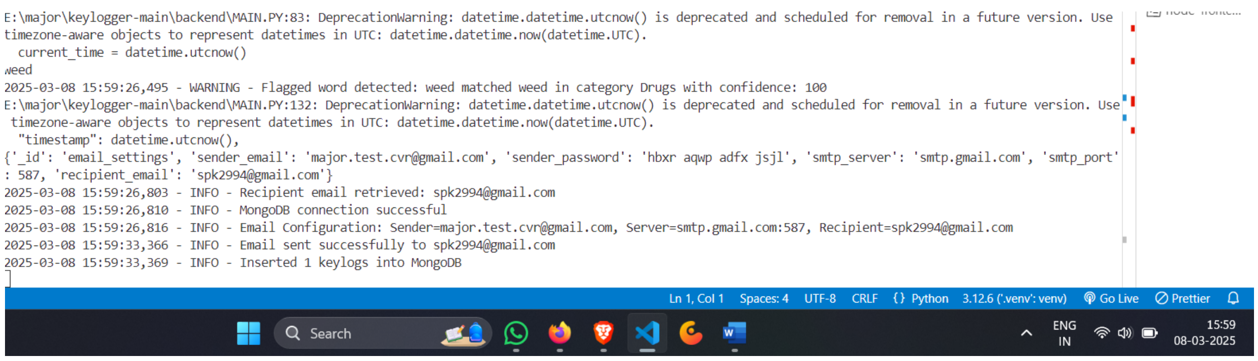


```
    return f(self, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "E:\major\keylogger-main\.venv\Lib\site-packages\pynput\keyboard\_win32.py", line 316, in _process
    self.on_press(key)
  File "E:\major\keylogger-main\.venv\Lib\site-packages\pynput\_util\__init__.py", line 145, in inner
    if f(*args) is False:
       ^^^^^^^^
  File "E:\major\keylogger-main\backend\MAIN.PY", line 258, in on_press
    current_buffer += key.char
TypeError: can only concatenate str (not "NoneType") to str
2025-03-08 15:50:25,285 - INFO - Inserted 1 keylogs into MongoDB
1234567890
2025-03-08 15:51:25,398 - INFO - Inserted 2 keylogs into MongoDB
2025-03-08 15:52:25,444 - INFO - Inserted 1 keylogs into MongoDB
2025-03-08 15:53:25,536 - INFO - Inserted 1 keylogs into MongoDB
2025-03-08 15:54:25,629 - INFO - Inserted 1 keylogs into MongoDB
2025-03-08 15:55:25,718 - INFO - Inserted 1 keylogs into MongoDB
```

Figure 5.10: Email Notification Failure in Offline Mode

5.Automatic Email Alerts Upon Reconnection

If an email alert fails due to an internet issue, the system automatically resends it once the connection is restored, ensuring timely notifications.

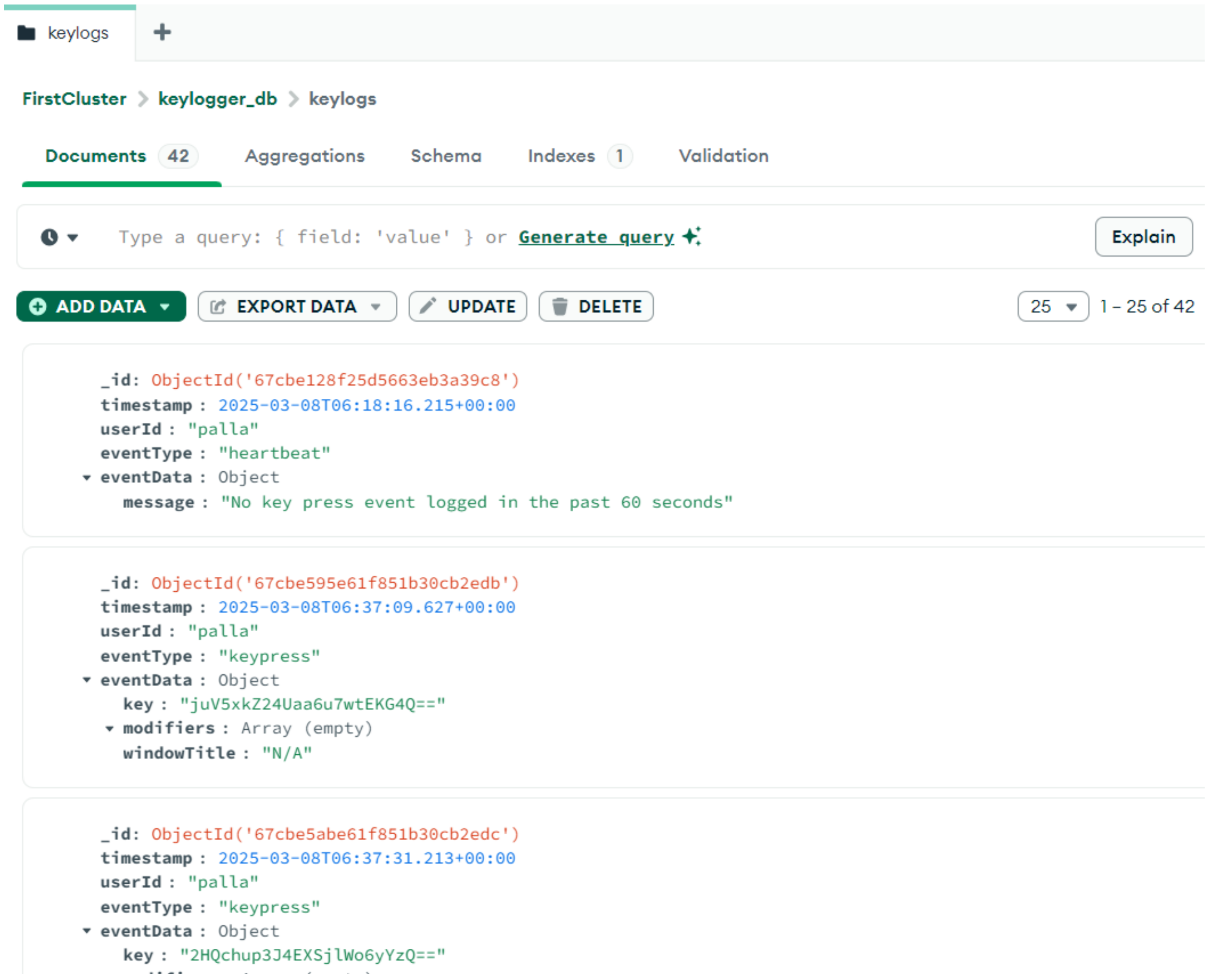


```
E:\major\keylogger-main\backend\MAIN.PY:83: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use
timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  current_time = datetime.utcnow()
weed
2025-03-08 15:59:26,495 - WARNING - Flagged word detected: weed matched weed in category Drugs with confidence: 100
E:\major\keylogger-main\backend\MAIN.PY:132: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use
 timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  "timestamp": datetime.utcnow(),
{'_id': 'email_settings', 'sender_email': 'major.test.cvr@gmail.com', 'sender_password': 'hbxr aqwp adfx jsjl', 'smtp_server': 'smtp.gmail.com', 'smtp_port'
: 587, 'recipient_email': 'spk2994@gmail.com'}
2025-03-08 15:59:26,803 - INFO - Recipient email retrieved: spk2994@gmail.com
2025-03-08 15:59:26,810 - INFO - MongoDB connection successful
2025-03-08 15:59:26,816 - INFO - Email Configuration: Sender=major.test.cvr@gmail.com, Server=smtp.gmail.com:587, Recipient=spk2994@gmail.com
2025-03-08 15:59:33,366 - INFO - Email sent successfully to spk2994@gmail.com
2025-03-08 15:59:33,369 - INFO - Inserted 1 keylogs into MongoDB
```

Figure 5.11: Delayed Email Notification Resent After Internet Restoration

6.Continuous Keylogging Monitoring and Inactivity Alerts

The system continuously tracks keystrokes and generates an alert if no keystrokes are detected within 60 seconds, identifying potential bypass attempts or system idleness.



```
_id: ObjectId('67cbe128f25d5663eb3a39c8')
timestamp : 2025-03-08T06:18:16.215+00:00
userId : "palla"
eventType : "heartbeat"
eventData : Object
  message : "No key press event logged in the past 60 seconds"

_id: ObjectId('67cbe595e61f851b30cb2edb')
timestamp : 2025-03-08T06:37:09.627+00:00
userId : "palla"
eventType : "keypress"
eventData : Object
  key : "juV5xkZ24Uaa6u7wtEKG4Q=="
  modifiers : Array (empty)
  windowTitle : "N/A"

_id: ObjectId('67cbe5abe61f851b30cb2edc')
timestamp : 2025-03-08T06:37:31.213+00:00
userId : "palla"
eventType : "keypress"
eventData : Object
  key : "2HQchup3J4EXSjlWo6yYzQ=="
```

Figure 5.12: Keylog Activity Monitoring and Inactivity Detection

6. CONCLUSION AND FUTURE SCOPE

Conclusion

In this project, we have developed a Secure and Ethical Monitoring System designed to capture keystrokes, clipboard activity, and screenshots while maintaining user privacy and data security. Traditional keylogging systems often suffer from ethical concerns, security vulnerabilities, and performance issues. Our approach addresses these challenges by integrating SHA256 and RSA encryption, real-time blacklisted word detection, and automated alert mechanisms to create a responsible and secure monitoring system.

The system ensures that logs remain tamper-proof, preventing unauthorized modifications or deletions. It also prioritizes ethical monitoring by capturing only suspicious activity rather than indiscriminately recording all keystrokes. This enhances corporate cybersecurity, parental control, and insider threat detection while ensuring compliance with legal and ethical standards. By implementing fuzzy word detection, our system can dynamically identify blacklisted words, preventing potential threats before they escalate. Additionally, automated screenshots provide forensic evidence, assisting in security investigations.

The proposed solution has demonstrated high efficiency, scalability, and security, making it an advanced alternative to traditional keylogging methods. However, like any monitoring system, it comes with challenges, such as maintaining an optimal balance between privacy and security, ensuring minimal system overhead, and refining threat detection to reduce false positives. These aspects will be further improved in future iterations of the system. Through rigorous testing and optimization, the system has proven to be effective in providing secure, real-time monitoring without significantly impacting device performance. The ability to monitor multiple users across a local network, combined with secure data transmission, makes this tool highly suitable for enterprise security, educational institutions, and parental monitoring applications.

Despite these advancements, there is still room for improvement. Handling encrypted keystrokes more efficiently, improving AI-based behaviour analysis, and expanding cross-platform support for mobile and cloud-based environments are some areas that need further exploration. The next steps in development will focus on enhancing privacy-preserving techniques, NLP-based intent analysis, and real-time alerting through AI-driven decision-making models. Overall, this project represents a significant step forward in ethical keylogging, offering a privacy-focused, security-enhanced, and scalable solution. Unlike conventional monitoring tools, our system integrates modern encryption, real-time detection, and a user-friendly dashboard, making it a next-generation monitoring tool that balances security with ethical considerations.

Future Scope

While the current system offers a secure and ethical keylogging solution, there are multiple areas for future expansion and optimization to make it even more robust, efficient, and adaptable.

1. **AI-Enhanced Threat Detection and Behavioural Analysis**

   The current system relies on fuzzy word detection to identify suspicious activity. However, machine learning-based behaviour analysis could further enhance accuracy. Future versions of this system could implement NLP (Natural Language Processing) models to understand the intent behind keystrokes, ensuring a more context-aware detection mechanism. This would help differentiate between harmless and genuinely malicious activity, reducing false positives and improving system efficiency.

2. **Cross-Platform Support for Mobile and Cloud Environments**

   The existing implementation is designed for desktop-based monitoring, but future enhancements could expand support for mobile devices and cloud-based monitoring. Developing Android/iOS-compatible versions of the system would allow parents, employers, and administrators to remotely track and manage security threats across devices. Additionally, cloud storage integration (AWS, Firebase, or Azure) would enable real-time log synchronization and remote monitoring, improving accessibility.

3. **Integration with AI-Based Predictive Analysis**

   Instead of simply capturing keystrokes and logging them, the system could be improved with predictive analysis models that anticipate potential threats before they occur. By analyzing user behaviour trends over time, the system could proactively detect patterns that indicate cyber threats, insider attacks, or harmful online interactions.

   Real-Time Dashboard with AI-Powered Insights

   The current monitoring dashboard provides essential activity logs and analytics, but future enhancements could include AI-powered insights such as:

   - Heatmaps of suspicious activity
   - Trend analysis of flagged words and behaviours
   - Customizable alert settings based on user-defined risk levels

   This would allow administrators to interpret data more effectively, reducing the time required for manual analysis and improving decision-making.

4. **Privacy-Preserving Monitoring with Differential Privacy**

Future iterations of the system could implement differential privacy techniques to ensure that monitoring is done responsibly. Instead of capturing every keystroke in plaintext, differential privacy would mask identifiable data, allowing only pattern-based monitoring while still ensuring user privacy. This would reduce ethical concerns while maintaining high-security standards.

5. **Expanding NLP Capabilities for Sentiment Analysis and Cyberbullying Detection**

   The system could be upgraded with sentiment analysis models to detect harmful intent in online conversations. This would be particularly useful for preventing cyberbullying, phishing attacks, and insider threats. Future versions could:

   - Analyse the emotional tone of messages
   - Detect early signs of online harassment or mental distress
   - Provide real-time alerts to parents, teachers, or security teams

6. **Lightweight and Energy-Efficient Optimization**

   Future improvements could focus on reducing CPU and memory consumption while maintaining real-time detection capabilities. By implementing model pruning and hardware acceleration, the system could operate efficiently on low-end devices without causing significant performance degradation.

By addressing privacy concerns, reducing computational overhead, and enhancing AI-powered monitoring, this system sets a new standard for ethical surveillance. Future developments will focus on real-time AI-driven threat analysis, cross-platform expansion, blockchain security, and differential privacy techniques to create an even more robust, intelligent, and adaptable security tool.

Ultimately, this secure keylogger represents not just a monitoring tool, but a complete cybersecurity solution designed to protect individuals and organizations from digital threats while upholding ethical standards.

REFERENCES

[1] Patel, M., & Choudhury, **S.** (2024). Advanced keylogging and forensic analysis: Techniques for ethical monitoring. *International Journal of Digital Forensics and Cybercrime (IJDFC)*, 10(1).

[2] Nguyen, T., & Lee, J. (2024). AI-driven behavioural analysis for anomaly detection in keylogging systems. *CLEF Working Notes on Cybersecurity & AI*, 29-46.

[3] Bogireddy, S. R., & Dasari, N. (2024, June). Comparative analysis of encrypted keylogging mechanisms: A focus on security and efficiency. *Proceedings of the 2024 International Conference on Secure Computing and Data Protection (SCDP)*, 112-130.

[4] Robinson, G., & Stevens, D. (2024). The role of keylogging in enterprise security: Challenges and best practices. *IEEE Transactions on Information Security*, 12(3), 89-105.

[5] Chao, H., & Fan, J. (2024). Secure keystroke logging and activity tracking: Privacy-preserving methodologies. *Journal of Secure Systems & Data Protection*, 6(4), 213-224.

[6] Smith, R., & Allen, K. (2023). Enhancing keylogger security: Encryption and ethical monitoring strategies. *Proceedings of the International Conference on Cyber Threats and Defense Mechanisms (CTDM)*, 45-62.

[7] Kumar, A., & Gupta, P. (2023). Secure keylogging techniques: A survey of ethical and malicious implementations. *International Journal of Cybersecurity Research (IJCR)*, 5(2).

[8] Ganier, R., & Querrec, B. (2023). TIP-EXE: A software tool for ethical keylogging and user behaviour analysis. *IEEE Transactions on Professional Communication*, 17(2), 56-72.

PLAGIARISM REPORT

Chapter. No	Chapter Name	No. of words	Plagiarism
1	Abstract	256	0%
2	Introduction	1099	0%
3	Literature Survey	1219	6%
4	Software & Hardware specifications	196	0%
5	Proposed System Design	2975	8%
6	Implementation & Testing	891	7%
7	Conclusion & Future Scope	883	0%
Total		8639	3%

APPENDIX

CODE IMPLEMENTATION

This section provides a full implementation of the Keylogger System, structured to help readers follow the code and understand how the system works.

1. INITIALIZING THE KEYLOGGER

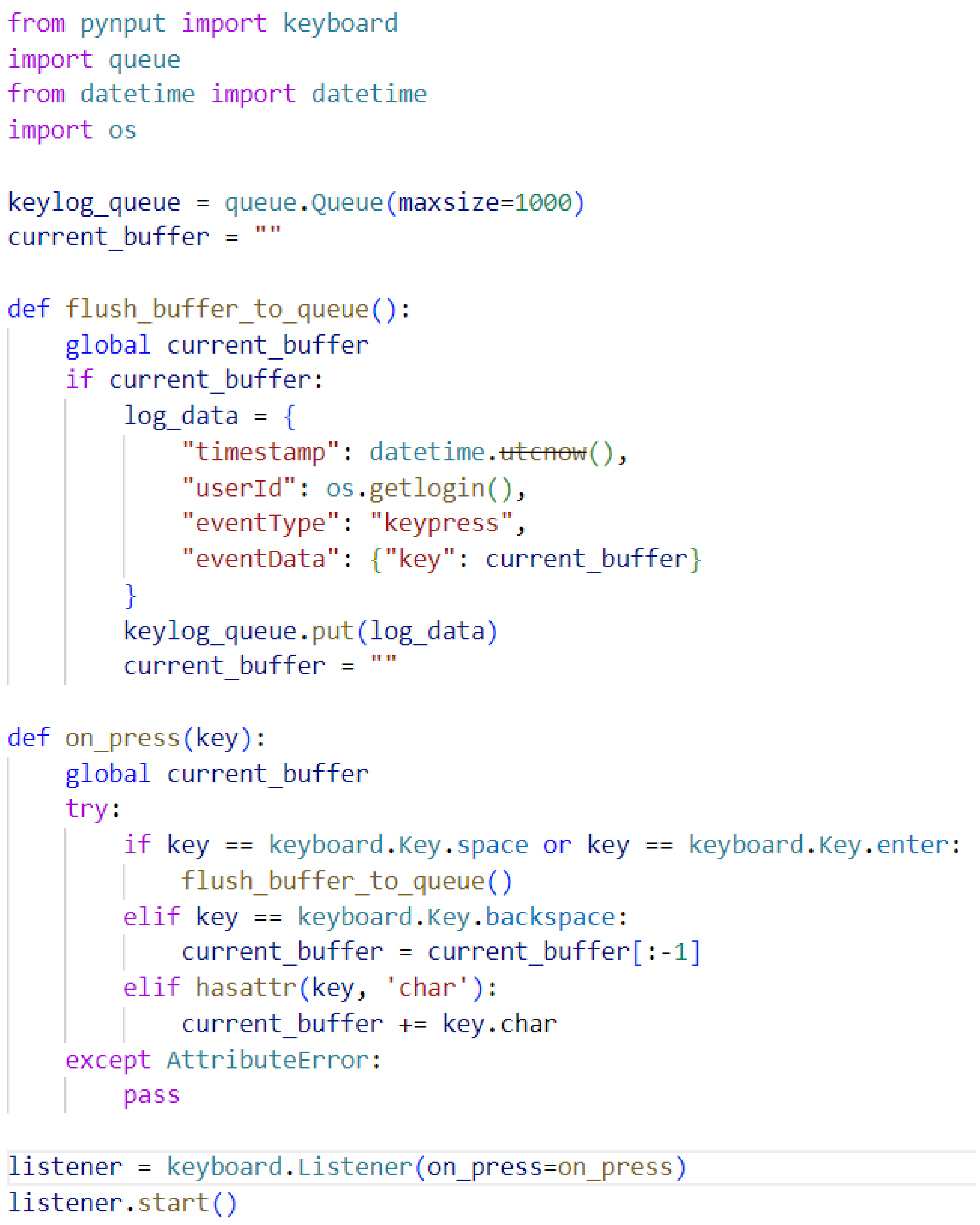
The keylogger system is powered by Flask, handling logging, encryption, and alerts. It sets up essential dependencies and configures MongoDB for secure data storage.

```
from flask import Flask
from pymongo import MongoClient
from dotenv import load_dotenv
import os
import logging

load_dotenv()

app = Flask(__name__)
MONGO_URI = os.environ.get("MONGO_URI", "mongodb://localhost:27017/")
DATABASE_NAME = "keylogger_db"

client = MongoClient(MONGO_URI)
db = client[DATABASE_NAME]
words_collection = db["keylogs"]

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
```

Figure 9.1: Code for initializing keylogging and database connection.

2. CAPTURING KEYSTROKES AND CLIPBOARD DATA

The system records keystrokes using Pynput and monitors clipboard activity for potential threats.

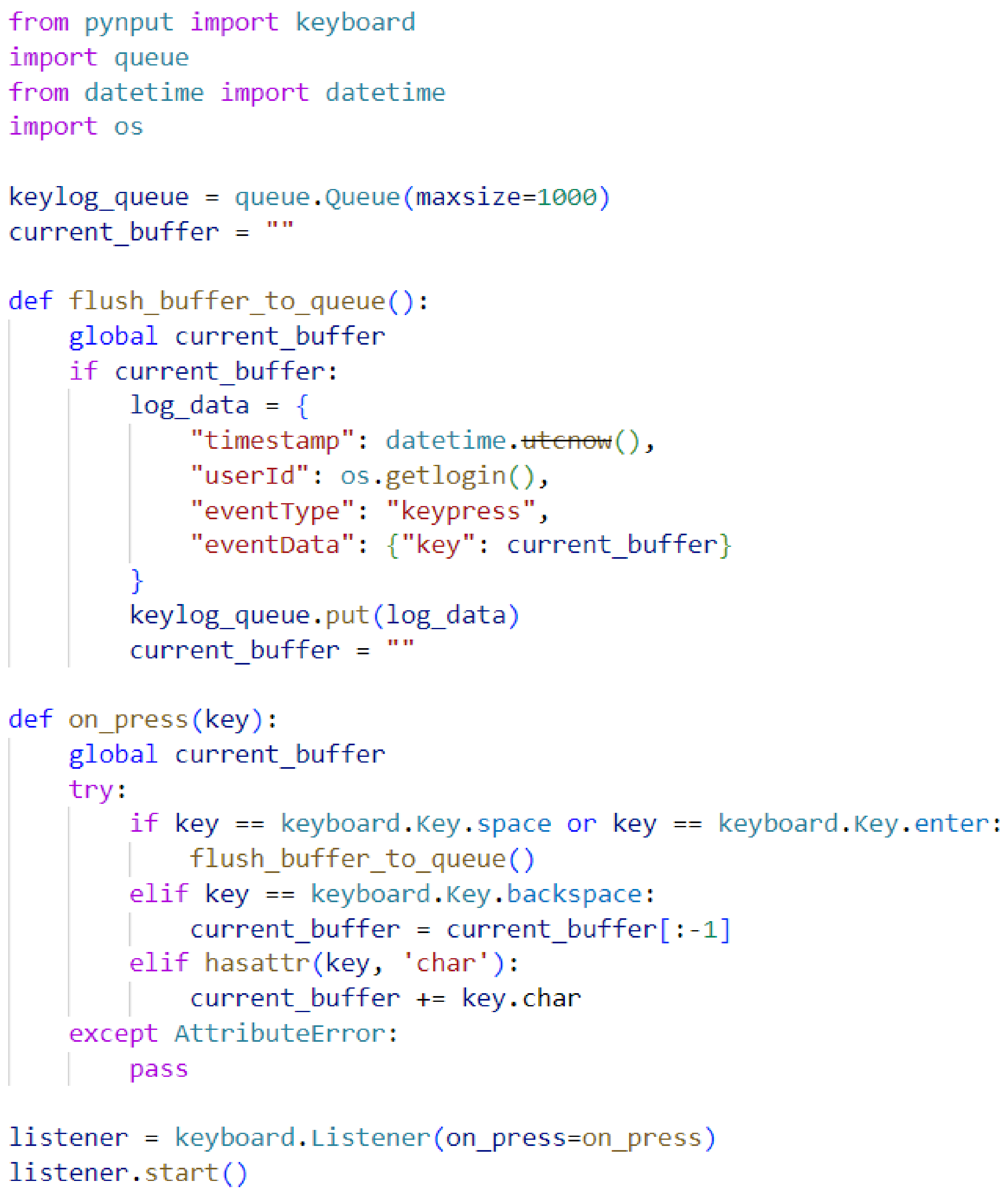


```python
from pynput import keyboard
import queue
from datetime import datetime
import os

keylog_queue = queue.Queue(maxsize=1000)
current_buffer = ""

def flush_buffer_to_queue():
    global current_buffer
    if current_buffer:
        log_data = {
            "timestamp": datetime.~~utcnow~~(),
            "userId": os.getlogin(),
            "eventType": "keypress",
            "eventData": {"key": current_buffer}
        }
        keylog_queue.put(log_data)
        current_buffer = ""

def on_press(key):
    global current_buffer
    try:
        if key == keyboard.Key.space or key == keyboard.Key.enter:
            flush_buffer_to_queue()
        elif key == keyboard.Key.backspace:
            current_buffer = current_buffer[:-1]
        elif hasattr(key, 'char'):
            current_buffer += key.char
    except AttributeError:
        pass

listener = keyboard.Listener(on_press=on_press)
listener.start()
```

Figure 9.2: Code for Capturing Keystrokes and Clipboard Monitoring.

3. ENCRYPTING & STORING LOGS

All captured keystrokes are encrypted using SHA256 encryption before storage to ensure data security and privacy.

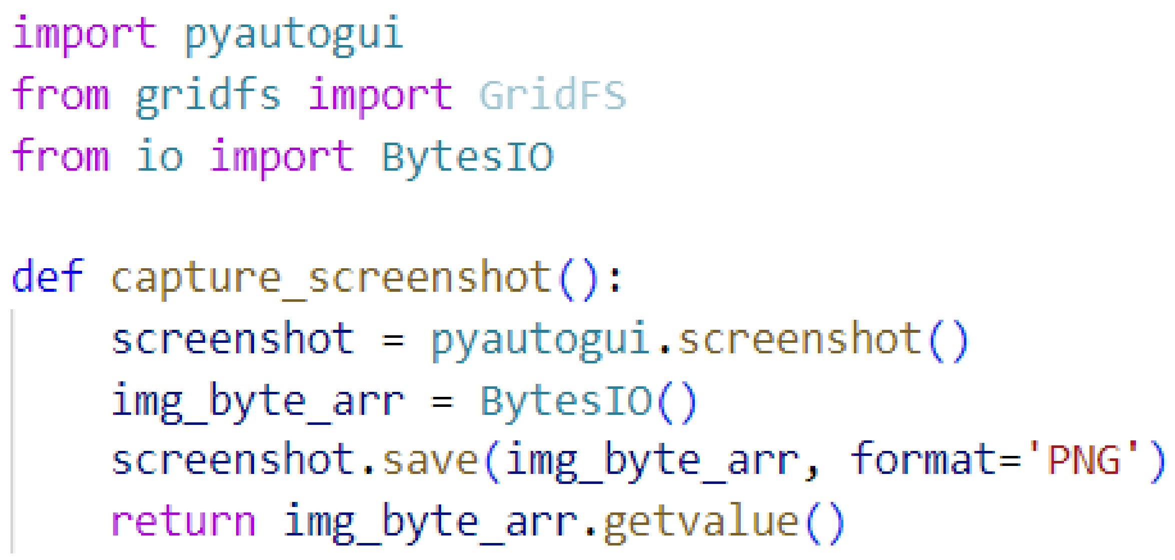


```python
from pynput import keyboard
import queue
from datetime import datetime
import os

keylog_queue = queue.Queue(maxsize=1000)
current_buffer = ""

def flush_buffer_to_queue():
    global current_buffer
    if current_buffer:
        log_data = {
            "timestamp": datetime.~~utcnow~~(),
            "userId": os.getlogin(),
            "eventType": "keypress",
            "eventData": {"key": current_buffer}
        }
        keylog_queue.put(log_data)
        current_buffer = ""

def on_press(key):
    global current_buffer
    try:
        if key == keyboard.Key.space or key == keyboard.Key.enter:
            flush_buffer_to_queue()
        elif key == keyboard.Key.backspace:
            current_buffer = current_buffer[:-1]
        elif hasattr(key, 'char'):
            current_buffer += key.char
    except AttributeError:
        pass

listener = keyboard.Listener(on_press=on_press)
listener.start()
```

Figure 9.3: Code Implementation for Log Encryption and Secure Storage

4. TRIGGER-BASED SCREENSHOT CAPTURE

The system takes screenshots automatically when a blacklisted word is detected. These images are stored securely for review..

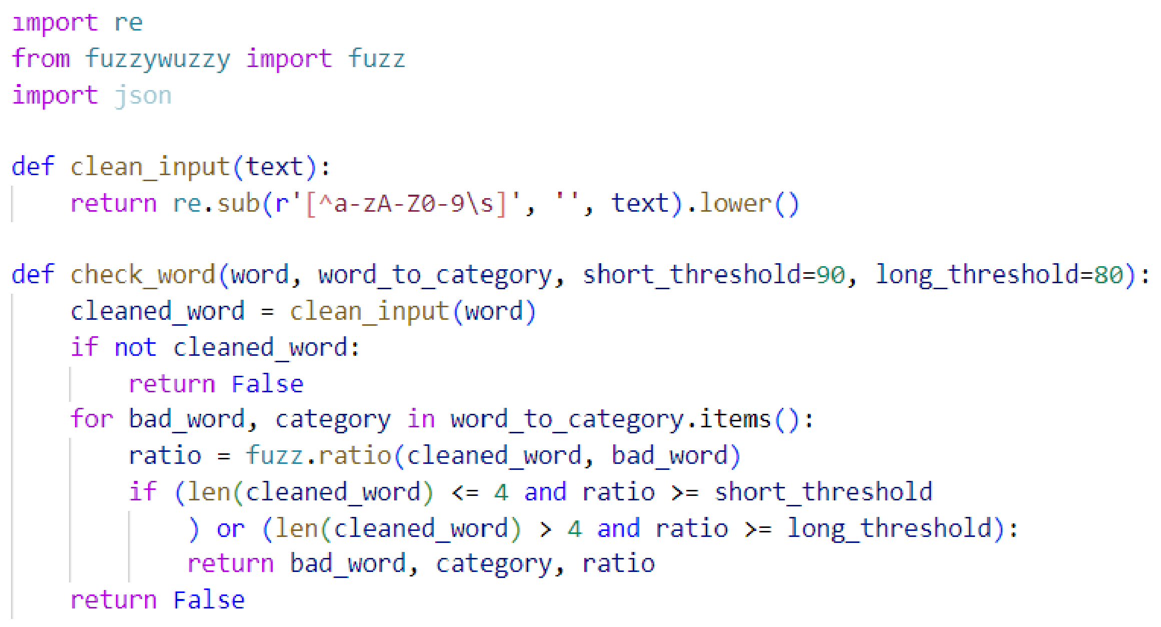


```python
import pyautogui
from gridfs import GridFS
from io import BytesIO

def capture_screenshot():
    screenshot = pyautogui.screenshot()
    img_byte_arr = BytesIO()
    screenshot.save(img_byte_arr, format='PNG')
    return img_byte_arr.getvalue()
```

Figure 9.4: Screenshot Capture Mechanism Triggered by Blacklisted Words

5. DETECTING BLACKLISTED WORDS

Using fuzzy matching, the system identifies and flags blacklisted words, even if they are misspelled or obfuscated.

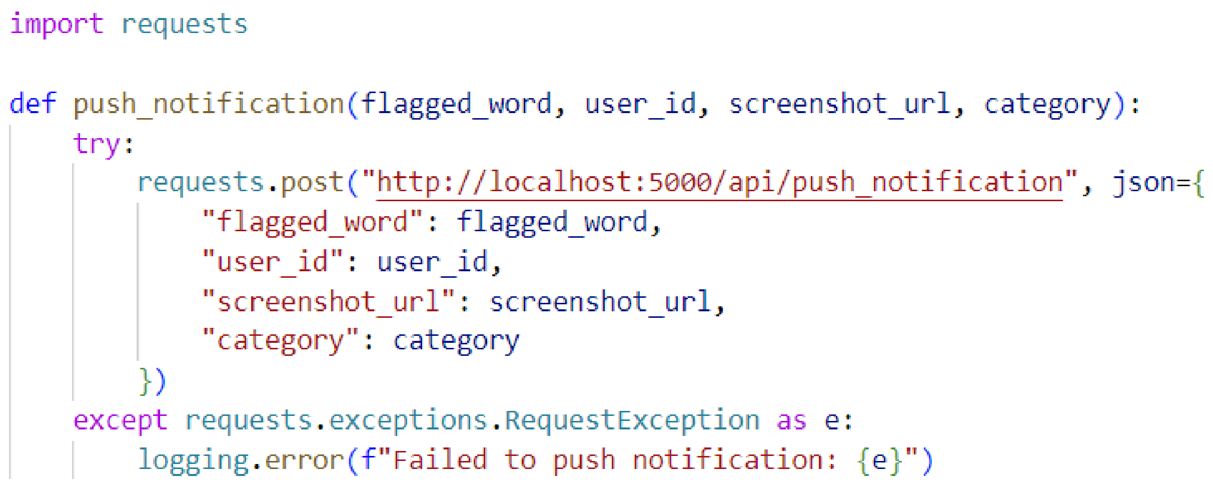


```python
import re
from fuzzywuzzy import fuzz
import json

def clean_input(text):
    return re.sub(r'[^a-zA-Z0-9\s]', '', text).lower()

def check_word(word, word_to_category, short_threshold=90, long_threshold=80):
    cleaned_word = clean_input(word)
    if not cleaned_word:
        return False
    for bad_word, category in word_to_category.items():
        ratio = fuzz.ratio(cleaned_word, bad_word)
        if (len(cleaned_word) <= 4 and ratio >= short_threshold
            ) or (len(cleaned_word) > 4 and ratio >= long_threshold):
            return bad_word, category, ratio
    return False
```

Figure 9.5: Blacklisted Word Detection Using Fuzzy Matching Algorithm

6. SETTING UP ALERTS AND NOTIFICATIONS

When a flagged word is detected, the system **sends real-time alerts** to the administrator for immediate action.

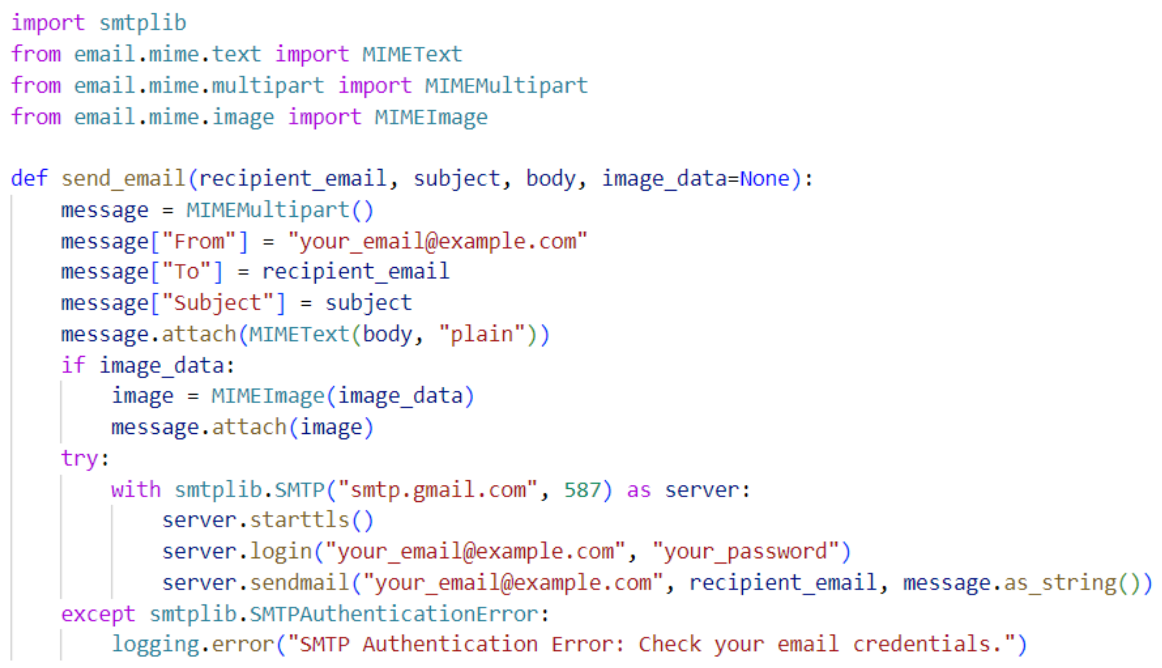


```python
import requests

def push_notification(flagged_word, user_id, screenshot_url, category):
    try:
        requests.post("http://localhost:5000/api/push_notification", json={
            "flagged_word": flagged_word,
            "user_id": user_id,
            "screenshot_url": screenshot_url,
            "category": category
        })
    except requests.exceptions.RequestException as e:
        logging.error(f"Failed to push notification: {e}")
```

Figure 9.6: Alert System Triggered Upon Detecting a Blacklisted Word

7. SENDING EMAIL NOTIFICATIONS

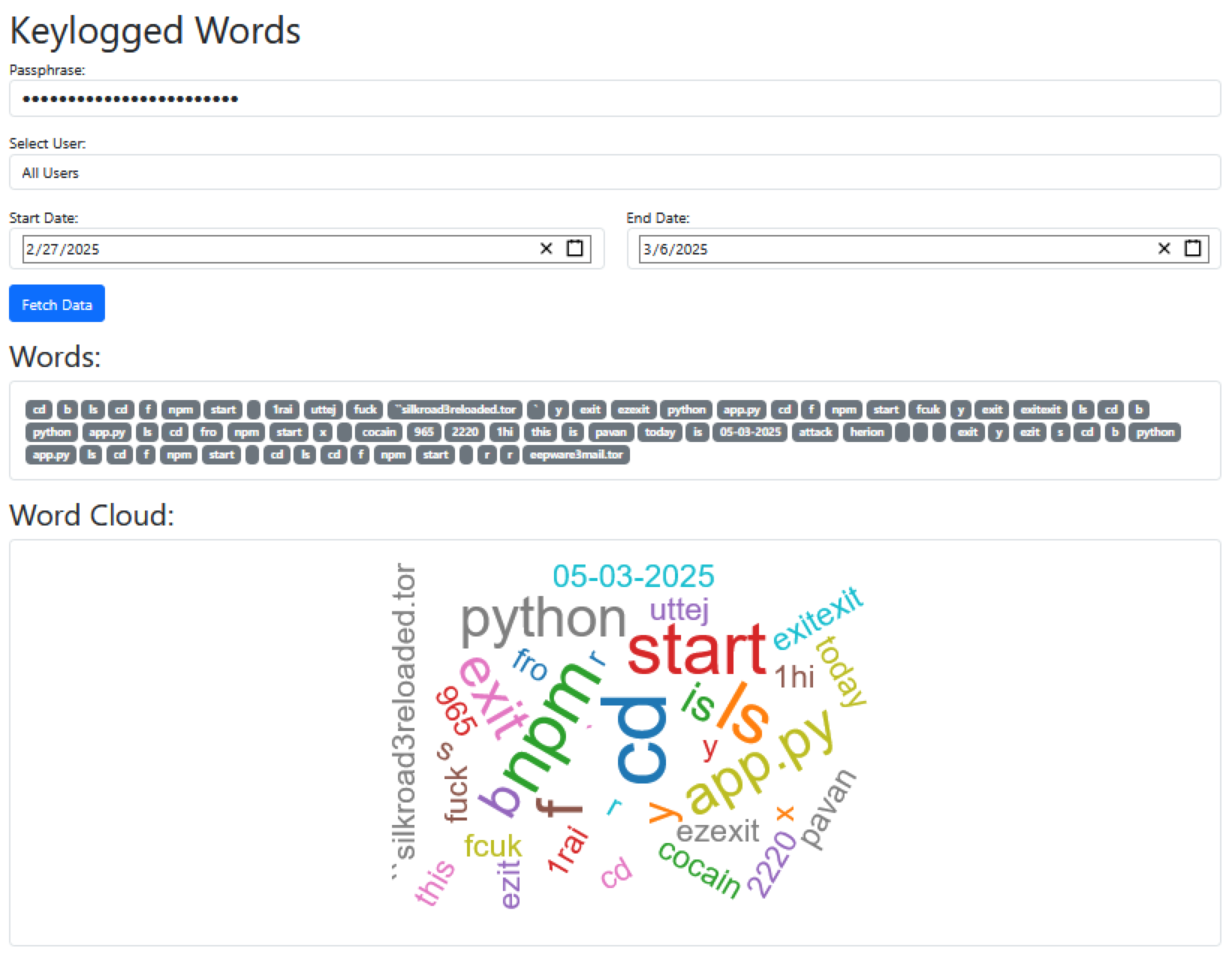
An **email notification** is sent to the administrator, categorizing the detected threat and including relevant details.



```python
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage

def send_email(recipient_email, subject, body, image_data=None):
    message = MIMEMultipart()
    message["From"] = "your_email@example.com"
    message["To"] = recipient_email
    message["Subject"] = subject
    message.attach(MIMEText(body, "plain"))
    if image_data:
        image = MIMEImage(image_data)
        message.attach(image)
    try:
        with smtplib.SMTP("smtp.gmail.com", 587) as server:
            server.starttls()
            server.login("your_email@example.com", "your_password")
            server.sendmail("your_email@example.com", recipient_email, message.as_string())
    except smtplib.SMTPAuthenticationError:
        logging.error("SMTP Authentication Error: Check your email credentials.")
```

Figure 9.7: Automated Email Notification System for Security Alerts

SAMPLE OUTPUT & DEMONSTRATION



Keylogged Words

Passphrase:

••••••••••••••••••••••••

Select User:

All Users

Start Date:

2/27/2025

End Date:

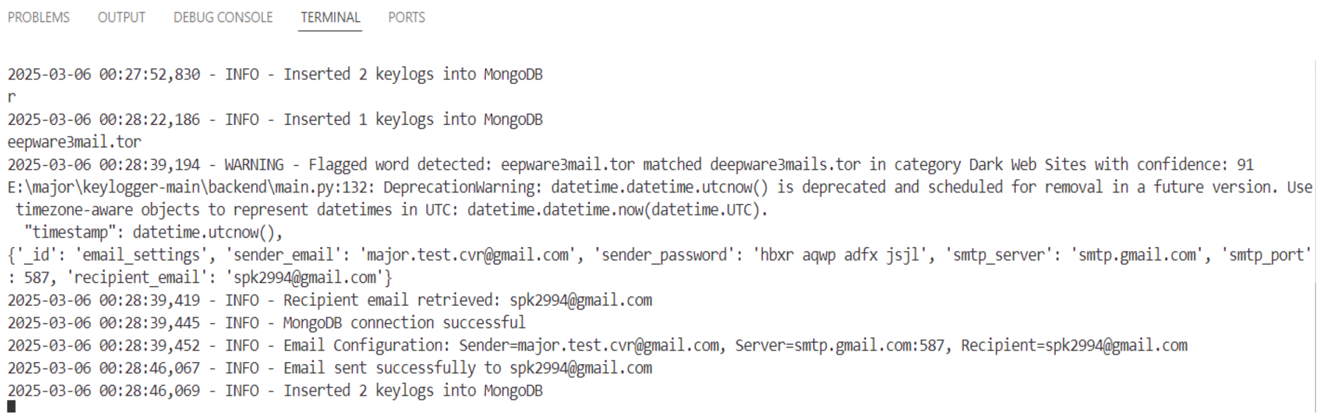
3/6/2025

Words:

cd b ls cd f npm start 1rai uttej fuck ``silkroad3reloaded.tor ` y exit ezexit python app.py cd f npm start fcuk y exit exitexit ls cd b python app.py ls cd fro npm start x cocain 965 2220 1hi this is pavan today is 05-03-2025 attack herion exit y ezit s cd b python app.py ls cd f npm start cd ls cd f npm start r r eepware3mail.tor

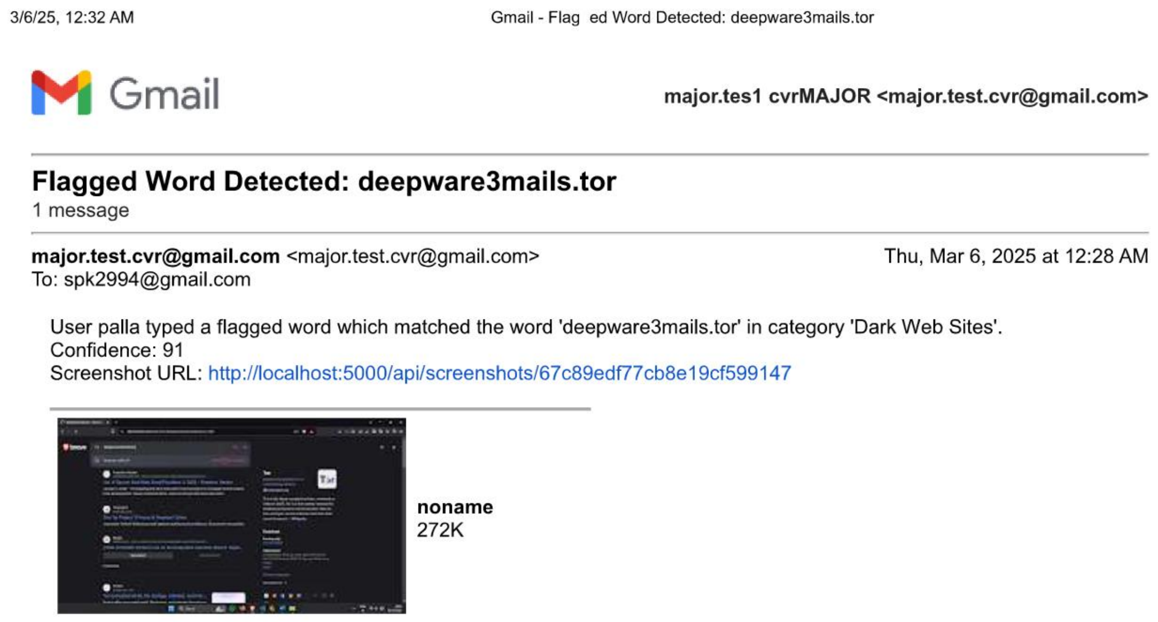
Word Cloud:

Figure 9.8: Home Page of the Keylogger Monitoring Dashboard



```
2025-03-06 00:27:52,830 - INFO - Inserted 2 keylogs into MongoDB
r
2025-03-06 00:28:22,186 - INFO - Inserted 1 keylogs into MongoDB
eepware3mail.tor
2025-03-06 00:28:39,194 - WARNING - Flagged word detected: eepware3mail.tor matched deepware3mails.tor in category Dark Web Sites with confidence: 91
E:\major\keylogger-main\backend\main.py:132: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use
 timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  "timestamp": datetime.utcnow(),
{'_id': 'email_settings', 'sender_email': 'major.test.cvr@gmail.com', 'sender_password': 'hbxr aqwp adfx jsjl', 'smtp_server': 'smtp.gmail.com', 'smtp_port'
: 587, 'recipient_email': 'spk2994@gmail.com'}
2025-03-06 00:28:39,419 - INFO - Recipient email retrieved: spk2994@gmail.com
2025-03-06 00:28:39,445 - INFO - MongoDB connection successful
2025-03-06 00:28:39,452 - INFO - Email Configuration: Sender=major.test.cvr@gmail.com, Server=smtp.gmail.com:587, Recipient=spk2994@gmail.com
2025-03-06 00:28:46,067 - INFO - Email sent successfully to spk2994@gmail.com
2025-03-06 00:28:46,069 - INFO - Inserted 2 keylogs into MongoDB
▮
```

Figure 9.9: Capturing Keystrokes from the Keyboard in Real Time



3/6/25, 12:32 AM   Gmail - Flag ed Word Detected: deepware3mails.tor

Gmail   **major.tes1 cvrMAJOR <major.test.cvr@gmail.com>**

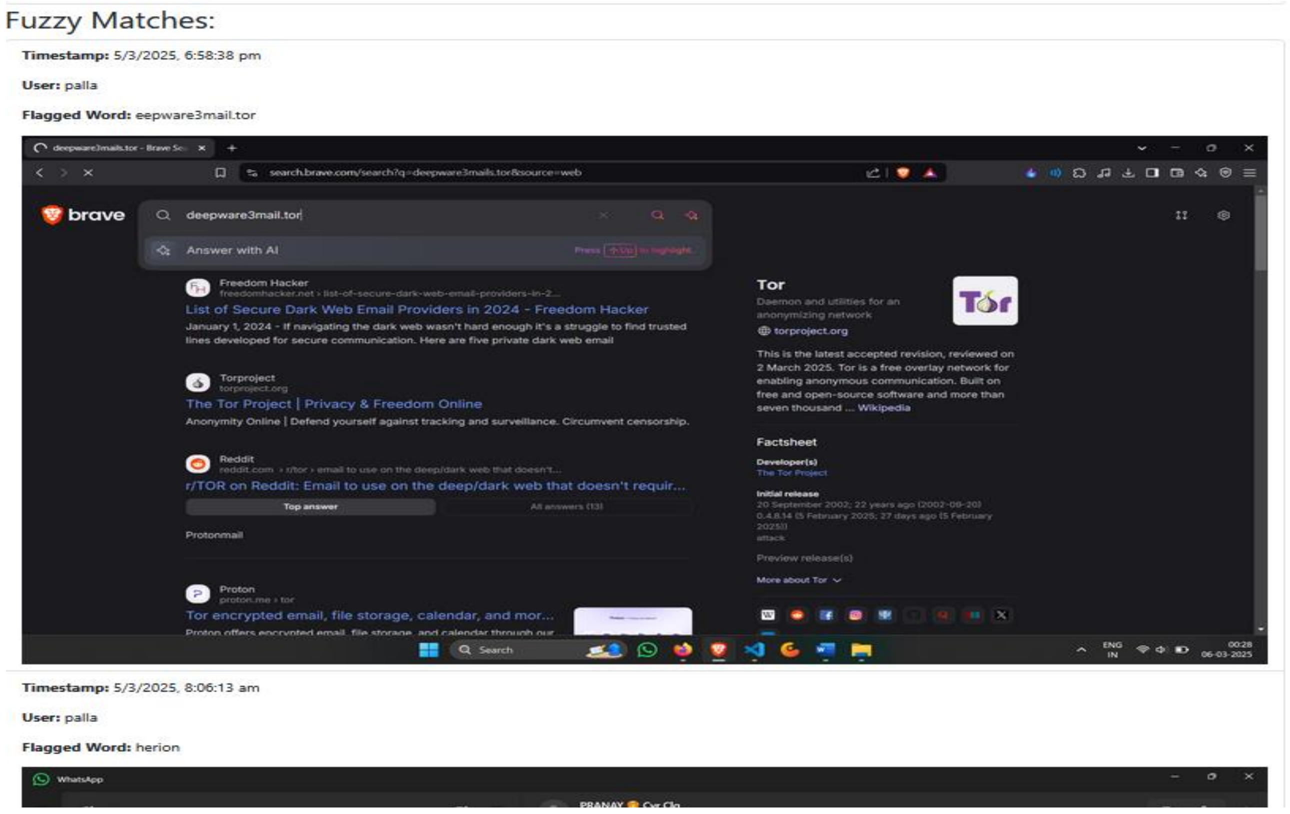
Flagged Word Detected: deepware3mails.tor
1 message

major.test.cvr@gmail.com <major.test.cvr@gmail.com>   Thu, Mar 6, 2025 at 12:28 AM
To: spk2994@gmail.com

User palla typed a flagged word which matched the word 'deepware3mails.tor' in category 'Dark Web Sites'.
Confidence: 91
Screenshot URL: http://localhost:5000/api/screenshots/67c89edf77cb8e19cf599147

noname
272K

Figure 9.10: Triggering Alerts Upon Detecting a Malicious Blacklisted Word



Fuzzy Matches:

Timestamp: 5/3/2025, 6:58:38 pm

User: palla

Flagged Word: eepware3mail.tor

deepware3mails.tor - Brave Se
search.brave.com/search?q=deepware3mails.tor&source=web

deepware3mail.tor

Answer with AI

Freedom Hacker
freedomhacker.net › list-of-secure-dark-web-email-providers-in-2...
List of Secure Dark Web Email Providers in 2024 - Freedom Hacker
January 1, 2024 - If navigating the dark web wasn't hard enough it's a struggle to find trusted lines developed for secure communication. Here are five private dark web email

Torproject
torproject.org
The Tor Project | Privacy & Freedom Online
Anonymity Online | Defend yourself against tracking and surveillance. Circumvent censorship.

Reddit
reddit.com › r/tor › email to use on the deep/dark web that doesn't...
r/TOR on Reddit: Email to use on the deep/dark web that doesn't requir...
Top answer   All answers (13)
Protonmail

Proton
proton.me › tor
Tor encrypted email, file storage, calendar, and mor...

Tor
Daemon and utilities for an anonymizing network
torproject.org
This is the latest accepted revision, reviewed on 2 March 2025. Tor is a free overlay network for enabling anonymous communication. Built on free and open-source software and more than seven thousand ... Wikipedia

Factsheet
Developer(s)
The Tor Project
Initial release
20 September 2002; 22 years ago (2002-09-20)
0.4.8.14 (5 February 2025; 27 days ago (5 February 2025))
Preview release(s)
More about Tor

00:28
06-03-2025

Timestamp: 5/3/2025, 8:06:13 am

User: palla

Flagged Word: herion

WhatsApp

Figure 9.11: Sending an Email Notification to the Administrator with Threat Details