Keylogger Detector: Enhancing User Security through Real-Time Threat Detection"

INDEX

Title:

Keylogger Detector: Enhancing User Security through Real-Time Threat Detection

Submitted by: TwinTech

1. Sakshi Shankar Jadhav (Roll number:248317)
2. Sakshi Mahesh Lanjekar (Roll number:248332)

Department Of Computer Science

College Name:-Mulund College Of Commerce (Autonomus)

Submitted to:

Digisuraksha Parhari Foundation

Date:

12th May 2025

ABSTRACT

The Keylogger Detection Tool project focuses on identifying and preventing keylogging attacks, which involve the covert recording of keystrokes on a computer or mobile device. Keyloggers are often used for malicious purposes, such as stealing sensitive information like passwords, credit card numbers, or personal messages. This project aims to develop a tool that can detect keyloggers in real-time by analyzing system processes, behavior patterns, and software interactions. The tool will use a combination of heuristic and signature-based techniques to identify suspicious activities that are typically associated with keylogging software. By providing users with an easy-to-use interface and real-time alerts, this tool helps protect against unauthorized data theft and enhances the security of personal

and organizational systems. The project will also explore different approaches for detecting keyloggers and assess their effectiveness in various scenarios. This tool is an essential part of a broader effort to improve cybersecurity and safeguard digital privacy.

Problem Statement

In today’s world, computers are used for everything from school work and communication to online shopping and banking. As a result, people often type personal and sensitive information like passwords, bank details, and private messages. Unfortunately, cybercriminals take advantage of this by using software called keyloggers. These are programs that secretly record every key you press on your keyboard, without showing any sign that they’re running. This means users often have no idea they are being watched.

The danger of keyloggers is that they can send this stolen data to someone else, which can lead to money theft, identity fraud, or other serious problems. The bigger issue is that many keyloggers are built in a way that hides them from regular antivirus software, especially the newer or customized ones. Most users don’t know how to find these hidden threats, and even tech-savvy users may miss them.

Because of this, there is a real need for a lightweight and easy-to-use tool that can check for such dangerous programs in real-time and alert users before any damage is done.

A tool like this can help everyday users protect their privacy and stay safe online.

Objective

The goal of this project is to create a software tool called a **Keylogger Detector** that helps users stay safe by finding and removing programs that secretly record keystrokes.

Here are the key aims of the project:

1. **Track Active Programs:**
   The tool will scan and watch all programs currently running on the computer.
2. **Look for Suspicious Behavior:**
   It will check for names or activities that match known keyloggers, such as tools that use words like “logger” or “hook”.
3. **Show Warnings in Real-Time:**
   If the tool finds something suspicious, it will quickly alert the user so they can take action.

4. **Allow Threat Removal:**
   Users will be able to stop and close the suspicious program directly from the tool.
5. **Display System Status:**
   The tool will also show useful system details like memory usage, number of running processes, and system uptime.
6. **Make It Simple to Use:**
   The entire tool is designed using Python and a basic window interface (Tkinter) to make it easy to use, even for people with limited computer knowledge.

The main goal is to give users a way to protect themselves against keyloggers by identifying them early and allowing safe removal—helping avoid personal data theft and improving digital safety.

Literature Review

In recent years, digital security has become a major concern due to the increasing number of online threats. Among these, **keyloggers** are one of the most dangerous because they operate silently in the background and steal user data without any visible warning. This section discusses what keyloggers are, how they work, the different types, and what previous researchers and developers have done to detect or prevent them.

What Are Keyloggers?

Keyloggers are software programs or hardware devices that are designed to record everything a user types on their keyboard. This includes login credentials, personal conversations, credit card details, and even confidential files. Once this data is collected, it can be sent to the hacker who uses it for identity theft, financial fraud, or unauthorized surveillance. Keyloggers are commonly used in cyber attacks and can be extremely hard to detect.

Types of Keyloggers

There are two main categories of keyloggers:

1. **Software Keyloggers:**
   These are programs installed secretly on a system. They work by hooking into the keyboard driver or by capturing keystrokes through system APIs. Some advanced versions can also take screenshots, track websites visited, or monitor clipboard activity.
2. **Hardware Keyloggers:**
   These are physical devices attached between the keyboard and the computer. They are harder to detect because they don’t run on the system itself. However, they require physical access to the target device, which limits their use.

Behavior of Keyloggers

Most keyloggers run silently, use very little system resources, and avoid appearing in task managers. Some even rename themselves as system processes to avoid suspicion. Others are installed as part of trojans or bundled with free software, making it difficult for average users to detect them. They often log data in hidden folders or send it directly to an external server.

Previous Research and Tools

Several studies and tools have been developed to deal with keyloggers. Here are a few key findings:

- **Antivirus Software:**
  Traditional antivirus programs scan for known threats using signature-based detection. However, many keyloggers use new names or encrypted code to avoid detection.
- **Anti-Keylogger Tools:**
  Special tools like Zemana AntiLogger and SpyShelter monitor for suspicious keyboard activity and alert the user. These tools work well but are not free and may require technical knowledge to use effectively.
- **Behavioral Analysis:**
  Some researchers have suggested using behavior-based detection, where the system looks at what a process is doing (e.g., monitoring keyboard input or logging files) instead of just checking its name. This is more effective against new or unknown keyloggers.
- **Machine Learning Techniques:**
  Recent research has introduced machine learning to detect keyloggers by identifying unusual system behavior. Although promising, these solutions often require a lot of system resources and training data, which can limit their use in everyday applications.

Gaps Identified

From reviewing the existing tools and research, we can identify several gaps:

- Many detection systems are expensive or require high system knowledge.
- Signature-based methods fail to detect new or customized keyloggers.
- Most tools do not alert users in real-time or give them control to stop the threat.
- Lightweight and beginner-friendly options are still lacking in the market.

Summary

To sum up, while there are tools and techniques available to detect keyloggers, most of them either require technical expertise or don’t effectively detect unknown threats. This project aims to fill that gap by building a lightweight, real-time keylogger detection tool using simple techniques like process scanning and pattern matching. The tool is meant to be easy to use and helpful for everyday users who want to keep their personal data safe without needing deep knowledge of cyber security.

Research Methodology

In any technical project, a well-planned and systematic approach is essential to achieve the goals successfully. For this **Keylogger Detector** project, a step-by-step research methodology was followed. This methodology includes understanding the problem, collecting data, analyzing keylogger behavior, designing the detection tool, implementing it using Python, and finally testing it in a real-world environment. Each step is explained below.

1. Problem Understanding and Requirements Analysis

The first step was to understand what keyloggers are, how they work, and why they are dangerous. We studied various types of keyloggers—both software and hardware-based—and looked into how they function on a system. We also studied user behavior and common security issues faced by non-technical users. The goal was to identify the gap between what users need (a simple detection tool) and what current tools offer (often complicated or incomplete).

Outcome:
We realized that there is a strong need for a lightweight, easy-to-use keylogger detection tool that can identify threats in real time and allow users to take action quickly.

2. Data Collection and Process Study

To build an effective detection system, it was important to study how keyloggers behave on a system. For this, we:

- Installed test keyloggers (open-source or simple Python-based keyloggers) in a safe and controlled environment (virtual machine).
- Observed their behavior using tools like Task Manager, Process Explorer, and command-line tools.
- Collected process names, file names, and common keywords used in known keylogger software.
- Identified patterns in how keyloggers run, such as:
  - Constant background activity
  - Use of suspicious process names (e.g., logger.exe, keyhook.exe)

  - Logging keystrokes into .txt or .log files
  - No visible window or interface

Outcome:
Created a list of suspicious process keywords that are commonly associated with keyloggers. These were later used in detection logic.

3. Design of Detection Algorithm

Once enough information was gathered, we designed a basic detection algorithm using the following principles:

- Monitor all active system processes.
- Check if any process name contains keywords like logger, key, hook, spy, etc.
- If a match is found, flag it as potentially dangerous.
- Display the detected process details (name, PID).
- Give the user the option to terminate the process directly from the interface.

We wanted the tool to be user-friendly and not require any deep cyber security knowledge to operate.

Outcome:
Designed a lightweight algorithm focused on pattern matching and process scanning for real-time detection.

4. Tool Development Using Python

The next step was to implement the algorithm using Python. We chose Python because:

- It is simple and beginner-friendly.
- It has powerful libraries for system monitoring (e.g., psutil, os, subprocess).
- Tkinter makes it easy to build GUI applications.

Key modules used:

- psutil for process monitoring
- tkinter for graphical user interface

- time, datetime, and os for system data
- PIL for displaying images in the interface

We added features such as:

- Start scan button
- System info display (CPU usage, memory, boot time)
- List of running processes
- Warning pop-ups if threats are found

Outcome:
Successfully developed the Keylogger Detector tool with a working GUI and detection logic.

5. Testing and Evaluation

After building the tool, we performed thorough testing in different scenarios:

- Tested with common processes to check for false positives.
- Installed dummy keyloggers to check if the tool detects them.
- Checked CPU and memory usage to make sure the tool is not heavy.
- Used the tool on different computers to check for performance and stability.

We also tested:

- Detection speed
- User response to alerts
- Accuracy of termination process

Outcome:
Tool was able to detect most test keyloggers accurately and allowed users to terminate them. A few false positives were seen (e.g., when software like AutoHotkey was running), but these could be managed with a whitelist feature in future versions.

6. User Feedback and Improvements

After internal testing, we showed the tool to a few users (friends or classmates) and asked them to test it. Based on their feedback, we made small changes to the interface and alert messages to make it more understandable and friendly.

Outcome:
Improved the tool's usability, added better progress indicators, and made error messages clearer.

Tool Implementation

Once all the research, planning, and design were complete, the next step was to bring the idea to life through actual coding. In this phase, we developed the **Keylogger Detector Tool** using Python. The process included building the program's interface, writing the logic for detecting keyloggers, and testing it on different systems. The tool was built with the goal of making it easy to use, effective in detecting keyloggers, and light enough to run on most computers without slowing them down.

Choice of Programming Language and Libraries

We selected Python for this project because it is one of the easiest programming languages to understand, and it has a wide range of libraries for both system-level tasks and creating graphical applications. Python also allows cross-platform development, meaning it can work on different operating systems, although our tool is focused mainly on Windows **systems**, as they are more commonly targeted by keyloggers.

Here are the main Python libraries we used:

- **Tkinter:** Used for building the **graphical user interface** (GUI). Tkinter is simple and built into Python, making it ideal for beginner-friendly applications.
- **psutil:** A very helpful module that lets us access information about the **running processes**, **CPU usage**, **memory**, and other system details. This is important for scanning the system for suspicious activity.
- **os and time:** Used to perform system operations and manage time-based tasks like loading animations and performance timers.
- **PIL (Pillow):** Used to display images, such as a welcome image, inside the application window.
- **datetime:** Helps to show the computer's boot time in a readable format.

Interface Design (GUI)

We designed the user interface using **Tkinter**. The GUI is made simple and clean so that even a person with very little technical knowledge can use it without any confusion. The layout was organized in such a way that every feature is clear and easy to access.

The interface contains the following main elements:

1. **Welcome Page:**
   Displays the name of the tool, a brief description, and several buttons:
   - Start Scan
   - View Running Processes
   - Show System Info
   - Exit
2. **Scanning Page:**
   When the user clicks "Start Scan," they are taken to a page where a loading bar shows the scanning progress in percentage. Labels also show estimated time and speed.
3. **Results Page:**
   After scanning is complete, this page shows a list of any suspicious processes found. It displays:
   - Process name
   - Process ID (PID)
   - The matching keyword (e.g., "logger," "hook," etc.)
4. **Terminate Button:**
   Allows users to stop any detected keyloggers from running on the system.
5. **System Information Page:**
   Shows system status such as:
   - CPU usage
   - Memory usage
   - Total number of running processes
   - System boot time
6. **Process Viewer:**
   Displays a list of all currently active processes. This feature can help users spot any unusual software even if it's not identified as a keylogger.

The interface was also made visually appealing by including a logo image on the welcome page and using consistent colors and fonts to enhance user experience.

Detection Logic (How It Works)

The main logic of the tool is to scan the system for **keyloggers** by checking the names of all running processes. Here's how it works:

- The tool uses psutil to get a list of all current processes on the computer.
- Each process name is compared to a **predefined list of suspicious keywords** related to known keylogger tools. These include:
  - keylogger
  - logger
  - spy
  - hook
  - recorder
  - refog
  - autohotkey (like ahk.exe)
  - kidlogger, logkeys, and others.
- If a match is found in the process name, it is flagged as a possible keylogger.
- The tool then alerts the user and displays the name and process ID (PID) of the detected process.
- The user is given an option to **terminate** the process directly from the tool by clicking a button.

This approach is simple but effective for detecting basic and known keyloggers. It may not catch every highly advanced or encrypted keylogger, but it's a great first step in user protection, especially for people without professional tools.

Process Termination Feature

When a suspicious process is found, the tool provides the user with an option to **terminate (stop) it** immediately. This feature is very important because it lets users take fast action instead of just being told something is wrong.

Here's how it works:

- When the user clicks "Terminate," the tool uses the psutil.Process().terminate() function.

- This tells the computer to stop that process from running.
- If successful, the user gets a message saying the process was ended.
- If it fails (for example, if the process is protected), the tool shows an error message with the reason.

This gives users more power and control over their own system, which is one of the main goals of the project.

Other Features Implemented

Besides just scanning and killing processes, we added a few more helpful features to improve the usefulness of the tool:

1. **System Health Info:**
   This gives the user real-time information about:
   - CPU usage in percentage
   - RAM (memory) usage
   - System uptime (how long the computer has been running)
2. **Process List Viewer:**
   Users can see all the currently running processes in a list. This is helpful for advanced users who want to investigate something manually.
3. **Confirmation on Exit:**
   A confirmation message appears when the user clicks the exit button. This avoids accidental closures.
4. **Progress Loader:**
   The scanning page includes a smooth progress bar and loading text to make the scan feel interactive and informative.

Testing and Debugging

After building the full tool, we tested it on several computers, including personal laptops and virtual machines. Here’s what we tested:

- Did the scan complete without crashing?
- Were the results accurate?

- Did the termination feature work correctly?
- Was the tool lightweight and fast?
- Did the interface work smoothly?

We also tested the tool by running known keylogger scripts in a safe testing environment. The tool was able to detect most of them using its keyword matching logic.

We did notice a few **false positives** (for example, it flagged harmless tools like AutoHotkey), but we considered this a good sign of the tool being cautious. In the future, we plan to improve this by adding a **whitelist feature** so users can mark safe processes.

Final Results of Implementation

After all coding, testing, and bug fixing, we were able to build a fully working **Keylogger Detection Tool** that includes:

- A simple, clean, and functional GUI
- Real-time scanning of processes
- Matching of process names against known keylogger patterns
- Ability to terminate suspicious programs
- Viewing system health and all active processes
- Error handling and user confirmations

The tool is light on resources, runs fast, and gives users the power to protect their own system without depending only on antivirus software.

Results and Observations

After developing the Keylogger Detector tool, we tested it in various conditions to evaluate how well it works in real-life situations. This section explains all the results we got from those tests and what we observed during the tool's operation. The main aim was to check whether the tool could detect keyloggers, run smoothly, and give users useful control over system security.

We carried out tests on different systems, under different scenarios, and with a few intentionally installed keylogger programs (only for safe educational testing). These results helped us understand both the strengths and the limitations of our tool.

1. Basic Functionality Test

We first checked whether the tool performs its basic tasks correctly. This included checking if:

- The tool opens without errors.
- Buttons work properly (like Start Scan, Show Processes, etc.).
- The scan runs from start to finish.
- Progress bar displays correctly.
- Results are shown after scanning.
- Detected threats can be terminated.

Observation:
The tool passed all these basic checks. It worked smoothly and without any crashes. The user interface was clean and easy to understand.

2. Detection Accuracy

To test the detection logic, we installed harmless test keyloggers (like basic Python-based ones or open-source tools such as Logkeys and KidLogger) in a safe, virtual environment. These were used to see whether the detector could identify their processes by name.

How we tested:

- Ran the tool on a system where no keylogger was installed → expected result: **no threats detected**.
- Ran the tool with a basic keylogger script active in the background → expected result: **keylogger detected**.
- Repeated with different names (like “spytool.exe,” “record.exe,” “ahk.exe”) to test name-based detection.

Observation:
The tool successfully detected most of the test keyloggers. It flagged processes containing keywords like "logger", "spy", "recorder", and "hook". The keyword-matching logic worked well. However, it also flagged tools like **AutoHotkey** that aren't always dangerous, which we noted as **false positives**.

3. False Positives and Limitations

During testing, we noticed that the tool sometimes flagged legitimate software that had names similar to those in the detection list. For example:

- AutoHotkey (ahk.exe) was flagged as a keylogger.
- Some log file creators were marked as suspicious.

Observation:
While this shows that the tool is being cautious (which is a good thing), it also points to the need for a “whitelist” feature in the future. This would allow users to mark safe programs and prevent unnecessary warnings.

4. Performance and Speed

We wanted the tool to be lightweight and fast. So, we observed how much system resources it used and how quickly it scanned.

- On average, the tool took **5–7 seconds** to complete the scan on systems with regular background processes.
- CPU usage during scanning stayed under **15–20%**, and memory usage was also low (under 100MB RAM).
- It didn’t slow down the system or affect other applications.

Observation:

The tool is **lightweight and efficient**. It works smoothly even on older computers with lower RAM or slower processors.

5. User Experience and Feedback

We gave the tool to a few non-technical users (friends, classmates) and asked them to try it. We watched how easily they could use the interface and asked them to give feedback.

Common feedback:

- Interface is simple and clear.
- Buttons are well-labeled and easy to understand.
- The scan process looks professional with the progress bar.
- It gives users a sense of control by allowing them to terminate suspicious processes.

Observation:

Users were able to run the scan, view results, and take action without needing technical help. This shows that the tool meets its goal of being **user-friendly**.

6. Process Termination Testing

After detecting a suspicious process, users can terminate it with one click. We tested this feature to check:

- Whether the process is really removed from memory.
- Whether system stability is affected after termination.

Observation:

The tool usually stopped detected processes quickly and safely, keeping the system stable. But for high-permission processes, it needed to run as administrator which is normal for Windows.

Ethical Impact & Market Relevance

Creating and using any cybersecurity tool—including a **Keylogger Detector**—comes with both ethical responsibilities and real-world relevance. In today's digital age, where online privacy and data protection are serious concerns, developing such tools not only benefits individual users but also contributes to broader digital safety. This section explains the **ethical importance** of the tool and how it fits into the **current market** for cybersecurity products.

Ethical Impact

Ethics in software development is all about making sure your product is used for the **right reasons**, and that it does not cause harm to users or systems. When it comes to keyloggers, the ethical line is very clear: keyloggers are almost always used to **invade privacy**, steal passwords, or **monitor someone without consent**, which is unethical and often illegal.

Our Keylogger Detector was designed with the **opposite purpose**—to **protect users** from such threats. Here's how it supports ethical use:

1. Protecting User Privacy

One of the most basic rights in the digital world is the **right to privacy**. Keyloggers violate this by recording everything a person types, including messages, bank details, and login credentials. Our tool helps users identify and remove these hidden programs, giving them back control over their personal information.

2. Empowering Non-Technical Users

Many computer users are not very tech-savvy. They may not know how to open Task Manager or understand what processes are running. This tool makes it easy for them to see if something suspicious is going on behind the scenes. Giving people this power is both ethical and helpful.

3. Promoting Cyber Safety

Instead of depending only on antivirus software (which may miss basic keyloggers), this tool adds another layer of safety. It encourages users to be more **proactive about their own cybersecurity**. This mindset—of checking your own system and staying aware—is exactly what modern digital ethics promotes.

4. No Harmful Intentions

The tool was not designed to spy, control, or manipulate any system. It doesn't collect any user data, and it doesn't send anything to a remote server. It is completely **offline** and **transparent**, meaning users can see exactly what it does.

5 Ethical Learning for Students

From an academic point of view, developing this tool also helped us understand how software can be misused—and how we, as ethical developers, can counter those misuses. This project taught us not only coding, but **responsible development**.

Market Relevance

Keylogger detection tools are **highly relevant in today's software market**, especially with the increasing number of cyber threats happening every day. Most people use online banking, email, shopping websites, and social media—all of which require typing in private data. This is exactly what keyloggers target.

Here's why this tool fits well into the current market:

1. Growing Cybersecurity Industry

The global cybersecurity market is growing rapidly, with businesses and individuals investing more in digital protection. According to reports, the market is expected to reach **hundreds of billions of dollars** in the next few years. Even small, specific tools like ours have a place in this industry.

2. Personal Use Demand

Most antivirus tools focus on general threats like viruses and malware. However, not many provide specific, easy-to-use tools for keylogger detection. That's where our project stands out. It offers a **simple solution** for individual users, especially for home or school use.

3. Educational Institutions and Labs

Our tool could be used in **educational settings**, such as in computer labs or classrooms, where monitoring software might be misused or unknowingly installed. Teachers or lab admins can use this tool to check systems regularly.

4. Small Offices and Startups

Small businesses may not be able to afford expensive cybersecurity packages. A lightweight and easy tool like ours can help these companies scan their systems without needing a full IT team.

5. Portable and Expandable

Since our tool is developed in Python and is lightweight, it can easily be turned into a **portable version**, or even integrated into larger software bundles in the future. This makes it suitable for market expansion or app store distribution with some improvements.

Limitations & Ethical Considerations

While our tool is helpful, it's important to understand its limitations:

- **Not 100% foolproof:** Very advanced or encrypted keyloggers may escape detection if their process names do not match known keywords.
- **False Positives:** As mentioned earlier, legitimate apps like AutoHotkey might be wrongly flagged. It is important to guide users properly so they don't remove essential processes by mistake.
- **User Responsibility:** The tool gives users power to terminate processes. This must be used responsibly to avoid shutting down critical system operations.Being open about these issues helps keep the tool ethical and promotes responsible use.

Future Scope

The development of the Keylogger Detector tool marks an important step toward improving digital safety and privacy for computer users. However, like all software, it is not perfect and can always be improved. This section looks ahead and discusses how the project can grow, what features can be added, and how it can be made even more helpful, powerful, and secure in the future.

1. Improving Detection Accuracy with AI or Heuristics

Right now, the tool uses **keyword-based detection**, where it checks for certain suspicious words (like “logger” or “spy”) in the names of running processes. This is useful, but not enough to catch more advanced or hidden keyloggers.

In the future, we can improve the tool’s brain by using:

- **Heuristic analysis:** This means analyzing how a program behaves, not just what it’s called. For example, if a program is secretly recording keystrokes or accessing keyboard inputs without permission, it could be flagged—even if its name is not suspicious.
- **Machine Learning (ML):** We can train a model using known keylogger behavior patterns and allow the tool to automatically detect new or unknown keyloggers that behave similarly.

This would make the detector smarter and reduce **false positives** and **false negatives**, improving both accuracy and safety.

2. Adding a Whitelist and Blacklist System

One of the challenges we faced during testing was that some **normal applications** were wrongly detected as threats. For example, programs like AutoHotkey, which are used for automation, were flagged just because of their process name.

To fix this, we can add two important features:

- **Whitelist:** Users can mark certain programs as "safe." These processes will be ignored in future scans, even if their names match the keylogger list.
- **Blacklist:** Advanced users or system admins can manually add suspicious programs to a list, even if they're not detected automatically.

This gives users **more control** over how the tool works and avoids unnecessary warnings.

3. Cross-Platform Support (Linux and macOS)

Currently, our tool works best on **Windows systems**, because Windows is most commonly affected by keyloggers. However, people also use **Linux** and **Mac** for personal and professional purposes.

In the future, we plan to:

- Research how processes are handled on Linux/macOS.
- Use Python's cross-platform libraries to adapt the tool.
- Make one version of the tool that can detect keyloggers across all major operating systems.

This would **expand the reach** of the tool and make it useful for even more users worldwide.

4. Real-Time Monitoring Feature

At the moment, the Keylogger Detector only works when the user **runs a scan manually**. That means if a keylogger is installed after the scan, the user won't be protected until they scan again.

A great improvement would be to add a **real-time background monitor** that:

- Watches for suspicious processes constantly.
- Sends an alert if a dangerous process starts.
- Gives the user an option to block or stop it immediately.

This would turn our project into a **real-time security tool**, like a basic anti-malware system, but lightweight and focused on keyloggers.

5. Reporting System and Threat Database

Another feature that can improve the tool is a **cloud-based threat database**. This would allow users to:

- **Report unknown threats** they've found.
- Check new processes against an **online list of reported keyloggers**.
- Get updates as more keylogger tools are discovered by the community.

This creates a kind of **community-powered security network**, which helps the tool stay up-to-date without needing large antivirus systems. It also encourages **cybersecurity collaboration**.

6. Mobile or Web-Based Companion App

Keyloggers also exist on mobile phones and tablets, especially on Android devices. In the future, we could develop a **mobile version** of this tool to help users:

- Detect apps that access the keyboard without permission.
- Monitor apps that record input in the background.
- Provide system health reports on mobile.

Additionally, a **web-based dashboard** could allow users to check the status of their computer remotely or send alerts to their phone if something suspicious is detected.

7. Better User Interface (UI) and Accessibility

Even though our current user interface is simple and clean, there's always room for improvement. In future versions, we can:

- Add **dark mode** for user comfort.
- Improve button icons and labels for clarity.
- Add **voice alerts** or **text-to-speech features** for visually impaired users.

- Translate the tool into **multiple languages** so more people can use it around the world.

These updates would make the tool more **inclusive, professional, and visually appealing**.

8. Open Source and Community Involvement

Right now, the tool is just part of a college project. But in the future, we can publish it as an **open-source project on platforms like GitHub**, so other developers can:

- Help improve the code.
- Add new features.
- Fix bugs.
- Use the tool in their own systems.

By making it open source, we can encourage learning and teamwork while improving the tool faster.

References on Keylogger Detection

1. Ortolani, S., Crispo, B. (2010). Bait Your Hook: A Novel Detection Technique for Keyloggers. RAID 2010. Available at: https://download.vusec.net/papers/raid-2010.pdf

2. Al-Naami, K. et al. (2016). Towards a Practical Unobservable Keylogger. NDSS Symposium. Available at: https://www.ndss-symposium.org/ndss2016/towards-practical-unobservable-keylogger/

3. Pawar, A. S., & Gavhane, D. N. (2023). Keylogger Detection and Prevention. IRJMETS, Vol 4, Issue 4. Available at: https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2023/37020/final/fin_irjmets1682541681.pdf

4. Kumar, R., & Vohra, R. (2014). Design and Implementation of Detection of Key Logger. IJEDR. Available at: https://www.ijedr.org/papers/IJEDR1402115.pdf

5. Karun, T. et al. (2020). Keylogger Detection using Memory Forensics and Network Monitoring. IJCA, Vol 177, No.11. Available at: https://ijcaonline.org/archives/volume177/number11/30941-2019919483/

6. How to Detect a Keylogger on a PC – YouTube (WindowsReport). Available at: https://www.youtube.com/watch?v=oEDC1N3r6w0

7. How to Detect & Remove Keylogger on Android & iPhone – YouTube (Stetson Doggett). Available at: https://www.youtube.com/watch?v=aeuQxw39yl4

8. Cybersecurity 101: How to Detect Keyloggers – YouTube (Security.org). Available at: https://www.youtube.com/watch?v=zRodnjTp3oU

9. How to Detect Keylogger on Computer? | Ethical Hacking – YouTube (Hack Pro). Available at: https://www.youtube.com/watch?v=up_laOV1DbA

10. Keylogger Data Exfiltration with Wireshark – YouTube (HackerSploit). Available at: https://www.youtube.com/watch?v=bswgaPzQEHM

11. https://youtu.be/CCGquS_rU9Q?si=ujrZx-PN-HfJ8cSq