**A Course Based Project Report**

**On**

**USB DETECTOR UTLITY APP**

**Submitted in partial fulfillment of requirement for the completion of the**

**Operating Systems Laboratory**

**II B.Tech Computer Science and Engineering**

**of**

**VNR VJIET**

**By**

**T.SUMANA – 23071A05K3**

**T.VAMSI KRISHNA – 23071A05K5**

**V.SRI HARSHITHA – 23071A05K6**

**V.NITYA AMULYA – 23071A05K7**

**V.CHAITHRA REDDY – 23071A05M0**

**2024-2025**



**VALLURIPALLI NAGESWARA RAO**

**VIGNANA JYOTHI INSTITUTE OF ENGINEERING &TECHNOLOGY**

**(AUTONOMOUS INSTITUTE)**

**NAAC ACCREDITED WITH ‘A++’ GRADE**

**NBA Accreditation for B. Tech Programs**

**Vignana Jyothi Nagar, Bachupally, Nizampet (S.O), Hyderabad 500090**

**Phone no: 040-23042758/59/60, Fax: 040-23042761**

**Email:postbox@vnrvjiet.ac.in Website: [www.vnrvjiet.ac.in](http://www.vnrvjiet.ac.in/)**

**A Project Report On**

**USB DETECTOR UTLITY APP**

**Submitted in partial fulfillment of requirement for the completion of the**

**Operating Systems Laboratory**

**II B.Tech Computer Science and Engineering**

**of**

**VNRVJIET**

**2024-2025**

**Under the Guidance**

**of**

**DR. D N VASUNDHARA**

**Assistant Professor**

**Department of CSE**





**VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING &**

**TECHNOLOGY**

**(AUTONOMUS INSTITUTE)**

**NAAC ACCREDITED WITH ‘A++’ GRADE**

**CERTIFICATE**

This is to certify that the project entitled “**USB DETECTOR UTLITY APP**” submitted in partial fulfillment for the course of Operating Systems laboratory being offered for the award of Batch (CSE-C) by VNRVJIET is a result of the bonafide work carried out by **23071A05K3**, **23071A05K5, 23071A05K6, 23071A05K7** and **23071A05M0** during the

year **2024-2025**.

This has not been submitted for any other certificate or course.

.

**Signature of Faculty Signature of Head of the Department**

# ACKNOWLEDGEMENT

An endeavor over a long period can be successful only with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We wish to express our profound gratitude to our honorable **Principal Dr. C.D.Naidu** and **HOD Dr.V.Baby**, **CSE Department**, **VNR Vignana Jyothi Institute of Engineering and Technology** for their constant and dedicated support towards our career moulding and development.

With great pleasure we express our gratitude to the internal guide **DR. D.N VASUNDHARA, Assistant Professor, CSE department** for his timely help, constant guidance, cooperation, support and encouragement throughout this project as it has urged us to explore many new things.

Finally, we wish to express my deep sense of gratitude and sincere thanks to our parents, friends and all our well-wishers who have technically and non-technically contributed to the successful completion of this course-based project.

# DECLARATION

We hereby declare that this Project Report titled “**USB DETECTOR UTLITY APP** “

submitted by us of Computer Science & Engineering in **VNR Vignana Jyothi**

**Institute of Engineering and Technology**, is a bonafide work undertaken by us and it is not submitted for any other certificate /Course or published any time before.

Name & Signature of the Students:

**T.SUMANA – 23071A05K3**

**T.VAMSI KRISHNA – 23071A05K5**

**V.SRI HARSHITHA – 23071A05K6**

**V.NITYA AMULYA – 23071A05K7**

**V.CHAITHRA REDDY – 23071A05M0**

**Date:**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S No | Topic | Pg no. |
| 1 | Abstract | 07 |
| 2 | Introduction | 08 |
| 3 | Methodology | 09 |
| 4 | Objectives | 10 |
| 5 | Flow of execution | 11 |
| 5 | Code | 12-15 |
| 6 | Output | 16-17 |
| 7 | Conclusion | 18 |
| 8 | Future Scope | 19 |
| 9 | References | 20 |

# ABSTRACT

This project focuses on enhancing the file system and device management capabilities of operating systems by simulating a utility that detects USB drive connections and disconnections in real-time. With the growing use of portable storage devices such as pen drives, external hard disks, and card readers, operating systems typically offer limited interaction or notifications beyond basic file browsing. This project addresses key challenges, including the lack of instant alerts for device connections, limited visibility into newly connected drives, and the difficulty in quickly accessing and verifying files on USB devices.

The system developed in this project offers real-time monitoring of USB devices, providing instant notifications when devices are connected or removed. It also includes a graphical user interface (GUI) that allows users to browse the contents of connected USB drives easily, improving data handling and operational control. Additionally, the system aims to enhance security by enabling users to identify potentially harmful files on USB devices. Through this utility, users can interact with USB drives efficiently, ensuring better system awareness and improved security for portable storage management.

This project aims to bridge the gap between users and the underlying operating system by offering a seamless and user-friendly way to interact with USB storage devices. By simulating real-time USB drive detection and management, it addresses the challenges of efficiently monitoring and interacting with external storage devices. The system leverages notifications to alert users immediately when a device is connected or disconnected, providing a smooth and intuitive experience. Additionally, by presenting a graphical interface, the project enhances the traditional file browsing process, allowing users to access their USB drives without having to navigate through multiple system directories manually.

The core functionality of the system revolves around detecting USB drive connections and disconnections. The application continuously monitors the system for any changes in connected devices and generates real-time notifications for users. This feature not only improves the responsiveness of the system but also enables users to quickly identify and interact with newly connected drives. Through this system, users no longer have to wait for the operating system’s default file explorer to update or manually search for connected devices, improving overall efficiency and convenience.

Security is another key aspect of this project, as it offers users more control over their connected USB devices. By providing immediate visibility into the contents of the connected drives, the system helps users verify the safety of files and folders, thus mitigating the risk of unauthorized or malicious file transfers. This feature is particularly relevant in environments where external drives may carry sensitive or potentially harmful data. Overall, this project offers a comprehensive solution for managing USB devices, enhancing both user experience and security in real-time.

# INTRODUCTION

An Operating System (OS) is the foundation of any computing device, providing essential services for computer programs and managing hardware resources. It facilitates user interaction with hardware, ensuring smooth communication between applications and system components. Among its critical functions are process management, memory allocation, device control, file system management, and security.

With the increasing reliance on external storage devices such as USB drives, external hard disks, and card readers, users often face difficulties in managing and interacting with these devices in a seamless manner. Most operating systems provide limited real-time interaction with external storage, typically offering basic file browsing without advanced notifications or monitoring capabilities. The lack of instant alerts, real-time file browsing, and security features creates challenges for users, especially in environments where security is crucial.

This project aims to address these limitations by providing a utility that detects USB drive connections and disconnections instantly, alerts users in real-time, and allows them to browse and interact with files through an intuitive graphical interface. By simulating the behavior of an OS utility for external storage management, this application enhances user experience, improves system awareness, and provides enhanced control over USB devices.

The importance of effective USB management cannot be overstated, especially in environments where external devices frequently exchange data with systems. Without a dedicated utility to manage these interactions, users often struggle with manually identifying connected devices or tracking changes in storage. This leads to inefficiencies, as users have to rely on system file explorers or other manual methods to access their USB devices, potentially wasting time. In addition, such methods fail to notify users in real-time when a device is connected or disconnected, creating missed opportunities for efficient file management. This project addresses these issues by automating the detection process, allowing users to stay informed and actively engage with their external devices without delay.

The project’s core functionality revolves around the real-time detection of USB drives as soon as they are connected to or removed from the system. Unlike traditional OS behavior, which may require users to refresh directories or wait for system recognition, this utility offers immediate feedback. By detecting these changes instantly, the system ensures that users are immediately aware of the status of their USB drives. This feature is particularly beneficial in busy environments where multiple USB devices are used frequently, such as in data analysis, troubleshooting, or file-sharing activities. It simplifies the process of monitoring external devices, providing enhanced efficiency for users who rely on USB drives for daily tasks.

Another significant benefit of this utility is the user-friendly graphical interface (GUI) that allows for easier browsing of files stored on connected USB drives. The interface is designed to be intuitive and straightforward, ensuring that users can quickly access files and folders without having to navigate through system directories manually.

# METHODOLOGY

To achieve the project goals, a systematic approach was adopted, leveraging both hardware and software components. The methodology involved the following steps:

**a)System Design and Architecture :**  
The core of the system is designed to detect USB drive connections and disconnections in real-time. This was achieved by continuously monitoring the system for any changes in the connected devices, using event-driven programming techniques. The system utilizes low-level APIs to detect and interact with USB devices, ensuring that the process is efficient and responsive.

**b)Real-Time Monitoring:**  
The application monitors USB ports for any device activity, detecting new connections and disconnections as they occur. When a USB drive is detected, the system triggers an event that notifies the user of the new connection. Similarly, when a USB drive is safely removed, the system generates a notification, ensuring users are informed of any device changes in real-time.

**c)Graphical User Interface(GUI):**  
A user-friendly GUI was developed to allow easy browsing of the contents of connected USB drives. The interface displays the available drives and their respective folders, enabling users to interact with files without navigating complex system directories. The file explorer is designed for simplicity and efficiency, with easy-to-use options for accessing, copying, or deleting files.

**d)Security Features:**  
The application integrates security mechanisms that provide users with information about the files present on the connected USB devices. The system ensures that any suspicious files can be flagged, enhancing security by preventing the accidental transfer of harmful data.

**e)NotificationSystem:**  
A notification system was incorporated to alert users when a USB device is connected or disconnected. These notifications appear in real-time, providing quick visibility of device changes without requiring the user to open file explorers manually. The notifications are configurable, allowing users to customize the level of alerts they wish to receive.

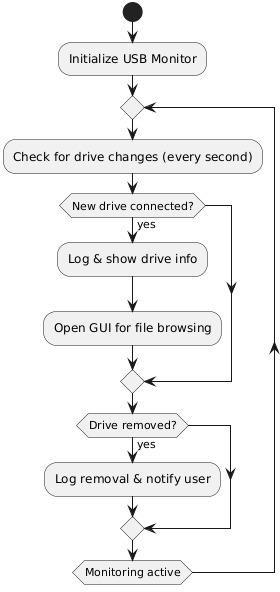
# OBJECTIVES

The primary objectives of this project are:

1. **Real-Time USB Drive Detection:**   
   The system will automatically detect and notify users of USB drive connections and disconnections instantly. By eliminating the need for manual detection or system refreshes, users will be immediately informed when a USB drive is inserted or removed, ensuring that no important data transfer goes unnoticed.
2. **User-Friendly File Browsing:**  
   The project will provide a simple and intuitive graphical interface that allows users to browse and interact with files on connected USB drives. This interface will be designed to make file management easier, enabling users to access, organize, and manage files on their USB devices without needing to navigate through complicated system folders.
3. **Enhanced Security:**  
   The system will integrate security features that scan the contents of USB devices and alert users about potentially harmful files. This includes identifying suspicious files, malware, or unauthorized transfers, which helps safeguard the system from threats and ensures that only verified and safe data is transferred to or from external devices.
4. **System Monitoring and Notification:**  
   The utility will continuously monitor USB devices for any changes, providing real-time updates. Notifications will be displayed whenever a device is connected or disconnected, ensuring users are always aware of the status of their connected storage. This real-time feedback will enhance system awareness and help prevent unauthorized or unnoticed device interactions.
5. **Improved User Experience:**  
   The system aims to simplify the process of managing external storage devices. By offering an efficient and organized way to browse, transfer, and delete files, users will experience less complexity in their interactions with USB drives. The user interface will be designed with ease of access and control in mind, making the overall experience smooth, efficient, and accessible to users of all skill levels.
6. **Customization and Personalization:**  
   The utility will allow users to personalize their notification preferences and the level of monitoring for connected devices. Users will be able to adjust settings for how and when they receive alerts, whether through pop-up messages, sound notifications, or log entries. This feature ensures that the system caters to individual preferences and provides a more tailored experience.

**FLOW OF EXECUTION**

**Flow Program Of USB DETECTOR UTLITY APP**



# IMPLEMENTATIONOFPROGRAM

**Code:**

import os

import time

import threading

import tkinter as tk

from tkinter import Listbox, messagebox, scrolledtext

import datetime

import sys

import shutil

try:

import win32file

except ImportError:

print("Win32 modules not found. Please install them using:")

print("pip install pywin32")

sys.exit(1)

try:

from colorama import init, Fore, Style

init()

COLOR\_SUPPORT = True

except ImportError:

COLOR\_SUPPORT = False

connected\_drives = set()

def list\_drives():

drives = []

bitmask = win32file.GetLogicalDrives()

for letter in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":

if bitmask & 1:

drives.append(f"{letter}:/")

bitmask >>= 1

return drives

def get\_file\_content(file\_path, binary=False):

try:

mode = "rb" if binary else "r"

with open(file\_path, mode) as f:

if binary:

content = f"Binary file, size: {os.path.getsize(file\_path)} bytes"

else:

content = f.read()

return content

except Exception as e:

return f"Error reading file: {str(e)}"

def is\_binary\_file(file\_path):

try:

with open(file\_path, 'rb') as f:

chunk = f.read(1024)

return b'\0' in chunk

except Exception:

return True

def log(message, msg\_type="INFO"):

timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

if COLOR\_SUPPORT:

color = {

"INFO": Fore.GREEN,

"WARNING": Fore.YELLOW,

"ERROR": Fore.RED,

"CONNECT": Fore.CYAN,

"DISCONNECT": Fore.MAGENTA

}.get(msg\_type, Fore.WHITE)

print(f"{color}[{timestamp}] [{msg\_type}] {message}{Style.RESET\_ALL}")

else:

print(f"[{timestamp}] [{msg\_type}] {message}")

def show\_files\_gui(drive\_path):

try:

files = os.listdir(drive\_path)

log(f"Found {len(files)} files/folders on {drive\_path}", "INFO")

except Exception as e:

log(f"Error reading drive {drive\_path}: {e}", "ERROR")

return

window = tk.Tk()

window.title(f"USB Contents - {drive\_path}")

window.geometry("800x600")

list\_frame = tk.Frame(window)

list\_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=False, padx=5, pady=5)

content\_frame = tk.Frame(window)

content\_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=5, pady=5)

path\_label = tk.Label(list\_frame, text=f"Drive: {drive\_path}", font=("Arial", 10, "bold"))

path\_label.pack(anchor="w", pady=(0, 5))

listbox = Listbox(list\_frame, width=30)

listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

scrollbar = tk.Scrollbar(list\_frame)

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

listbox.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=listbox.yview)

content\_label = tk.Label(content\_frame, text="File Content:")

content\_label.pack(anchor="w")

text\_area = scrolledtext.ScrolledText(content\_frame, wrap=tk.WORD)

text\_area.pack(fill=tk.BOTH, expand=True)

for file in files:

listbox.insert(tk.END, file)

def on\_select(event):

if listbox.curselection():

selected = listbox.get(listbox.curselection()[0])

full\_path = os.path.join(drive\_path, selected)

content\_label.config(text=f"File: {selected}")

if os.path.isdir(full\_path):

text\_area.delete(1.0, tk.END)

text\_area.insert(tk.END, f"[Directory] {selected}\n\nContents:\n")

try:

for item in os.listdir(full\_path):

text\_area.insert(tk.END, f"- {item}\n")

except Exception as e:

text\_area.insert(tk.END, f"Error listing directory: {str(e)}")

else:

is\_binary = is\_binary\_file(full\_path)

content = get\_file\_content(full\_path, binary=is\_binary)

text\_area.delete(1.0, tk.END)

text\_area.insert(tk.END, content)

log(f"Accessed: {full\_path}", "INFO")

listbox.bind('<<ListboxSelect>>', on\_select)

window.mainloop()

def monitor\_usb():

global connected\_drives

connected\_drives = set(list\_drives())

log("Starting USB detection system...", "INFO")

log(f"Currently connected drives: {', '.join(connected\_drives) if connected\_drives else 'None'}", "INFO")

while True:

time.sleep(1)

current = set(list\_drives())

if current - connected\_drives:

new\_drives = current - connected\_drives

for new\_drive in new\_drives:

log(f"USB drive connected: {new\_drive}", "CONNECT")

try:

total, free = shutil.disk\_usage(new\_drive)[0::2]

log(f"Drive {new\_drive} - Size: {total//(1024\*3)}GB, Free: {free//(1024\*3)}GB", "INFO")

except Exception:

log(f"Could not get disk information for {new\_drive}", "WARNING")

threading.Thread(target=show\_files\_gui, args=(new\_drive,)).start()

if connected\_drives - current:

removed\_drives = connected\_drives - current

for removed\_drive in removed\_drives:

log(f"USB drive removed: {removed\_drive}", "DISCONNECT")

messagebox.showinfo("USB Removed", f"Drive {removed\_drive} disconnected.")

connected\_drives = current

if \_name\_ == "\_main\_":

try:

log("USB Detector v1.0 starting...", "INFO")

monitor\_usb()

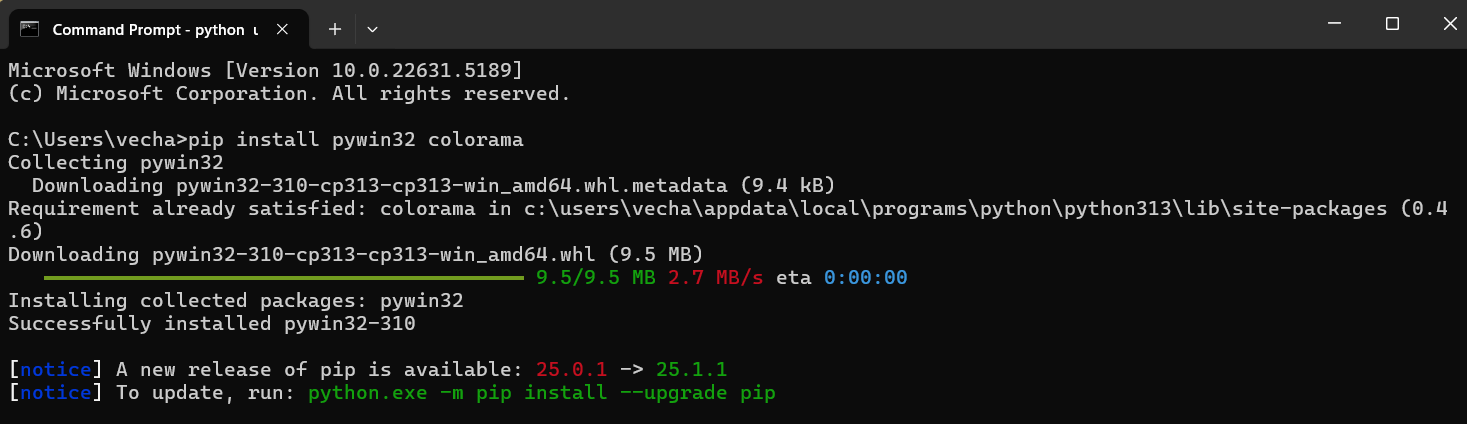
except KeyboardInterrupt:

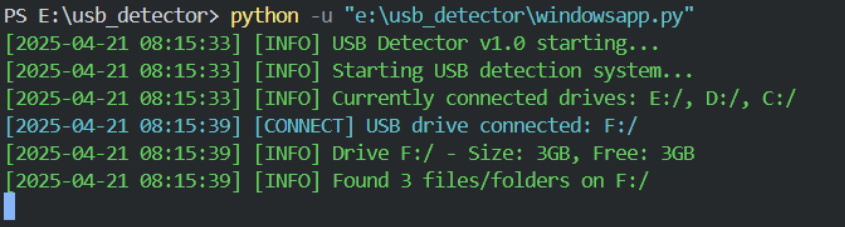
log("Program terminated by user.", "INFO")

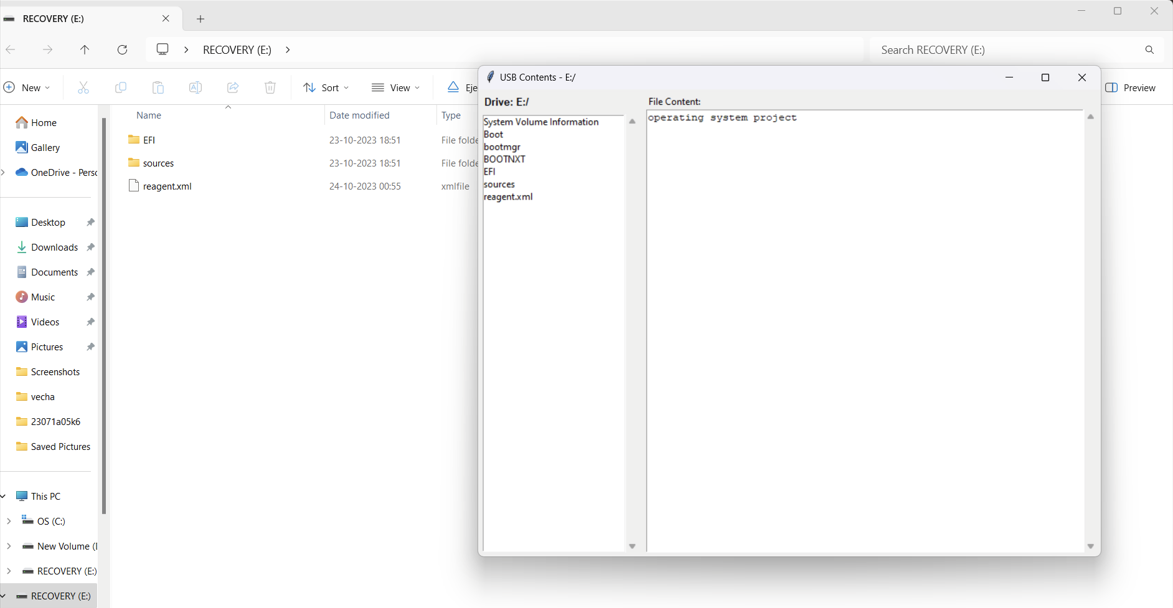
except Exception as e:

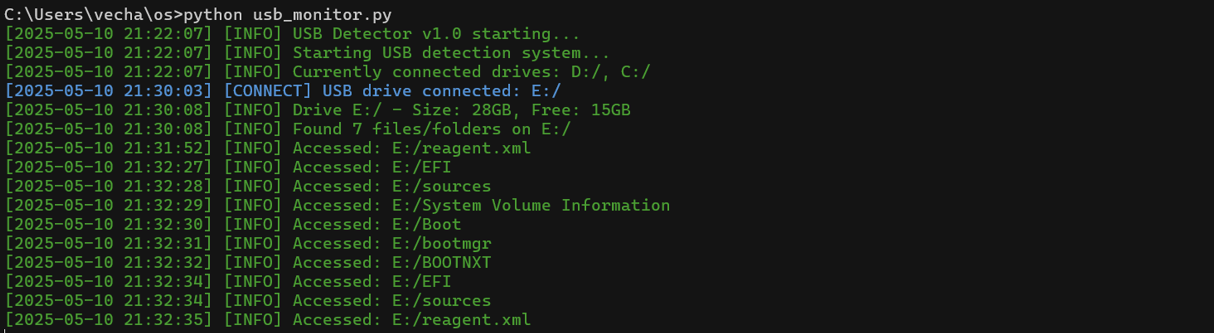
log(f"Unexpected error: {str(e)}", "ERROR")

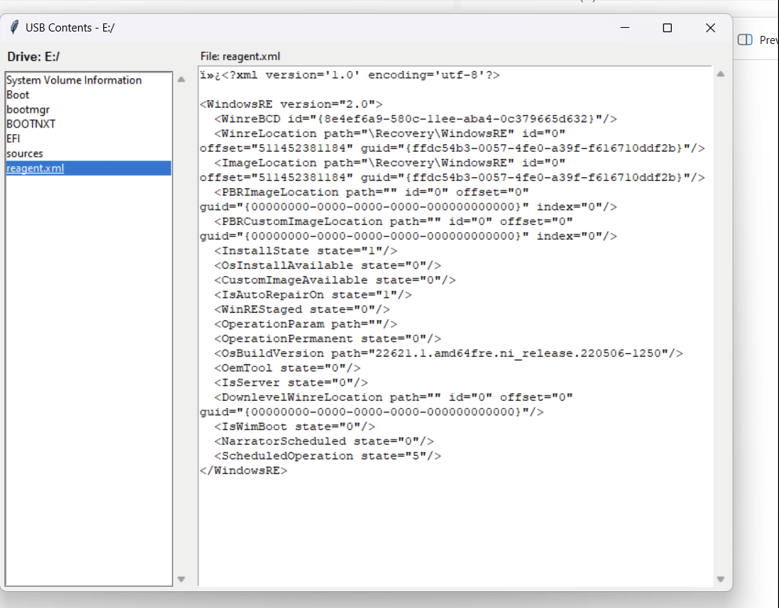
# OUTPUT:

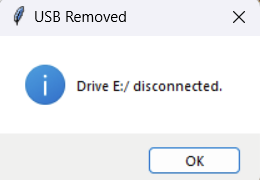












# REFERENCES

**References**

1. **Silberschatz, A., Galvin, P. B., & Gagne, G. (2018).** *Operating System Concepts* (10th Edition). Wiley.  
   – A comprehensive guide on operating system fundamentals, including file systems and device management.
2. **Stallings, W. (2017).** *Operating Systems: Internals and Design Principles* (8th Edition). Pearson.  
   – Offers detailed insights into OS architecture and device handling mechanisms.
3. **Microsoft Developer Network(MSDN).** <https://learn.microsoft.com>  
   – Documentation and API references for handling USB devices and file system events on Windows.
4. **Linux Kernel Documentation.** <https://www.kernel.org/doc/>  
   – Technical resources on USB subsystem and udev device management in Linux.
5. **PyUSB Documentation.** <https://github.com/pyusb/pyusb>  
   – A Python library for USB access, useful for detecting and interacting with USB devices.
6. **Tkinter GUI Library – Python Docs.** <https://docs.python.org/3/library/tkinter.html>  
   – Official documentation for creating graphical user interfaces in Python.
7. **Real-Time USB Monitoring Using Python.** GeeksforGeeks.  
   <https://www.geeksforgeeks.org/python-monitor-usb-events/>  
   – Article explaining how to monitor USB connections using Python libraries.
8. **Security Risks of USB Devices:** Kaspersky Lab Reports.  
   <https://www.kaspersky.com/blog/usb-device-risks/>  
   – Insight into the security challenges posed by external USB devices.
9. **Udev – Device Manager for Linux.** https://wiki.archlinux.org/title/Udev  
   – Documentation on using udev rules for detecting and managing USB devices in Linux-based systems.
10. **USB Device Security Best Practices.** SANS Institute Whitepapers.  
    https://www.sans.org/white-papers/usb-device-security/  
    – A detailed whitepaper discussing USB vulnerabilities and preventive measures in secure environments.

# CONCLUSION

In conclusion, this project addresses a critical gap in how modern operating systems handle external USB storage devices by introducing a specialized utility that offers real-time detection, monitoring, and management of USB drives. Traditional OS file explorers often lack immediate feedback, advanced security features, or user-friendly interfaces for managing USB connections. This project bridges that gap by simulating the behavior of an intelligent OS utility focused on improving user experience, security, and system interaction. By implementing features such as instant USB connection/disconnection alerts, an intuitive graphical file browser, and proactive security notifications, the application empowers users to interact with external storage devices more efficiently and safely. It simplifies the process of data access and enhances system awareness, especially in environments where the timely handling of USB drives is crucial. Overall, the utility not only strengthens the functionality of traditional operating systems but also introduces modern solutions for better storage management. It lays a solid foundation for future enhancements like advanced scanning, backup integration, or cloud syncing, ensuring that the system remains adaptable and relevant in evolving computing environments.

Furthermore, the project demonstrates the practical application of operating system concepts such as device management, file system handling, and user interface design. By bridging theoretical knowledge with real-world utility, it enhances understanding of how background processes and hardware interactions can be presented in a user-centric way. This not only provides technical value but also promotes better digital hygiene among users by making them more aware of external device interactions and potential risks.

The modular and extensible nature of the application also opens the door for future scalability. Features such as file transfer logging, user authentication for drive access, and integration with antivirus software can be added to enhance its utility further. This makes the project a strong foundation for more complex systems, such as enterprise-level USB monitoring solutions or educational tools for teaching OS functionalities. It also aligns with the growing need for endpoint security in both personal and corporate computing environments.

In summary, the USB monitoring and file browsing utility delivers a comprehensive solution to an everyday computing challenge. It combines real-time responsiveness with a clean, user-friendly interface, while also addressing the security concerns posed by external drives. As portable storage devices continue to be an integral part of data sharing and mobility, this project ensures that users remain in control, informed, and protected whenever they connect a USB device to their sy

# FUTURESCOPE

The proposed USB monitoring and file browsing utility presents a strong foundation with significant potential for further development and enhancement. In the future, the system can be expanded to include **advanced security features** such as automatic scanning of USB contents using antivirus APIs, detection of hidden or encrypted files, and integration with system-level firewalls to block suspicious activity. These additions would make the utility more suitable for use in enterprise environments where data protection is critical.

Another promising direction is the **integration with cloud storage platforms**, enabling automatic backup or synchronization of files from connected USB drives to services like Google Drive, Dropbox, or OneDrive. This feature would allow users to seamlessly transfer important data to the cloud for safekeeping, reducing the risk of data loss from accidental removal or device corruption. Additionally, users could configure selective sync settings to upload only specific file types or folders.

From a usability perspective, the utility can be made **cross-platform and mobile-compatible**, allowing users to monitor USB device activity from their smartphones or tablets through a companion app. Real-time alerts and logs could be synced via Bluetooth, Wi-Fi, or cloud messaging. Also, the GUI can be enhanced to support drag-and-drop file transfers, dark mode themes, and accessibility features to cater to a wider range of users. These improvements would not only enhance functionality but also expand the reach and adoption of the application in both personal and professional computing environments.

In addition to security and usability enhancements, the utility can evolve to support **multi-device and network-wide USB monitoring**. In organizational settings like schools, offices, or laboratories, system administrators could monitor all USB activities across multiple machines from a central dashboard. This would include real-time connection logs, file transfer history, and device whitelisting or blacklisting features. Such centralized management would improve accountability and reduce the risks associated with unauthorized data transfers or malware injections.

Lastly, with the growing focus on automation and artificial intelligence, future versions of this utility can incorporate **AI-powered recommendations and automation scripts**. For example, based on user behavior, the system could suggest actions like automatically opening specific folders when a known device is connected or generating file access reports at scheduled intervals. Machine learning algorithms could also help detect anomalies in file structures or transfer patterns, enabling preemptive alerts for unusual activity. These intelligent features would make the tool not only reactive but also predictive, significantly enhancing its overall value and efficiency.