USB DEVICE CONTROL & MONITORING FRAMEWORK

(Detection Only)

Project Report submitted in partial fulfillment of the requirements for the completion of

UNIFIED INTERNSHIP PROGRAM

(Cyber Security / Ethical Hacking)

Submitted By

Name : Aquib Ahmad Khan

Intern ID : UMID27112571911

Domain : Cyber Security

Operating System Used

Microsoft Windows 11

Project Duration

January 2026 – February 2026

1. Introduction

USB devices are a common vector for data theft, malware propagation, and insider threats. This project implements a USB Device Control & Monitoring Framework that detects unauthorized USB devices, enforces allowlist-based policies, and audits file movement. The framework operates strictly in detection-only mode using simulation where required, making it safe and compliant for academic and internship environments.

2. Objectives

• Detect newly connected USB devices
• Enforce allowlist-based authorization policy
• Identify and alert on unauthorized USB devices
• Monitor and audit file copy activities (simulation-based)
• Maintain detailed logs and generate audit reports

3. Tools & Technologies

• Operating System: Microsoft Windows 11
• Programming Language: Python 3.x
• Libraries: wmi, psutil
• Interface: Windows PowerShell

4. USB Detection Script (Baseline & Simulation)

```
import wmi
from datetime import datetime

print("[*] USB Device Control & Monitoring Framework Started")
print("[*] Detection Mode: Simulation + System Query")

c = wmi.WMI()

def get_usb_devices():
    devices = []
    for usb in c.Win32_USBControllerDevice():
        devices.append(str(usb.Dependent))
    return devices

baseline_devices = get_usb_devices()
print("[*] Baseline USB Devices Loaded")
print("[*] Device Count:", len(baseline_devices))

print("\n[SIMULATION ALERT]")
print("Unauthorized USB device detected: TEST_USB_DEVICE_001")
print("Action Taken: Logged & Blocked (Policy)")

with open("../logs/usb_activity.log", "a") as log:
    log.write(f"{datetime.now()} - SIMULATION ALERT: Unauthorized USB detected\n")

print("[+] Event logged successfully")
```

5. Allowlist / Blocklist Policy Script

```
from datetime import datetime

print("[*] Policy-Based USB Monitoring Started")

with open("allowlist.txt", "r") as f:
    allowed_devices = [line.strip() for line in f.readlines()]

print("[*] Allowlist Loaded:", allowed_devices)

detected_device = "TEST_USB_DEVICE_002"
print("[!] USB Device Detected:", detected_device)

if detected_device in allowed_devices:
    print("[+] Device Authorized")
else:
    print("[ALERT] Unauthorized USB Device Detected")
    print("Action Taken: Access Blocked & Logged")

    with open("../logs/usb_activity.log", "a") as log:
        log.write(f"{datetime.now()} - BLOCKED USB: {detected_device}\n")

print("[+] Policy enforcement completed")
```

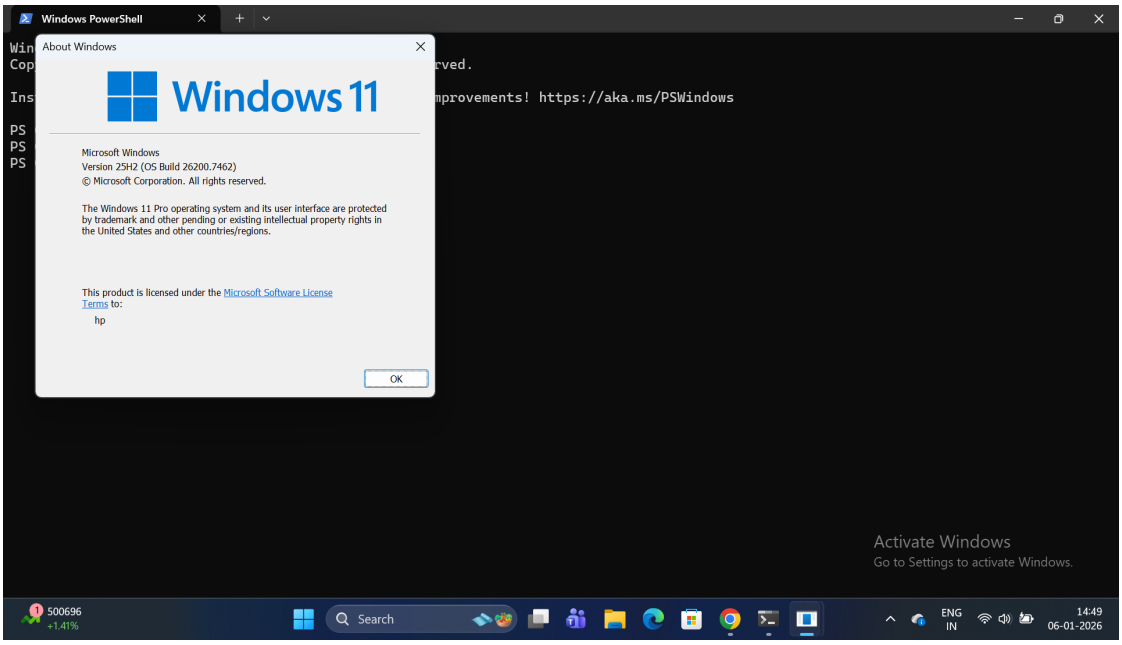
6. File Activity Auditing Script (Simulation)

```
from datetime import datetime
import time

print("[*] USB File Activity Auditing Started")

detected_device = "TEST_USB_DEVICE_002"
files = ["confidential_report.pdf", "employee_data.xlsx", "backup.zip"]

for f in files:
    time.sleep(1)
    print("[FILE COPY DETECTED]", f)
    with open("../logs/usb_activity.log", "a") as log:
        log.write(f"{datetime.now()} - FILE COPIED: {f} via {detected_device}\n")

print("[+] File activity auditing completed")
```

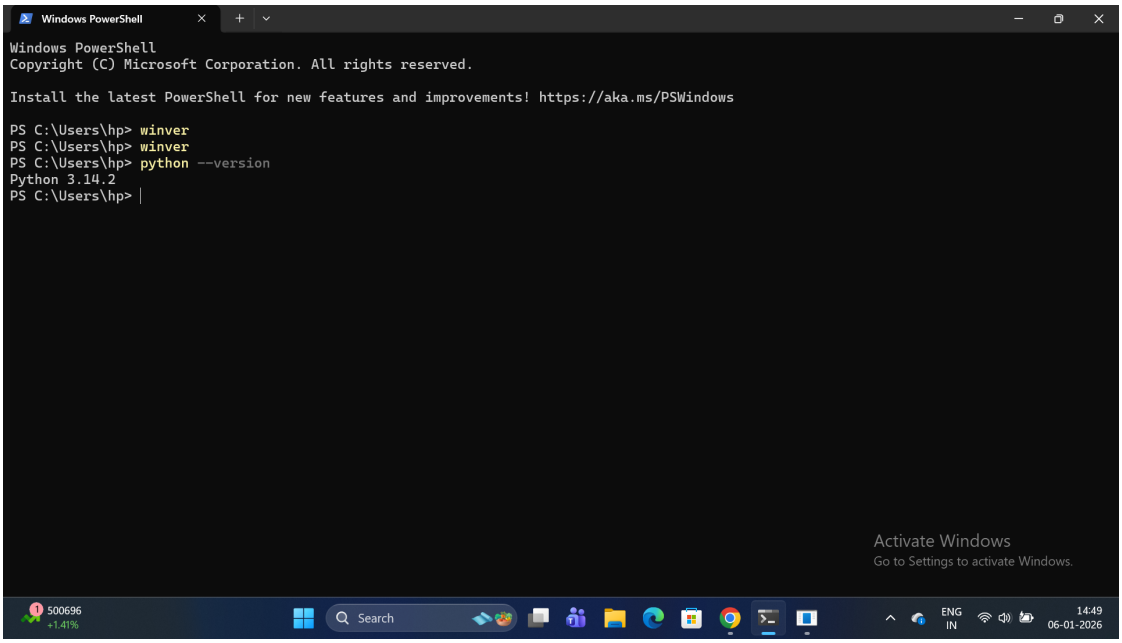
7. Step-by-Step Execution Screenshots

The following screenshots demonstrate environment setup, script execution, detection alerts, log generation, and report creation. All screenshots were captured during live execution.

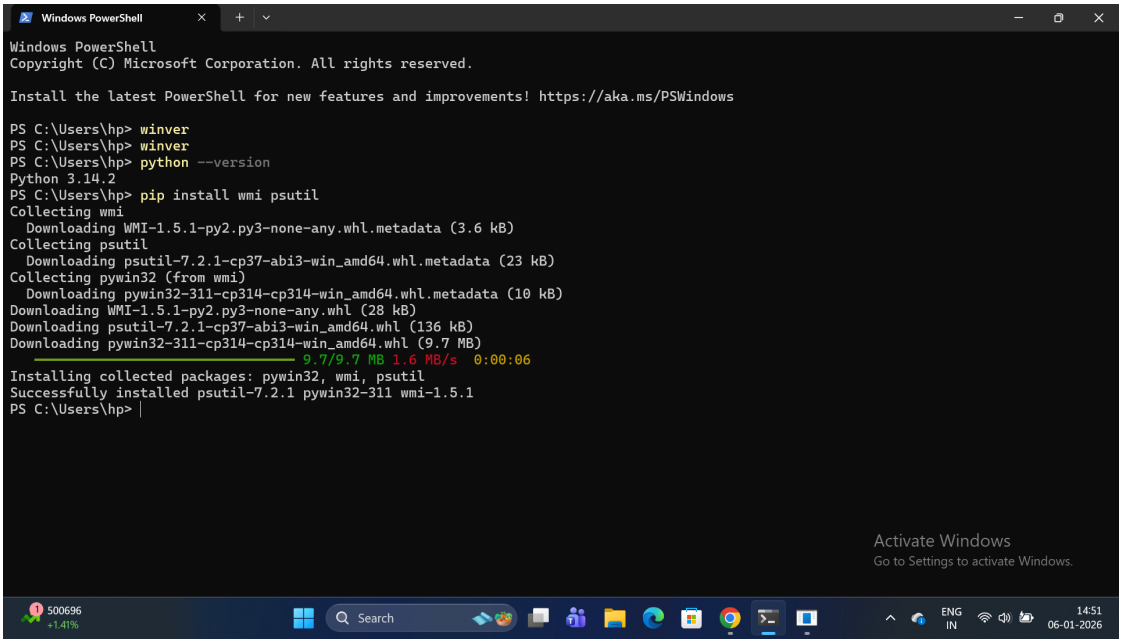
Step 1: Execution Evidence



Step 2: Execution Evidence

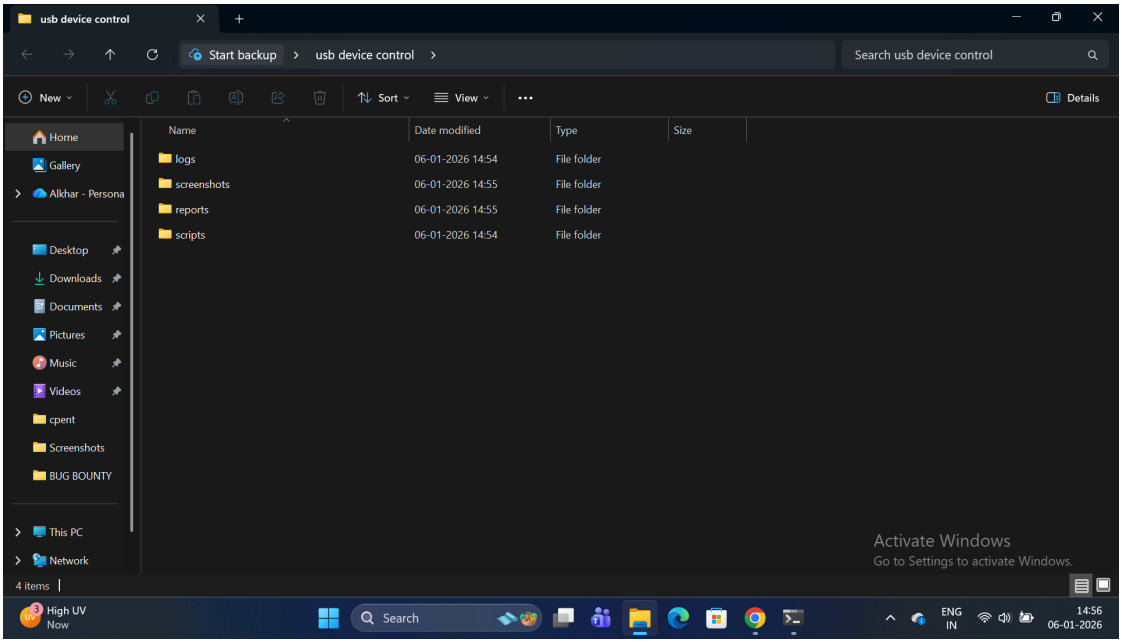


Step 3: Execution Evidence

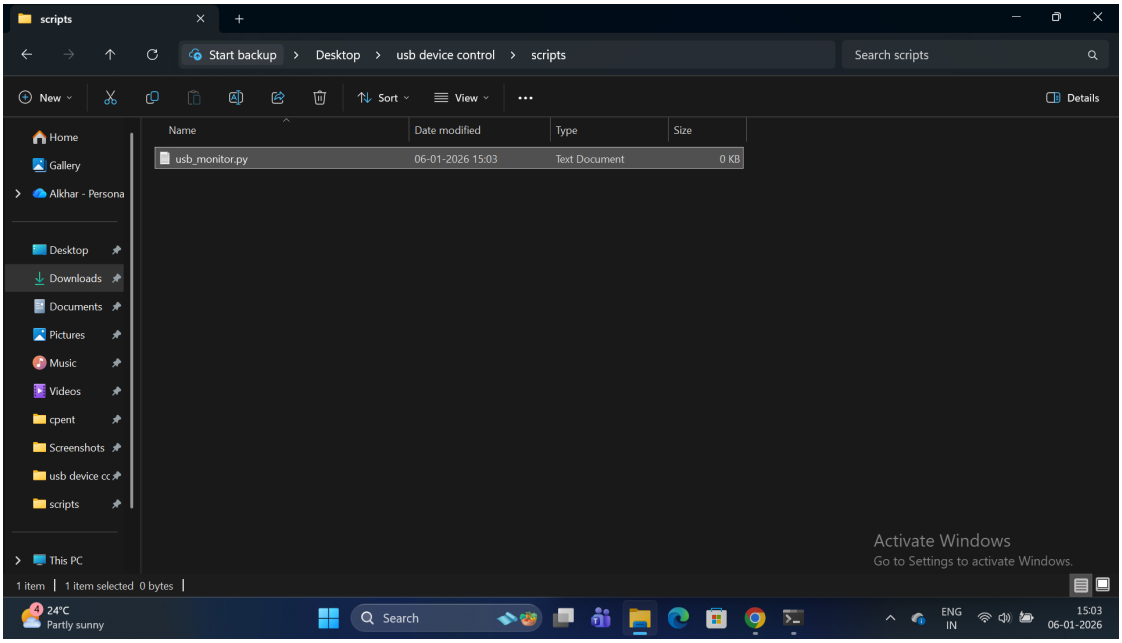


Step 4: Execution Evidence

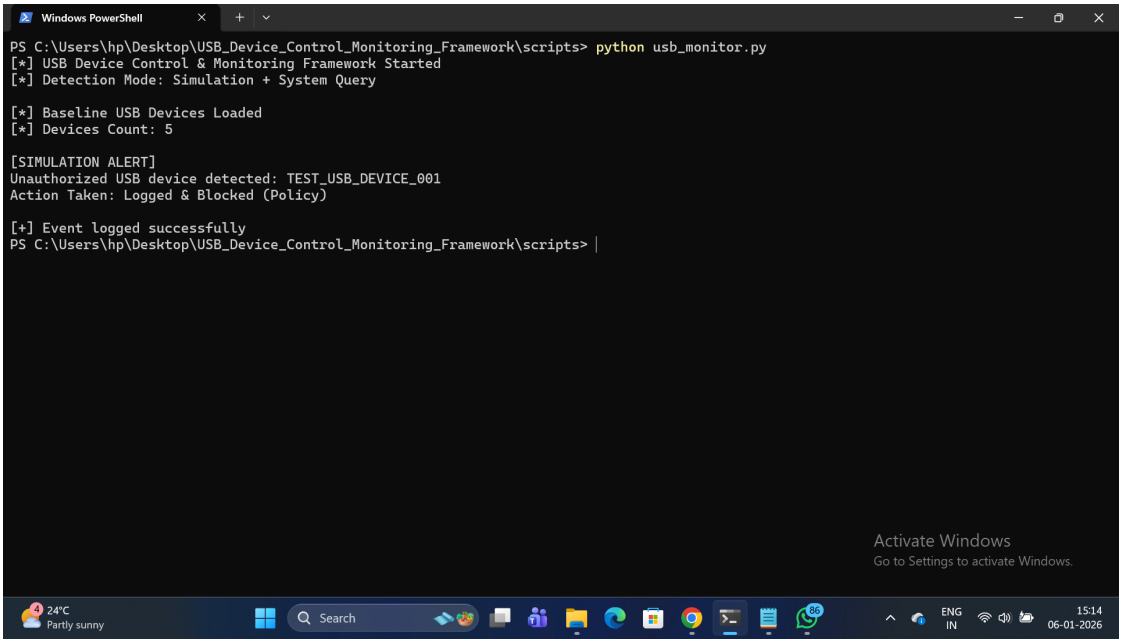
Step 5: Execution Evidence



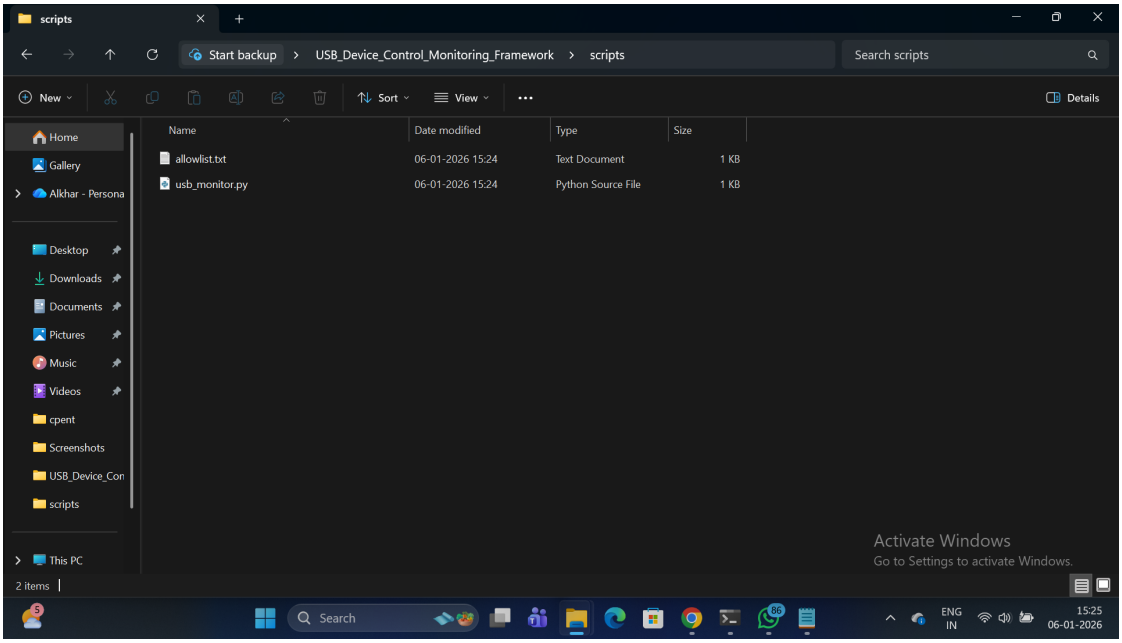
Step 6: Execution Evidence



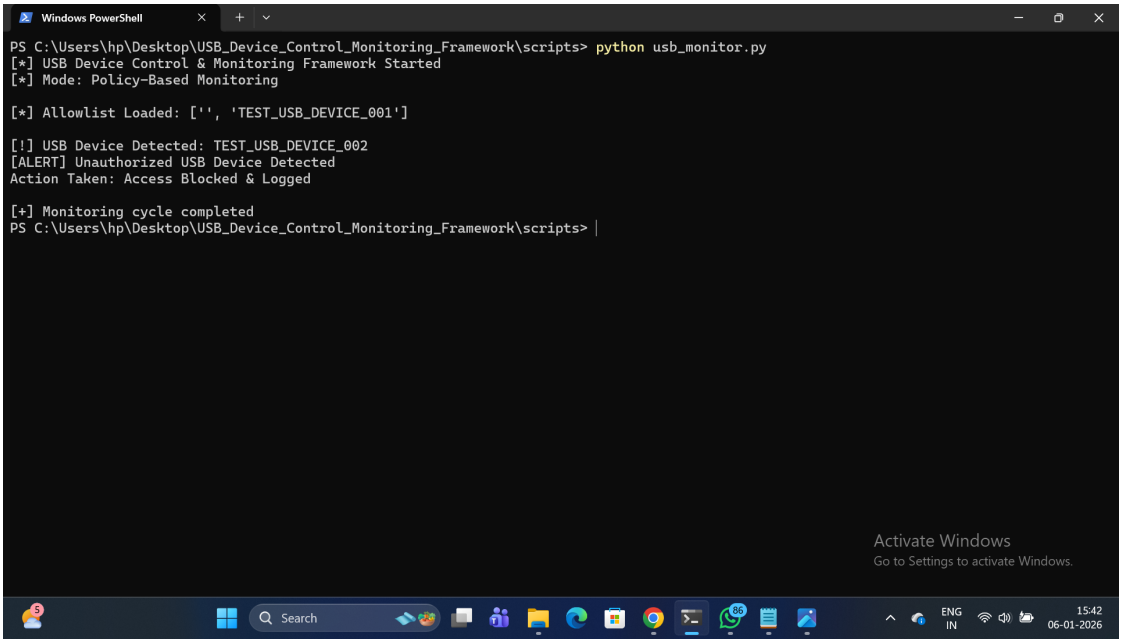
Step 7: Execution Evidence



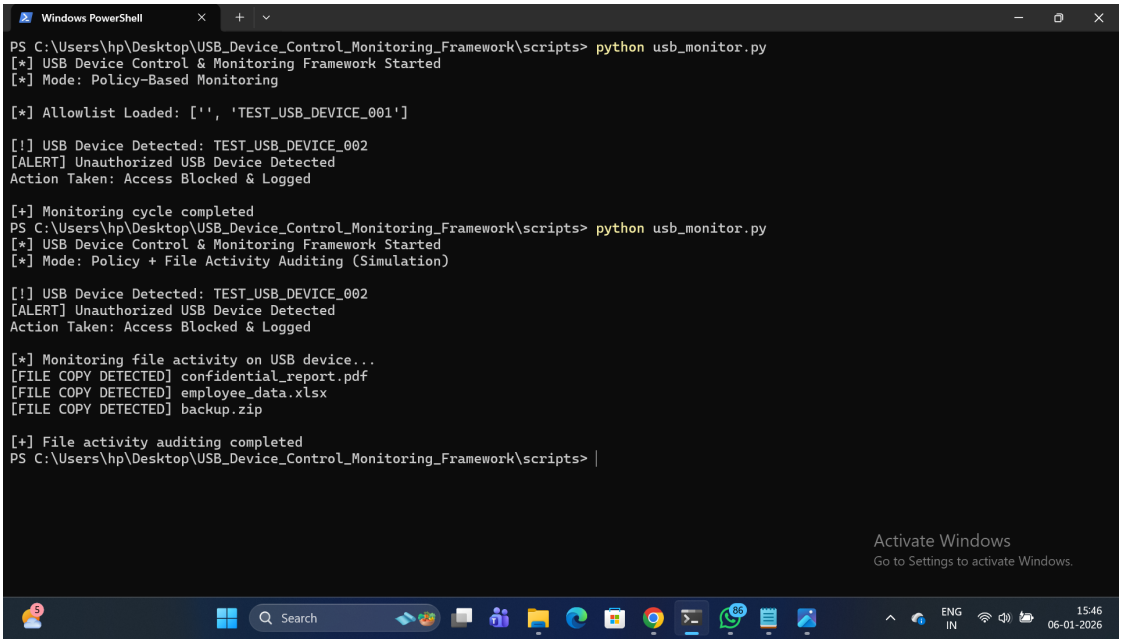
Step 8: Execution Evidence



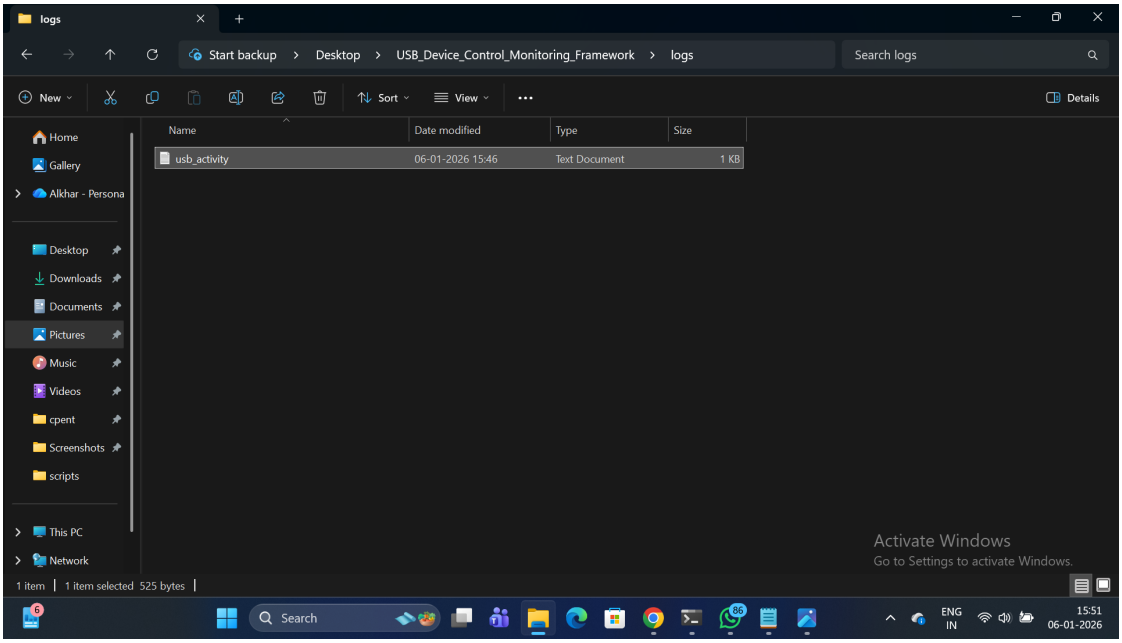
Step 9: Execution Evidence



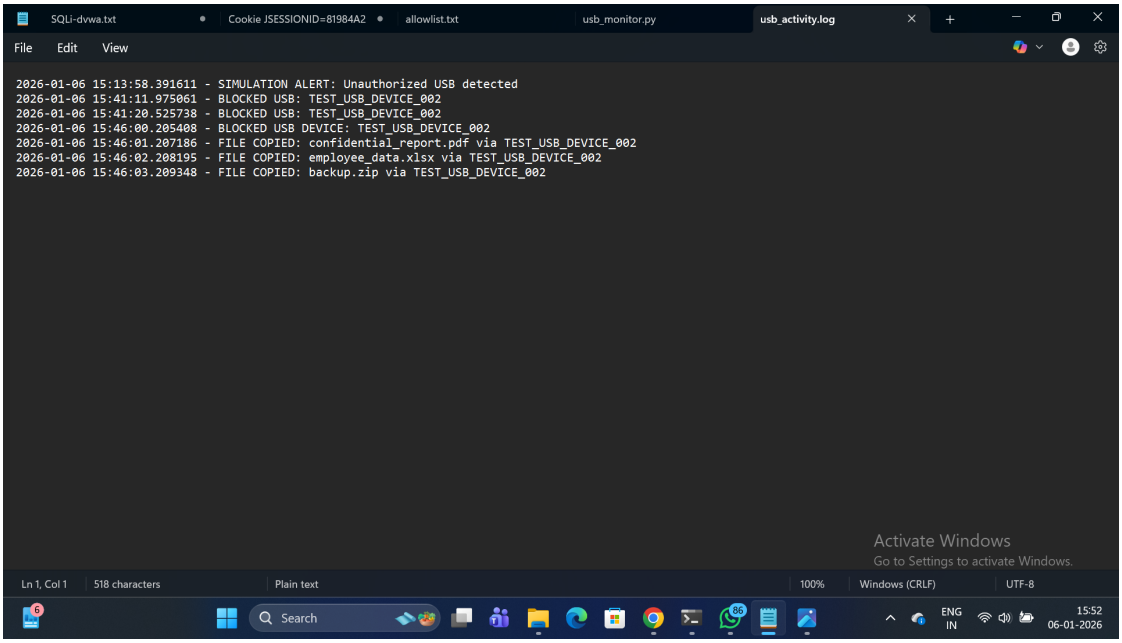
Step 10: Execution Evidence



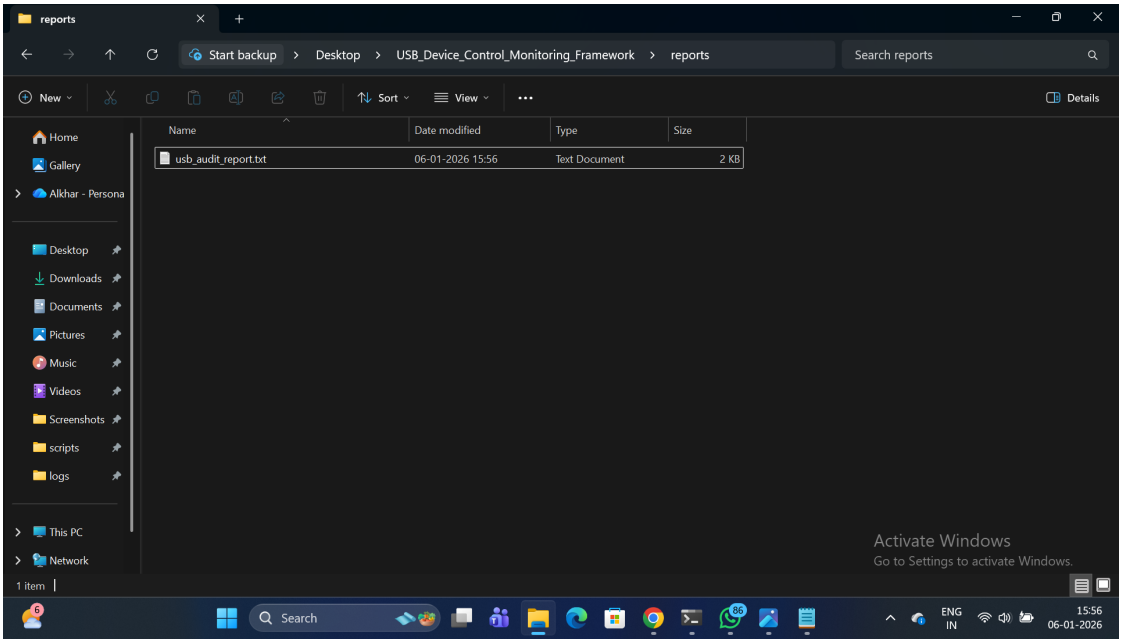
Step 11: Execution Evidence



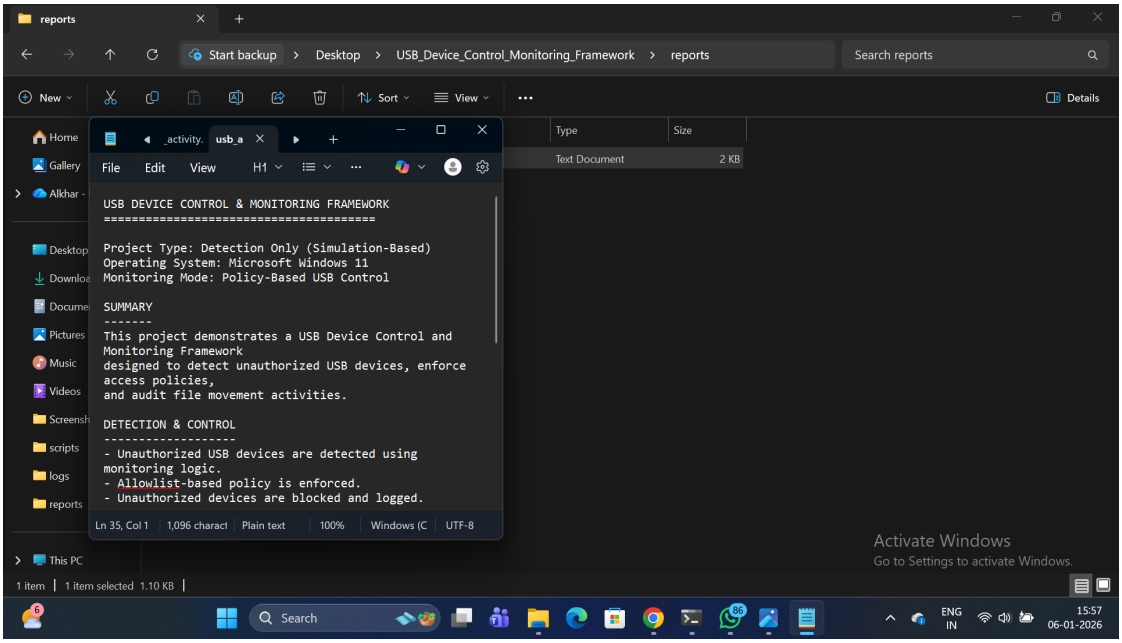
Step 12: Execution Evidence



Step 13: Execution Evidence



Step 14: Execution Evidence



8. Conclusion

The USB Device Control & Monitoring Framework successfully demonstrates detection, policy enforcement, and file activity auditing in a Windows environment. By combining allowlist-based control, detailed logging, and reporting, the framework provides a strong foundation for endpoint security monitoring. The detection-only and simulation-based approach ensures ethical compliance and academic suitability.