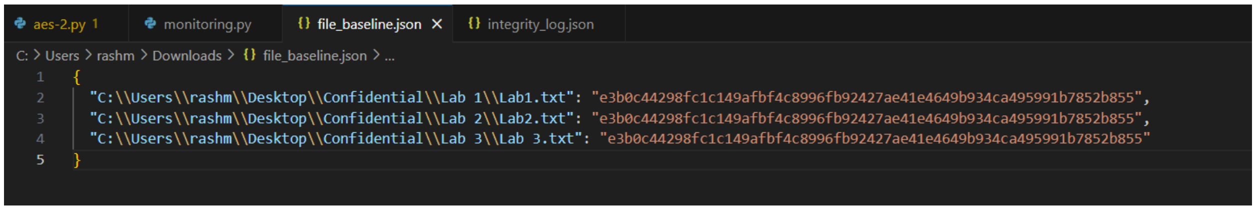
Group#8

- **Rashmika Adusumilli - 11770425**
- **SAI TEJA UPPU - 11691244**
- **Mounika Chavagani - 11737704**
- **Esther Eze - 11886695**

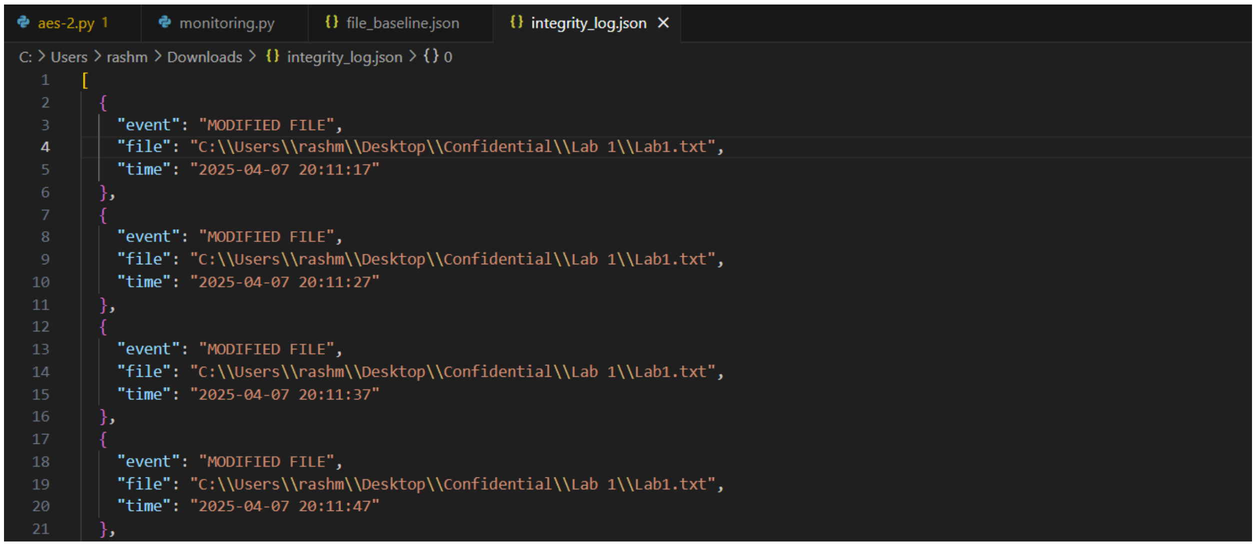
Monitoring:

As part of this step, in order to mimic other file logging tools, we have written a python code which can keep on checking the hash values of the files for every 10s and identify if they are changed or not. When the application started executing, it creates a file called file_baseline and stores the initial hash valves of all the existing files.



```json
{
  "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 1\\Lab1.txt": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 2\\Lab2.txt": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 3\\Lab 3.txt": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
}
```

For every 10s, the program will keep checking to ensure that hash files are changed or not. If any file is changed, then it is logged into a file called integrity_log.json and keep updating with the time when it identified the change. Here when the program is running, we have changed the text in Lab1.txt file. So it kept logging that there was a change.

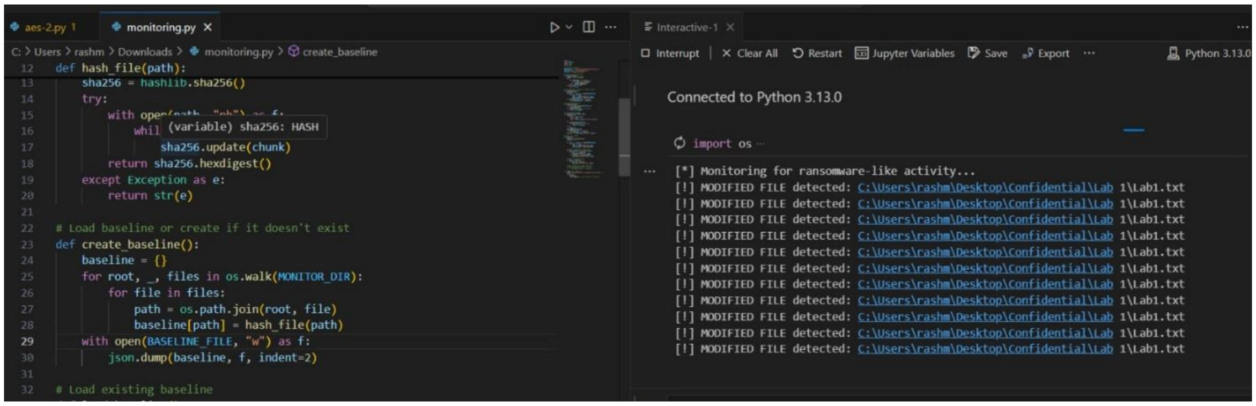


```json
[
  {
    "event": "MODIFIED FILE",
    "file": "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 1\\Lab1.txt",
    "time": "2025-04-07 20:11:17"
  },
  {
    "event": "MODIFIED FILE",
    "file": "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 1\\Lab1.txt",
    "time": "2025-04-07 20:11:27"
  },
  {
    "event": "MODIFIED FILE",
    "file": "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 1\\Lab1.txt",
    "time": "2025-04-07 20:11:37"
  },
  {
    "event": "MODIFIED FILE",
    "file": "C:\\Users\\rashm\\Desktop\\Confidential\\Lab 1\\Lab1.txt",
    "time": "2025-04-07 20:11:47"
  },
```

Here is the python code for the above:

```
import os
import hashlib
import json
import time

# Directory to monitor
MONITOR_DIR = r"C:\Users\rashm\Desktop\Confidential"
BASELINE_FILE = "file_baseline.json"
LOG_FILE = "integrity_log.json"

# Hashing function (SHA-256)
def hash_file(path):
    sha256 = hashlib.sha256()
    try:
        with open(path, "rb") as f:
            while chunk := f.read(8192):
                sha256.update(chunk)
        return sha256.hexdigest()
    except Exception as e:
        return str(e)

# Load baseline or create if it doesn't exist
def create_baseline():
    baseline = {}
    for root, _, files in os.walk(MONITOR_DIR):
        for file in files:
            path = os.path.join(root, file)
            baseline[path] = hash_file(path)
    with open(BASELINE_FILE, "w") as f:
        json.dump(baseline, f, indent=2)

# Load existing baseline
def load_baseline():
    if not os.path.exists(BASELINE_FILE):
        create_baseline()
    with open(BASELINE_FILE, "r") as f:
        return json.load(f)

# Log suspicious changes
def log_event(event_type, path):
    log_entry = {
        "event": event_type,
        "file": path,
        "time": time.strftime("%Y-%m-%d %H:%M:%S")
    }
    if os.path.exists(LOG_FILE):
        with open(LOG_FILE, "r") as f:
            logs = json.load(f)
    else:
        logs = []
    logs.append(log_entry)
    with open(LOG_FILE, "w") as f:
        json.dump(logs, f, indent=2)
    print(f"[!] {event_type} detected: {path}")

# Monitor for changes
def monitor():
    baseline = load_baseline()
    current = {}

    for root, _, files in os.walk(MONITOR_DIR):
        for file in files:
            path = os.path.join(root, file)
            current[path] = hash_file(path)

    # Check for modified or new files
    for path, hash_val in current.items():
        if path not in baseline:
            log_event("NEW FILE", path)
        elif hash_val != baseline[path]:
            log_event("MODIFIED FILE", path)

    # Check for deleted files
    for path in baseline:
        if path not in current:
            log_event("DELETED FILE", path)

    # Update baseline if needed (optional)
    # with open(BASELINE_FILE, "w") as f:
    #     json.dump(current, f, indent=2)

if __name__ == "__main__":
    print("[*] Monitoring for ransomware-like activity...")
    while True:
        monitor()
        time.sleep(10)  # Scan interval (in seconds)
```

Here is the screenshot of the program running and updating the file:



```python
def hash_file(path):
    sha256 = hashlib.sha256()
    try:
        with open(path, "rb") as f:
            whil (variable) sha256: HASH
                sha256.update(chunk)
        return sha256.hexdigest()
    except Exception as e:
        return str(e)

# Load baseline or create if it doesn't exist
def create_baseline():
    baseline = {}
    for root, _, files in os.walk(MONITOR_DIR):
        for file in files:
            path = os.path.join(root, file)
            baseline[path] = hash_file(path)
    with open(BASELINE_FILE, "w") as f:
        json.dump(baseline, f, indent=2)

# Load existing baseline
```

Connected to Python 3.13.0

```
import os ···
```

```
[*] Monitoring for ransomware-like activity...
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
[!] MODIFIED FILE detected: C:\Users\rashm\Desktop\Confidential\Lab 1\Lab1.txt
```