

## Appendix B: System Metrics and Process Monitoring

**Student Name:** Syed Farhan

**Student ID:** 253UA25557

**Course:** AAC6164 – Fundamental of Computer Science

**Project Title:** Linux-Based Monitoring, Analytics, and Reporting System Using Python

### 1. Introduction

This appendix documents the individual technical contribution of Student B, which focuses on system performance monitoring using Python and Linux commands. The objective of this module is to collect, analyse, and display real-time information about CPU usage and running processes.

### 2. Tools and Technologies Used

The following tools and technologies were used:

- Python 3
- Linux (Ubuntu)
- Libraries: subprocess, csv, datetime
- Commands: top, ps aux

### 3. System Design and Implementation

The system monitoring tool was developed using the Python subprocess module to execute Linux system commands. The output of each command is captured and processed to extract relevant performance metrics.

For CPU monitoring, the top -bn1 command is used to obtain CPU usage statistics, including user time, system time, and idle time.

For process monitoring, the ps aux command is used to list all active processes. The script sorts the processes based on CPU and memory usage and displays the top five processes for each category.

Both monitoring functions are combined into a single script named system\_monitor.py to improve usability and maintainability.

## 4. System Logging

A logging mechanism was implemented using CSV format. The `system_logger.py` script records system performance data including timestamp, CPU usage, memory usage, disk usage, uptime, and process count into `system_log.csv`. This allows historical data analysis and performance tracking.

## 5. Discussion of Results

When executed, the system monitoring script displays current CPU usage, total number of processes, and the top processes by CPU and memory consumption. The logging script successfully generates a CSV file containing system performance records.

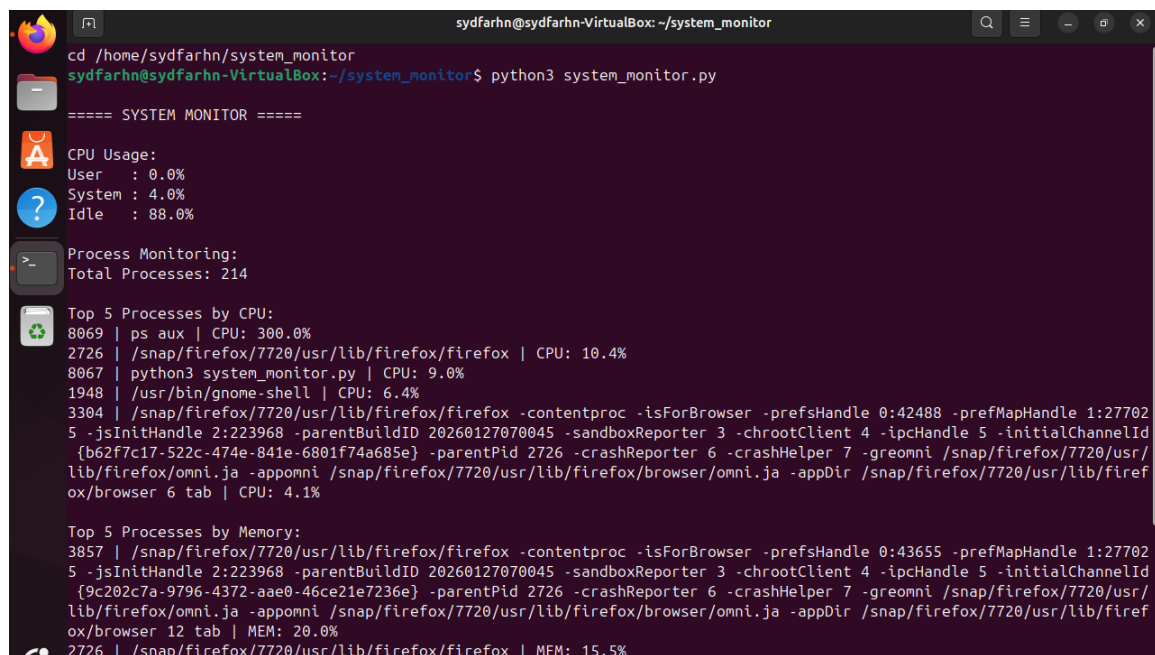
This confirms that the system functions as intended and meets the requirements of system metrics monitoring.

## 6. Conclusion

This appendix demonstrates the successful completion of Student B's responsibilities in system performance monitoring and logging. The implementation reflects practical skills in system administration, scripting, and data analysis.

## 7. Supporting Evidence

Figure B1: Execution of `system_monitor.py`



```
sydfarhn@sydfarhn-VirtualBox: ~/system_monitor
cd /home/sydfarhn/system_monitor
sydfarhn@sydfarhn-VirtualBox:~/system_monitor$ python3 system_monitor.py

===== SYSTEM MONITOR =====

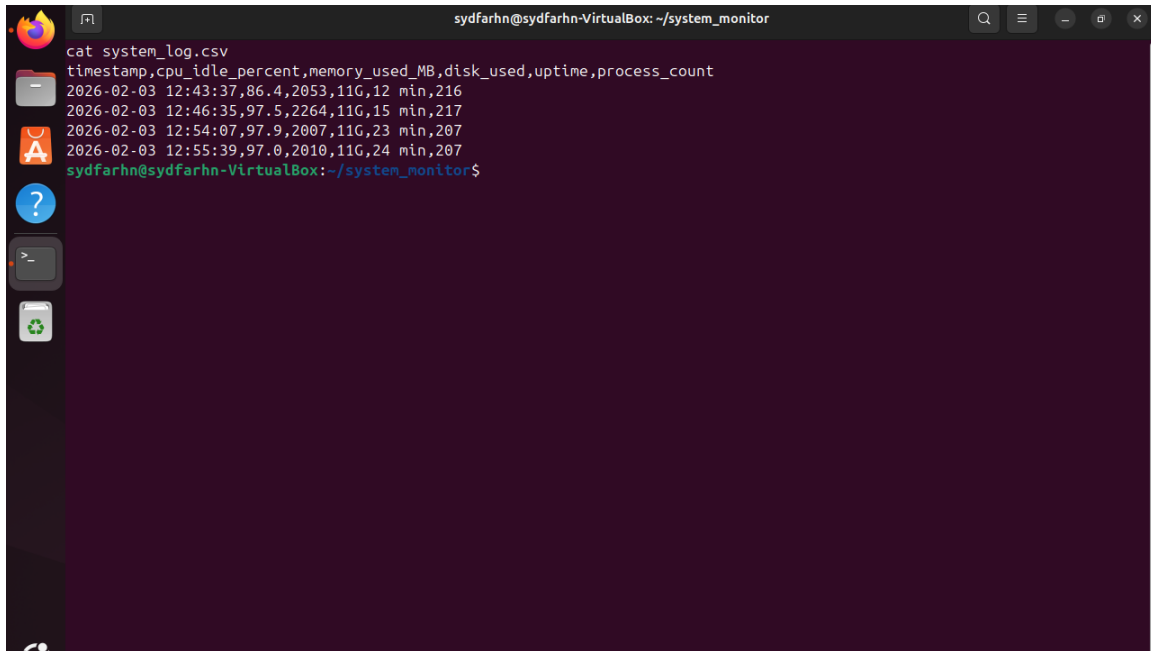
CPU Usage:
User   : 0.0%
System : 4.0%
Idle   : 88.0%

Process Monitoring:
Total Processes: 214

Top 5 Processes by CPU:
8069 | ps aux | CPU: 300.0%
2726 | /snap/firefox/7720/usr/lib/firefox/firefox | CPU: 10.4%
8067 | python3 system_monitor.py | CPU: 9.0%
1948 | /usr/bin/gnome-shell | CPU: 6.4%
3304 | /snap/firefox/7720/usr/lib/firefox/firefox -contentproc -isForBrowser -prefsHandle 0:42488 -prefMapHandle 1:27702
5 -jsInitHandle 2:223968 -parentBuildID 20260127070045 -sandboxReporter 3 -chrootClient 4 -ipcHandle 5 -initialChannelId {b62f7c17-522c-474e-841e-6801f74a685e} -parentPid 2726 -crashReporter 6 -crashHelper 7 -greomni /snap/firefox/7720/usr/lib/firefox/omni.ja -appomni /snap/firefox/7720/usr/lib/firefox/browser/omni.ja -appDir /snap/firefox/7720/usr/lib/firefox/browser 6 tab | CPU: 4.1%

Top 5 Processes by Memory:
3857 | /snap/firefox/7720/usr/lib/firefox/firefox -contentproc -isForBrowser -prefsHandle 0:43655 -prefMapHandle 1:27702
5 -jsInitHandle 2:223968 -parentBuildID 20260127070045 -sandboxReporter 3 -chrootClient 4 -ipcHandle 5 -initialChannelId {9c202c7a-9796-4372-aae0-46ce21e7236e} -parentPid 2726 -crashReporter 6 -crashHelper 7 -greomni /snap/firefox/7720/usr/lib/firefox/omni.ja -appomni /snap/firefox/7720/usr/lib/firefox/browser/omni.ja -appDir /snap/firefox/7720/usr/lib/firefox/browser 12 tab | MEM: 20.0%
2726 | /snap/firefox/7720/usr/lib/firefox/firefox | MEM: 15.5%
```

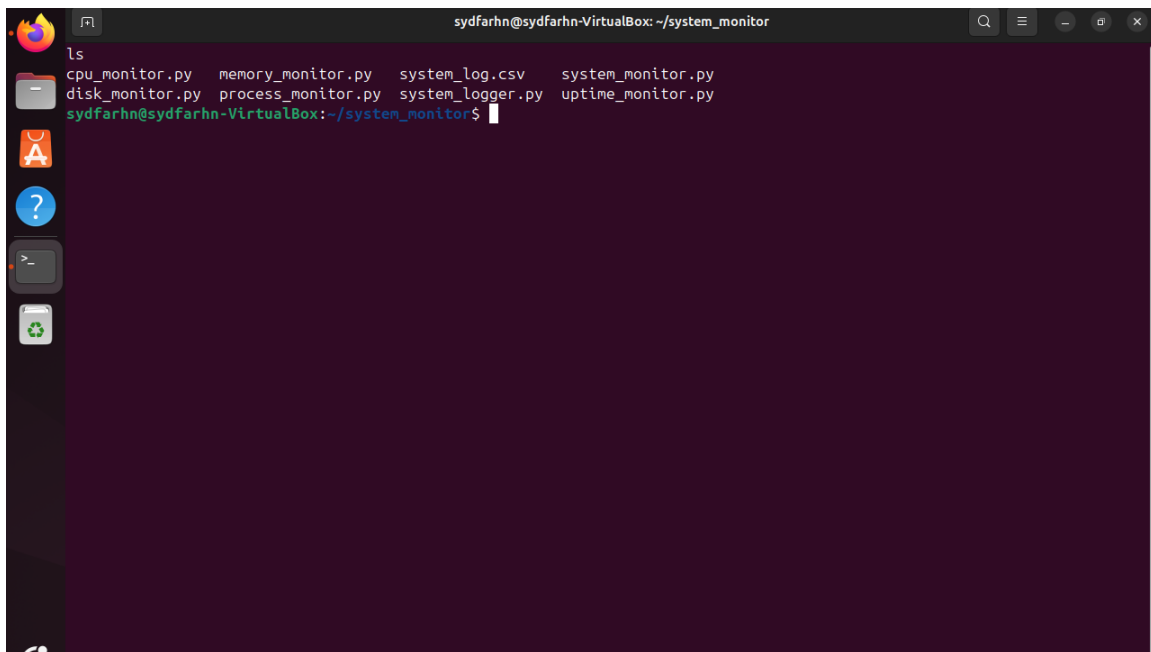
Figure B2: Generated CSV Log File



A terminal window titled 'sydfarhn@sydfarhn-VirtualBox: ~/system\_monitor'. The user has executed the command 'cat system\_log.csv'. The output displays the header of the CSV file and four rows of system monitoring data. The data includes timestamps, CPU idle percentage, memory used in MB, disk used, uptime, and process count.

```
cat system_log.csv
timestamp,cpu_idle_percent,memory_used_MB,disk_used,uptime,process_count
2026-02-03 12:43:37,86.4,2053,11G,12 min,216
2026-02-03 12:46:35,97.5,2264,11G,15 min,217
2026-02-03 12:54:07,97.9,2007,11G,23 min,207
2026-02-03 12:55:39,97.0,2010,11G,24 min,207
sydfarhn@sydfarhn-VirtualBox:~/system_monitor$
```

Figure B3: Project Directory Structure



A terminal window titled 'sydfarhn@sydfarhn-VirtualBox: ~/system\_monitor'. The user has executed the command 'ls'. The output lists the files in the directory: 'cpu\_monitor.py', 'memory\_monitor.py', 'system\_log.csv', 'system\_monitor.py', 'disk\_monitor.py', 'process\_monitor.py', 'system\_logger.py', and 'uptime\_monitor.py'.

```
ls
cpu_monitor.py  memory_monitor.py  system_log.csv  system_monitor.py
disk_monitor.py process_monitor.py system_logger.py uptime_monitor.py
sydfarhn@sydfarhn-VirtualBox:~/system_monitor$
```