Team:

Mohamed Tamer 221007879

Youssef Khaled 221027758

Youssef Tarek 221027715

Submitted to:

Eng: Ahmed Gamal

Task Manager Documentation

Features

1. Main Menu with Options

- Uses zenity to display a graphical menu for user selection.

- Options include:

-Graphs: Display real-time system monitoring graphs.

- CPU: Show CPU information and usage stats.

- GPU: Display GPU stats such as utilization, memory usage, and temperature.

- Disk: Show disk usage and space details.

- Memory: Display memory stats such as total, used, and available memory.

- Network: Show network statistics like packets sent/received.

- Exit: Exit the application.

2. System Resource Monitoring

- *CPU*: Displays CPU utilization, number of processes/threads, speed, and temperature.

- *GPU*: Shows GPU utilization, temperature, and memory details (requires nvidia-smi).

- *Disk*: Provides disk space information (used, available, and total).

- *Memory*: Displays memory stats such as total, used, free, cached, and available memory.

- *Network*: Monitors network stats like packets transmitted/received and dropped packets.

3. *Real-Time Graphs*

- Displays real-time graphs for:

- *CPU usage* (percentage)

- *Memory usage* (percentage)

- *Disk usage* (percentage)

- *Network usage* (data transfer in MB/s)

- Uses *Matplotlib* for graph rendering and *Tkinter* for GUI.

4. Script-Based Implementation

- Shell scripts (.sh files) handle backend data collection for each resource.

- Python scripts (graphs.py) handle graph generation and real-time visualization.

Python Libraries:

- psutil
- matplotlib.pyplot
- matplotlib.animation
- collections.deque

Requirements

- *Software Dependencies*
- 1. *System Tools*:
- - zenity (for GUI dialogs)
- - bash (for script execution)
- - psutil (for Python-based system monitoring)
- - nvidia-smi (for GPU monitoring, requires NVIDIA drivers)
-
- 2. *Python Libraries* (specified in requirements.txt):
- - matplotlib
- - tkinter
- - psutil
-
- 3. *Linux Environment*:
- - Assumes Linux-based tools (e.g., top, df, free, sensors).
-
- #### *Docker Requirements*
- - Docker installed on the host system.

- Docker image for the application with all dependencies pre-in

File Structure

```
task_manager/
├─── CPU/
│   └─── cpu_info.sh        # Script for CPU information
├─── Disk/
│   └─── disk_info.sh       # Script for Disk information
├─── GPU/
│   └─── gpu_info.sh        # Script for GPU information
├─── Memory/
│   └─── memory_info.sh     # Script for Memory information
├─── Network/
│   └─── network_info.sh    # Script for Network information
├─── main.sh               # Main script orchestrating the application
├─── Dockerfile            # Docker configuration
├─── graphs.py             # Python script for generating graphs
└─── README.md             # Project documentation Installation
```

Usage

For Monitoring System Resources

- Ideal for users who want a graphical and terminal-based way to monitor their system's CPU, GPU, memory, disk, and network usage.

For Troubleshooting

- Users can monitor and identify system bottlenecks like high CPU usage, memory leaks, or disk space issues.

For Learning and Demonstration

- Showcases integration of shell scripts, Python, and Docker for building a full-stack monitoring tool.

- Demonstrates the use of GUI-based tools (zenity, Tkinter) within a Dockerized application.

For Development/Customization

- Developers can extend the functionality by adding new scripts for additional metrics (e.g., power consumption, system logs).

- Easy to containerize and deploy for different environments.

References:

GitHub: **https://github.com/AAST-Projects/Task_Manager.git**

Docker Hub:

https://hub.docker.com/repository/docker/mohamedeid123/task_manager/tags