

# Subject: Open Source Technologies(OST)

|  |  |  |
| --- | --- | --- |
| Sr no | PRN | Name |
| 1 | 23070123006 | Abhishek Patawari |
| 2 | 23070123045 | Devkarthik Suresh |
| 3 | 23070123069 | Otniel Jhirad |
| 4 | 23070123049 | Deepti Emmi |
| 5 | 23070123014 | Akshit Mathur |

# System Health Monitor



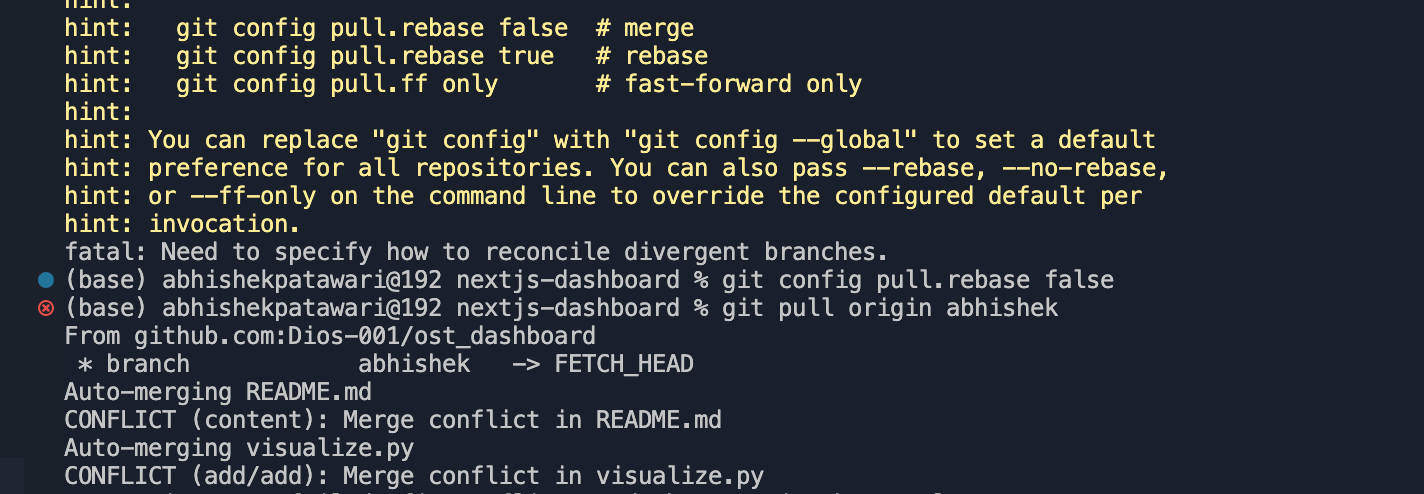
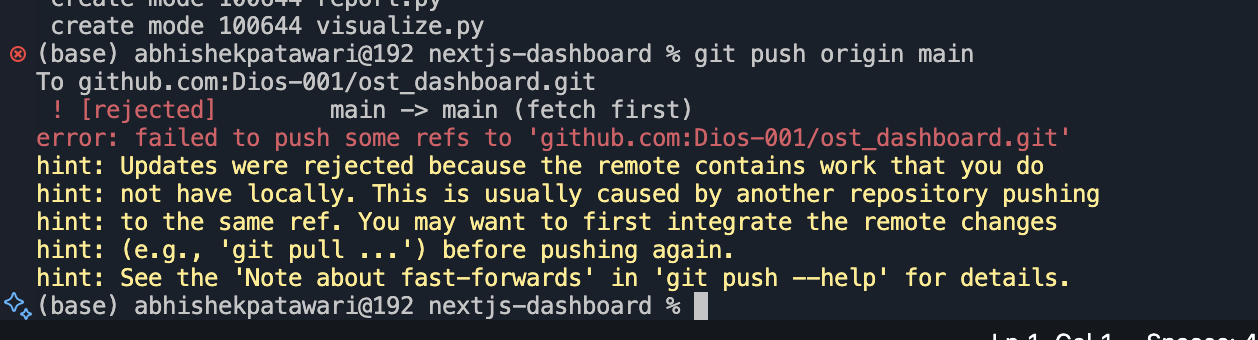
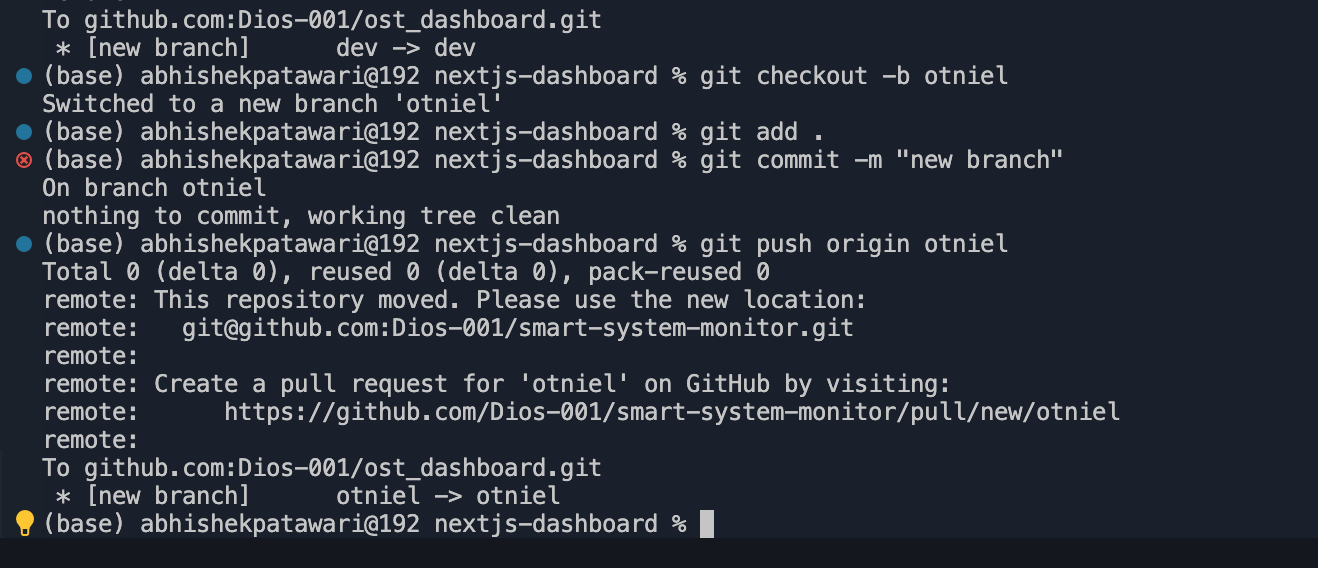
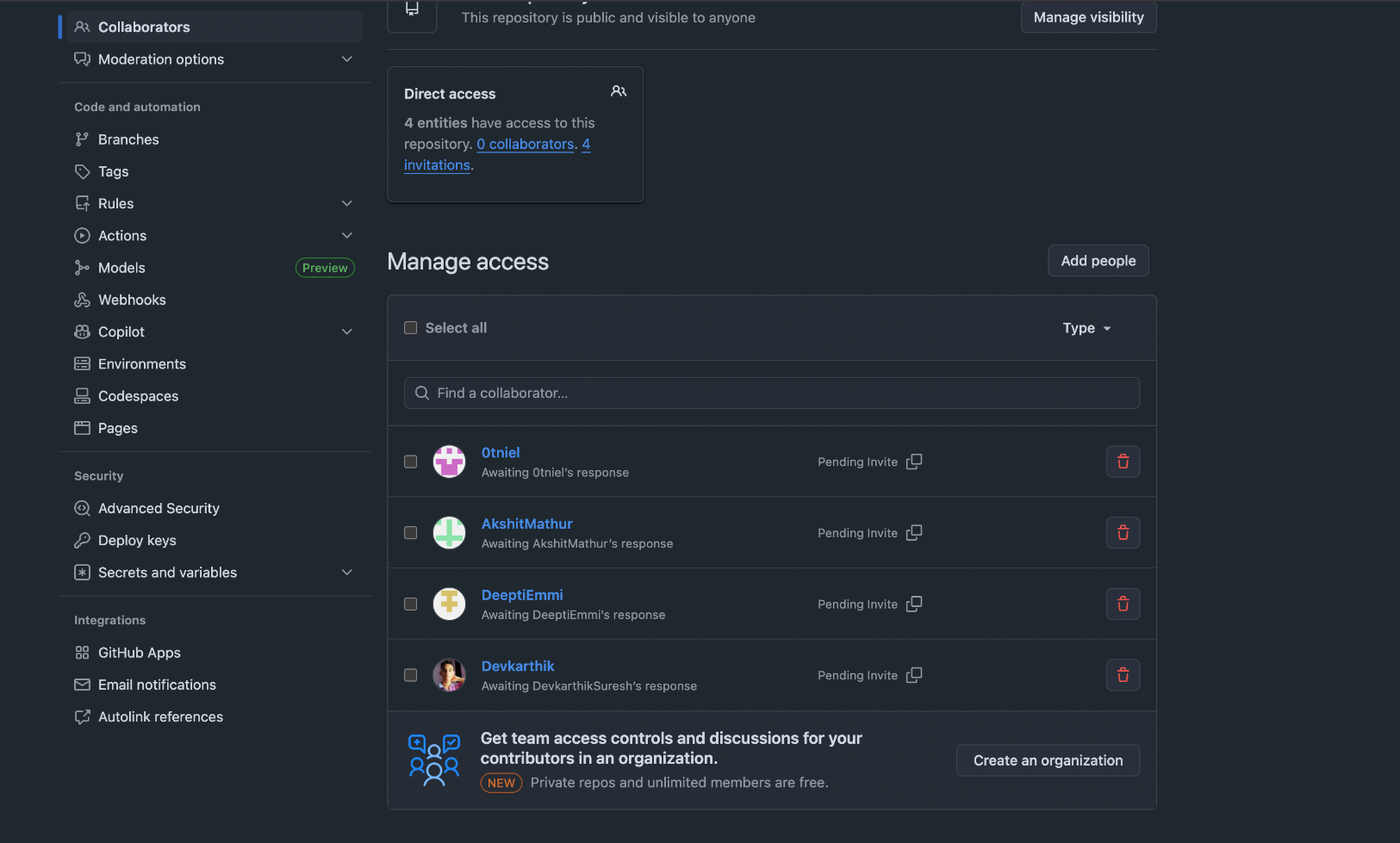
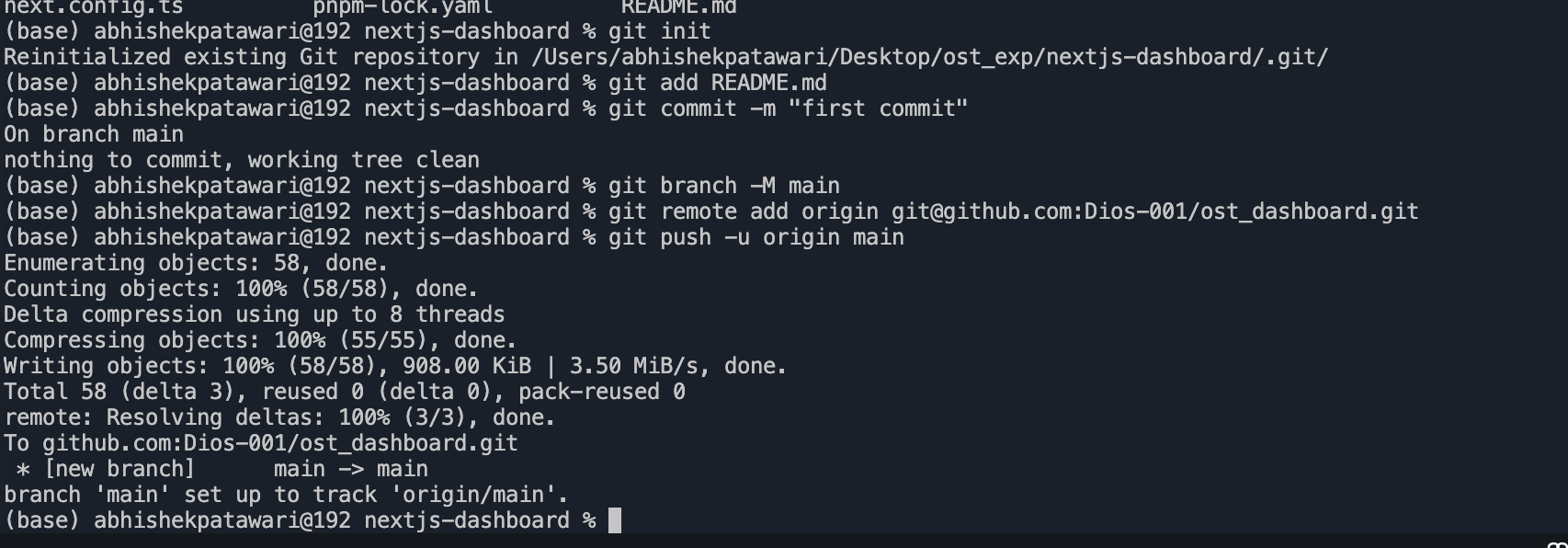
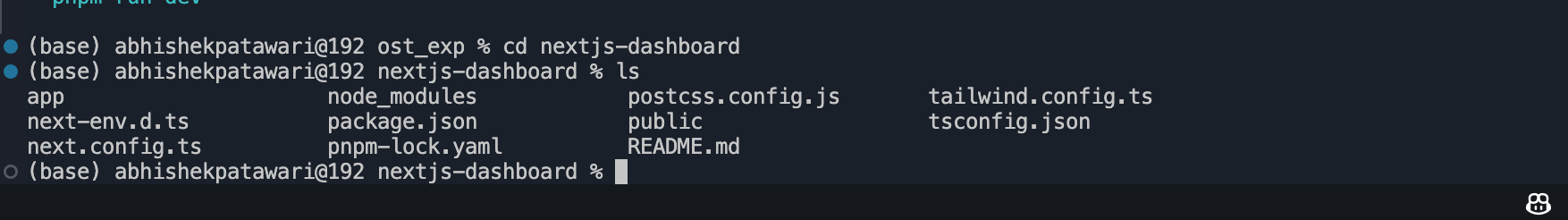
**Description:**  
The System Health Monitor is a lightweight automation project designed to track and display critical system performance metrics such as CPU usage, memory consumption, disk space, network statistics, and active processes.  
The project runs entirely using **Shell scripting** on **Ubuntu/Linux**, and provides both **real-time monitoring** and **log file generation** for further analysis.

It helps users and administrators quickly check system health, automate periodic logging using cron jobs, and generate summarized reports.  
This project also demonstrates **collaborative software development using GitHub**, where contributors worked on separate modules (CPU, memory, network, and automation) and raised GitHub issues for each feature before merging their changes.

### **Tools/Technologies Used:**

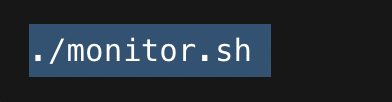
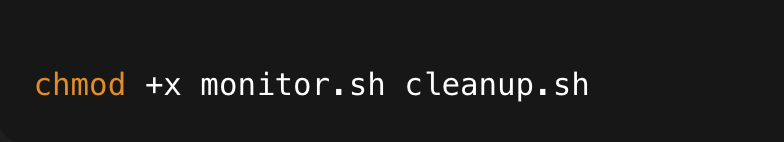
|  |  |
| --- | --- |
| **Category** | Tools/Technologies |
| **Operating System** | Ubuntu 22.04 LTS |
| **Programming Language** | Bash (Shell Scripting) |
| **Version Control** | Git & GitHub |
| **Automation** | Cron Jobs |
| **Editor / Terminal** | VS Code, GNOME Terminal |
| **Additional Utilities** | |  |  | | --- | --- | |  | top, free, df, uptime, ps, awk, grep | |

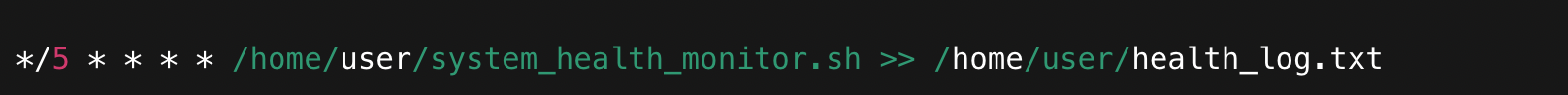
**Screenshots of Execution:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA black background with green and pink text

Description automatically generated**

**How to Run the Project:**

#### **Step 1: Clone the Repository**

git clone https://github.com/<your-org>/system-health-monitor.git

cd system-health-monitor

#### **Step 2: Give Execute Permissions**

chmod +x system\_health\_monitor.sh

#### **Step 3: Run the Script**

./system\_health\_monitor.sh

#### **Step 4: Optional – Automate Logging Every 5 Minutes**

crontab -e

\*/5 \* \* \* \* /home/user/system-health-monitor/system\_health\_monitor.sh >> /home/user/health\_l

**Actual Output v/s Expected Output:**

|  |  |  |
| --- | --- | --- |
| Parameter | Expected Output | Actual Output |
| CPU Usage Display | Shows % CPU Usage | Accurate |
| Memory Usage Display | Shows % Memory Usage | Matches “free” output |
| Disk Usage Display | Shows % used | Matches “df -h” output |
| Network Status | Shows “UP/DOWN” | Correct Detection |
| Log File | Appends Entries | Working as intended |

Note: Installation of necessary python libraries are necessary

**Unique Feature or Enhancement**

This project goes beyond standard system monitoring by combining **automation, analytics, and collaborative scripting** into one integrated tool.

* Unlike basic top or htop utilities, this suite **automatically logs** system metrics, **sends email alerts**, and **visualizes performance trends** without user intervention.
* The added **System Health Analyzer** gives a quick performance summary—calculating average and peak CPU, memory, and disk usage, transforming raw logs into meaningful insights.
* Each feature (logging, alerting, visualization, summary) runs through a **cron-based automation pipeline**, making the solution fully autonomous.
* Team members implemented modular scripts so that the system can be expanded easily for other metrics like temperature or network usage.

| **Student Name** | **Role / Contribution** | **GitHub Profile** | **Key Commits / Pull Requests** |
| --- | --- | --- | --- |
| **Abhishek Patawari** | Designed and implemented monitor.sh automation script, configured cron jobs, handled file permissions, and integrated all modules. | [github.com/abhishekpatawari](https://github.com/abhishekpatawari) | PR #11, #12 |
| **Deepti Emmi** | Developed logger.py for system data logging, managed file handling using config.json, optimized I/O, and ensured stable logging performance. | [github.com/DeeptiEmmi](https://github.com/DeeptiEmmi) | PR #9, #16,#19 |
| **Akshit Mathur** | Built alert.py with configurable email notifications using SMTP; set up CPU threshold alerting logic via JSON parameters. | [github.com/AkshitMathur](https://github.com/AkshitMathur) | PR #10 |
| **Otniel Jhirad** | Created visualize.py to plot CPU, memory, and disk usage trends using matplotlib; implemented labeling and export of plots. | [github.com/0tniel](https://github.com/0tniel) | PR #8 |
| **Devkarthik Suresh** | Added analyzer.py to summarize system logs—computes average and peak usage values; generates concise performance reports and integrates analytics into the system. | [github.com/DevkarthikSuresh](https://github.com/DevkarthikSuresh) | PR #19, #20 |

**Submission Timeline and Project Tracking**

The project was executed and tracked systematically over a short duration using **GitHub issues, commits, and pull requests** to ensure timely progress and collaboration.

* Project planning, role allocation, and branch creation for each member.
* Core script development (logger.py, alert.py, monitor.sh) and initial testing.
* Implementation of visualization (visualize.py) and system analyzer module (analyzer.py).
* Integration, final testing, and documentation (README, report preparation).

Commit timestamps and GitHub milestones clearly reflect continuous contributions by all team members, demonstrating proper time management and coordinated teamwork leading up to the submission deadline.

**GitHub Repository Link:** [**https://github.com/Dios-001/smart-system-monitor**](https://github.com/Dios-001/smart-system-monitor)

**Commit History:**

**README File:**

The project **System Health Logger** is an streamlined automation tool which helps to monitor as well as track system performance parameters which include such as CPU, memory, disk, and network usage. Developed with Shell scripting and Python on Ubuntu/Linux, it offers real-time monitoring, creates logs, and generates automated reports through scheduled cron jobs.

**Features:**

* Real-time system monitoring
* Auto log generation and cleanup
* Visual reports and alerts
* Scheduled automation via cron
* Lightweight and modular design

**How To Run:**

**Bash:-**

*chmod +x monitor.sh cleanup.sh*

*./monitor.sh*

**Cron Setup:-**

*crontab -e*

*\*/10 \* \* \* \* /home/$USER/system-health-logger/monitor.sh*

**Project Files:**

|  |  |
| --- | --- |
| **File Name** | **Description** |
| .next/ | Build and runtime files |
| .DS\_Store | macOS system file |
| .env | Environment settings |
| README.md | Project Documentation |
| alert.py | Sends performance alerts |
| cleanup.sh | Removes old logs |
| config.json | Stores config data |
| logger.py | Handles system logs |
| monitor.sh | Main monitoring script |
| report.py | Generates reports |
| Systemanalyzer.py | Analyses performance |
| Visualize.py | Creates visual graphs |

**Git Commands**

git init

git add .

git commit -m "initial commit"

git push origin main

**Contributors**

1. Abhishek Patawari (23070123006)
2. Otniel Jhirad (23070123069)
3. Devkartik Suresh (23070123045)
4. Deepti Emmi (23070123049)
5. Akshit Mathur (23070123014)

**Summary:**

This project helps in **automation**, and **system health tracking** through modular scripting, visualization, and GitHub-based workflow management.

**Git Commands Used:**

While development of the project we used several Git commands for version control as well as collabartion. These commands helped us in initializing the repository, committing changes, creating branches, and pushing updates to GitHub.

|  |  |
| --- | --- |
| **Command** | **Purpose** |
| git init | Initializes a new Git repository in the project directory |
| git add | Add files to staging area |
| git commit -m "Initial commit" | Save changes with a message |
| git push | Upload changes to Github |
| git pull | Get updates from Github |
| git branch | Create or view branches |

These Git commands ensured smooth version control, enabling efficient tracking of changes and collaboration among team members.

**Shell Scripting and ClI Commands Used**

Shell commands were extensively used during project setup and testing. These commands provided control over file handling, permissions, and program execution.

|  |  |
| --- | --- |
| **Command** | **Purpose** |
| ls | list files |
| cd | Change Directory |
| mkdir | Create folder |
| chmod +x | Give execute permission |
| cat | View file content |
| grep | Search inside files |
| ./run.sh | Run shell script |
| output.sh | Redirect output to file |

These CLI operations allowed easy testing, automation, and debugging of the program without needing a GUI environment.

**Use of .sh Scripts and Cron Jobs**

A shell script (run.sh) was created to automate the execution process. This script simplifies the workflow by running all required commands sequentially.

Sample run.sh Script:

Code

*#!/bin/bash*

*echo "Starting Automated Log Analyzer..."*

*python3 log\_parser.py /var/log/syslog*

*echo "Report successfully generated in output/report.txt"*

Additionally, **cron jobs** were used to schedule the execution of this script automatically at specific time intervals, ensuring consistent and timely analysis.

**Example of Cron Job Entry:**

*0 0 \* \* \* /home/user/log-analyzer/run.sh*

This line runs the script every day at midnight (00:00 hrs) to generate a daily log report automatically.