Enhanced System Health Monitors

Open Source Technology Project

Group Members:

Pratiksha Phadtare -22070123084
Purvansha Gehlod -2207012388
Sanskriti Jha -22070123101
Sanika Desai -22070123098

ENTC B 2022-26
Department of ENTC

Faculty Details:

Dr. Sudhanshu S. Gonge

Assistant Professor,
Department of CSE

Outline

1. Motivation
2. Background information/Related works
3. Proposed Method
4. Results
5. Contribution
6. Future Work
7. Link of GitHub Code
8. References

01 Motivation

Why is system monitoring crucial?

Maintaining System Performance

Modern systems handle multiple tasks, which can cause resource strain (e.g., high CPU or memory usage) if not monitored effectively.

Preventing Failures

Unchecked resource usage can lead to bottlenecks, crashes, or slowdowns, especially in systems with critical applications.

Early Issue Detection

Monitoring helps identify issues before they become catastrophic (e.g., disk space running out, overheating CPUs).

Objectives:

01 Provide real-time insights into system performance.

02 Monitor multiple system metrics (CPU, Memory, Disk, Network) in one place.

03 Trigger alerts when CPU or memory usage exceeds defined thresholds.

04 Offer users a simple, readable, and customizable tool for regular system health checks.

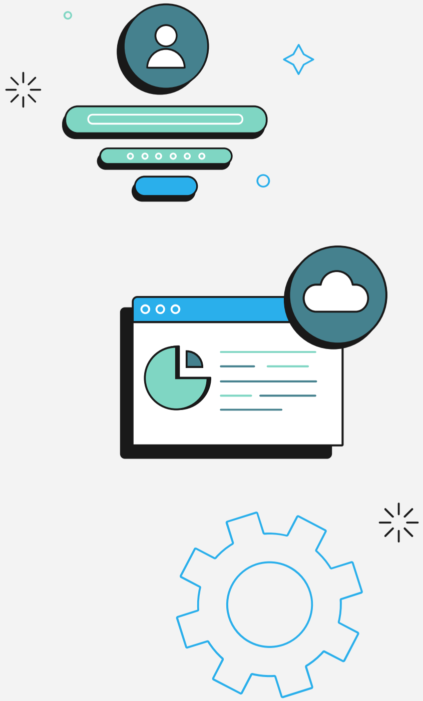
Why an Enhanced Script?

Existing Tools

While tools like `top`, `htop`, and `vmstat` provide information, they are either complex or lack customization.

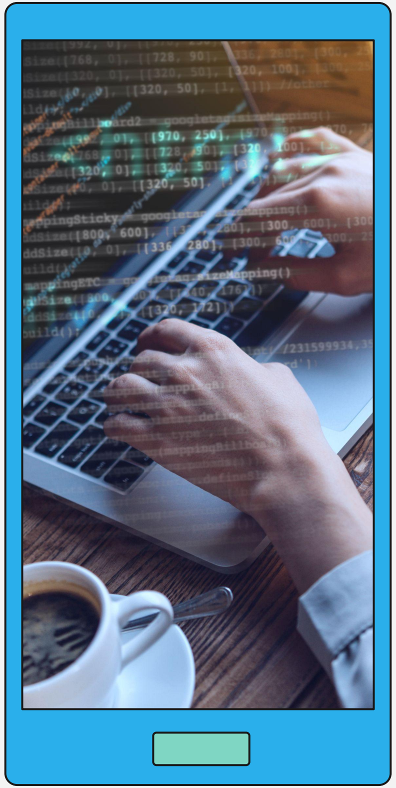
Our Approach

Develop a lightweight, script-based tool that is easy to use and can be customized based on user-defined thresholds.



02

Background Information/Related Works



Existing System Monitoring Solutions

- **Common Tools**: `top`, `htop`, `free`, `df`, `ifconfig`, `mpstat`
- **Limitations**:
  - **Complexity**: Tools like `top` and `htop` provide excessive detail, making them harder to interpret for non-technical users.
  - **Lack of Alerts**: Many existing tools don’t automatically notify users when resource usage exceeds critical levels.
  - **Scattered Information**: Users must run multiple tools to get a comprehensive picture of the system's health.

Related Projects

- **Linux Dash**: A web-based tool providing real-time insights on CPU, RAM, and network stats.

Limitation: Requires a web server, making it less suitable for users preferring simple scripts.

- **Glances**: A Python-based monitoring tool displaying CPU, memory, and disk usage.

Limitation: Can overwhelm users with too much information and requires additional setup.

Our Project

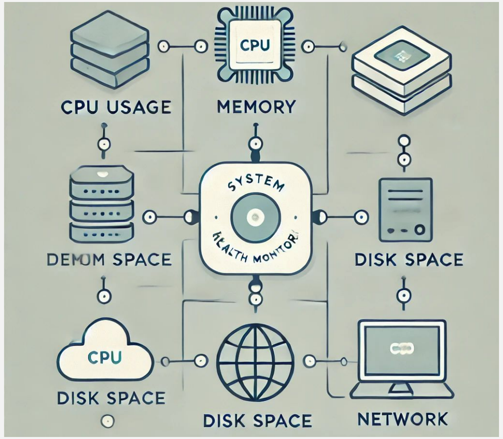
- Simplicity: Script-based, easy to use, no web server needed.
- Customization: Users can set custom thresholds for alerts.

03

Proposed methods

Overview of the Enhanced Health Monitor Script

- **Objective:**
  - Monitor system performance in real-time.
  - Key checks: CPU, memory, disk usage, load average, network activity, and processes.
  - Alerts: Warns if CPU or memory usage exceeds 80%.
- **Programming Language:** Bash Script
- **Visuals:**



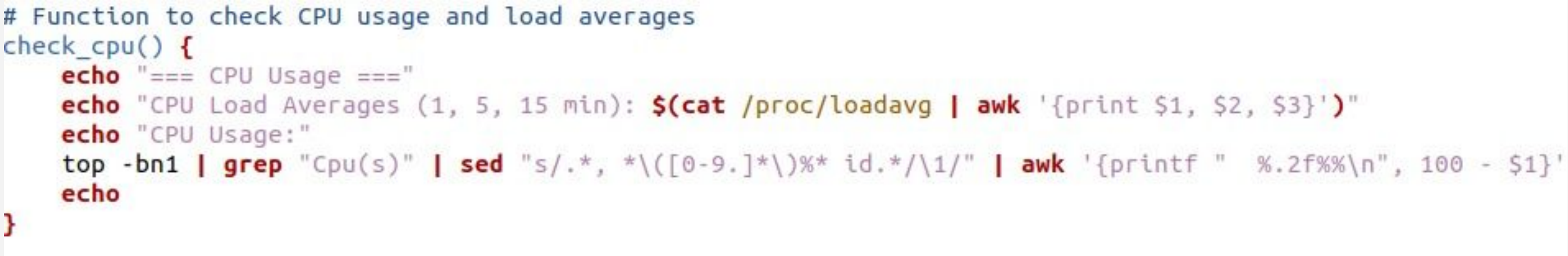
Breakdown of Code and Key Insights

Threshold Setup:

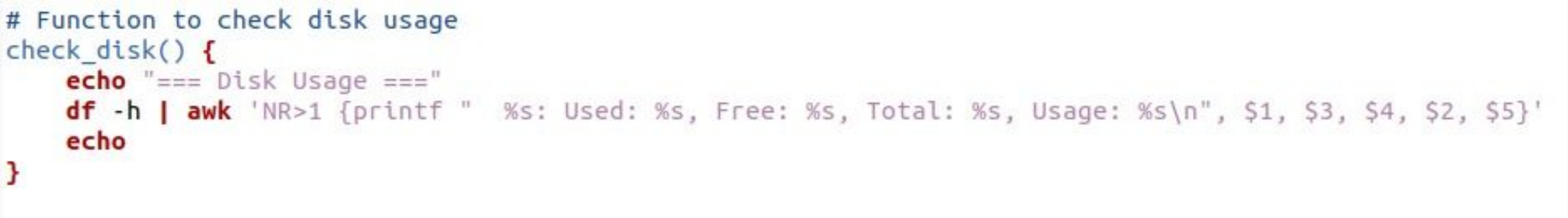
  - CPU_THRESHOLD = 80%, MEMORY_THRESHOLD = 80%

- **Key Functions:**
  - `check_cpu()`: Monitors CPU usage with `mpstat`.
  - `check_memory()`: Uses `free` to get memory usage stats.
  - `check_disk()`: Reports disk space with `df -h`.
  - `check_load()`: Extracts system load averages from `/proc/loadavg`.
  - `check_network()`: Displays network activity using `ip -s link`.
  - `check_processes()`: Lists top memory-hogging processes

Snippets of each function with labels or annotations.

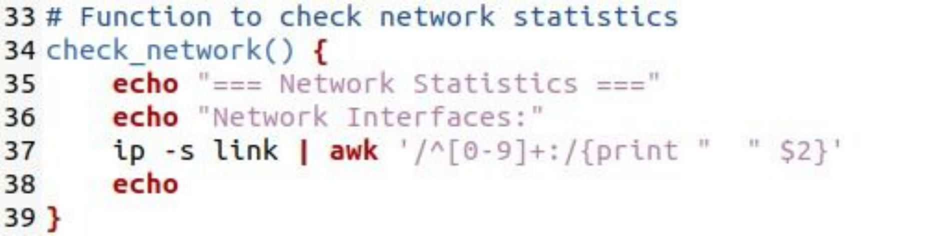


```
# Function to check CPU usage and load averages
check_cpu() {
    echo "=== CPU Usage ==="
    echo "CPU Load Averages (1, 5, 15 min): $(cat /proc/loadavg | awk '{print $1, $2, $3}')"
    echo "CPU Usage:"
    top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{printf "  %.2f%%\n", 100 - $1}'
    echo
}
```

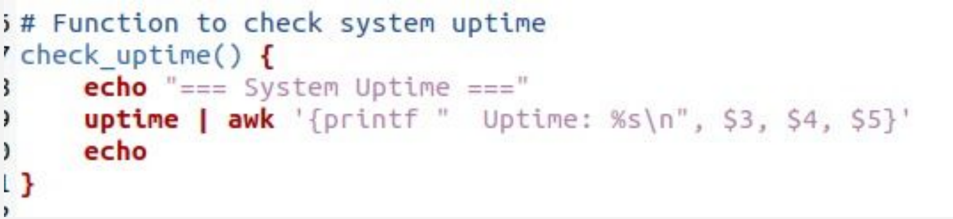


```
# Function to check disk usage
check_disk() {
    echo "=== Disk Usage ==="
    df -h | awk 'NR>1 {printf "  %s: Used: %s, Free: %s, Total: %s, Usage: %s\n", $1, $3, $4, $2, $5}'
    echo
}
```

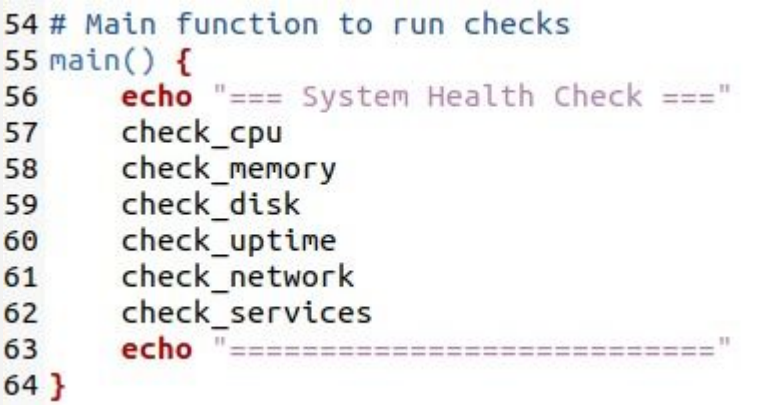
Snippets of each function with labels or annotations.



```bash
# Function to check network statistics
check_network() {
    echo "=== Network Statistics ==="
    echo "Network Interfaces:"
    ip -s link | awk '/^[0-9]+:/{print "  " $2}'
    echo
}
```



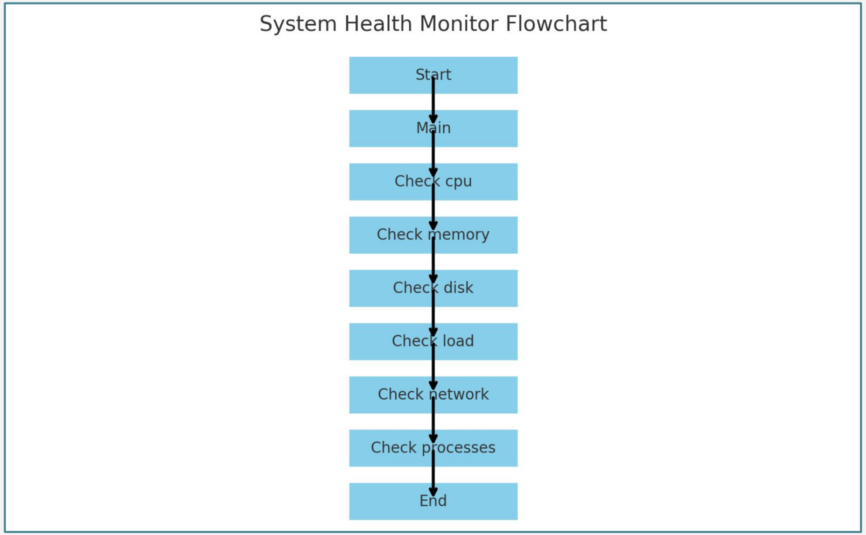
```bash
# Function to check system uptime
check_uptime() {
    echo "=== System Uptime ==="
    uptime | awk '{printf "  Uptime: %s\n", $3, $4, $5}'
    echo
}
```

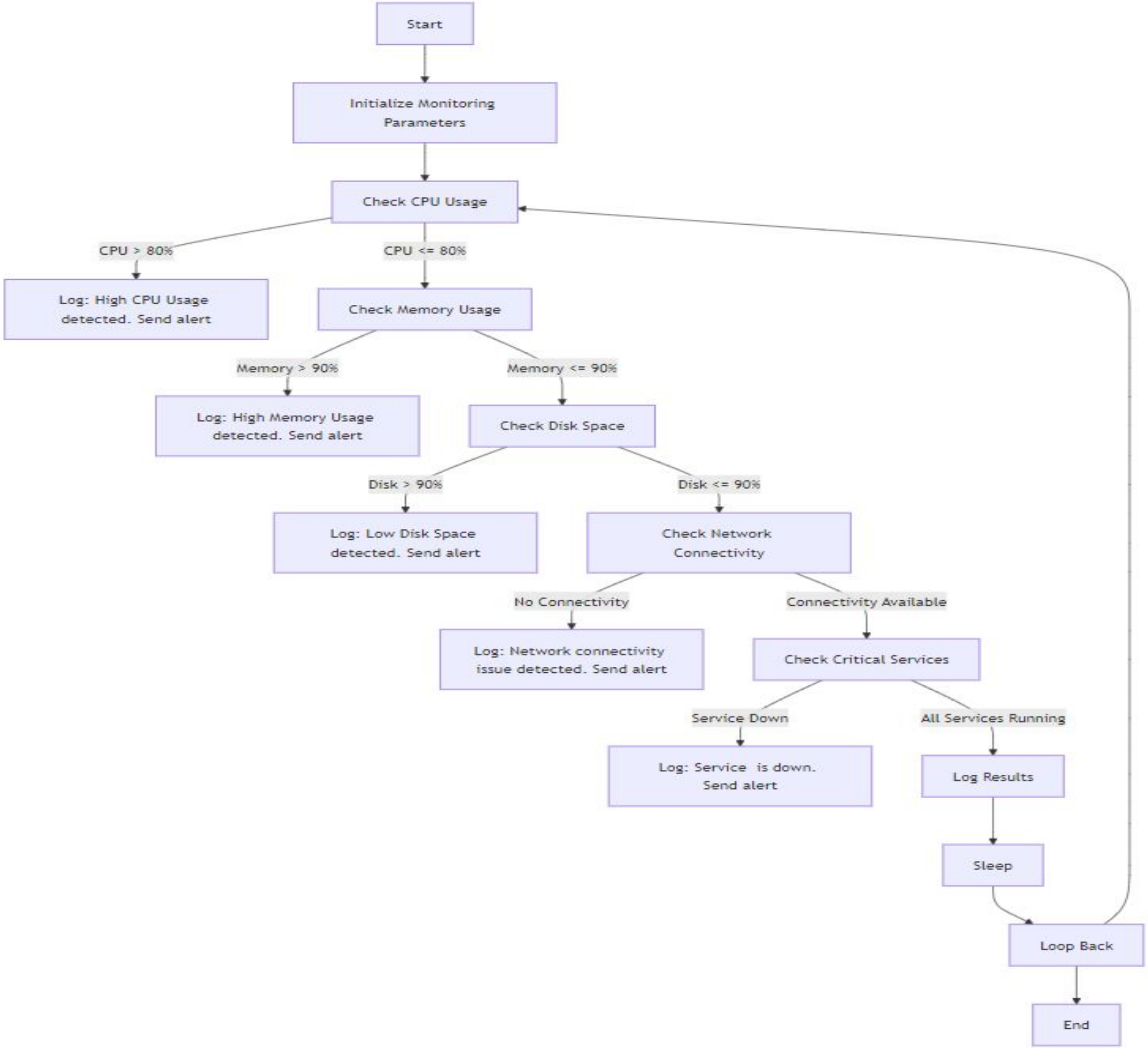


```bash
# Main function to run checks
main() {
    echo "=== System Health Check ==="
    check_cpu
    check_memory
    check_disk
    check_uptime
    check_network
    check_services
    echo "==========================="
}
```

How the Script Works

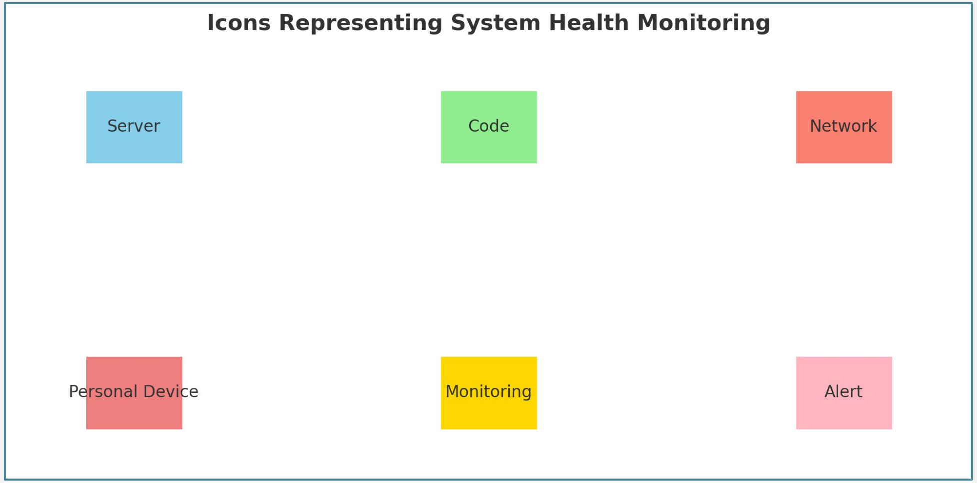
Execution Flow: All functions are called sequentially in the main block. Outputs a detailed report with real-time data.
Alerts: Warnings appear if CPU or memory usage exceeds 80%. Data Flow: CPU & **Memory Usage:** Tracks system load to identify bottlenecks. Network Usage: Observes transmitted and received bytes.
Processes: Displays top processes consuming resources.
Visuals:




Start
Initialize Monitoring Parameters
Check CPU Usage
CPU > 80%
CPU <= 80%
Log: High CPU Usage detected. Send alert
Check Memory Usage
Memory > 90%
Memory <= 90%
Log: High Memory Usage detected. Send alert
Check Disk Space
Disk > 90%
Disk <= 90%
Log: Low Disk Space detected. Send alert
Check Network Connectivity
No Connectivity
Connectivity Available
Log: Network connectivity issue detected. Send alert
Check Critical Services
Service Down
All Services Running
Log: Service is down. Send alert
Log Results
Sleep
Loop Back
End

Applications of the Health Monitor System

Administrators: Monitor servers to prevent downtime.
Developers: Track resource usage for performance optimization.
Network Engineers: Analyze network traffic to detect anomalies.
Personal Use: Monitor laptops or PCs for memory or process issues.
Educational Use: Learn about automation and system monitoring via Bash scripting.



04

Results

System Health Overview

- **Purpose**: To monitor critical system metrics in real-time.
- **Checks Performed**:
  - CPU Usage
  - Memory Usage
  - Disk Space Usage
  - Load Average
  - Network Usage
  - Running Processes
- **Importance**: Helps maintain system performance and prevent potential issues.

CPU and Memory Usage

CPU Usage:
Method: mpstat command.
Example Output:

```
===== CPU Usage =====
CPU Usage: 75.00%
WARNING: CPU usage exceeds 80%
```

Significance: High CPU usage can lead to performance degradation.
Memory Usage:
Method: free -h command.
Example Output:

```
===== Memory Usage =====
Used: 3.5G / Total: 8G (43.75%)
WARNING: Memory usage exceeds 80%
```

Significance: Prevents out-of-memory errors and application crashes.

Disk Space and Load Average

Disk Space Usage:

Method: df -h command.
Example Output:

```
===== Disk Space Usage =====
Used: 30G / Total: 100G (30.00%)
```

Significance: Ensures sufficient space for applications and processes.

Load Average:
Method: Reading /proc/loadavg.
Example Output:

```
===== Load Average =====
Load Average (1 min, 5 min, 15 min): 0.75, 0.50, 0.25
```

Significance: Indicates CPU load; high values may signal overload.

Network Usage and Running Processes

Network Usage:

Method: ip -s link.

Example Output:

```
===== Network Usage =====
Received: 1024 bytes
Transmitted: 2048 bytes
```

Significance: Identifies data transmission bottlenecks.

Running Processes:

Method: ps aux --sort=-%mem.

Example Output:

```
===== Running Processes =====
process1: 12.00%
process2: 8.50%
```

Significance: Allows management of resource-intensive applications.

05

Contributions

Issue Resolution and Tracking

Key Points:

- **Identifying Bugs:**
  - Conducted thorough testing to identify potential bugs in the script.
  - Examples of bugs found and resolved.
- **Tracking Issues:**
  - Utilized issue tracking tools (e.g., GitHub Issues) to log identified problems.
  - Documented steps for reproducibility, making it easier for others to understand the issues.

Resolutions Implemented: Collaborated with team members to devise and implement effective solutions, ensuring that fixes were thoroughly tested before deployment.
Continuous Monitoring: After deploying fixes, monitored system performance to verify that issues were resolved and to prevent recurrence.

Tracking and Raising Issues

- **Description:** Maintained a detailed log of issues and feature requests.
- **Process:**
  - Used GitHub's issue tracking to report and monitor bugs.
  - Collaborated with team members to prioritize issues based on severity.
- **GitHub Issue Tracking:** Used GitHub's issue tracking system to maintain a clear and detailed log of reported issues and feature requests, ensuring transparency and accountability.
- **Structured Reporting:** Created structured issue reports that included descriptions, labels (bug, feature, enhancement), and severity levels, aiding in prioritization.
- **Team Collaboration:** Worked closely with team members to assess the urgency of issues, facilitating discussions on resource allocation and timelines for resolutions.

Enhancements in Functionality and Performance

Description: Contributed to optimizing code and improving performance.

Key Contributions:

- Reduced code complexity by optimizing algorithms.
- Enhanced response time of system checks.
- Improved overall efficiency of the script.

Performance Analysis: Conducted analyses to identify performance bottlenecks linked to tracked issues, leading to informed decisions on optimization efforts.

System Optimization: Implemented code changes that enhanced system performance, resulting in faster response times and reduced resource consumption.

Optimizing Lines of Code (LOC):

Code Review: Participated in code reviews to identify areas for improvement, focusing on reducing complexity and redundancy in the codebase.

Best Practices Implementation: Ensured that coding best practices were followed, including modularization and documentation, to improve maintainability.

LOC Reduction: Achieved a measurable reduction in LOC through refactoring and consolidating functions, enhancing overall code quality and readability.

06

Future Works

Opportunities for Enhancement

Enhanced Data Visualization:

- **Problem:** Current system statistics are presented in a textual format, which may not be intuitive for all users.
- **Opportunity:** Develop graphical representations (charts/graphs) of CPU, memory, and disk usage for clearer insights and quicker assessments of system health.

Integration with Cloud Monitoring Tools:

- **Problem:** Limited monitoring capabilities for systems spread across multiple locations or using cloud services.
- **Opportunity:** Explore integration with popular cloud monitoring platforms (e.g., AWS CloudWatch, Azure Monitor) to provide a comprehensive view of resources across environments.

Automated Alerts and Notifications:

- **Problem:** Current alerts are manually checked and may result in delayed responses to critical issues.
- **Opportunity:** Implement automated alerts via email or messaging apps (e.g., Slack) to notify system administrators of critical thresholds being breached.

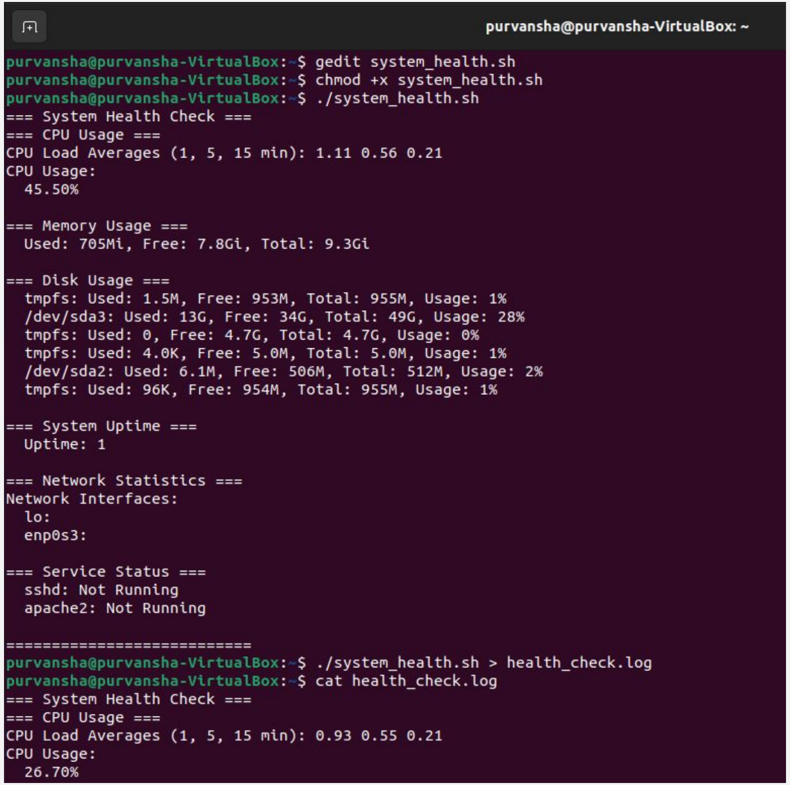
User Customization Options:

- **Problem:** The existing tool has fixed thresholds for alerts, which may not suit every user's needs.
- **Opportunity:** Develop user-configurable settings for thresholds and notifications, allowing customization based on individual system requirements.

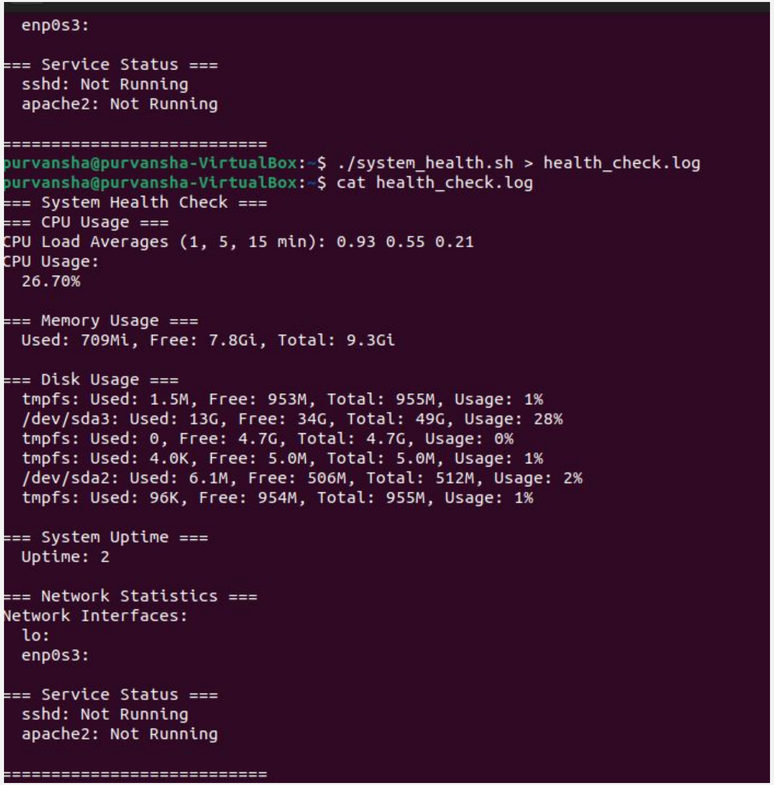
07

Screenshot and Link of GitHub Code

SCREENSHOT OF TERMINAL COMMAND:

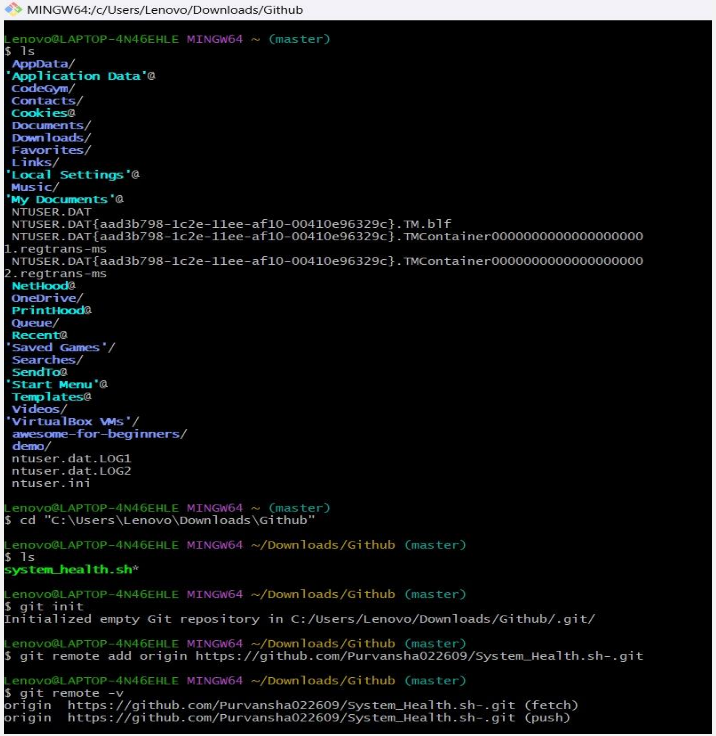


```
purvansha@purvansha-VirtualBox:~$ gedit system_health.sh
purvansha@purvansha-VirtualBox:~$ chmod +x system_health.sh
purvansha@purvansha-VirtualBox:~$ ./system_health.sh
=== System Health Check ===
=== CPU Usage ===
CPU Load Averages (1, 5, 15 min): 1.11 0.56 0.21
CPU Usage:
  45.50%

=== Memory Usage ===
  Used: 705Mi, Free: 7.8Gi, Total: 9.3Gi

=== Disk Usage ===
  tmpfs: Used: 1.5M, Free: 953M, Total: 955M, Usage: 1%
  /dev/sda3: Used: 13G, Free: 34G, Total: 49G, Usage: 28%
  tmpfs: Used: 0, Free: 4.7G, Total: 4.7G, Usage: 0%
  tmpfs: Used: 4.0K, Free: 5.0M, Total: 5.0M, Usage: 1%
  /dev/sda2: Used: 6.1M, Free: 506M, Total: 512M, Usage: 2%
  tmpfs: Used: 96K, Free: 954M, Total: 955M, Usage: 1%

=== System Uptime ===
  Uptime: 1

=== Network Statistics ===
Network Interfaces:
  lo:
  enp0s3:

=== Service Status ===
  sshd: Not Running
  apache2: Not Running

===========================
purvansha@purvansha-VirtualBox:~$ ./system_health.sh > health_check.log
purvansha@purvansha-VirtualBox:~$ cat health_check.log
=== System Health Check ===
=== CPU Usage ===
CPU Load Averages (1, 5, 15 min): 0.93 0.55 0.21
CPU Usage:
  26.70%
```



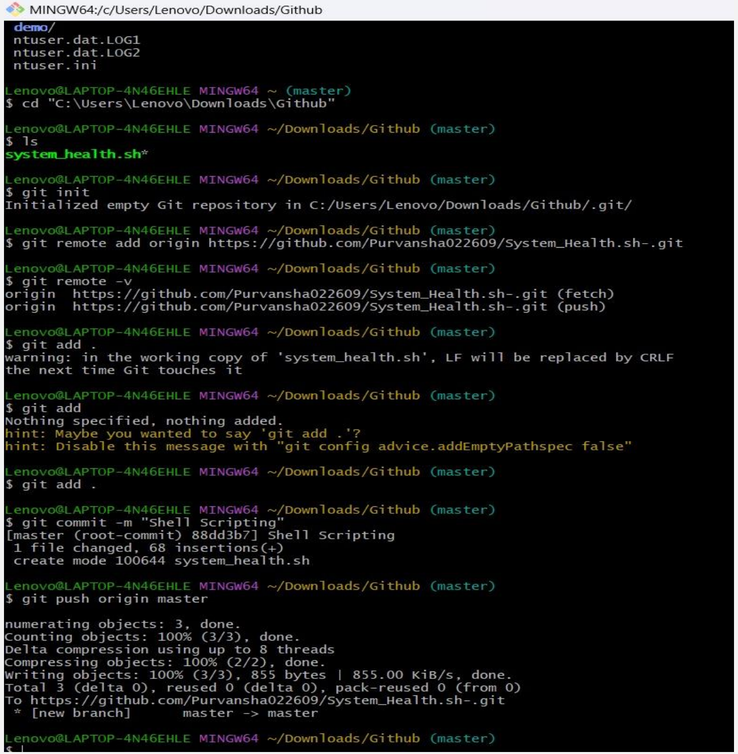
```
  enp0s3:

=== Service Status ===
  sshd: Not Running
  apache2: Not Running

===========================
purvansha@purvansha-VirtualBox:~$ ./system_health.sh > health_check.log
purvansha@purvansha-VirtualBox:~$ cat health_check.log
=== System Health Check ===
=== CPU Usage ===
CPU Load Averages (1, 5, 15 min): 0.93 0.55 0.21
CPU Usage:
  26.70%

=== Memory Usage ===
  Used: 709Mi, Free: 7.8Gi, Total: 9.3Gi

=== Disk Usage ===
  tmpfs: Used: 1.5M, Free: 953M, Total: 955M, Usage: 1%
  /dev/sda3: Used: 13G, Free: 34G, Total: 49G, Usage: 28%
  tmpfs: Used: 0, Free: 4.7G, Total: 4.7G, Usage: 0%
  tmpfs: Used: 4.0K, Free: 5.0M, Total: 5.0M, Usage: 1%
  /dev/sda2: Used: 6.1M, Free: 506M, Total: 512M, Usage: 2%
  tmpfs: Used: 96K, Free: 954M, Total: 955M, Usage: 1%

=== System Uptime ===
  Uptime: 2

=== Network Statistics ===
Network Interfaces:
  lo:
  enp0s3:

=== Service Status ===
  sshd: Not Running
  apache2: Not Running

===========================
```

SCREENSHOT OF GIT BASH COMMANDS:



MINGW64:/c/Users/Lenovo/Downloads/Github

```
Lenovo@LAPTOP-4N46EHLE MINGW64 ~ (master)
$ ls
 AppData/
'Application Data'@
 CodeGym/
 Contacts/
 Cookies@
 Documents/
 Downloads/
 Favorites/
 Links/
'Local Settings'@
 Music/
'My Documents'@
 NTUSER.DAT
 NTUSER.DAT{aad3b798-1c2e-11ee-af10-00410e96329c}.TM.blf
 NTUSER.DAT{aad3b798-1c2e-11ee-af10-00410e96329c}.TMContainer0000000000000000000
1.regtrans-ms
 NTUSER.DAT{aad3b798-1c2e-11ee-af10-00410e96329c}.TMContainer0000000000000000000
2.regtrans-ms
 NetHood@
 OneDrive/
 PrintHood@
 Queue/
 Recent@
'Saved Games'/
 Searches/
 SendTo@
'Start Menu'@
 Templates@
 Videos/
'VirtualBox VMs'/
 awesome-for-beginners/
 demo/
 ntuser.dat.LOG1
 ntuser.dat.LOG2
 ntuser.ini

Lenovo@LAPTOP-4N46EHLE MINGW64 ~ (master)
$ cd "C:\Users\Lenovo\Downloads\Github"

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ ls
system_health.sh*

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git init
Initialized empty Git repository in C:/Users/Lenovo/Downloads/Github/.git/

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git remote add origin https://github.com/Purvansha022609/System_Health.sh-.git

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git remote -v
origin  https://github.com/Purvansha022609/System_Health.sh-.git (fetch)
origin  https://github.com/Purvansha022609/System_Health.sh-.git (push)
```

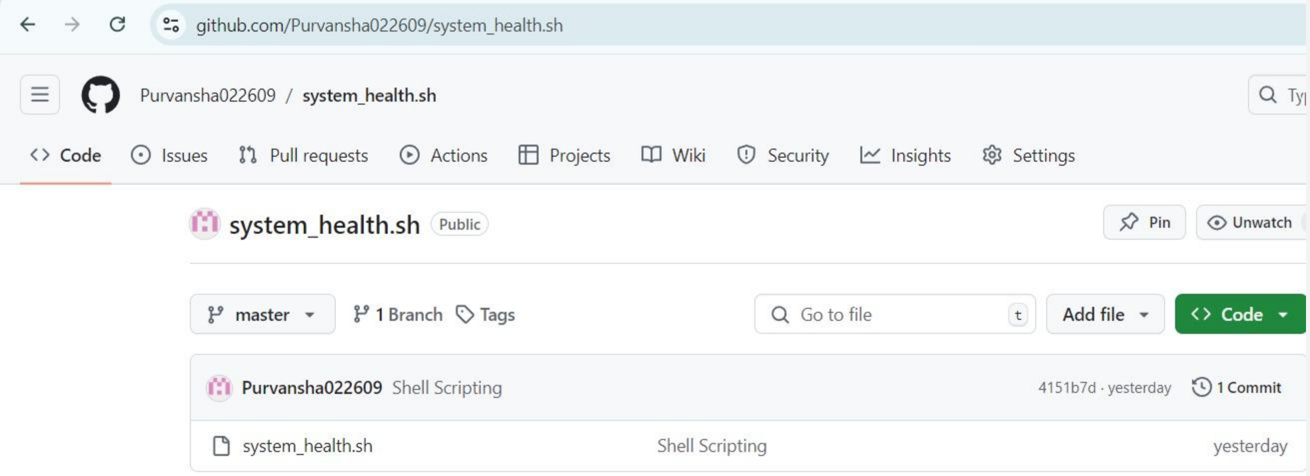


MINGW64:/c/Users/Lenovo/Downloads/Github

```
 demo/
 ntuser.dat.LOG1
 ntuser.dat.LOG2
 ntuser.ini

Lenovo@LAPTOP-4N46EHLE MINGW64 ~ (master)
$ cd "C:\Users\Lenovo\Downloads\Github"

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ ls
system_health.sh*

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git init
Initialized empty Git repository in C:/Users/Lenovo/Downloads/Github/.git/

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git remote add origin https://github.com/Purvansha022609/System_Health.sh-.git

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git remote -v
origin  https://github.com/Purvansha022609/System_Health.sh-.git (fetch)
origin  https://github.com/Purvansha022609/System_Health.sh-.git (push)

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git add .
warning: in the working copy of 'system_health.sh', LF will be replaced by CRLF
the next time Git touches it

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Disable this message with "git config advice.addEmptyPathspec false"

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git add .

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git commit -m "Shell Scripting"
[master (root-commit) 88dd3b7] Shell Scripting
 1 file changed, 68 insertions(+)
 create mode 100644 system_health.sh

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ git push origin master

numerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 855 bytes | 855.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Purvansha022609/System_Health.sh-.git
 * [new branch]      master -> master

Lenovo@LAPTOP-4N46EHLE MINGW64 ~/Downloads/Github (master)
$ |
```

LINK OF GITHUB CODE:

https://github.com/Purvansha022609/system_health.sh



Purvansha022609 Shell Scripting 4151b7d · yesterday 1 Commit

system_health.sh	Shell Scripting	yesterday

08

References

Bash Scripting Documentation:

- GNU Bash Reference Manual. Retrieved from GNU Bash Documentation.

System Monitoring Tools:

- Linux System Monitoring: Tools and Techniques. Retrieved from DigitalOcean.

mpstat Utility:

- Linux Manual Pages: mpstat. Retrieved from Linux Man Pages.

free Command Documentation:

- Linux Manual Pages: free. Retrieved from Linux Man Pages.

df Command Documentation:

- Linux Manual Pages: df. Retrieved from Linux Man Pages..

GitHub Issues and Collaboration:

- GitHub Documentation: Issues. Retrieved from GitHub Docs.

THANK YOU