System Health Monitoring Dashboard

CS 474 - Individual Project

Kelin Argueta

**SYSTEMS PROJECTS - Directions and Mandatory Requirements**

1. Maintain a Git repository that is updated weekly and use a repository platform (e.g., GitHub) that forces you to use command-line tools to manage and update your project rather than web-front ends (e.g., GitLab).
2. Include the following components and features:
   1. Scripting (e.g., bash in Linux or PowerShell in Windows)
   2. Some kind of a UI (not console) front-end
   3. An API interaction
3. You can follow a tutorial that builds a project similar to or even the same as your selected project. However, if this is the case, you must include additional requirements (to be determined after you have submitted your project description).

---------------------------------------------------------------------------------------------------------------

**Question 1**  
**a) Project Description**

The System Health Monitoring Dashboard is a web-based application designed to monitor and display critical system metrics, including **CPU usage, memory, process, and load average**. The tool will gather real-time data from the host system and presents it through an intuitive user interface (UI), providing users with insights into their system’s health and performance.

**Functions:**

• Monitor CPU, memory, disk usage, and system temperature.

• Display real-time metrics through a web UI.  
• Send alerts via SMS when specified thresholds are exceeded.

**External API:**

I will use Flask, and Twilio as an external API to notify users when their system metrics cross defined thresholds.

**Tech Stack:**

• Backend: Python with Flask to build the web dashboard.  
• Frontend: JavaScript, HTML  
• Scripting: Bash scripts for gathering system health data.

• Database: MySQL for storing data.  
• Version Control: Git for repository management.

**b) Technologies to Learn**

• Flask  
• JavaScript  
• Bash

• Twilio

**Question 2**  
a) Learning Resources

1. Flask Dashboard Tutorial  
<https://flask.palletsprojects.com/en/3.0.x/>

This website provides documentation that covers all aspects of using the Flask framework, including routing, templates, and deployment.

2. Twilio SMS API Tutorial <https://www.twilio.com/docs/iam/connect/quickstart/python>

This provides step-by-step instructions on how to send and receive SMS messages using the Twilio API in Python.

3. Bash Scripting Tutorial https://www.tutorialspoint.com/unix/unix-using-variables.htm  
This tutorial covers Bash scripting, essential for collecting system metrics like CPU, memory, and disk usage.

4. Learn PowerShell https://blog.netwrix.com/powershell-scripting-tutorial/ https://www.youtube.com/watch?v=IHrGresKu2w

These links provide a beginner-friendly tutorial to understand the basics of PowerShell scripting in Windows.

5. JavaScript Basics https://github.com/shaijut/Mozilla-Javascript-Tutorial-Offline/blob/master/JavaScript% 20MDN%20Web%20Docs.doc  
http://edu-9.de/uploads/books/javascript-handbook.pdf

These links provide free tutorials from MDN Web Docs that cover the fundamentals of JavaScript, including how to write and execute scripts for web development.

**Question 3**  
Project Tasks Breakdown (5 Weeks)

|  |  |
| --- | --- |
| **wk** | **Tasks to be completed** |
| 1 | 1. Set up the project repository and initialize project structure. 2. Write Bash/PowerShell scripts to collect system metrics (**CPU usage, memory, process, and load average**.) 3. Research the external API (Twilio) for notification integration. 4. Implement backend using Flask (Python). |
| 2 | 1. Set up the API integration with Twilio. 2. Test the API notification with sample data. 3. Build the front-end UI to display system metrics. 4. Integrate the backend with the UI to display real-time system data (e.g., CPU, memory, disk usage). |
| 3 | 1. Test the UI for proper data retrieval and display. 2. Finalize the UI the and improve design. 3. Add additional features, such as threshold configuration for alerts. 4. Implement logging of system data. |
| 4 | 1. Conduct full testing and resolve any bugs. 2. Finalize project for submission, ensuring Git repository is updated. 3. Review and complete documentation (e.g., README.dm, user instructions) |

Table 1: Example of week-based planned task completion for the duration of the entire project.

**Question 4**  
Set up a Git public repository on GitHub.

Public Repository URL: https://github.com/arguetakelin/CS\_474\_System\_Health\_Monitoring\_Dashboard.git.

**Question 5**Implementation and completion of your individual project.

**Question 6**

Presentation and demo of your individual project.

**Implementation Process:**

1. Run this command:

for i in {1..20}; do top -l1 | head -n 5 >> top\_output.txt; echo "----" >> top\_output.txt; sleep 30; done

1. Using bash, parse, extract the fields you want to use and add them to a text CSV file separated by a comma.
2. Setup a MySQL database and create a table to store the values (you can do a CSV upload to the table) of the CSV file.
3. Setup a one-page webpage so you can first press a button and see the contents of the database.
4. Create a chart by querying the database and showing it on the webpage.