TechComp System Health Monitoring Bot

Test Documentation v1.0

Jamie Irvine 50103233

Table of Contents

[Use Case 2](#_Toc182864329)

[Test Scenario 3](#_Toc182864330)

[User Story 3](#_Toc182864331)

[Test Cases 4](#_Toc182864332)

[Further Program Development 7](#_Toc182864333)

# Use Case

TechComp, a company specializing in providing automated solutions for system monitoring, creates a "System Health Monitoring Bot" that tracks CPU and memory usage.

**Actors**

* End User: A person who inputs their system's CPU and memory usage to check system health.

**Prerequisites**

* The end user knows their current CPU usage (in percentage) and memory usage (in GB).
* The program runs in a Python environment.

**Steps**

1. The program prompts the user to enter their CPU usage percentage.
2. The program prompts the user to enter their memory usage in GB.
3. The program determines the system's CPU and memory health by using the below criteria:

CPU Usage:

* Less than 40% = "Underutilized"
* 40% - 75% = "Optimal"
* More than 75% = "Overloaded"

Memory Usage:

* Less than 4GB = "Underutilized"
* 4GB - 8GB = "Optimal"
* More than 8GB = "Overloaded"

4. The program displays results based on user’s input.

5. The user reviews the results and may make changes to their system based on the results

**Benefits**

* Allows users to monitor their system health and make changes if required
* Could be used by TechComp to sell PC upgrades to users such as better CPU or RAM

# Test Scenario

## User Story

**User story for my program is as follows:**

As a computer user, I wish to assess my system's health in order to understand whether my CPU and memory are performing optimally.

**Description:**

The system health assessment tool should allow me to input my current CPU (%) and memory (GB) usage. Based on these inputs, it will categorize the usage as "Underutilized," "Optimal," or "Overloaded".

**Acceptance Criteria:**

The program should prompt me, the user, to enter my CPU usage in percentage and memory usage in GB.

The program determines the system's CPU and memory health by using the below criteria:

CPU Usage:

* Less than 40% = "Underutilized"
* 40% - 75% = "Optimal"
* More than 75% = "Overloaded"

Memory Usage:

* Less than 4GB = "Underutilized"
* 4GB - 8GB = "Optimal"
* More than 8GB = "Overloaded"

The program outputs the health status for both CPU and memory.

**Example:**

I am using the program,

I input my CPU usage as 60% and memory usage as 2 GB,

The program should display:

CPU Status: Optimal (60%)

Memory Status: Underutilized (2 GB)

## Test Cases

The first test was to inspect my code for syntax errors and ensure that my syntax listed the correct conditions for ‘Underutilized, Optimal & Overloaded.’

A screenshot of a computer program

Description automatically generated

I then completed a series of test cases to ensure that my program outputs the correct expected results.

The results of my completed test cases are listed in the below test table:

| **Test Case** | **CPU Usage (%)** | **Memory Usage (GB)** | **Expected Output** | **Actual Output** |
| --- | --- | --- | --- | --- |
| 1 | 30 | 2 | CPU: Underutilized, Memory: Underutilized | CPU: Underutilized, Memory: Underutilized |
| 2 | 60 | 6 | CPU: Optimal, Memory: Optimal | CPU: Optimal, Memory: Optimal |
| 3 | 85 | 9 | CPU: Overloaded, Memory: Overloaded | CPU: Overloaded, Memory: Overloaded |
| 4 | 50 | 11 | CPU: Optimal, Memory: Overloaded | CPU: Optimal, Memory: Overloaded |

I have included screenshots of my test case results below:

Test Case 1:

A screenshot of a computer program

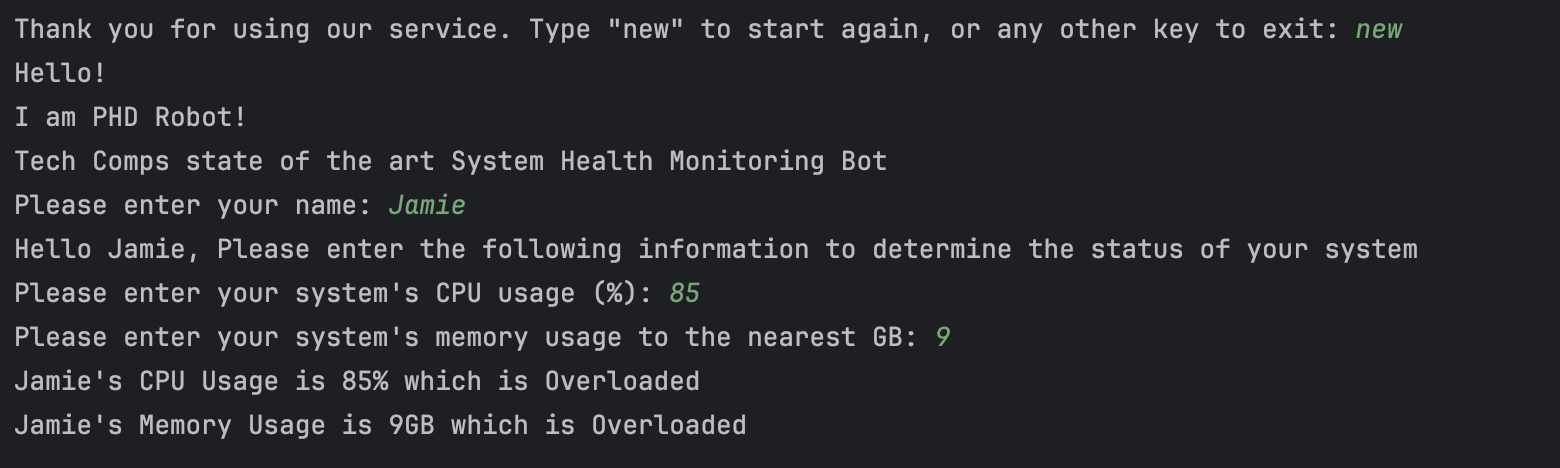
Description automatically generated

Test Case 2:

A screenshot of a computer program

Description automatically generated

Test Case 3:



Test Case 4:

A screenshot of a computer program

Description automatically generated

All my test cases outputted the correct expected result.

For further testing I attempted to enter non expected characters as the program user. I designed the program to validate the users input to only accept 0-100% for CPU usage and only accept integers for memory usage.

Here are the results of my tests:

A screenshot of a computer

Description automatically generated

As expected, the program produced an error message when non numerical characters were entered and when a percentage higher than 100 was entered.

I also checked my interrupter ‘PyCharm’ for run time errors and the interrupter is currently displaying no run time errors.

A screenshot of a computer

Description automatically generated

# Further Program Development

A screen shot of a computer

Description automatically generated

Initially I coded my program to ask the user to enter an integer for CPU usage, this caused an error as sometimes during testing I would enter a % symbol after the integer which would cause the program to crash.

To resolve this I used the .endswith method to write an IF statement. If users input ends with %, then the program would use the .strip method to remove the last index of users inputted string.

After removing the % symbol from users inputted string, the program converts the users cpu\_input to an integer by using an IF statement, if cpu\_input.isdigit() then cpu\_input would convert to an integer listed as cpu\_usage variable. It was important to convert the users string to an integer so as not to impact my IF, ELIF, ELSE statements which are used to determine CPU and memory status.

A screenshot of a computer program

Description automatically generated

These amendments to my code now allow the user to enter their CPU usage as either numerical digits (integers), or as numerical digits along with a % symbol such as 80%.

I then used this learning to apply the same logic to the user’s memory input. If the user enters gb or GB along with their memory figure, the program removes the GB and converts the string to an integer. A screenshot of a computer program

Description automatically generated

I tested the above changes for both CPU and Memory inputs and both worked as expected.

I also added and tested a loop function whereby the program asks the user to type ‘new’ if they would like to start again. When user types ‘new’, the program will start again from the beginning. This can be seen in my test case screenshots above.

If user enters any other input at this stage the program will end as seen below:

A black screen with white text

Description automatically generated