Health monitoring system

NAME: Karthik Manoj
ROLL No: 44

INTRODUCTION

The Health Monitoring System is a user-friendly C-based application designed to help individuals track their essential health metrics: blood pressure, heart rate, and blood glucose levels. Through a simple menu interface, users can record, display, save, and load their health data, making it easy to monitor their health over time. This tool provides an efficient way to manage personal health information, ensuring users have quick access to their important health parameters whenever needed.

OVER-VIEW

The Health Monitoring System operates through a simple menu-driven interface, providing users with clear options to interact with their health data. The system ensures ease of use with straightforward prompts and responses, making it accessible for users with basic computer skills. Features are ; Record Health Parameters: Users can input their current health metrics directly into the program. Display Health Parameters: The recorded metrics can be displayed on the screen for quick review. Save Health Parameters to File: Users can save their health data to a text file for future reference. Load Health Parameters from File: Previously saved health data can be loaded from a text file, allowing users to track their health over time.

OBJECTIVE

The objective of the Health Monitoring System project is to develop a simple and user-friendly application that enables individuals to:1. RECORD their essential health metrics, including blood pressure, heart rate, and blood glucose levels.2. DISPLAY their recorded health data for immediate review and monitoring.3. SAVE health data to a file for future reference and continuity.4. LOAD previously saved health data to track changes and trends over time.5. PROMOTE BETTER HEALTH MANAGEMENT by providing an accessible tool for regular health monitoring, thus facilitating timely medical intervention and informed decision-making.

SYSTEM REQUIREMENT

Hardware Requirements:- Processor: Intel Pentium IV or equivalent- RAM: Minimum 512 MB- Storage: Minimum 50 MB of free disk space- Display: VGA or higher resolution monitorSoftware Requirements:- Operating System: Compatible with Windows, macOS, or Linux- Compiler: GCC (GNU Compiler Collection) or any C compiler supporting ANSI C standards- Text Editor/IDE: Any text editor (e.g., Visual Studio Code, Sublime Text) or IDE (e.g., Code::Blocks, Eclipse)

DESIGN AND DEVELOPEMENT

The design and development of the Health Monitoring System involve creating a straightforward application in C to manage essential health metrics: blood pressure, heart rate, and blood glucose levels. The system utilizes a structured approach with a struct to store these parameters, ensuring efficient data handling. It features a menu-driven interface that allows users to record, display, save to file, and load from file seamlessly. Error handling is implemented to manage file operations and validate user inputs, ensuring reliability. Development focuses on clarity and simplicity, aiming for easy navigation and effective health data management for users.

PROGRAM LOGIC

1. Initialization: Define a structure to store health parameters. - Initialize variables for user input and file handling.
2. Menu Interface: - Display options to record, display, save, load data, or exit.
3. Functions: - Record: Capture and store user-inputted health data. - Display: Show current health parameters. - Save: Write data to a text file. - Load: Read data from the file to update parameters.
4. File Handling: - Manage file operations to store and retrieve data.
5. Error Handling: - Check for file operation errors and validate user inputs.
6. Execution: - Continuously loop through menu options until the user chooses to exit.
7. Termination: - Exit the program upon user request

```
Define structure HealthParameters {
    Integer bp
    Integer heartRate
    Float Glucose
}

Function main():
    Initialize HealthParameters params
    Integer choice

    Loop:
        Display menu options:
            1. Record Health
Parameters
            2. Display Health
Parameters
            3. Save Health Parameters
to File
            4. Load Health Parameters
from File
            5. Exit

        Get user choice and store in
choice

        Switch (choice):
            Case 1:
                Call
recordParameters(params)
                Break
            Case 2:
                Call
displayParameters(params)
                Break
            Case 3:
                Call
saveParameters(params)
                Break
            Case 4:
                Call
loadParameters(params)
                Break
            Case 5:
                Call exitProgram()
                Break
            Default:
                Display "Invalid
choice. Please try again."

Function recordParameters(params):
    Display "Enter blood pressure:"
    Input params.bp
    Display "Enter heart rate:"
    Input params.heartRate
    Display "Enter blood glucose
level:"
    Input params.Glucose

Function displayParameters(params):
    Display "Current Health
Parameters:"
    Display "Blood Pressure: " +
params.bp
    Display "Heart Rate: " +
params.heartRate
    Display "Blood Glucose Level: " +
params.Glucose
```

```
params.heartRate
    Display "Blood Glucose Level: " +
params.Glucose

Function saveParameters(params):
    Open file "health_parameters.txt"
in write mode
    Write params.bp, params.heartRate,
params.Glucose to file
    Close file

Function loadParameters(params):
    Open file "health_parameters.txt"
in read mode
    Read params.bp, params.heartRate,
params.Glucose from file
    Close file

Function exitProgram():
    Display "Exiting..."
    Terminate program
```

TEST CASE

1. Record Parameters: - Input valid blood pressure, heart rate, and blood glucose values. - Verify stored values in the system.
2. Display Parameters: - Check if current health parameters display correctly. - Ensure formatting and data accuracy.
3. Save Parameters to File: - Save health parameters to a file (health_parameters.txt). - Confirm data is correctly written to the file.
4. Load Parameters from File: - Load health parameters from health_parameters.txt. - Verify loaded data matches stored parameters.
5. Invalid Input Handling: - Test system response to non-numeric inputs or invalid formats during parameter recording. - Ensure error messages are displayed appropriately.
6. Menu Navigation: - Navigate through menu options (Record, Display, Save, Load, Exit). - Confirm smooth transition between

```
Health Monitoring System
1. Record Health Parameters
2. Display Health Parameters
3. Save Health Parameters to File
4. Load Health Parameters from File
5. Exit
Enter your choice: 1
Enter blood pressure: 100
Enter heart rate: 75
Enter blood glucose level: 120

Health Monitoring System
1. Record Health Parameters
2. Display Health Parameters
3. Save Health Parameters to File
4. Load Health Parameters from File
5. Exit
Enter your choice: 2

Health Parameters:
Blood Pressure: 100
Heart Rate: 75
Blood Glucose Level: 120.00
```

```
Health Monitoring System
1. Record Health Parameters
2. Display Health Parameters
3. Save Health Parameters to File
4. Load Health Parameters from File
5. Exit
Enter your choice: 3
Health parameters saved to file.

Health Monitoring System
1. Record Health Parameters
2. Display Health Parameters
3. Save Health Parameters to File
4. Load Health Parameters from File
5. Exit
Enter your choice: 4
Health parameters loaded from file.

Health Monitoring System
1. Record Health Parameters
2. Display Health Parameters
3. Save Health Parameters to File
4. Load Health Parameters from File
5. Exit
Enter your choice: 5
Exiting...
```

```
1  100 75 120.00
2
```

CONCLUSION

The Health Monitoring System effectively manages and displays health parameters such as blood pressure, heart rate, and blood glucose. It ensures data integrity through reliable file operations and handles user inputs with robust error handling. Overall, the system provides a user-friendly interface for monitoring and storing essential health metrics.

Future enhancements

1. Graphical User Interface (GUI): - Integrate a GUI for a more intuitive user experience.
2. Data Analytics: - Implement data analytics to provide insights and trends over time.
3. Cloud Integration: - Enable cloud storage and synchronization for remote access to health data.
4. Alerts and Notifications: - Introduce alerts for abnormal health readings based on predefined thresholds.
5. Integration with Wearable Devices: - Support integration with wearable health devices for real-time data syncing.
6. Multi-user Support: - Extend functionality to support multiple users with personalized health profiles.
7. Mobile Application: - Develop a mobile app version for on-the-go health monitoring and management.