Health Management System (Project Report)

Student Name	**Piyush Sindhu**
Roll Number	590028080
Semester	First Semester
Course Subject	Programming In C
Course Code	CSEG1032

Abstract

This Project Implements a Modular **Health Management System** Using the C Programming Language.

The System Is Designed To

- Manage Basic Health Records.
- Calculate Body Mass Index (BMI) With Standard Category Classification.
- Provides Exercise Recommendations Based on User Goals.

It Demonstrates the Application of Structured Programming, Modular Design & Practical Algorithms to Solve Real World Problems in Health Tracking.

The Program Is Menu Driven and Allows Users To

- Add, View & Delete Health Records.
- It Calculates BMI Using the Standard Formula and Classifies Results into Categories Such as Underweight, Normal, Overweight & Obese (Class I–III).
- Additionally, The System Provides Exercise Recommendations for Weight Gain, Weight Loss or Weight Maintenance Depending on The User's Baseline Activity Level and Target Weight.

This Project Highlights Key Programming Concepts Such as Structures, Functions, File Handling & Modularity.

It Serves as a Foundation for Building Scalable Health Applications and Demonstrates How C Can Be Used to Design Practical, Real World Systems.

Problem Definition

Problem Statement

Health Management Is an Essential Aspect of Modern Life. Many Individuals Struggle to Track Their Health Data, Calculate BMI & Plan Exercise Routines Effectively.

Manual Tracking Is Error Prone and Inefficient, Leading to Poor Health Outcomes.

Objectives

- To Design a Modular C Program That Manages Health Records.
- To Implement BMI Calculation and Classification.
- To Provide Exercise Recommendations Based on User Goals.
- To Demonstrate Structured Programming & Modular Design Principles.

Scope

The System Includes

- Basic Record Management (Name, Gender, Age & ID).
- BMI Calculation & Classification.
- Exercise Recommendation System.
- Sample Input/Output for Testing.

System Design & Algorithm

Architecture Overview

The System Is Divided into Modules

- **main.c** → Controls Program Flow and User Interaction.
- **basic_data.c** → Handles Record Management (Add, View & Delete).
- **bmi.c** → Performs BMI Calculation & Classification & Provides Exercise Recommendations.

Data Structures

- **Struct Record** → Stores Name, Gender, Age & ID (Declared In basic_data.h).
- **Struct Health** → Stores Weight, Height & BMI Category (Declared In bmi.h).

- **Struct Exercise** → Stores Target Weight, Activity Level & Goal Type (Declared In bmi.h).

Functions

- addRecord () → Adds a New Health Record.
- displayRecords () → Displays Existing Records.
- deleteRecord () → Removes One Record by ID or Deletes All Available Records.
- calculateBmi () → Computes BMI & Classifies Its Corresponding Category, Also Provides Personalized Exercise Recommendations.

Algorithm (Add Record)

- Prompt User for Name, Gender, Age & ID.
- Store Details in a Struct Record.
- Append Record to File Using fwrite () With File Handling.
- Confirm Success & Return to Main Menu.

Algorithm (Calculate BMI)

1. Input Weight (In Kg) And Height (In M).
2. Compute BMI Using Weight / (Height × Height) & Inline Functions.
3. Compare BMI With Thresholds
    - BMI < 18.5 → Underweight
    - BMI <= 24.9 → Normal
    - BMI <= 29.9 → Overweight
    - BMI <= 34.9 → Obese Class I
    - BMI <= 39.9 → Obese Class II
    - BMI > 39.9 → Obese Class III

Display BMI Value and Category.

Algorithm (Exercise Recommendation)

1. Input Goal Weight & Baseline Activity Level.

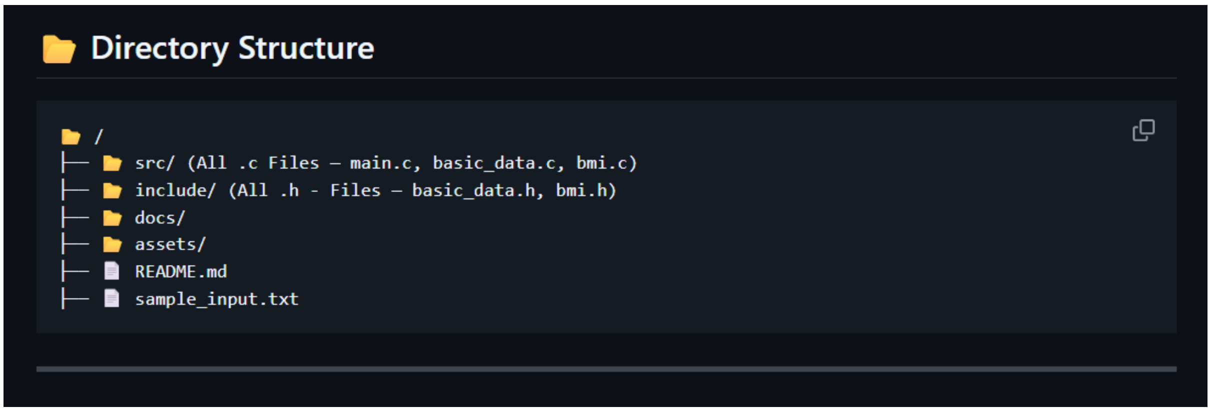
2. If goal > current & BMI Category (Underweight) → Recommend Gain Weight Plan.
3. If goal < current & BMI Category (Obese Class I & II) → Recommend Lose Weight Plan.
4. If goal ~ current → Recommend Maintain Weight Plan.
5. For Obese Class III (The Program Prints a Statement Mentioning That Seek Immediate Medical Advise).
6. Display Exercise Recommendation.

Implementation Details

Concepts Demonstrated

- 🏗 **Structures & Functions** — Used to Define Health Records and Organize Related Operations.
- 💾 **File Operations** — (fopen, fwrite, fread, remove, rename) For Persistent Storage & Record Management.
- 🔗 **Pointer Based String Manipulation** — Handling User Input and String Operations Efficiently.
- 📦 **Modular Programming Using Header Files** - Separating Interfaces .h From Implementations .c Files.
- **Menu Driven Program Design** — Guiding Users Through Options with a Clear Interface.
- 🛡 **Use of Inline Functions** — Safer BMI & Weight Calculations Compared to Macros.

GitHub Repository Structure



Directory Structure

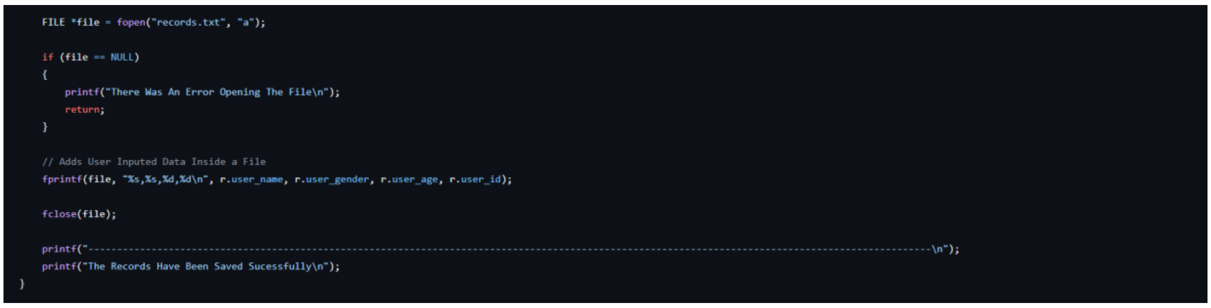
```
/
├── src/ (All .c Files – main.c, basic_data.c, bmi.c)
├── include/ (All .h - Files – basic_data.h, bmi.h)
├── docs/
├── assets/
├── README.md
├── sample_input.txt
```

Each Module Is Implemented with Clear Separation of Concerns

- src – Contains All The (.c) Files.
- Include – Contains All The (.h) Files.
- docs – Contains the Project Report File.
- assets – Program Screenshots & Diagrams.
- README.md – Gives a Brief Overview About This Project.
- sample_input.txt – Provides Sample Test Cases for All Possible Outputs.

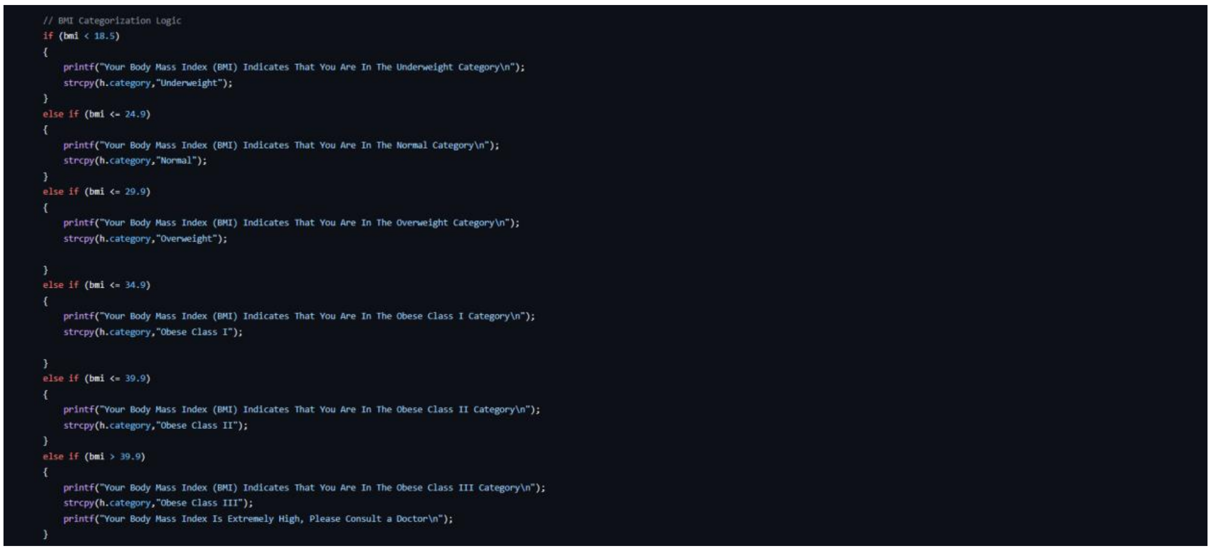
Code Snippets

Snippet Adding a New Record



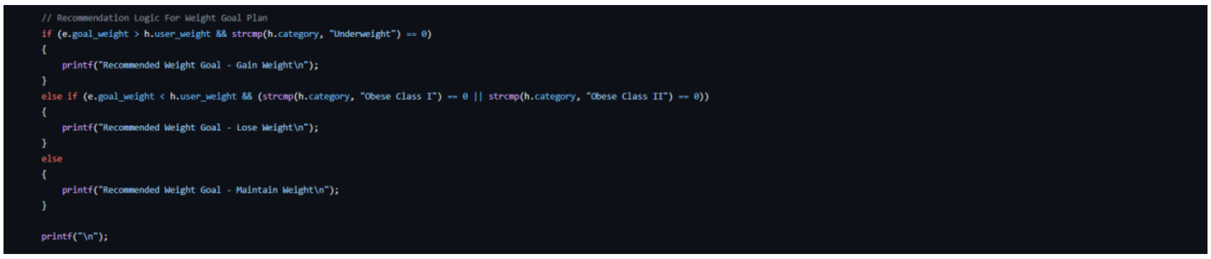
```c
    FILE *file = fopen("records.txt", "a");

    if (file == NULL)
    {
        printf("There Was An Error Opening The File\n");
        return;
    }

    // Adds User Inputed Data Inside a File
    fprintf(file, "%s,%s,%d,%d\n", r.user_name, r.user_gender, r.user_age, r.user_id);

    fclose(file);

    printf("----------------------------------------------------------------------------------------------------------------------------------\n");
    printf("The Records Have Been Saved Sucessfully\n");
}
```

Snippet Body Mass Index (BMI) Logic



```c
// BMI Categorization Logic
if (bmi < 18.5)
{
    printf("Your Body Mass Index (BMI) Indicates That You Are In The Underweight Category\n");
    strcpy(h.category,"Underweight");
}
else if (bmi <= 24.9)
{
    printf("Your Body Mass Index (BMI) Indicates That You Are In The Normal Category\n");
    strcpy(h.category,"Normal");
}
else if (bmi <= 29.9)
{
    printf("Your Body Mass Index (BMI) Indicates That You Are In The Overweight Category\n");
    strcpy(h.category,"Overweight");

}
else if (bmi <= 34.9)
{
    printf("Your Body Mass Index (BMI) Indicates That You Are In The Obese Class I Category\n");
    strcpy(h.category,"Obese Class I");

}
else if (bmi <= 39.9)
{
    printf("Your Body Mass Index (BMI) Indicates That You Are In The Obese Class II Category\n");
    strcpy(h.category,"Obese Class II");
}
else if (bmi > 39.9)
{
    printf("Your Body Mass Index (BMI) Indicates That You Are In The Obese Class III Category\n");
    strcpy(h.category,"Obese Class III");
    printf("Your Body Mass Index Is Extremely High, Please Consult a Doctor\n");
}
```

Snippet Weight Goal Recommendation



```c
// Recommendation Logic For Weight Goal Plan
if (e.goal_weight > h.user_weight && strcmp(h.category, "Underweight") == 0)
{
    printf("Recommended Weight Goal - Gain Weight\n");
}
else if (e.goal_weight < h.user_weight && (strcmp(h.category, "Obese Class I") == 0 || strcmp(h.category, "Obese Class II") == 0))
{
    printf("Recommended Weight Goal - Lose Weight\n");
}
else
{
    printf("Recommended Weight Goal - Maintain Weight\n");
}

printf("\n");
```

Testing & Results

The System Was Tested Using the Inputs Provided Inside (sample_input.txt)

- **BMI Calculation -** Correctly Classified Test Cases into Underweight, Normal, Overweight & Obese Categories.
- **Exercise Recommendation -** Suggested Appropriate Plans for Weight Gain, Loss & Maintenance.
- **Record Management -** Successfully Added, Viewed & Deleted Records.

Screenshots Of Execution Are Included In (assets/).

Conclusion & Future Work

The Health Management System Successfully Demonstrates the Use of C Programming to Build a Modular, Menu Driven Application.

It Highlights the Importance of Structured Design, Modularity & Practical Application of Algorithms.

The Project Provided Valuable Learning in File Handling, Structures & Modular Programming Serving as a Foundation for More Advanced Health Applications.

🚀 Future Work

- 🍎 **Dietary Recommendations Based** on Body Mass Index (BMI).
- 🗄 **Database Integration** for Persistent Storage.
- 🏃 **Expanded Exercise Library** with Intensity Scaling.
- 🌐 **GUI Or Web Interface** for Better Usability.

⚠ Limitations

- 🚫 **No Dynamic Memory Allocation** — Records Are Handled Using Static Structures & Arrays Only.
- 📉 **Limited Exercise Recommendation Library** — Suggestions Are Basic and Not Comprehensive for All BMI Categories or Fitness Goals.
- 📄 **Records Stored in Plain Text Files** — No Database Integration, Which Limits Scalability and Advanced Querying.
- 🔄 **Single User Focus** — The System Does Not Yet Support Multiple User Accounts or Concurrent Record Management.
- 🎨 **Console Based Interface Only** — No Graphical User Interface (GUI) Or Web Interface for Improved Usability.

📚 References

- 🚫 No External References Were Used in The Development of This Project.
- ✍ All Code and Documentation Were Created Independently.

👥 Team Details

- 🏵 **Member** - MahiruAmane.
- □ **Note** — This Project Was Completed **Individually**.

Appendix

Sample Input Test Cases

Please Refer to the (sample_input.txt) File for All Available Test Cases Categorized as Follows

- BMI Calculation
- Weight Management
- Exercise Recommendation

Output Screenshots

Please Refer to the (assets/) Folder for All Available Screenshots Demonstrating the Complete Operation of This Program.

----- End Of Project Report -----