

Project 1 - Health Data Management System

1 Project Introduction

The system is designed to help users record, analyze and manage health data, display health trends through charts, and provide personalized health advice based on user data. The system is suitable for fitness enthusiasts, patients with chronic diseases (such as high blood pressure) and medical personnel. Users can input data such as weight, blood pressure, heart rate, exercise, sleep, etc., set daily health reminders, and import data from smart devices or export reports for doctors' reference.

2 Business Requirements

Goal: To help users fully understand their health status and adjust their lifestyle in time according to data feedback to prevent or improve health problems.

Key Features:

- **Health Data Recording:** Users can input data such as weight, blood pressure, heart rate, exercise, and sleep.
- **Data Visualization:** Use charts and trend graphs to help users visually understand the changes in their health status.
- **Health Reminders:** Provide daily reminders (e.g., to drink water, take medication, or exercise) to improve the efficiency of health management.
- **Personalized Health Advice:** Based on data analysis, offer tailored advice on diet, exercise, and lifestyle adjustments.
- **Data Import/Export:** Support importing data from smart devices and exporting health reports for reference by doctors.

Legend: ■ for nouns, ■ for verbs

3 Nouns-Verbs

Nouns:

- User
- Health Data
- health problems
- health status
- feedback
- weight
- blood Pressure
- heart rate
- exercise
- sleep
- charts
- graphs
- Reminders
- health management

- health Advice
- diet
- lifestyle adjustments
- health report
- doctors
- data analysis
- reference
- smart device

Verbs:

- Record
- input
- understand
- provide
- improve
- offer
- support
- import
- export
- manage
- prevent
- adjust

4 Target Audience

- **Fitness Enthusiasts:**

Those who pay attention to sports data, calorie consumption and training effects, and pursue body management and health improvement.

- **Chronic Disease Patients (e.g., High Blood Pressure Patients):**

Individuals who need to monitor indicators such as blood pressure regularly and hope to receive abnormal reminders and personalized suggestions.

- **Healthcare Professionals (Doctors):**

Professionals who hope to obtain patient health data reports to formulate treatment plans more comprehensively.

5 Rules

- **Data privacy and security:** All user data must be encrypted and stored in compliance with relevant laws and regulations.
- **Data accuracy:** User input data must be formatted, and imported device data must match the system data format.
- **Scheduled reminders:** The health reminder function must support user-defined time and reminder content.
- **User experience:** The interface is simple and easy to use, suitable for users of different

ages and health backgrounds.

6 Challenge Questions

- How to ensure privacy and security during data transmission and storage?
- How to achieve efficient integration of data from multiple sources (manual input and device import)?
- How to design a system that can meet the dynamic recording needs of fitness enthusiasts and is suitable for regular monitoring of patients with chronic diseases?
- How to provide accurate health advice? How to ensure the accuracy and personalization of data analysis algorithms?
- How to design reports that allow doctors to quickly obtain and analyze patient health data?

7 Classes, Attributes and Associations

Classes and attributes:

Class	Attributes
User	userID, name, age, gender, contactInfo
FitnessEnthusiast	workoutPreferences, fitnessGoals
Patient	medicalHistory, conditions, medications
Doctor	specialization, licenseNumber, department
HealthData	dataID, date, weight, bloodPressure, heartRate, exercise, sleep, remarks
HealthReminder	reminderID, type, content, time, status
HealthAdvice	adviceID, category, content
Report	reportID, dateRange, summary, exportStatus
DataManager	users, healthDataRecords, reports, reminders, advices
ReportBuilder	reportID, dateRange, summary, exportStatus
UserFactory	None

Associations¹:

- **User and Its Extended Classes:**

The User class serves as the base class for all users. Its extended classes include FitnessEnthusiast, Patient, and Doctor. These extended classes inherit all the basic attributes and associations defined in the User class

- **Association between User and HealthData:**

Each User can have multiple HealthData records.

This is a one-to-many composition relationship, which means that each HealthData instance must be associated with a User. If a User is deleted, all of the corresponding HealthData records will also be deleted.

- **Association between HealthData and HealthReminder:**

Each HealthData can be associated with multiple HealthReminders.

This relationship is a one-to-many aggregation, indicating that while HealthReminders are connected to a HealthData, they can exist and be managed independently within the system.

- **Association between HealthData and HealthAdvice:**

Each HealthData can have multiple HealthAdvice entries.

This one-to-many composition is used to store or generate personalized health advice for the user, based on their recorded health data.

- **Association between Doctor and Report:**

Each Doctor can generate multiple Reports.

This is a one-to-many relationship used to record and present the user's health data and trends over specific time periods.

- **Association between Doctor and Patient:**

As extensions of the User class, Doctor and Patient have a special relationship where a Doctor can be associated with multiple Patients.

This one-to-many relationship facilitates the management and monitoring of multiple patients' health data by a single doctor, allowing doctors to generate, view, and analyze health reports for each patient.

8 User Personas and User Stories

User personas

Persona 1: Alex (Fitness Enthusiast)

Age: 28

Background: A fitness enthusiast who is dedicated to body management and achieving optimal training results.

Usage Scenario: He aims to optimize his workout plan through detailed record-keeping and trend analysis.

Persona 2: Ms. Susan (High Blood Pressure Patient)

Age: 55

Background: A patient with high blood pressure who needs to monitor her blood pressure levels regularly.

Usage Scenario: She requires timely alerts for abnormal readings and actionable advice to manage her condition effectively.

Persona 3: Dr. Wang (Doctor)

Age: 40

Background: A doctor who needs to quickly access comprehensive health data of his patients.

Usage Scenario: He uses the system to review patients' overall health trends, which helps him in formulating effective treatment plans.

User stories

Persona 1: Fitness enthusiast (Alex)

- Record exercise data:

As a fitness enthusiast, I hope to enter exercise type, duration, calorie consumption and other data every day to track fitness progress.

- View trend charts:

As a fitness enthusiast, I hope to view exercise data trends through charts to intuitively understand the training effect.

- Receive personalized suggestions:

As a fitness enthusiast, I hope the system can provide diet and training optimization suggestions based on my data to continuously improve my fitness plan.

Persona 2: Hypertensive patient (Ms. Susan)

- Record blood pressure data:

As a hypertensive patient, I hope to enter blood pressure data regularly to monitor my health in real time.

- Abnormal reminder:

As a hypertensive patient, I hope that when the blood pressure value is abnormal, the system can send reminders in time to help me adjust my lifestyle or seek medical treatment.

- Trend analysis and suggestions:

As a hypertensive patient, I hope the system can generate a blood pressure change trend chart and provide personalized health suggestions to better manage my condition.

Persona 3: Doctor (Dr. Wang)

- View patient data:

As a doctor, I hope to easily view the patient's historical health data to understand their health trends.

- Export health report:

As a doctor, I hope that I can export detailed health reports so that I can refer to them during diagnosis and treatment.

- Data statistics and chart display:

As a doctor, I hope that the system can provide statistical analysis and chart display functions to help me quickly grasp the overall data distribution and change trends.

Dimensions²

Health Needs / Goals

- Fitness Enthusiast (Alex) focuses on performance and improvement.
- Patient (Ms. Susan) focuses on monitoring and stability.
- Doctor (Dr. Wang) focuses on comprehensive data and patient treatment.

Frequency of System Usage

- Fitness Enthusiast: Uses daily to track workouts and view trends.
- Patient: Uses regularly (e.g., daily or weekly) to record vitals and receive alerts.
- Doctor: Uses intermittently when reviewing multiple patients' data or generating reports.

Technical Proficiency

- **Fitness Enthusiast:** Generally tech-savvy, comfortable with recording data and using charts.
- **Patient:** May vary; might need simpler reminders or instructions.
- **Doctor:** Professionally motivated to use the system, moderate to high proficiency.

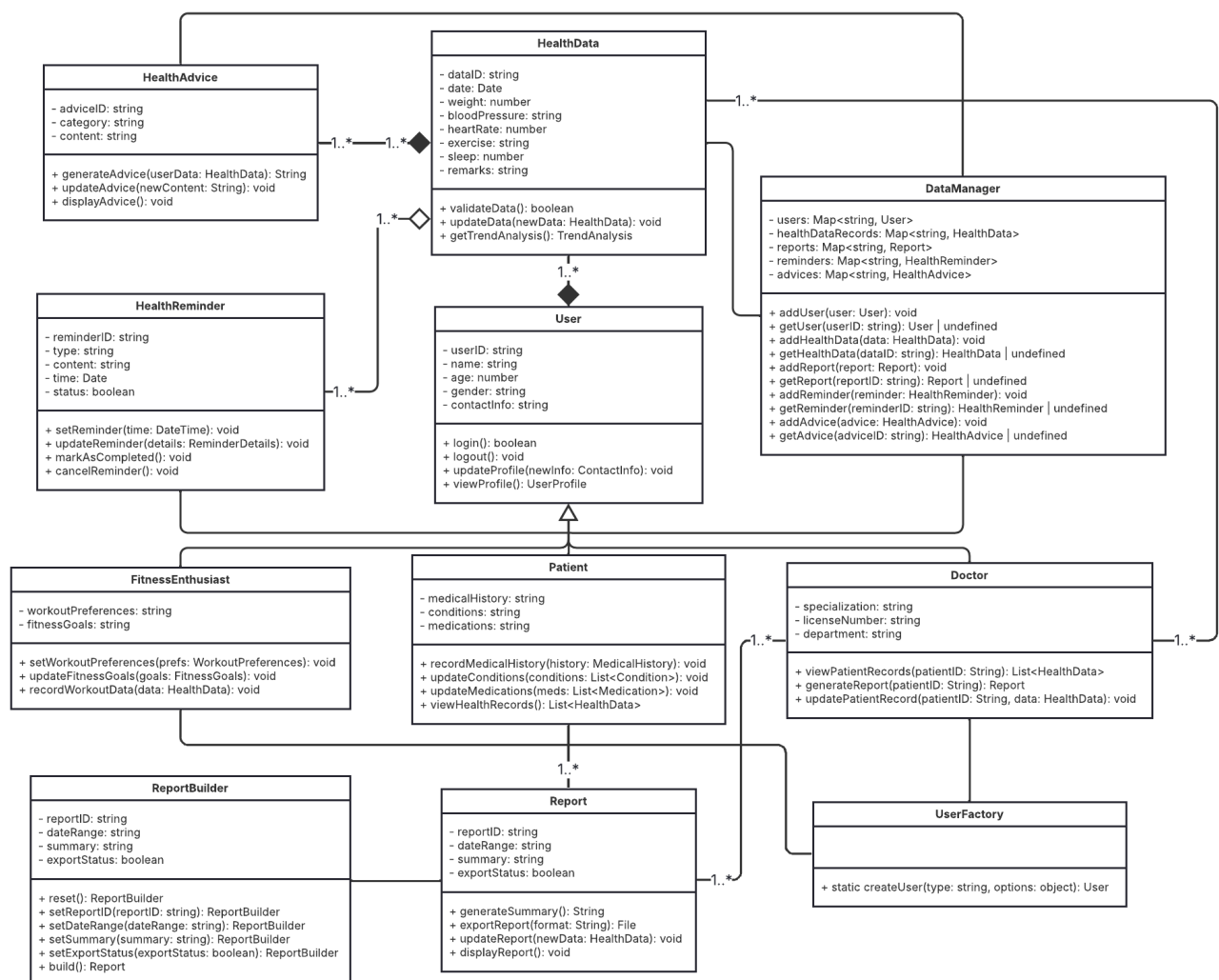
Age / Demographics

- Fitness Enthusiast (younger adult, actively pursuing fitness goals).
- Patient (middle-aged to senior, chronic condition management).
- Doctor (professional adult with medical expertise).

Primary Motivation

- Fitness Enthusiast: Optimize personal fitness routine and progress.
- Patient: Prevent complications and manage chronic disease.
- Doctor: Provide effective diagnosis and treatment, handle patient reports efficiently.

9 UML Class Diagram



10 Interface Low Level Mockups

Health Data Management System			
<div>Data Entry</div>			
Date: <input type="text" value="MM/DD/YYYY"/>			
exercise type: <input type="text" value="Swimming/etc."/>			
Duration: <input type="text" value="Min"/>			
calorie consumption: <input type="text" value="Cal"/>			
<div>Submit</div>			
Data Entry	Trend Charts	Health Advice	Settings

Exercise Data Entry Interface

(User story 1.1) User can enter exercise data at data entry interface, such as date, exercise type, duration and calorie consumption. User can submit data after that.

Health Data Management System			
<div>Line/Bar chart</div>			
<div>Filter</div> <div>Time range:</div> <div>From: <input type="text"/></div> <div>To: <input type="text"/></div>			
Data Entry	Trend Charts	Health Advice	Settings

Trend Charts Interface

(User story 1.2) User can view their data trend with line/bar chart at trend charts interface. User can also filter time range of the data trends.

Health Data Management System			
<p>Diet advice:</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p>Title</p> <p>Description</p> <p style="text-align: center;">View details</p> </div> <p>Exercise advice:</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p>Title</p> <p>Description</p> <p style="text-align: center;">View details</p> </div>			
Data Entry	Trend Charts	Health Advice	Settings

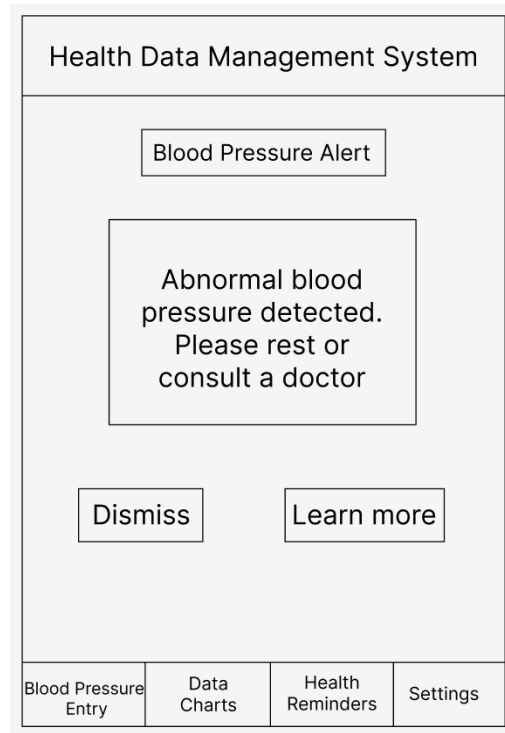
Personalized Advice Interface

(User story 1.3) User can get the health advice like diet advice and exercise advice. In each of advice, there are title, description and view details of advice.

Health Data Management System			
<div style="border: 1px solid black; padding: 5px; display: inline-block;">Blood Pressure Entry</div>			
<p>Time: <div style="border: 1px solid black; padding: 5px; display: inline-block;">HH/MM/SS MM/DD/YYYY</div></p> <p>systolic blood pressure: <div style="border: 1px solid black; width: 150px; height: 20px; display: inline-block;"></div></p> <p>Diastolic blood pressure: <div style="border: 1px solid black; width: 150px; height: 20px; display: inline-block;"></div></p> <p>Heart rate: <div style="border: 1px solid black; width: 150px; height: 20px; display: inline-block;"></div></p> <p style="text-align: center;"><div style="border: 1px solid black; padding: 5px; display: inline-block;">Save</div></p>			
Blood Pressure Entry	Data Charts	Health Reminders	Settings

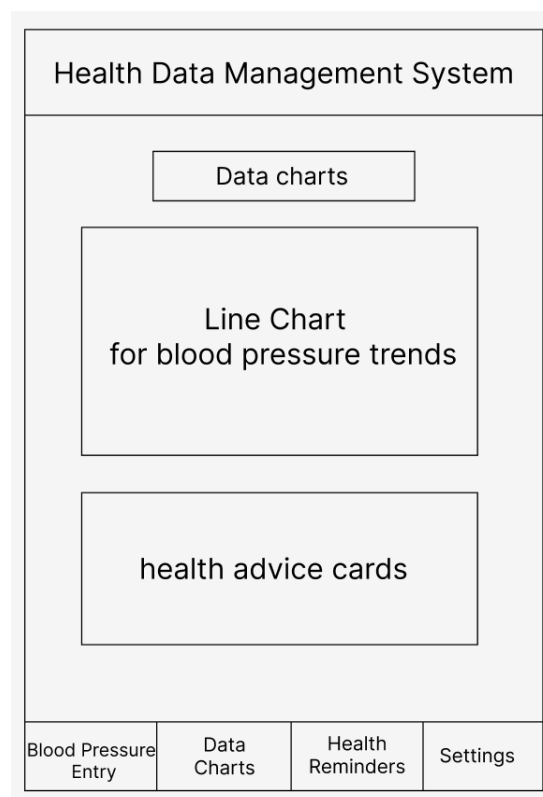
Blood Pressure Data Entry Interface

(User story 2.1) Patient can enter blood pressure and heart rate data with the timestamp in blood pressure entry interface.



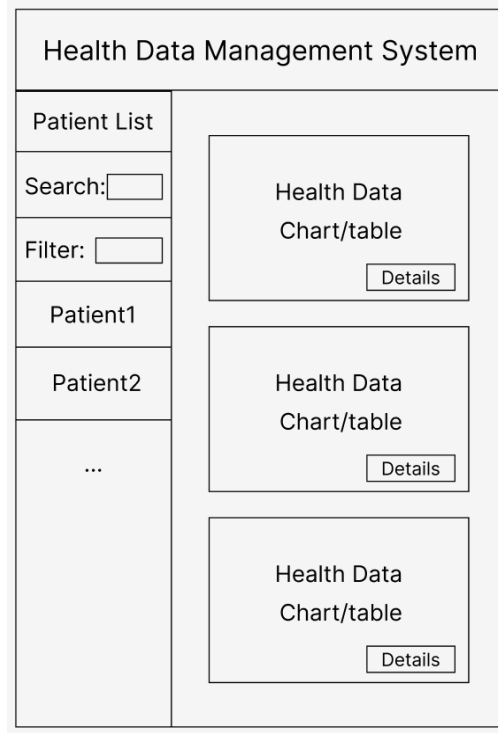
Abnormality Alert Interface

(User story 2.2) Patient can receive blood pressure alert when the data is abnormal(out of the normal range), patient can choose dismiss this alert or learn more to contact with doctor.



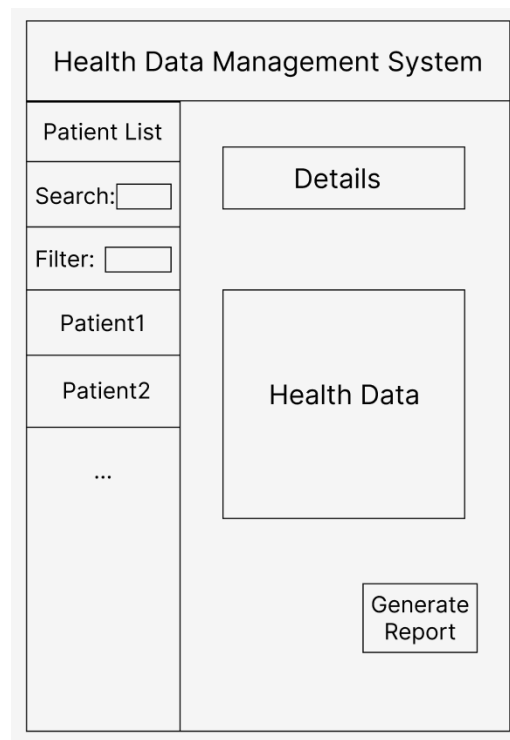
Trend Analysis and Advice Interface

(User story 2.3) Patient can view the health reminders with line chart for blood pressure trends and can receive a health advice cards to remind patient.



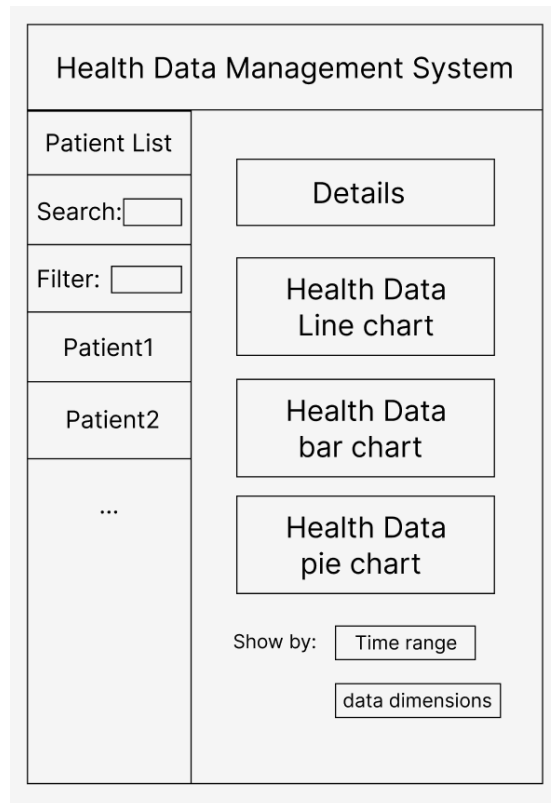
Patient Data View Interface

(User story 3.1) Doctor can search or filter to find the patient. Then, doctor can view the detailed health data with chart/table of patient. Doctor can also choose one of the data to view details.



Health Report Export Interface

(User story 3.2) Doctor can choose the details and generate report based on the specific health data and send to patient.



Data Statistics and Chart Display Interface

(User story 3.3) Doctor can view the details of health data with different charts. They will be shown by choose time range or data dimensions.

11 Links

UML diagram:

https://lucid.app/lucidchart/6e96ad6e-f806-4245-bce5-dbc2ef646ab6/edit?viewport_loc=-1058%2C-136%2C4037%2C1988%2CHWEp-vi-RSFO&invitationId=inv_3a486691-75e4-4b83-9ab6-b73c318c027e

Mockups:

<https://www.figma.com/design/H1qa6dEVO7DNxJd36jqh2o/Project1?node-id=0-1&t=zXNWRvht3yommbV9-1>

12 Acknowledgment

I used external tools to build the UML diagrams and Mockups.

I used GenAI of ChatGPT-4 for this project.

Prompt1:

What are the associations between each class?

Response:

Associations:

User and Its Extended Classes:

The User class serves as the base class for all users. Its extended classes include FitnessEnthusiast, Patient, and Doctor. These extended classes inherit all the basic attributes

and associations defined in the User class

Association between User and HealthData:

Each User can have multiple HealthData records.

This is a one-to-many composition relationship, which means that each HealthData instance must be associated with a User. If a User is deleted, all of the corresponding HealthData records will also be deleted.

Association between HealthData and HealthReminder:

Each HealthData can be associated with multiple HealthReminders.

This relationship is a one-to-many aggregation, indicating that while HealthReminders are connected to a HealthData, they can exist and be managed independently within the system.

Association between HealthData and HealthAdvice:

Each HealthData can have multiple HealthAdvice entries.

This one-to-many composition is used to store or generate personalized health advice for the user, based on their recorded health data.

Association between Doctor and Report:

Each Doctor can generate multiple Reports.

This is a one-to-many relationship used to record and present the user's health data and trends over specific time periods.

Association between Doctor and Patient:

As extensions of the User class, Doctor and Patient have a special relationship where a Doctor can be associated with multiple Patients.

This one-to-many relationship facilitates the management and monitoring of multiple patients' health data by a single doctor, allowing doctors to generate, view, and analyze health reports for each patient.

I built the UML diagram to analyze these associations and find that they are correct.

Prompt2:

How about the rankings of user personas dimensions?

Response:

Health Needs / Goals

Fitness Enthusiast (Alex) focuses on performance and improvement.

Patient (Ms. Susan) focuses on monitoring and stability.

Doctor (Dr. Wang) focuses on comprehensive data and patient treatment.

Frequency of System Usage

Fitness Enthusiast: Uses daily to track workouts and view trends.

Patient: Uses regularly (e.g., daily or weekly) to record vitals and receive alerts.

Doctor: Uses intermittently when reviewing multiple patients' data or generating reports.

Technical Proficiency

Fitness Enthusiast: Generally tech-savvy, comfortable with recording data and using charts.

Patient: May vary; might need simpler reminders or instructions.

Doctor: Professionally motivated to use the system, moderate to high proficiency.

Age / Demographics

Fitness Enthusiast (younger adult, actively pursuing fitness goals).

Patient (middle-aged to senior, chronic condition management).

Doctor (professional adult with medical expertise).

Primary Motivation

Fitness Enthusiast: Optimize personal fitness routine and progress.

Patient: Prevent complications and manage chronic disease.

Doctor: Provide effective diagnosis and treatment, handle patient reports efficiently.

I check this answer and related to the user stories and user personas to verify the result of this response.

Prompt(JSDoc part):

How can I generate Jsdoc in my project folder?

Response:

How to Generate JSDoc in Your Project Folder

To generate JSDoc documentation for your project, follow these steps:

1] Install JSDoc (If Not Installed)

If you haven't installed JSDoc globally or in your project, do so using npm:

Install JSDoc globally

```
npm install -g jsdoc
```

This allows you to run jsdoc from any project.

Or Install JSDoc locally in your project

```
npm install --save-dev jsdoc
```

This installs JSDoc as a development dependency for your project.

2] Generate Documentation

Navigate to your project root folder and run the following command:

```
npx jsdoc -c jsdoc.json
```

or, if you installed JSDoc globally:

...

I followed these steps and generated docs successfully.

Prompt(OOP.md part):

Please generate the explanation and counter examples of these OOP applications:

Response:

OOP Patterns Explanation and Hypothetical “Break” Examples

This document provides:

1. **Code examples** of three classes (`DataManager`, `ReportBuilder`, `UserFactory`) that demonstrate good object-oriented design and certain design patterns.
2. **Explanations** of why these are good OOP applications.
3. **Hypothetical scenarios** showing how each concept could be broken if misused.

1. Source Code

Below is the **ES2024 (ES Module)** code for the three classes. They follow the UML class diagrams you provided, and each class demonstrates a particular OOP/design pattern concept:

1. **`DataManager`**: A **Singleton** class for in-memory data management.
2. **`ReportBuilder`**: A **Builder** class for step-by-step construction of `Report` objects.
3. **`UserFactory`**: A **Factory** class for creating different user objects based on a type string.

...

I checked these responses by myself and most of them are correct. I corrected some wrong places and put them in my markdown file.

Prompt(JSDoc Part):

Here is my user.js file, please show me how to add jsdoc to this file.

Response:

A file

// User.js

// Single Responsibility Principle(SRP)

/**

* @module User

* @description Abstract class representing a generic user.

*/

/**

* @abstract

*/

...

I checked and learned how to add jsdoc in js file and add them in the rest of the files.

Prompt(Unit test part):

Please show me how can I create the test file of my project with the jest method.

Response:

```
/ tests/project1.test.js
import { DataManager } from "../solutions/DataManager.js";
import { UserFactory } from "../solutions/UserFactory.js";
import { Doctor } from "../solutions/Doctor.js";
import { Patient } from "../solutions/Patient.js";
import { FitnessEnthusiast } from "../solutions/FitnessEnthusiast.js";
import { HealthData } from "../solutions/HealthData.js";
import { HealthReminder } from "../solutions/HealthReminder.js";
import { HealthAdvice } from "../solutions/HealthAdvice.js";
import { Report } from "../solutions/Report.js";
import { ReportBuilder } from "../solutions/ReportBuilder.js";

describe("Health Data Management System - Unit Tests", () => {
  // If DataManager is a singleton, use DataManager.getInstance().
  // Otherwise, we'll instantiate a new DataManager for each test suite.
  let dm;

  beforeAll(() => {
    // For singleton usage:
    // dm = DataManager.getInstance();
    // For non-singleton usage:
    dm = new DataManager();
  });
  ...
}
```

I checked and learned how to use the jest method to do the unit test of my project.

Prompt(Code running part):

How can I use npm start and npm test to run my cases in the terminal?

Response:

How to Use npm start and npm test to Run Your Project and Tests in the Terminal
To properly configure your project to use npm start for running the project and npm test for running test cases, follow these steps:

1) Ensure You Have a package.json File

First, check if your project has a package.json file. If not, create one by running:

```
npm init -y
```

This will generate a basic package.json file.

2] Set Up the scripts Section in package.json

Edit package.json and add the following "scripts" section:

```
"scripts": {  
  "start": "node ./main.js",  
  "test": "jest --config=jest.config.js"  
}  
"start" → Runs your main.js file using Node.js.  
"test" → Runs your Jest test cases.  
...
```

I followed these steps and run the cases by npm start and npm test successfully.