Holly Rossmann
SWENG 837

August 11, 2024
Final Project

Health Monitoring System

Link to Video Presentation: https://youtu.be/-UxlcA45hE8

1. Problem Statement and Requirements

1.1 Business Requirements

1.1.1 Problem Statement

In recent years, there has been a growing need for personal, continuous health monitoring that helps to improve preventative care, manage chronic diseases, and quickly alert patients and their doctors during health emergencies. Traditional responses from healthcare settings have lacked the ability to track real-time health metrics and alerts, which often lead to delayed responses and low quality patient care.

Thus, I have decided to create a Health Monitoring System, that is a cloud-based platform to collect, store, and analyze health data from wearable devices, providing insights and alerts to both patients and healthcare providers, as listed on the course website.

1.1.2 System Functionalities

- **Real-time Data Collection**
  - The system will collect and store health metric data from wearable devices, such as heart rate, activity levels, and sleep patterns.
- **Data Analysis and Insights**
  - The system will analyze health data to provide insights, trends, and predictions
- **Alerts and Notifications**
  - The system will send real-time alerts to both patients and healthcare providers for abnormal / strange health indicators
- **Patient & Provider Dashboards**
  - The system will provide user-friendly interfaces for patients and healthcare providers to access health data and insights
- **Integration with Healthcare Systems**
  - The system will allow integration between different healthcare applications

1.1.3 Target Users & Needs

- **Patients**
  - Needs continuous health monitoring, insights into their health status, and alerts for abnormal conditions to manage their health
- **Healthcare Provider Facilities**

    - Receives patient data for monitoring and informed decision-making. Acts as the central hub for managing alerts and coordinating care
- **Doctors**
    - Receives alerts from the facility, reviews patient data, and makes medical decisions

1.1.4 Business Goals

- **Improve Patient Outcomes**
    - Enhance patient care by providing insights and interventions, leading to better health outcomes
- **Increase Treatment Efficiency**
    - Streamline healthcare operations by reducing hospital visits and enabling remote monitoring and consultations
- **Enhance Patient Engagement**
    - Empower patients to take an active role in managing their health through personalized insights and recommendations
- **Expand Market Reach**
    - Offer healthcare solutions to a broader audience, including patients with chronic conditions and/or those seeking preventive care

1.2 Non-Functional Requirements

1.2.1 Performance Requirements

- **Scalability**
    - The system should be able to handle an increasing number of users and devices without degrading performance
- **Response Time**
    - Health data should be processed and insights delivered almost instantly to ensure timely interventions
- **Throughput**
    - The system should support a high volume of data transactions and interactions simultaneously

1.2.2 Security Requirements

- **Authentication and Authorization**
    - Implement robust authentication mechanisms to ensure that only authorized users can access the system
- **Data Encryption**
    - Use encryption protocols to protect sensitive health data both in transit and at rest
- **Compliance**
    - Ensure compliance with healthcare regulations and standards (e.g., HIPAA)

1.2.3 Maintainability Requirements

- **Code Modularity**
    - Design the system with a modular architecture to facilitate easy updates and maintenance

- **Documentation**
    - Provide comprehensive documentation for system components, APIs, and user guides
- **Testing Strategies**
    - Implement automated testing and continuous integration to ensure the system's reliability and quality

1.2.4 Other Non-Functional Requirements

- **Usability**
    - Ensure the system is intuitive and easy to use for all user types, with accessible interfaces and straightforward navigation
- **Reliability**
    - The system should be available and operational 99.9% of the time, with minimal downtime

2. Actors Table

Type	Actor	Goal Description
Primary	Patient	Monitor personal health metrics, receive alerts, and access insights, user of a wearable device
Primary	Wearable Device	Hosts the Healthcare Monitoring System for a patient. Needs to read and collect patient health data
Primary	Healthcare Facility	Receives patient data for monitoring and informed decision-making. Acts as the central hub for managing alerts and coordinating care.
Primary	Doctor	Receives alerts from the facility, reviews patient data, and makes medical decisions.
Supporting	System Administrator	Manage system configurations, user access, and ensure security.
Supporting	Data Analyst	Analyze data to identify trends and generate reports for healthcare improvements.
Supporting	Technical Support Staff	Provide assistance and troubleshooting for users and devices.
Offstage	Insurance Company	Utilize health data to assess patient health and customize insurance plans.

Offstage	Government Regulator	Ensures compliance with healthcare regulations, audits system usage, and enforces data protection laws.

Figure 1: System Actors Table

3. Use Cases

Use Case 1

Use Case Section	Comment
Use Case Name	Collect Health Data
Scope	Healthcare Monitoring System
Level	User Goal
Primary Actors	Patient
Supporting Actor	Wearable Device
Stakeholders and Interests	**Patient:** Wants to monitor their health data in real-time and receive actionable insights.
Preconditions	- The patient is equipped with a compatible wearable device that is connected to the system.
Success Guarantee	The patient's health data is continuously monitored and accessible to authorized users.
Main Success Scenario	1. The Patient emits data to the Wearable Device 2. The Device transmits the Patient data to the healthcare monitoring system in real time. 3. The system stores the data
Extensions	2a. If data transmission fails, the system logs the issue and attempts reconnection.
Special Requirements	- Ensure data privacy and secure transmission. - Provide user-friendly interfaces for data access.

Figure 2: 'Collect Health Data' - Use Case

Use Case 2

Use Case Section	Comment
Use Case Name	Send Health Alert
Scope	Healthcare Monitoring System

Level	User Goal
Primary Actor	Wearable Device
Supporting Actor	Healthcare Facility, Data Analyst
Stakeholders and Interests	**Healthcare Facility:** Wants to ensure alerts are assigned to the right professional **Doctor:** Receives alerts to provide timely medical responses and decisions
Preconditions	- The system is set up with alert thresholds for different health metrics. - The Patient is already emitting data to the Wearable Device
Success Guarantee	Alerts are accurately generated and assigned to appropriate healthcare professionals.
Main Success Scenario	1. The Wearable Device transmits Patient data to the System 2. The System analyzes the data, and flags an abnormal health metric 3. The system sends the data to a data analyst to confirm the abnormal health metric 4. When confirmed, the the analyst sends out an alert to the Patient and their Healthcare Facility
Extensions	5a. If no doctor is available, the facility follows escalation protocols to ensure timely response.
Special Requirements	Provide customizable alert thresholds for different patients.

Figure 3: 'Send Health Alert' - Use Case

Use Case 3

Use Case Section	**Comment**
Use Case Name	Address Patient Alert
Scope	Healthcare Monitoring System
Level	User Goal
Primary Actor	Doctor
Supporting Actors	Patient
Stakeholders and Interests	**Doctor:** Wants to access insights to make informed treatment decisions

	Patient: Wants to be notified of changes in patient records or health treatment plans
Preconditions	- The patient's data is stored in the system. - The healthcare facility has already assigned alert to a doctor
Success Guarantee	Doctors can access accurate and actionable patient insights.
Main Success Scenario	1. The Doctor logs into the System. System handles authentication. 2. The doctor browses the "Alerts" section of the system and selects one. 3. The system displays the patient's health metrics and insights related to the alert. 4. The doctor adjusts treatment plans based on alert 5. The doctor documents changes in the patient's care record. 6. The system sends notification to the Patient regarding the change in record
Extensions	3a. If insights are unavailable, the system notifies the doctor and suggests a follow-up analysis.
Special Requirements	- Ensure that insights are presented in a clear and actionable manner.

Figure 4: 'Address Patient Alert' - Use Case

Use Case 4

Use Case Section	**Comment**
Use Case Name	Add Health Goal
Scope	Healthcare Monitoring System
Level	User Goal
Primary Actor	Patient
Supporting Actor	Technical Support Staff *(extension)*
Stakeholders and Interests	**Patient:** Wants to set and track personal health goals for better management of their health. **Technical Support Staff:** Wants to quickly address any issues with data retrieval to give users a pleasant experience
Preconditions	- The patient has access to the healthcare monitoring system and a profile set up with basic health data.
Success Guarantee	Patients can set, track, and update their personal health goals, with

	the system providing feedback on progress.
Main Success Scenario	1. The Patient logs into the system. System handles authentication. 2. The patient selects the “Health Goals” section within the system. 3. The patient selects “Add New Goal” section within the system and adds in the details of the new health goal 4. The system saves the goal and sends a notification back to the Patient
Extensions	3a. If the patient is unsure about setting goals, the system offers recommendations based on health data. 5a. If data from devices is unavailable, the system alerts the Technical Support Staff
Special Requirements	- Ensure the system provides personalized and actionable feedback. - Offer motivational tips and reminders to help patients stay on track.

Figure 5: ‘Add Health Goal’ - Use Case

Use Case 5

Use Case Section	**Comment**
Use Case Name	Update Personal Information
Scope	Healthcare Monitoring System
Level	User Goal
Primary Actor	Patient
Supporting Actors	System Administrator
Stakeholders and Interests	**Patient:** Wants to keep their personal and health information up-to-date for accurate monitoring and care. **System Administrator:** Wants to be aware of any high-level changes to patient information to identify any that are of concern
Preconditions	- The patient has an account in the healthcare monitoring system.
Success Guarantee	The patient's personal and health information is accurately updated and stored in the system.

Main Success Scenario	1. The Patient logs into the system. System handles authentication. 2. The Patient navigates to the “Profile Settings” section of the system 3. The patient updates personal information (e.g., contact details, emergency contacts) and health information (e.g., allergies, medical conditions). 4. The patient chooses “Save” and the system saves the updated information. 5. A confirmation of the successful update is sent to the Patient and to the system administrator
Extensions	4a. If the information cannot be saved, the system alerts the patient.
Special Requirements	- Ensure data privacy and secure handling of personal information. - Provide an intuitive user interface for easy updates.

Figure 6: ‘Update Personal Information’ - Use Case

4. Use System Use Case Diagram

Create a visual representation of the system's actors (users and external systems) and their interactions with the system.

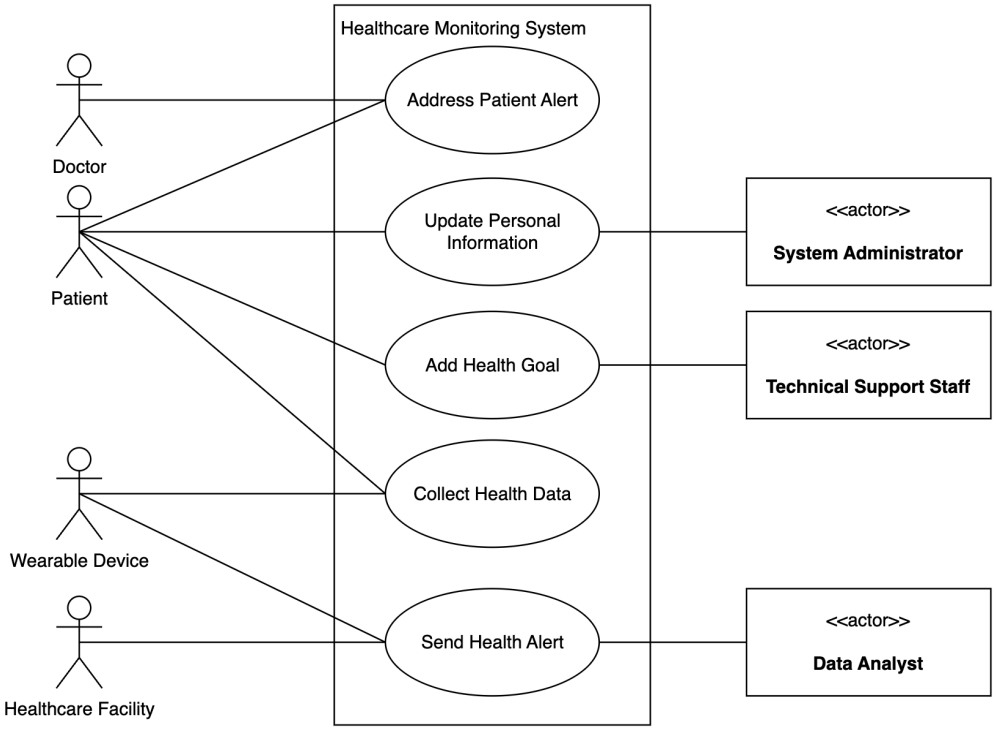

Figure 7: System Use Case Diagram

5. Use Case Sequence Diagrams

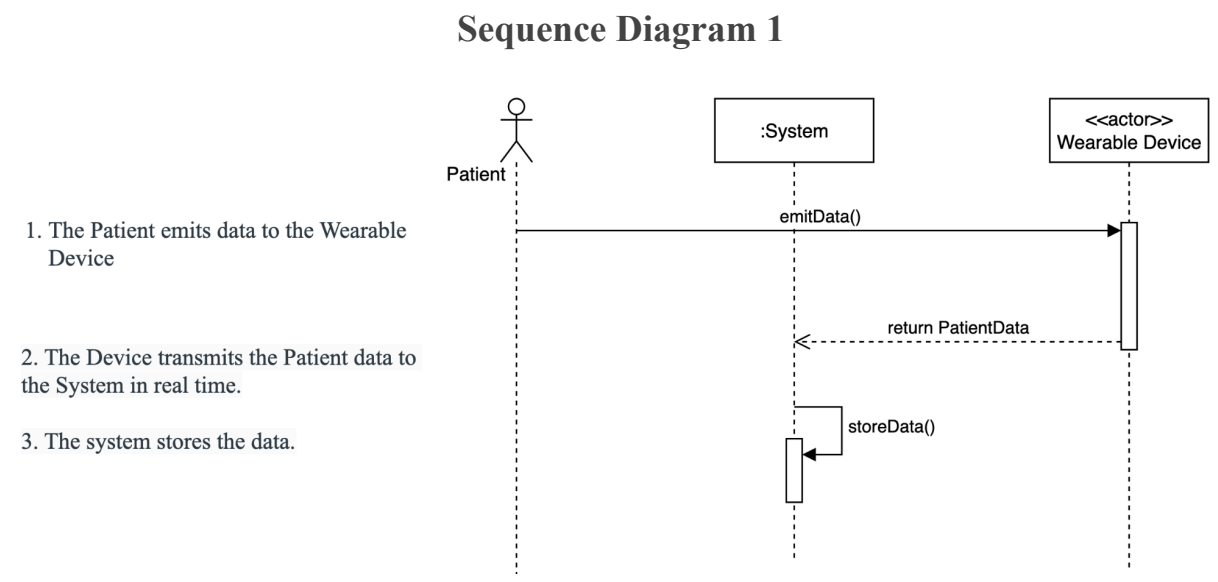

Figure 8: 'Collect Health Data' Sequence Diagram

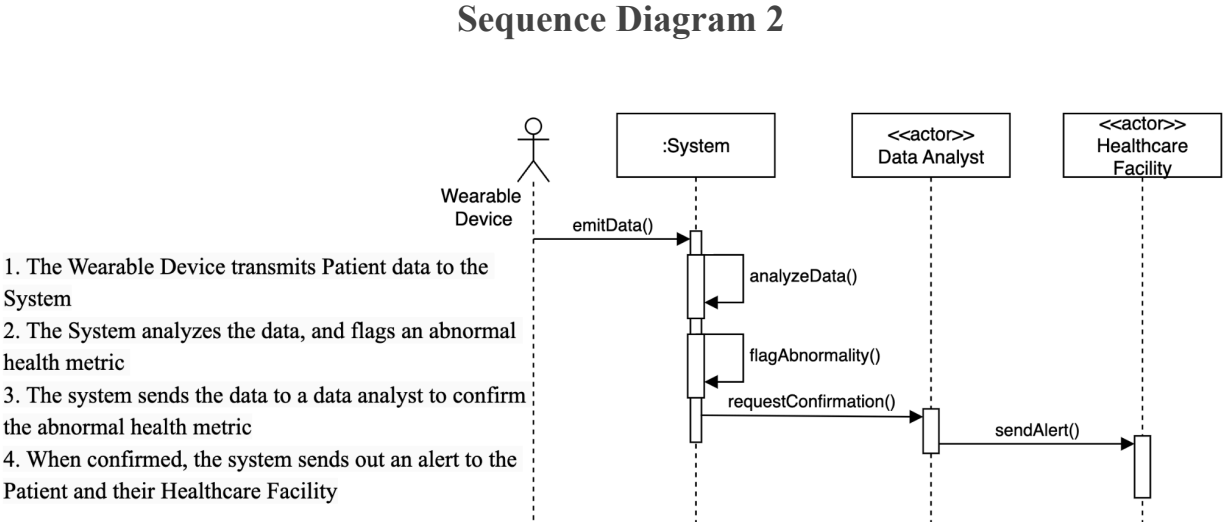

Figure 9: 'Send Health Alert' Sequence Diagram

Sequence Diagram 3

1. The Doctor logs into the System. System handles authentication.

2. The doctor browses the “Alerts” section of the system and selects one.

3. The system displays the patient’s health metrics and insights related to the alert.

4. The doctor adjusts treatment plans based on alert
5. The doctor documents changes in the patient's care record.
6. The system sends notification to the Patient regarding the change in record

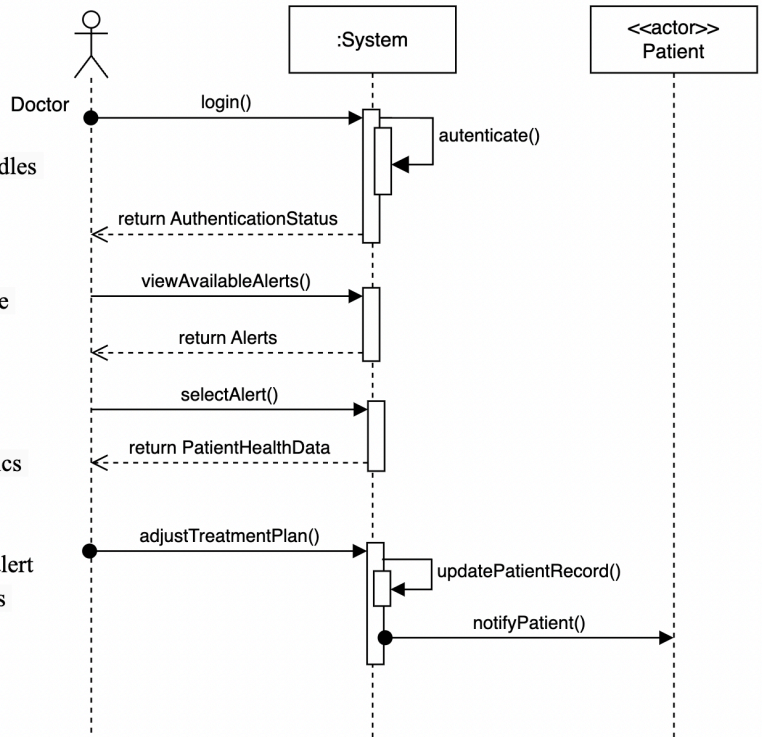

Figure 10: ‘Address Patient Alert’ Sequence Diagram

Sequence Diagram 4

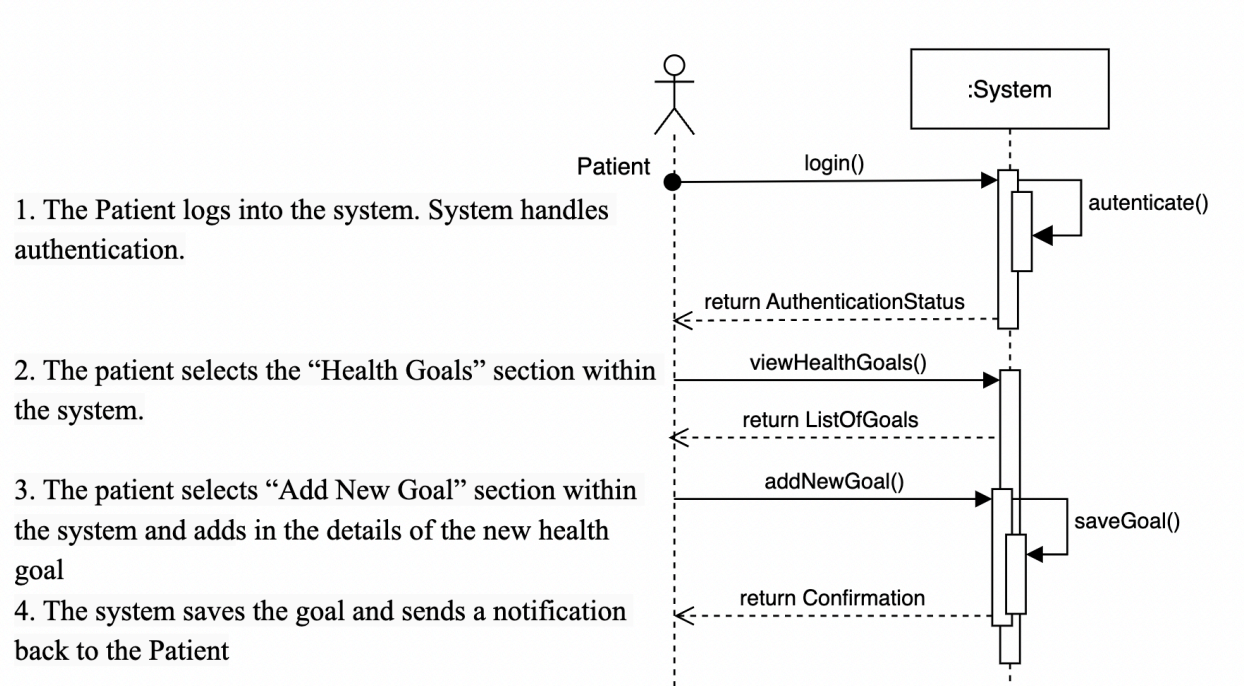

Figure 11: ‘Add Health Goal’ Sequence Diagram

Sequence Diagram 5

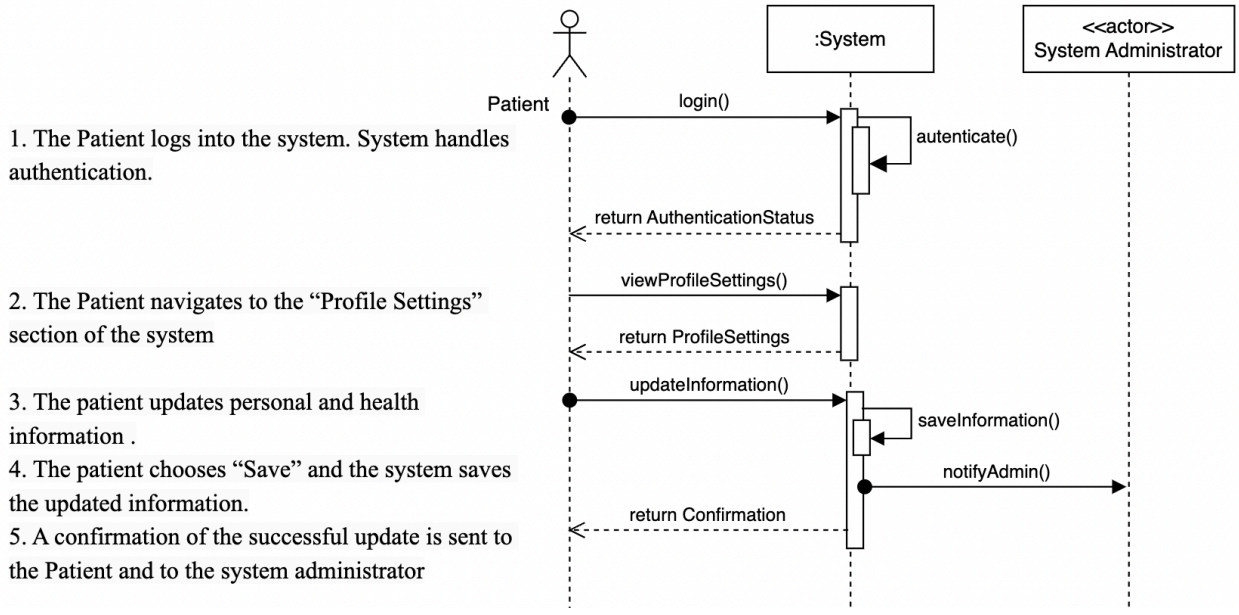

Figure 12: ‘Update Personal Information’ Sequence Diagram

6. Domain Model

With our actors and key use cases defined, I can now move onto creating the Domain Model for the system. Here, I will identify key entities and their relationships within the problem domain, independent of any specific technology.

To begin, I created a table to list out possible conceptual categories & class examples that would go along with my system. Any example in red means that the possible class has been pruned - it is either vague, irrelevant, redundant or an attribute of a class, instead of the class itself.

Conceptual Class / Category	Example for the Health Monitoring System
Physical or Tangible Objects	Wearable Device, Sensor, Smartphone, Medical Equipment, Tablet
Specifications, Designs or Descriptions of Things	Device specifications, Health Metric Thresholds, Alert Configurations, User Profile Settings
Places	Healthcare Facility, Home, Clinic, Emergency Room, Pharmacy
Transactions	Health Data Transmission, Alerts, Patient Registration, Health Data Metrics, Goal Setting, Appointment Schedule
Transaction Line Items	Appointment, Registration Information, Alert Line Item, Appointment Line Item, Goal Line Item
Roles of People	Patient, Doctor, System Administrator, Technical Support Staff, Data Analyst
Containers of Other Things (Physical or Information)	Patient Folders, Wearable App Store, Alert Queue, Health Record Repository
Things in a Container	Patient Files, Apps, User Accounts
Other Computers/Systems (external)	Health Metrics Database, Cloud Server, Insurance System
Abstract Noun Concepts	Health, Wellness, Monitoring, Alert, Emergency
Organizations	Healthcare Facility, Insurance Company, Device Manufacturer, Health Agency
Events	Health Alert Triggered, Data Collected, Goal Achieved, Alert Resolved, System Update
Processes	Data Analysis, Health Monitoring, Alert Management, Goal Tracking, User Authentication

Rules and Policies	Data Privacy Policy, Alert Escalation Policy, Device Usage Policy, Data Retention Policy
Catalogs	Metrics Database, Patient Database, Healthcare Facility Directory, Insurance Companies List
Records of Finance, Work, Contracts, Legal Matters, etc.	Patient Medical Records, Appointment Ledger, Billing Statements, Insurance Claims, Lab Test Results
Financial Instruments and Services	Insurance Policies, Payment Plan, Billing Statements, Cash, Check, Line of Credit
Manuals, Books, Documents, Reference Papers	User Manual, Medical Reference Guide, Policy Document, Training Manual, Appointment Schedule
Measurements and Metrics	Heart Rate, Blood Pressure, Step Ct., Calorie Intake, Sleep Quality
Communication Channels	Email Notifications, SMS Alerts, Mobile App Notification, Web Portal, In-App Messaging

Figure 13: Initial Class Brainstorm for Domain Model

After the pruning was completed, we are now left with the "Good" classes that will be brought into the Domain Model.

Good Classes (Retained)
Alert
Appointment Schedule
Data Analyst
Data Analyzer
Doctor
Goal
Goal Line Item
Health Metrics
Healthcare Facility
Patient
Patient Medical Records
System Administrator
Wearable Device

Figure 14: 'Good' Classes for Domain Model

After this determination, I was able to go ahead and implement the Domain Model, which is shown below.

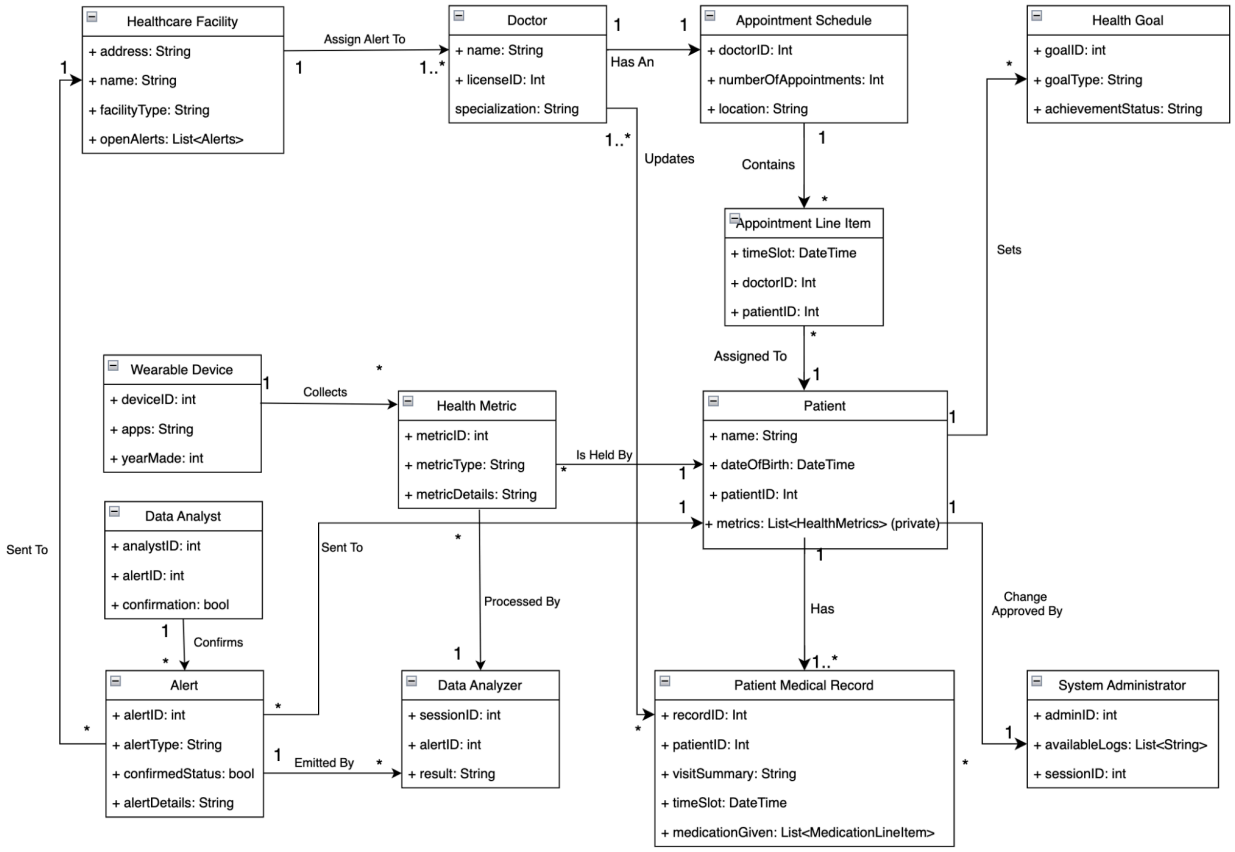

Figure 15: System Domain Model

7. System Design Class Diagram

Next, I created the System Design Class Diagram, which translates the domain model into a set of classes, their attributes, and relationships, reflecting the system's functionality. Note that this is for the system as a whole, not a specific use case. It encompasses all functions and software data needed for each use case.

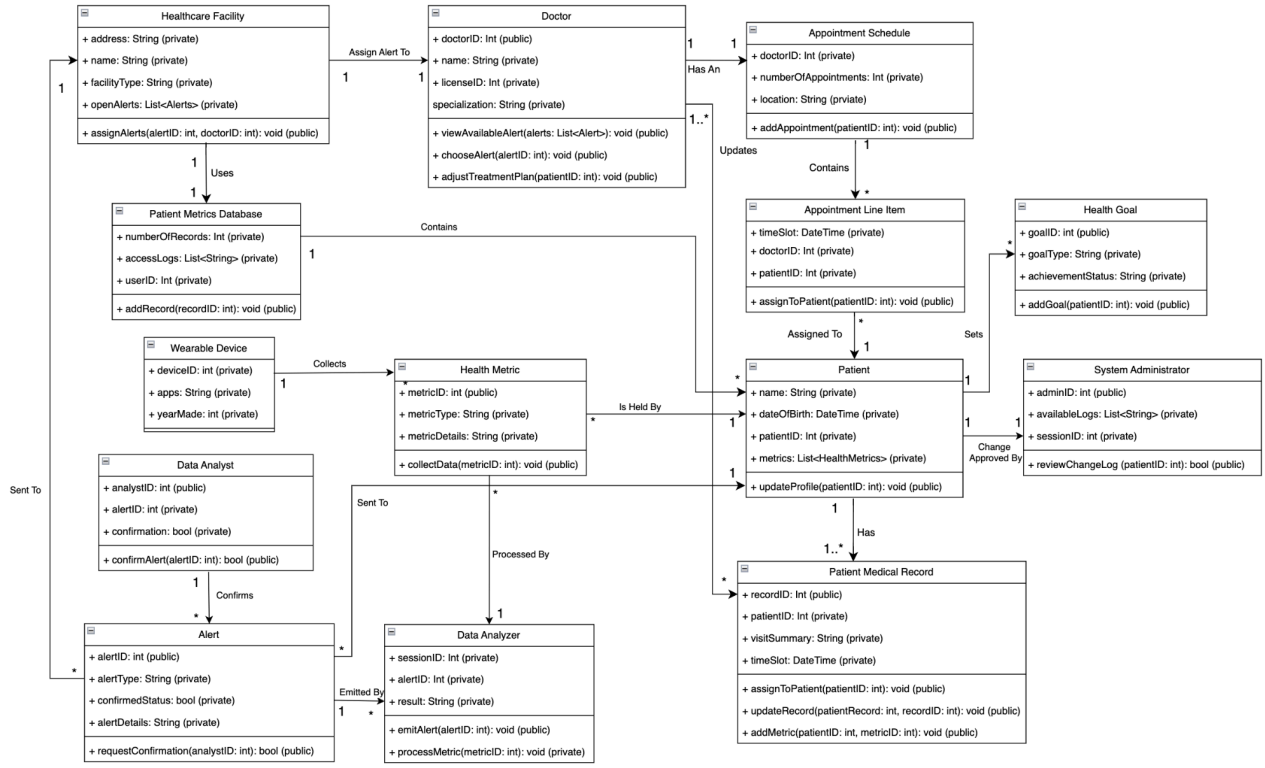

Figure 16: System Design Class Diagram

8. State Diagram

The State Diagram I chose to design was for the “Alert” object. Here I illustrate the possible states and transitions that an Alert can undergo throughout its lifecycle within the system.

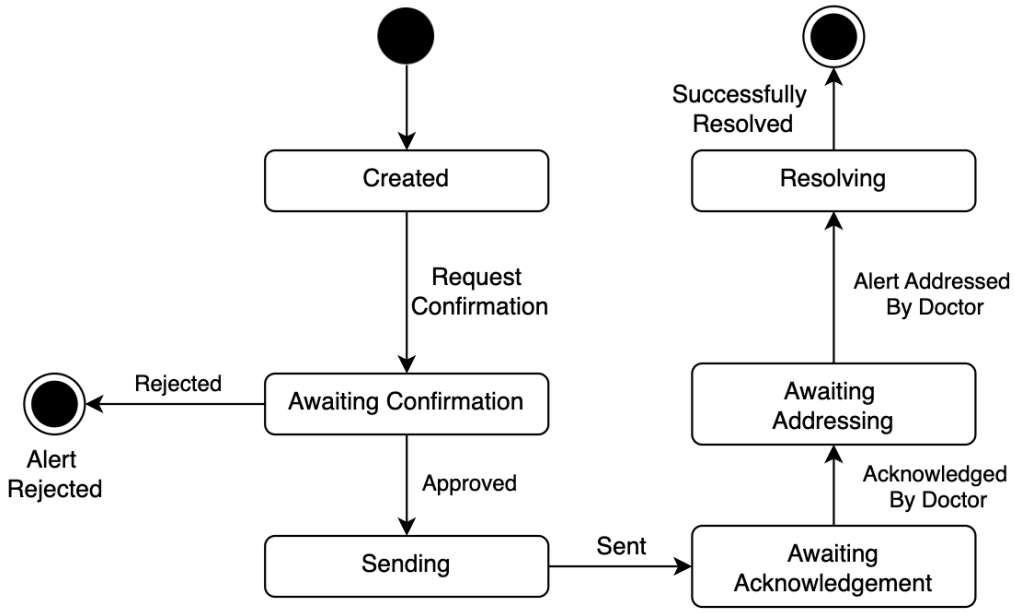

Figure 17: ‘Alert’ State Diagram

9. Activity Diagram

Like the State Diagram in the last section, I chose to focus on the process that an Alert undergoes, and this Activity Diagram visually represents the activities and flows within a specific process, highlighting the responsibility of different actors using swimlanes. In particular, it focuses on Use Cases 2 and 3, as they relate to alerts.

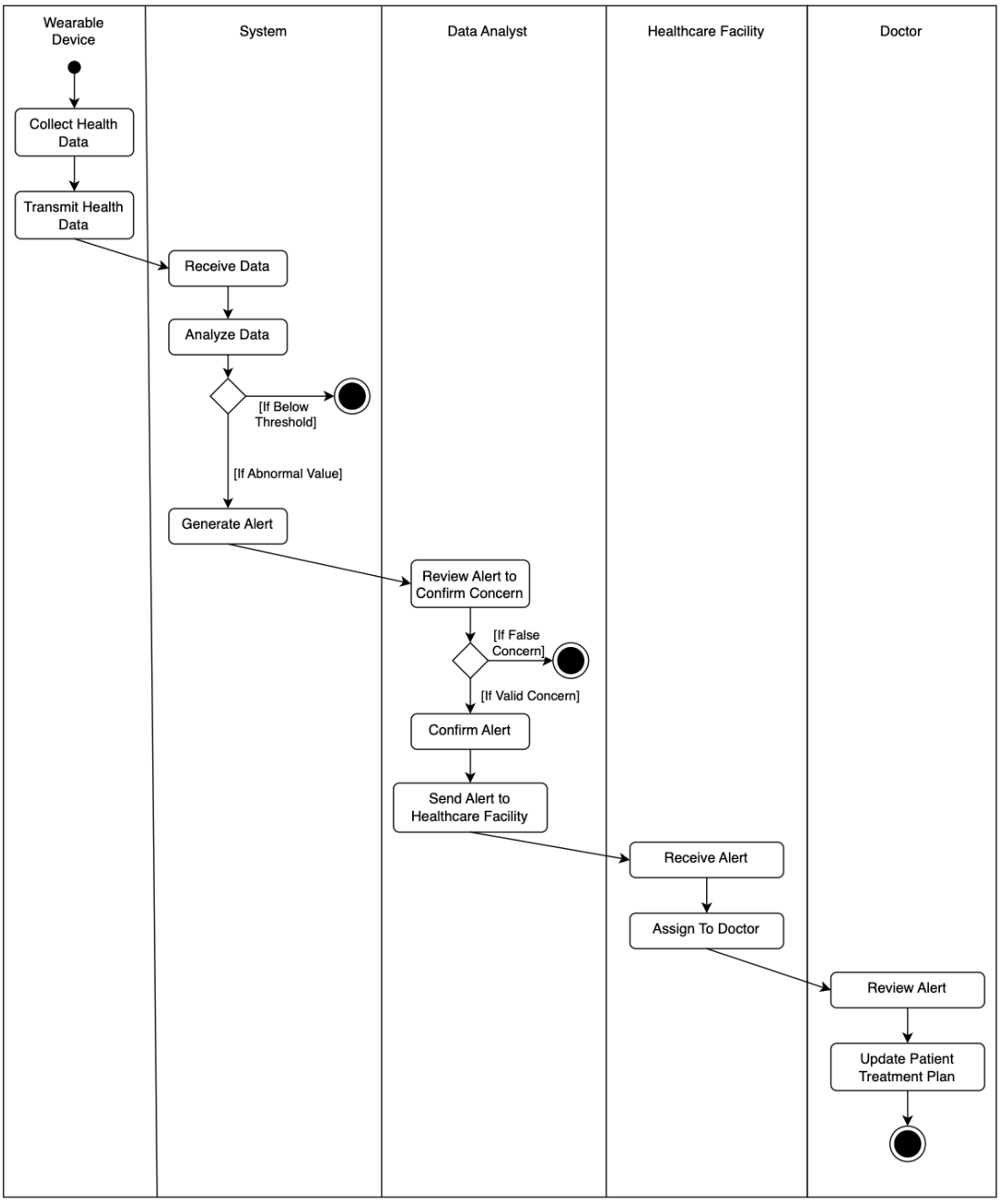

Figure 18: 'Alert' Activity Diagram

10. Component Diagram

Below is the system's Component Diagram, which depicts the system's physical components and their dependencies, providing a high-level architectural view. There are 5 main components and subsystems here - Wearable Device, Alert Management, Healthcare System, Goal Management, and System Administrator.

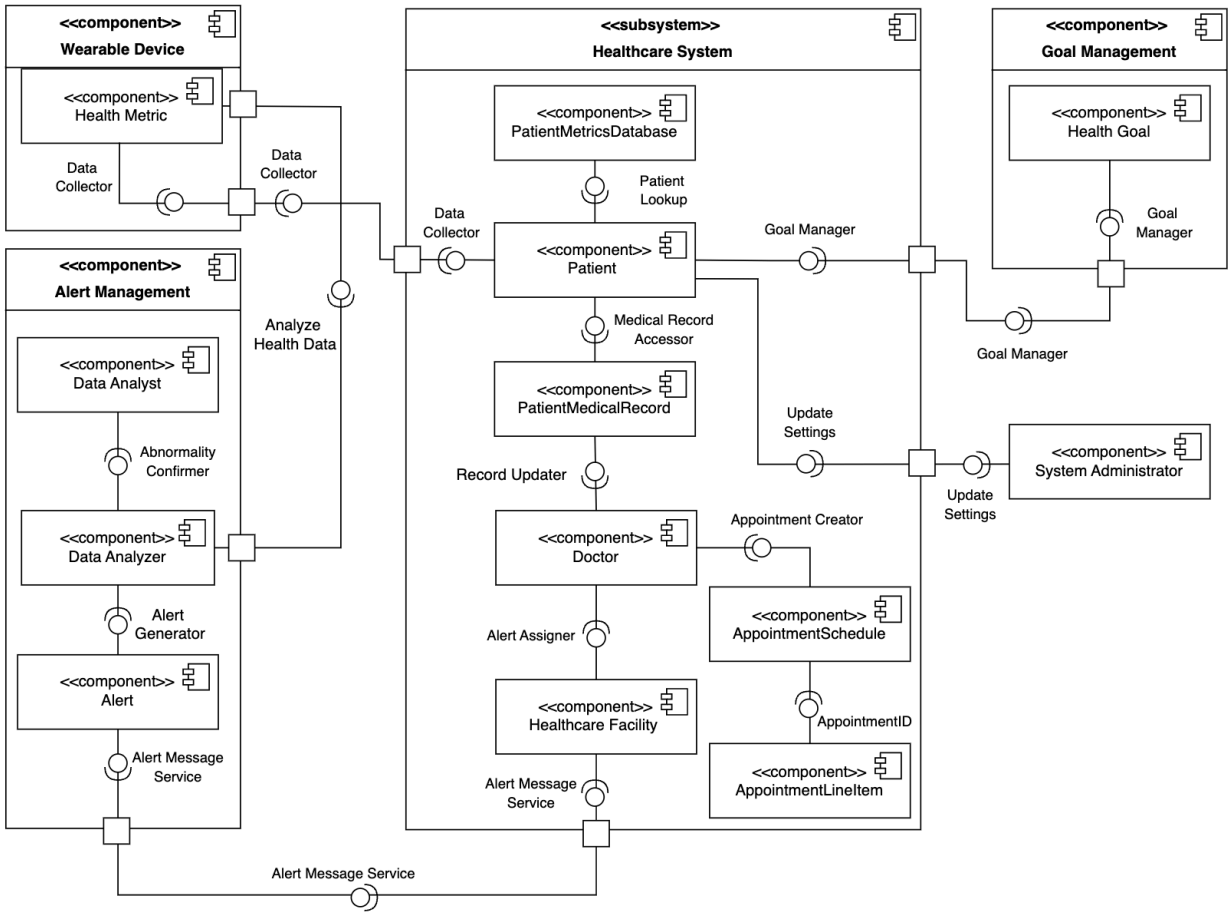

Figure 19: System Component Diagram

11. Cloud Architecture Deployment Diagram

Below is the System Cloud Architecture Deployment Diagram. Here, I have decided to go with a Client-Server architecture, as it has centralized data management and processing, allowing healthcare facilities and doctors to access patient data and alerts from various locations. This architecture supports scalability, since additional client devices (like wearables and user interfaces) can be added without significantly affecting the server-side processes. Also, it helps with security and data integrity by promoting controlled access and robust data encryption on the server.

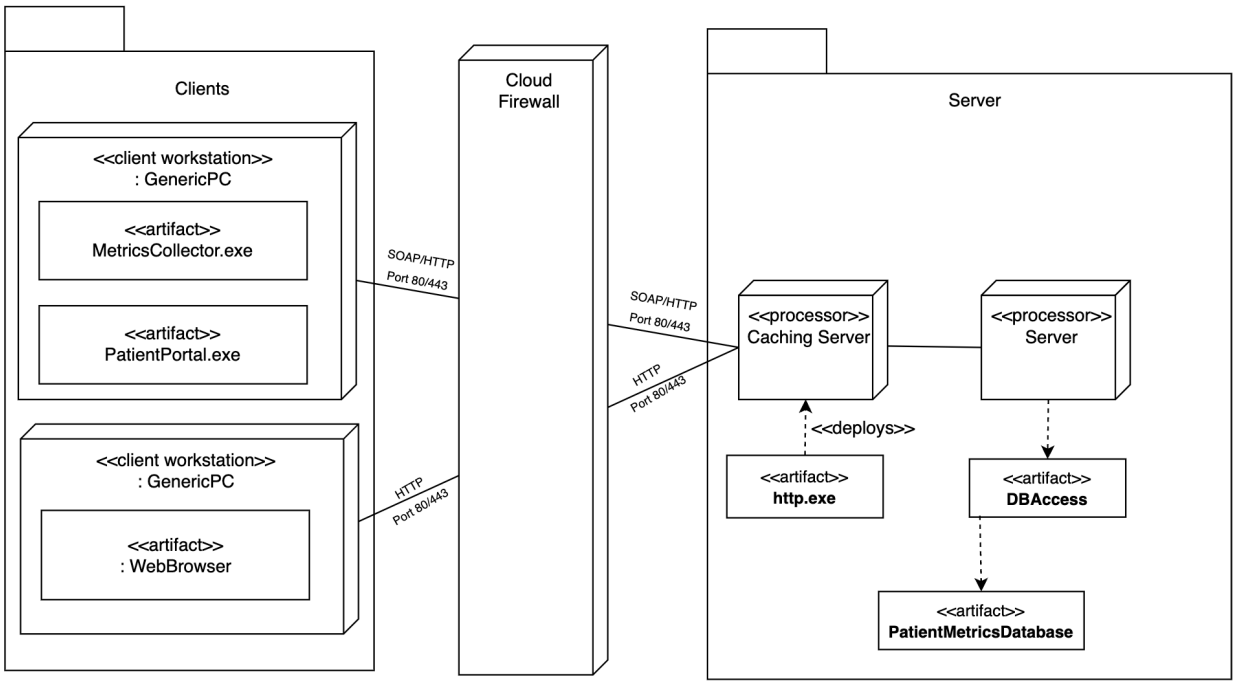

Figure 20: Cloud Architecture Deployment Diagram

12. Skeleton Classes & Table Definition

12.1 Skeleton Classes

Alert Class

```
class Alert {
private:
    int alertId;
    string alertType;
    string alertMessage;
    bool confirmedStatus;

public:
    // Constructor
    Alert(int id, string type, string message, bool status);

    // Getters and Setters
    int getAlertId();
    void setAlertId(int id);
    string getAlertType();
    void setAlertType(string type);
    string getAlertMessage();
    void setAlertMessage(string message);
    bool getConfirmedStatus();
    void setConfirmedStatus(bool acknowledged);

    // Methods
    void requestConfirmation();
};
```

Appointment Schedule Class

```
class AppointmentSchedule {
private:
    int appointmentId;
    int patientId;
    int doctorId;
    int numberOfAppointments;
    String location;

public:
    // Constructor
    AppointmentSchedule(int id, int patientId, int doctorId, int numAppt,
string location);
```

```
    // Getters and Setters
    int getAppointmentId();
    void setAppointmentId(int id);
    int getPatientId();
    void setPatientId(int id);
    int getDoctorId();
    void setDoctorId(int id);
    int getNumberOfAppointments();
    void setNumberOfAppointments(int num);
    string getLocation();
    void setLocation(string location);

    // Methods
    void addAppointment(patientID);
};
```

Patient Class

```
class Patient {
private:
    int patientId;
    datetime dateofBirth;
    string name;
    list<healthmetrics> metrics;
public:
    // Constructor
    Patient(int id, datetime dob, string name, list<healthmetrics> metrics);

    // Getters and Setters
    int getPatientId();
    void setPatientId(int id);
    string getName();
    void setName(string name);
    string getDOB();
    void setDOB(datetime DOB);
    list<healthmetrics> getMetrics();
    void setMetrics(list<healthmetrics> metrics);

    // Methods
    void updateProfile(patientID);
};
```

Doctor Class

```
class Doctor {
private:
    int doctorId;
    string name;
    int licenseID;
    string specialization;
public:
    // Constructor
    Doctor(int id, string name, string specialization, int licenseID);

    // Getters and Setters
    int getDoctorId();
    void setDoctorId(int id);
    string getName();
    void setName(string name);
    int getLicenceID();
    void setLicenceID(int id);
    string getSpecialization();
    void setSpecialization(string spec);

    // Methods
    void viewAvailableAlert(alerts);
    void chooseAlert(alertID);
    void adjustTreatmentPlan(patientID);
};
```

Wearable Device Class

```
class WearableDevice {
private:
    int deviceId;
    string apps;
    int yearMade;
public:
    // Constructor
    WearableDevice(int id, string apps, int yearMade);

    // Getters and Setters
    int getDeviceId();
    void setDeviceId(int id);
    string getApps();
    void setApps(string apps);
    int getYearMade();
    void setYearMade(int year);

    // Methods
    void collectData(metricID); };
```

12.2 Table Definitions

Alerts Table

```
CREATE TABLE Alerts (
    alertId INT PRIMARY KEY,
    patientId INT,
    alertType TEXT,
    alertMessage TEXT,
    confirmationStatus BOOLEAN,
    FOREIGN KEY (patientId) REFERENCES Patients(patientId)
);
```

Appointments Table

```
CREATE TABLE AppointmentSchedules (
    appointmentId INT PRIMARY KEY,
    patientId INT,
    doctorId INT,
    numberOfAppointments INT,
    location TEXT,
    FOREIGN KEY (patientId) REFERENCES Patients(patientId),
    FOREIGN KEY (doctorId) REFERENCES Doctors(doctorId)
);
```

Patients Table

```
CREATE TABLE Patients (
    patientId INT PRIMARY KEY,
    dateOfBirth TEXT,
    name TEXT,
    metricId INT,
    FOREIGN KEY (metricId) REFERENCES Metrics(metricId)
);
```

Doctors Table

```
CREATE TABLE Doctors (
    doctorId INT PRIMARY KEY,
    name TEXT,
    specialization TEXT,
    licenseId INT
);
```

Wearable Devices Table

```
CREATE TABLE WearableDevices (
    deviceId INT PRIMARY KEY,
    apps TEXT,
    yearMade INT
);
```

13. Design Patterns

There are multiple design patterns and principles that I took into consideration throughout the process of the system architecture design.

To begin with, the SOLID Principle of "Single Responsibility" was used, as I wanted to ensure that each component of the system had a distinct responsibility. This meant that the Alert Management Component should only handle the alert-related tasks, and the Goal Management Component should only handle the goal-related tasks. This separation makes the system easier to maintain and extend as needed, since changes to one component will have low impact on others.

Also, I utilized the GRASP Principle of "Creator," as I wanted to make sure that the system components that needed to instantiate and/or manage certain objects would be responsible for creating them. An example here would be the Alert Management Component creating and managing alert instances. This ensures that the object creation aligns with the component's responsibilities, and I was hoping that by doing this, I could help simplify the system's design.

The last pattern I will touch on falls under the GOF / Cloud Native category. I wanted to bring in Event Driven Architecture. This means that the Alerts and notifications were handled in a way where events trigger corresponding actions or processes. By having real-time processing and responses from certain triggers, I was hoping to improve the system's responsiveness and scalability.