Software Design Specifications Document

Baymax – Remote Patient Monitoring System

By

Bisma Ijaz - 351995

Vimal - 343007

Syed Fazail Haider - 346062

Supervisor:
Dr. Muhammad Daud Abdullah Asif

Co-Supervisor:
Dr. Wajid Mumtaz

Bachelor of Science in Computer Science (2020-2024)

Department of Computing
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences & Technology

Table of Contents

List of Sequence Diagrams

List of State Transition Diagrams

List of Activity Diagrams

List of Data Flow Diagrams

Revision History

Name	Date	Reason for changes	Version

Application Evaluation History

Comments (by committee) ***include the ones given at scope time both in doc and presentation**	**Action Taken**

Supervised by

Dr. Muhammad Daud Abdullah Asif

Signature_____

1. Introduction

1.1 Purpose

The product named ‘Baymax’ is a remote patient monitoring system (RPMS) composed of two integral parts: the ‘Baymax Kit’ which is a medical monitoring kit containing sensors and microprocessors, and the ‘Baymax Portal’ which is an integrated Web-based platform developed using the MERN stack. MERN stands for MongoDB, Express.js, React, and Node.js, whose details are as follows:

1. ***M****ongoDB*: a NoSQL database layer in the MERN stack. It is highly scalable and ideal for handling large amounts of data and provides high performance.
2. ***E****xpress.js*: a flexible and lightweight Node.js web application framework that provides a set of features to allow simple development of scalable server-side web and mobile applications.
3. ***R****eact*: *a* frontend JavaScript library for building modern, dynamic, and interactive user interfaces.
4. ***N****ode.js*: a JavaScript runtime environment which allows developers to run JavaScript on the server side, enabling the development of scalable and high-performance web applications.

This Software Design Specifications (SDS) document describes the ‘Baymax Portal’ that aims to provide services for the doctors and patients alike for the purpose of remote patient monitoring. The product aims to facilitate timely medical intervention by:

1. connecting patients with doctors and hospitals,
2. transmitting patient body vitals from the Baymax kit,
3. and using machine learning based logic for predictions to aid diagnosis of clinical events.

1.2 Document Overview

In this document, design aspects of 'Baymax' are outlined with a focus on its Design Methodology and Software Process Model. The System Overview provides a comprehensive understanding of the project's structure and functionalities. Architectural Design describes the high-level design principles which guide the system's development. Next, the Design Models help explain the conceptual frameworks that are used. The Data Design section provides details of the types of data that are stored in the system, the relationships between them and the ways of data organization.

Lastly, Human Interface Design describes the user interaction aspects of the 'Baymax Portal.' This document serves as a vital resource for understanding the structural and design components of the 'Baymax Portal.'

1.3 Product Scope

'Baymax Portal' represents an innovative healthcare system that aims to reshape the landscape of remote patient monitoring in Pakistan. It aims to address these issues by providing an integrated web-based platform that offers a range of services to patients, doctors, and administrators. It is specifically designed to serve a diverse range of individuals including but not limited to:

- those who require remote medical care,
- the elderly,
- those in need of regular checkups,
- individuals managing chronic conditions,
- expectant mothers,
- residents of remote areas,
- and individuals with mobility challenges like paralysis.

In pursuit of its mission, Baymax has set clear objectives and goals, which include:

1. Enhancing remote patient monitoring through utilization of the 'Baymax Kit' to provide patients with a reliable and convenient method for continuous health tracking.

2. Establishing efficient communication channels between patients and healthcare providers, reducing response times, and improving overall patient care.

3. Leveraging the power of machine learning to improve the accuracy of medical condition predictions by analysing correlations among various bio signals to enhance the precision of diagnoses and empower healthcare providers to offer more effective care to patients.

4. Securely storing and analysing patient data in the cloud.

5. Streamlining administrative processes and providing an efficient healthcare management platform connecting doctors, e, and patients.

1.4 Implementation Progress

Here, light will be shed upon the progress of our implementation of project modules:

1. Login webpage for patients, doctors, and hospital administrators has been developed
2. Separate registration webpages for patients, doctors, and hospital administrators developed
3. Webpages to add, delete and update profiles of patients and doctors developed
4. Dashboard webpages for patients, doctors, and hospital administrators developed
5. Appointment setting webpages for patients and appointment management webpages for doctors and hospital administrators developed
6. Webpages for sensor management and sensor data display developed
7. Webpages for chat between patients and doctors developed
8. Home, About Us, Contact Us and Services webpages developed
9. Webpages for viewing patients, doctors and hospitals developed
10. Webpages for display of notifications developed
11. Database schema has been made
12. Controllers have been implemented

2. Design Methodology and Software Process Model

For our Baymax project, the design methodology chosen is the Object-Oriented Programming (OOP) design methodology. There are multiple reasons for this choice:

1. OOP is ideal for management of complex systems due to higher level of maintainability and lower redundancy due to objects and classes
2. OOP allows for the encapsulation of code into objects, which allows reusability and modularity along with better project organization.

3. By representing real world entities as objects, OOP allows abstraction of complex systems which makes the system easier to manage and understand.
4. OOP hides the internal details of objects and shows only the details that are necessary. This concept of encapsulation allows more robustness and fewer dependencies.
5. New classes and objects can be made which means that modules can be extended and changes can be made in later future depending upon the requirements of the stakeholders.

Moreover, the process model being followed is the Iterative and Incremental Model. This model is chosen for several reasons which are as follows:

1. There is room for flexibility and adaptability, particularly in the context of healthcare applications where requirements are subject to change from stakeholders depending upon the need
2. The iterative approach allows for timely and frequent evaluation and feedback, ensuring that the evolving needs of both medical practitioners and patients are addressed effectively.
3. The incremental development enables the delivery of functional components in stages, allowing stakeholders including patients, doctors, hospitals, other healthcare providers (individuals and organizations), government, industry experts, to evaluate tangible progress and provide feedback throughout the development lifecycle.

The choice of an Iterative and Incremental Model aligns with the dynamic nature of healthcare systems, allowing the Baymax Project to accommodate changing requirements, integrate feedback, and ensure that the final product meets the highest standards of functionality, usability, and reliability in a constantly evolving healthcare landscape.

3. System Overview

'Baymax Portal' is a self-contained product which operates independently with its ecosystem but interacts with external components to provide a comprehensive healthcare solution. It interfaces with health monitoring device kit 'Baymax Kit,' utilizes cloud services for data management, displays patients' bio-signals' data to doctors, offers real-time chat features, and integrates external machine learning algorithms for clinical events predictions.

3.1 Functionality

Here is a high-level summary of the major functions that the Baymax Portal must perform or allow users to perform:

- Register new users as patients, doctors, or hospital administrators
- Create and manage patient and doctors profiles with personal and medical information
- Provide customized dashboards for patients, doctors, and hospital administrators with role-specific functionalities
- Enable real-time chat between patients and doctors
- Facilitate appointment booking and management
- Transmit vital signs and health data from monitoring hardware from patient side to doctor side
- Utilize machine learning algorithms to predict clinical events
- Present predictive analysis to doctors for diagnosis and alert doctors/hospitals as per requirement
- Securely store and manage patient data in a cloud-based system
- Ensure data privacy and compliance with regulations

3.2 Architectural Design

For our Baymax project, a combination of Microservices Architecture and Cloud-Native Architecture would be best. There are several advantages that favor this decision:

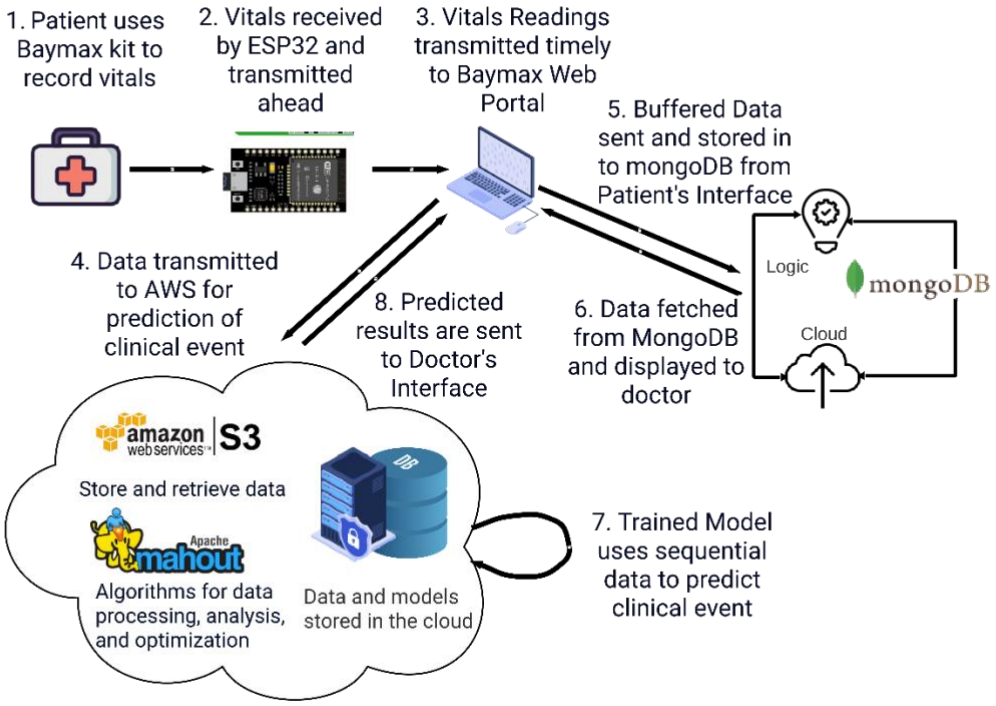
1. **Microservices Architecture**

- ***Modularity:*** Baymax can be divided into independent microservices, such as user registration, data transmission, machine learning predictions, and more. Each microservice can be developed, deployed, and scaled independently.
- ***Scalability:*** Microservices allow for scalable deployment, which is crucial for handling varying workloads, especially in a healthcare system where the demand for remote patient monitoring services can vary greatly.
- ***Maintainability:*** Each microservice can be updated and maintained without affecting the entire system, making it easier to manage and evolve over time.
- ***Fault Tolerance:*** Each microservice operates independently, which means that if one microservice fails, it does not necessarily impact the entire system. This isolation enhances fault tolerance and allows for more straightforward resolution of faults.

2. Cloud-Native Architecture:

- ***Scalability:*** Leveraging cloud services, in this case Amazon Web Services (AWS), enables automatic scalability based on demand. This is essential for handling potential spikes in usage and ensuring a responsive system.
- ***Flexibility:*** Cloud-native architecture provides flexibility in terms of resource allocation, allowing Baymax to adapt to changing requirements efficiently.
- ***Reliability and Availability:*** Cloud platforms typically offer high reliability and availability. This is crucial for a healthcare system where uninterrupted access to patient data and services is mandatory.
- ***Cost Effectiveness:*** Cloud platforms offer a pay-as-you-go model, where resources are allocated based on actual usage. This can lead to cost savings for our project.

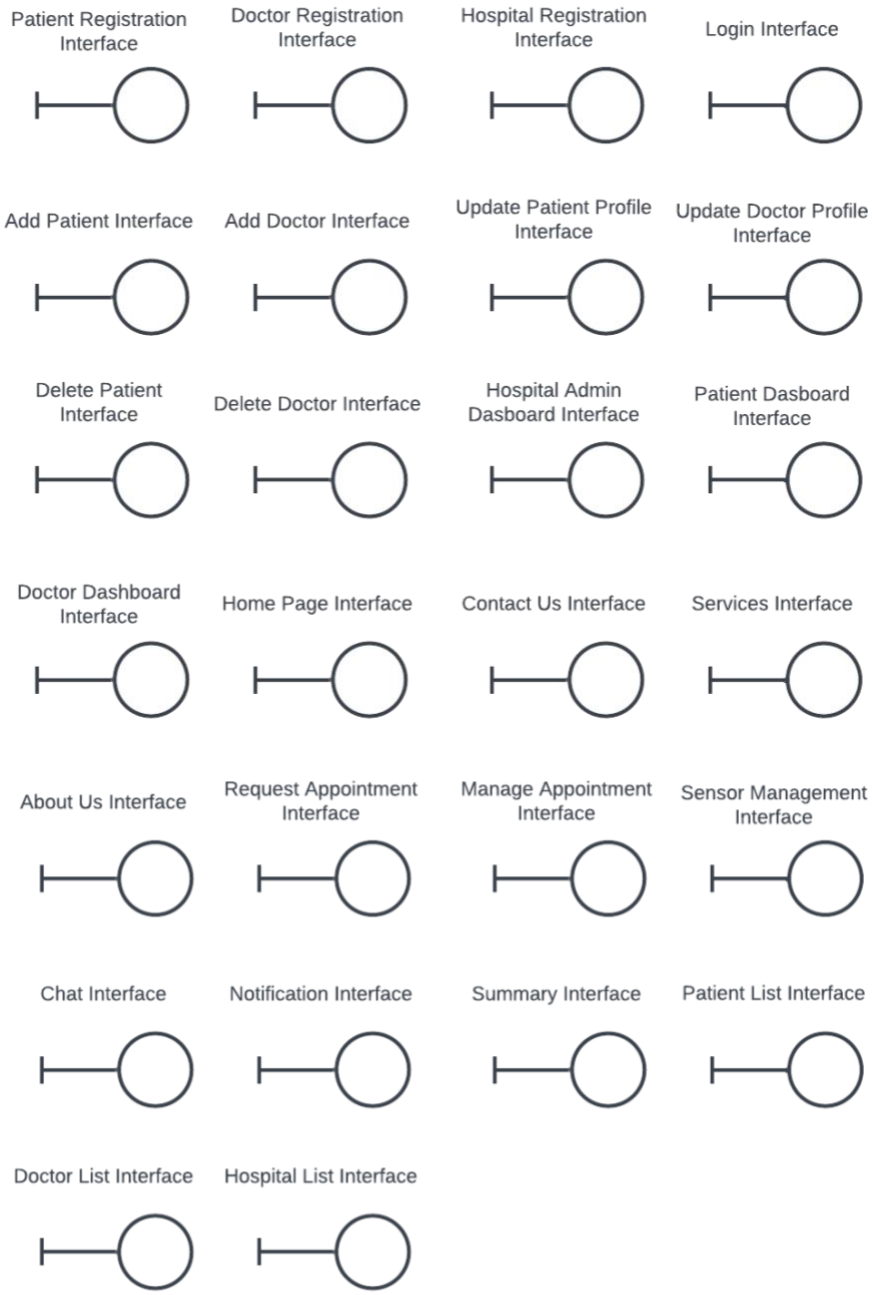
Given below is our system architecture design:


1. Patient uses Baymax kit to record vitals
2. Vitals received by ESP32 and transmitted ahead
3. Vitals Readings transmitted timely to Baymax Web Portal
5. Buffered Data sent and stored in to mongoDB from Patient's Interface
Logic
mongoDB
Cloud
6. Data fetched from MongoDB and displayed to doctor
8. Predicted results are sent to Doctor's Interface
4. Data transmitted to AWS for prediction of clinical event
amazon web services S3
Store and retrieve data
Apache mahout
Algorithms for data processing, analysis, and optimization
DB
Data and models stored in the cloud
7. Trained Model uses sequential data to predict clinical event

4. Analysis Model

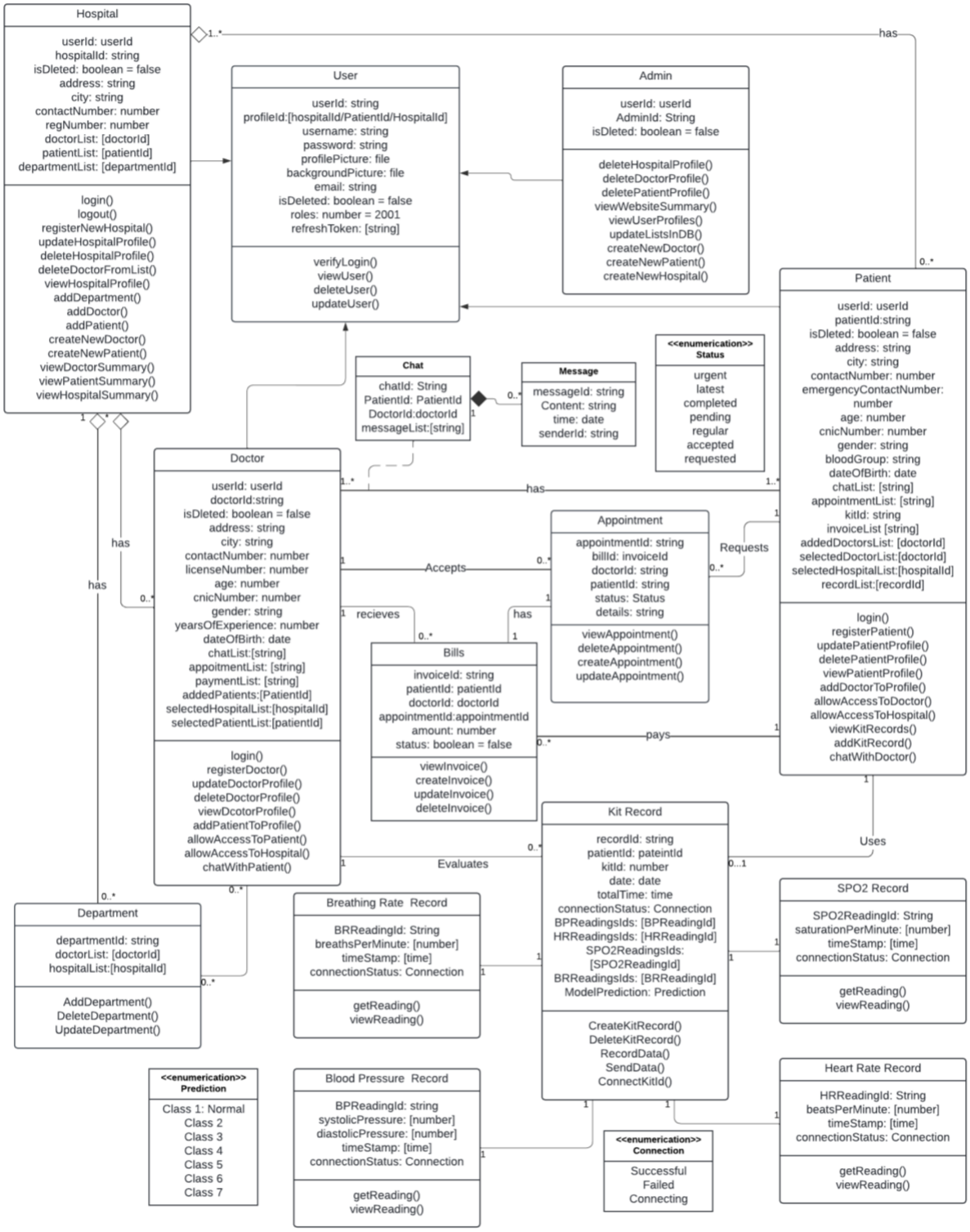
4.1 Interface Objects

The interface objects represent the main interfaces of the system. Given below is the list of different interface objects used in our system. Many more will be added later on according to requirements.



5. Design Models

5.1 Class Diagram



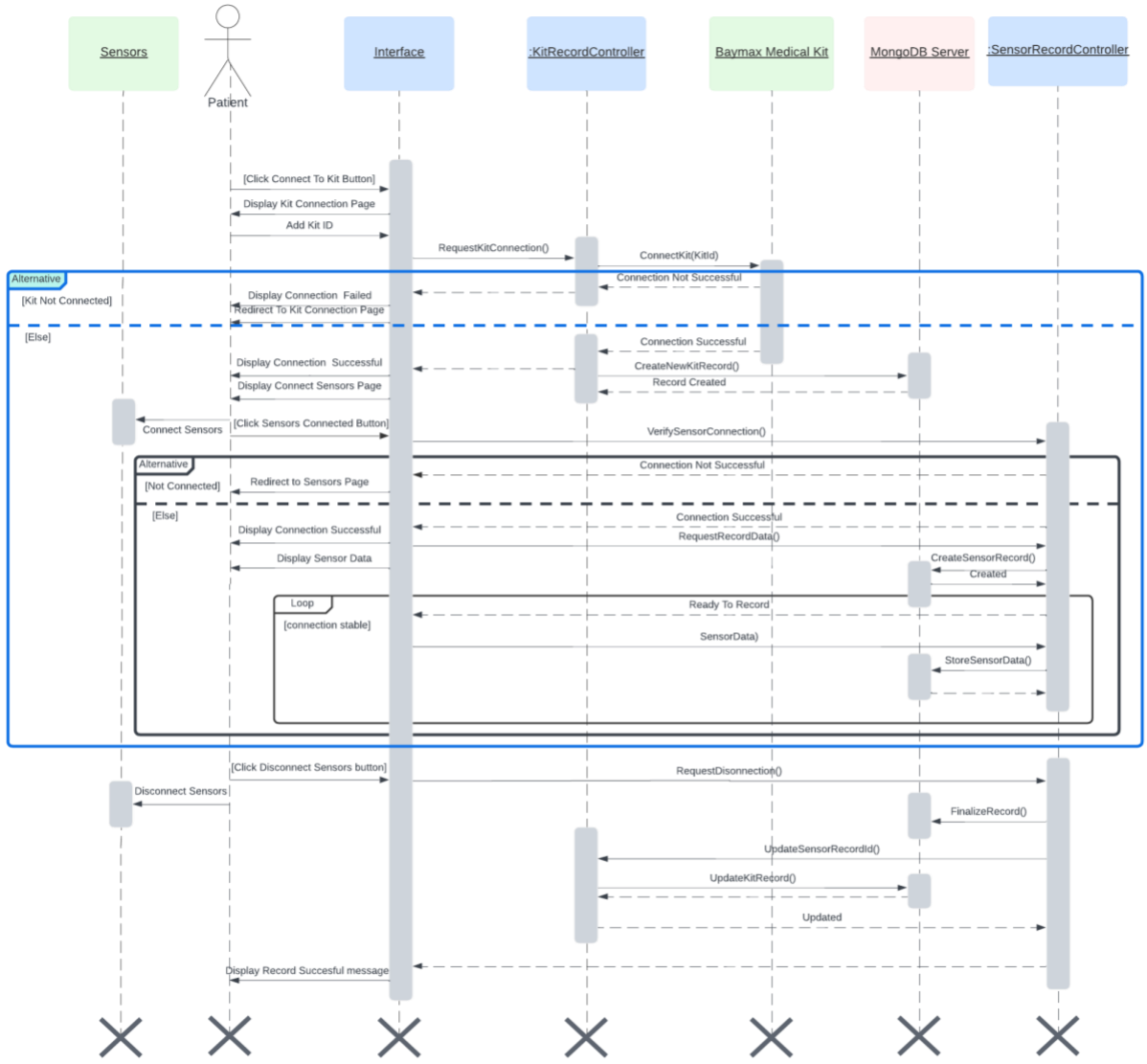
The class diagram elaborates how all classes are linked to each and the functions that each class has. Due to space constraint all functions cannot be mentioned here, only the main ones are highlighted.

5.2 Sequence Diagrams

The sequence diagrams elaborate the sequence of actions that take place between different frontend, backend, physical components of the system and the user.

1. Kit Connection and Data Acquisition

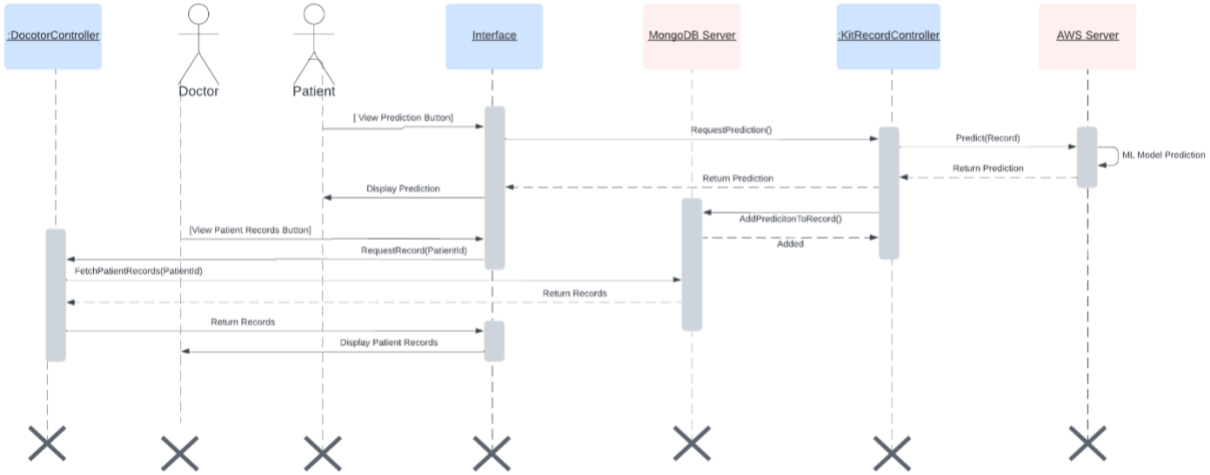
It describes the procedure of connecting the Baymax kit and transferring data to backend.



Sequence Diagram 1 Kit Connection and Data Acquisition

2. Model Prediction and Doctor Evaluation

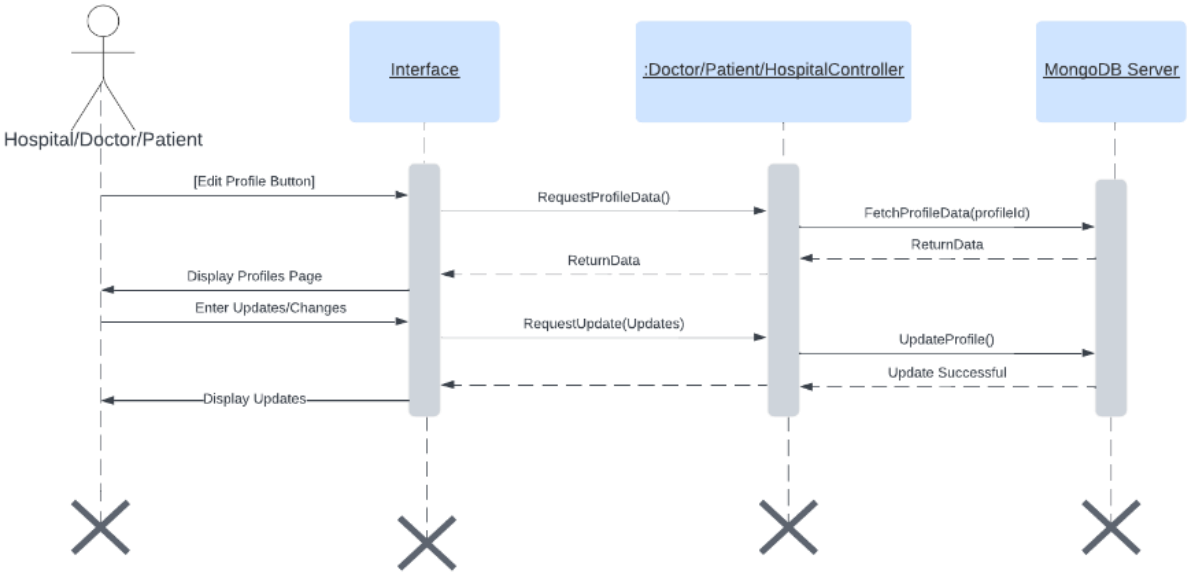
It shows how the model resided in AWS tests the data and predicts the condition.



Sequence Diagram 2 Model Prediction and Doctor Evaluation

3. Update/Edit Profile

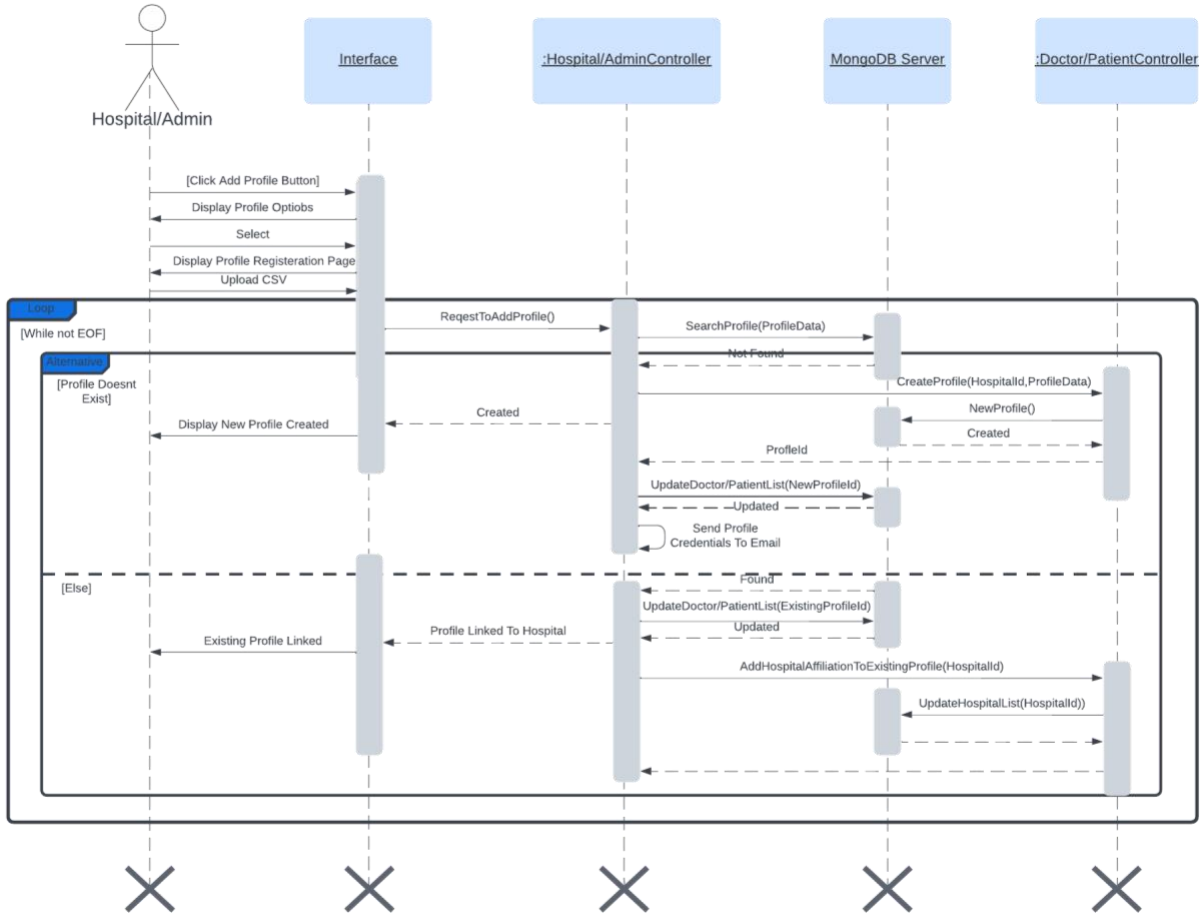
It shows how the profile is Updated/Edited.



Sequence Diagram 3 Update/Edit Profile

4. Add Doctor/Patient Profile by Hospital/Admin

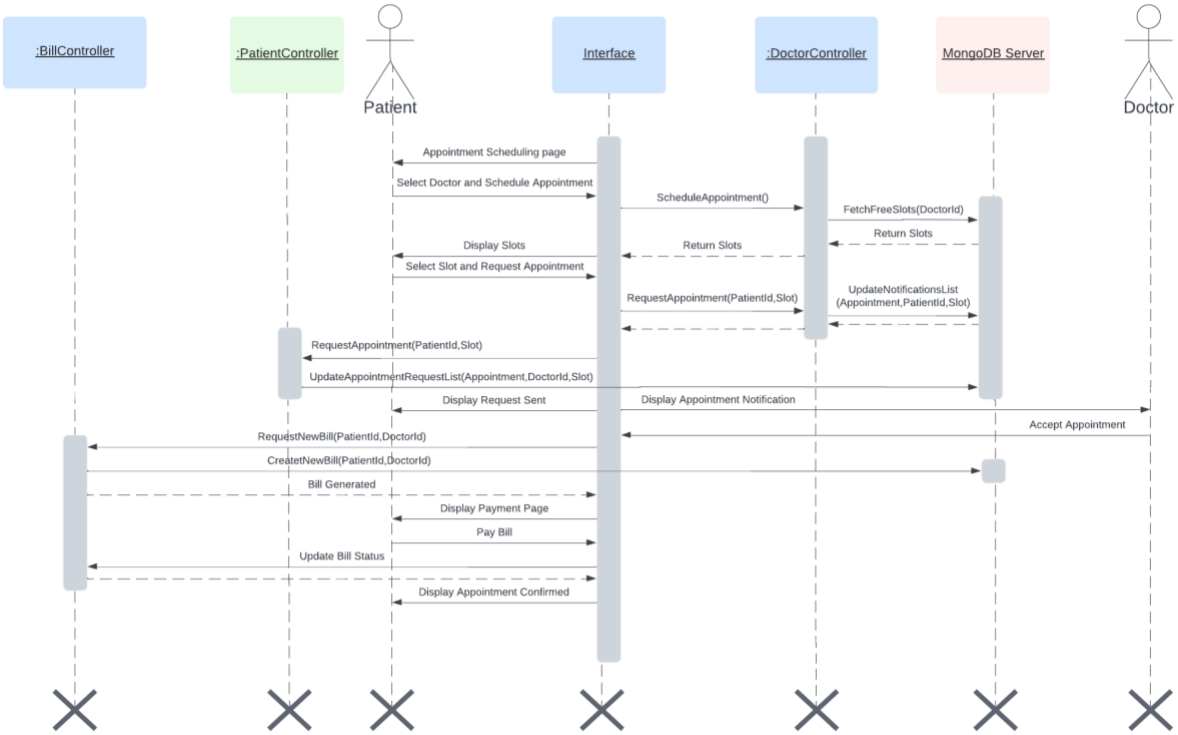
This diagram shows how the Admin/hospital will add a doctor/patient profile.



Sequence Diagram 4 Add Doctor/Patient Profile by Hospital/Admin

5. Appointment Scheduling and Bill Payment

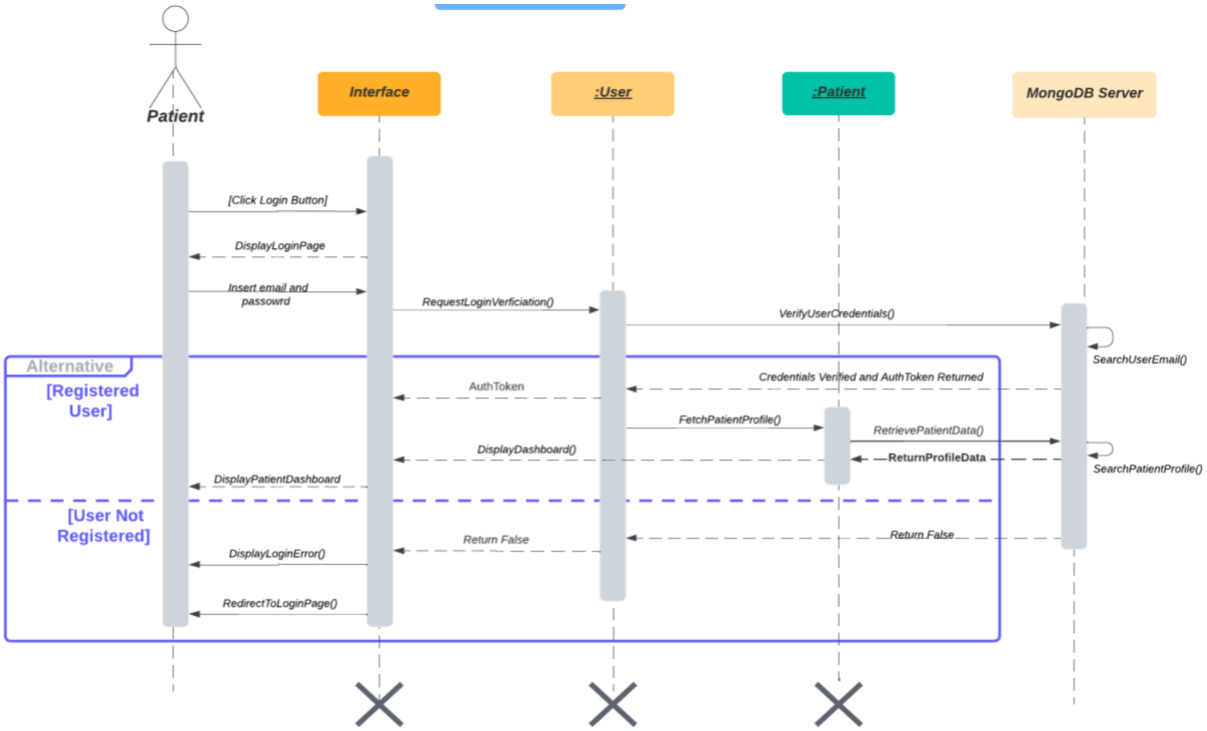
It describes the procedure of scheduling an appointment and the bill payment of that appointment.



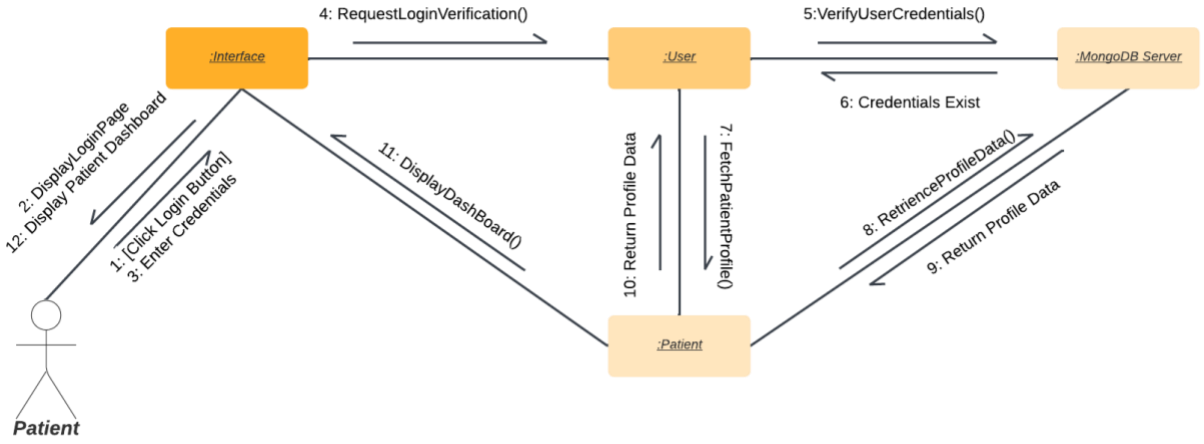
Sequence Diagram 5 Appointment Scheduling and Bill Payment

6. Login

It describes the login procedure to the webpage.

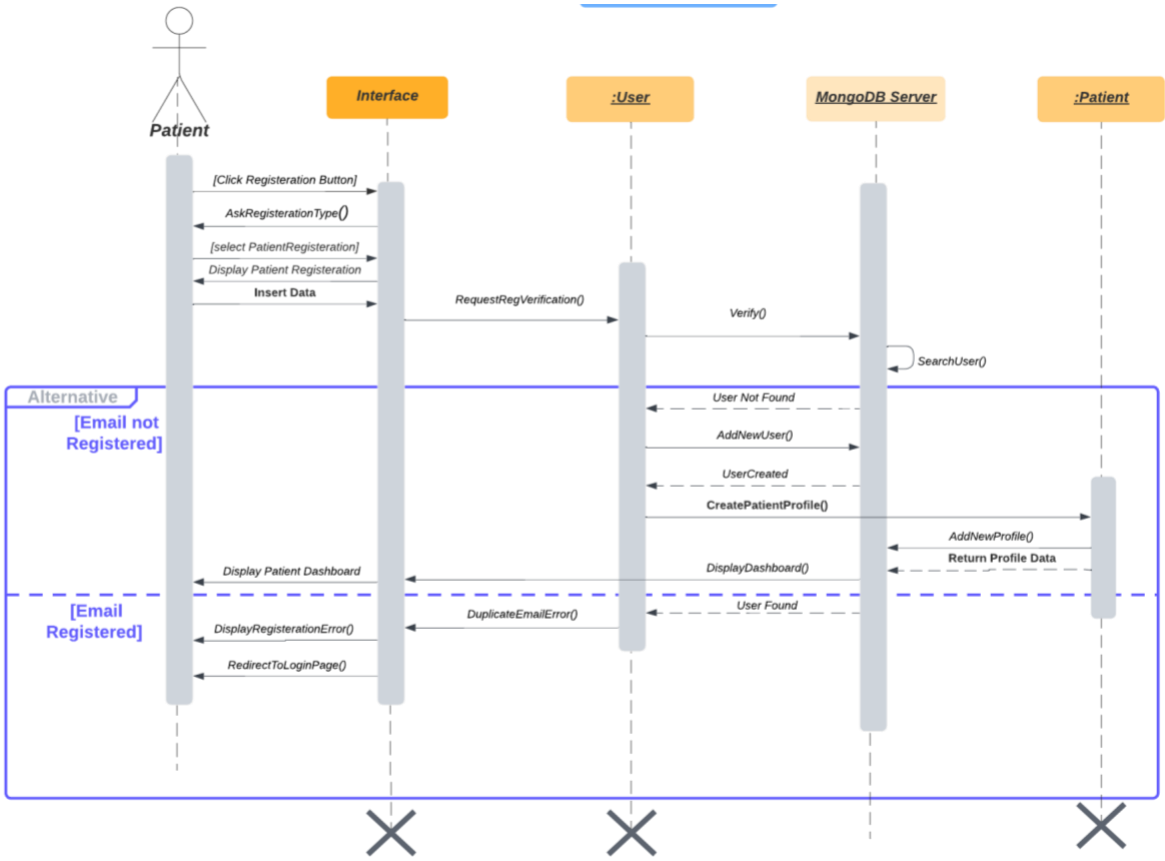


Sequence Diagram 6 Login

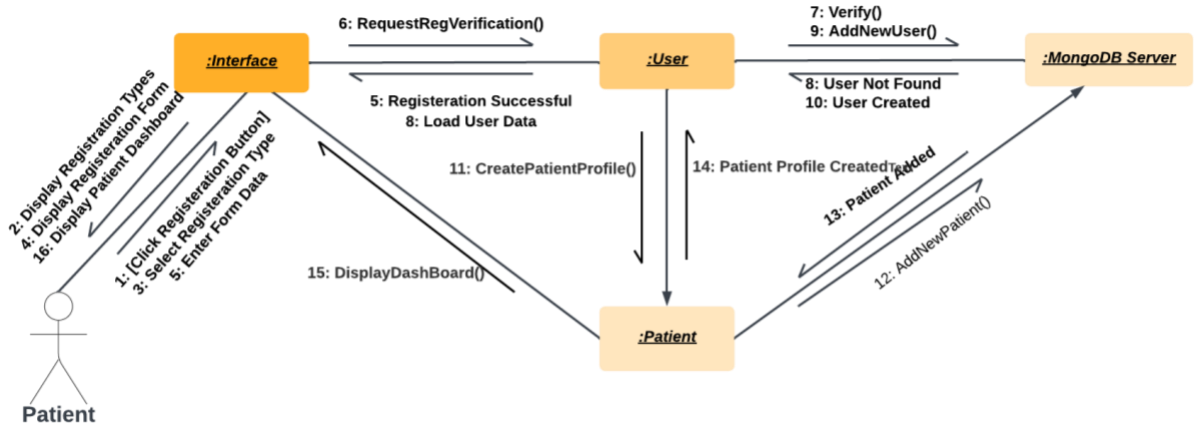


7. Registration

It describes the registration procedure.

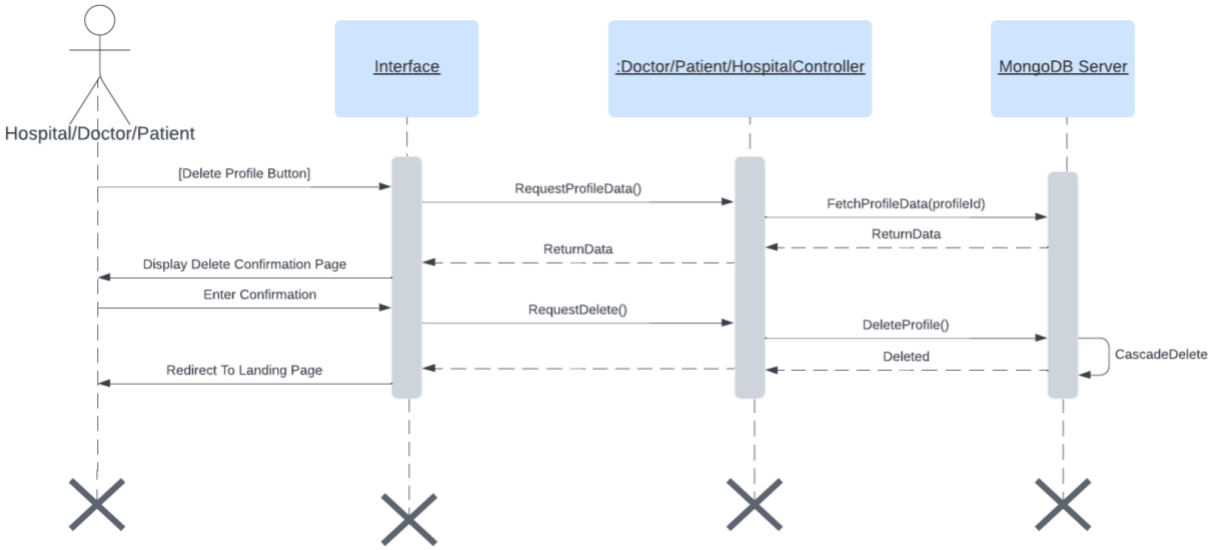


Sequence Diagram 7 Registration



8. Delete Profile

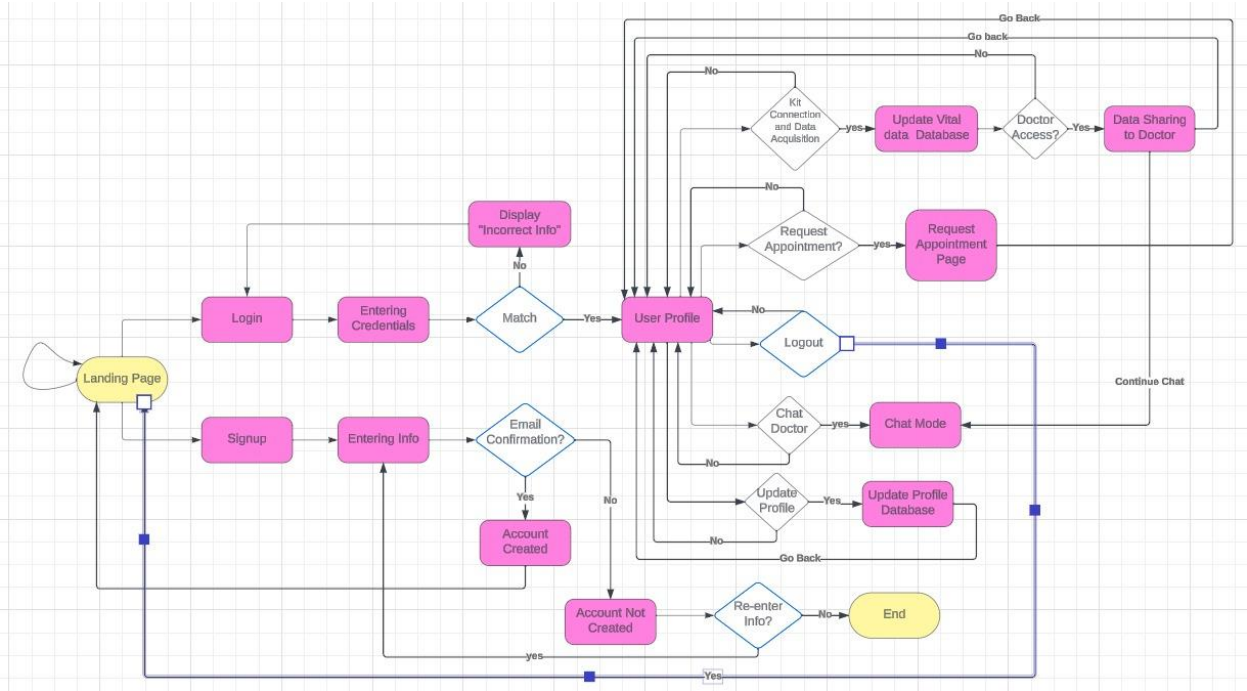
It describes the deletion process of a profile.



Sequence Diagram 8 Delete Profile

5.3 State Transition Diagram

State transition diagrams highlight on the different states of the system.



State Transition Diagram 1

The sub-diagrams for '[Kit Connection and Data Acquisition](), ' '[Update Profile](),' and '[Data Sharing to Doctor]()' are shared under the heading of Activity and Sequence Diagrams.

5.4 Activity Diagrams

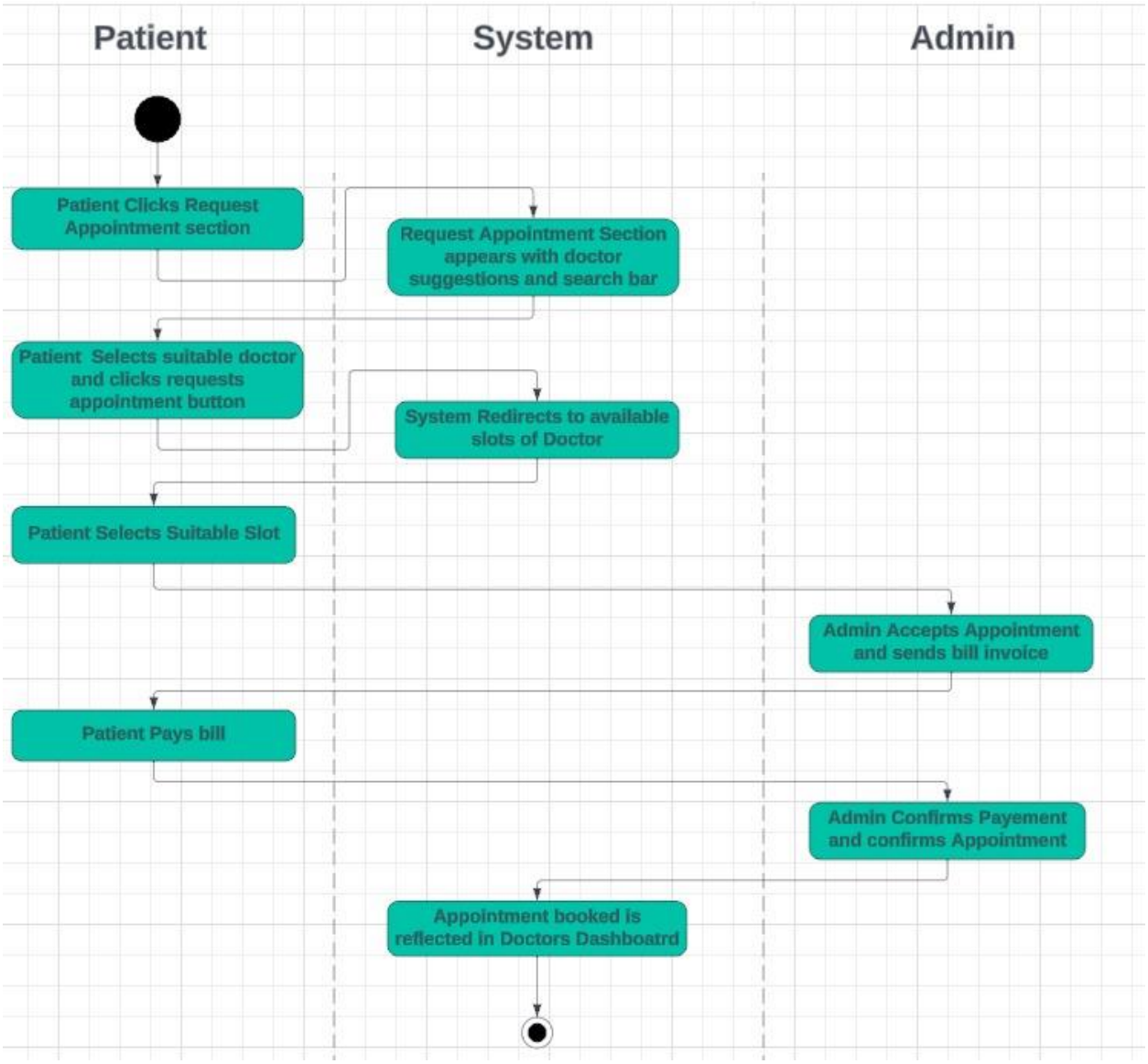
1. Share Patient Data with Doctor

Activities to do to share vital data of patient to the doctor.

Activity Diagram 1 Share Patient Data with Doctor

2. Appointment Booking

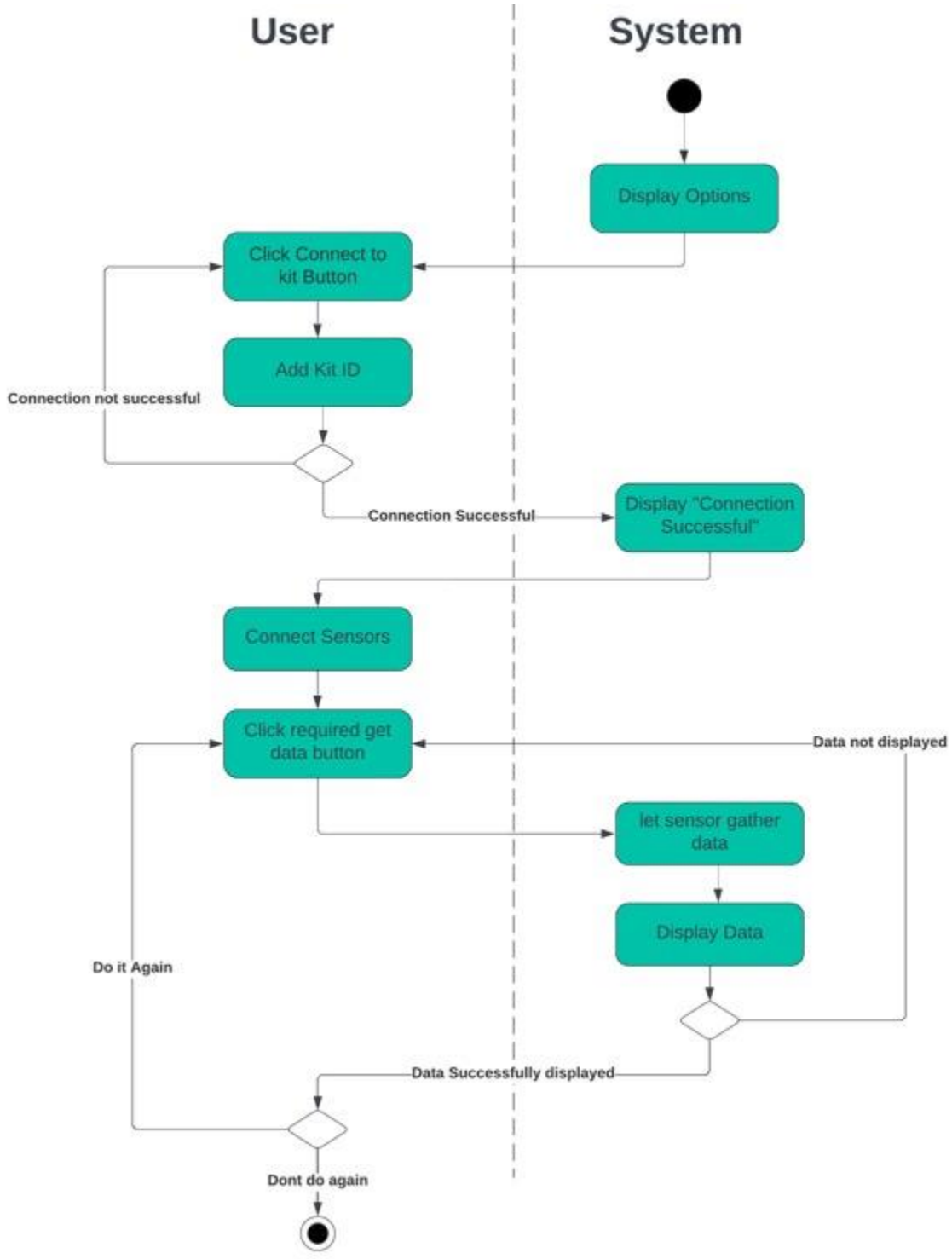
Activities to perform to book an appointment to a hospital doctor/doctor.



Activity Diagram 2 Appointment Booking

3. Kit Connection and Data Acquisition

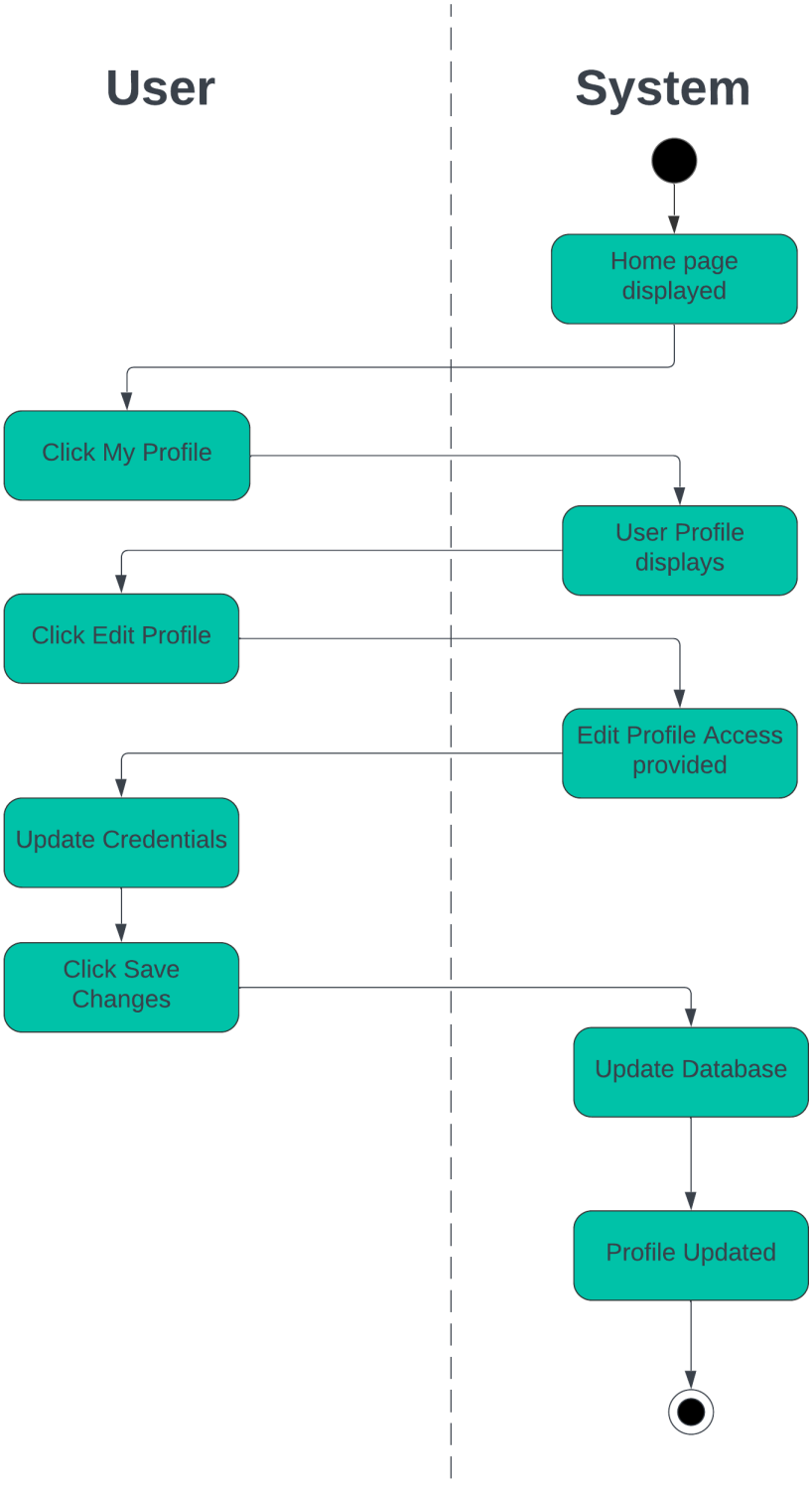
Sub Diagram to follow activities to connect Baymax kit and Acquire Data.



Activity Diagram 3 Kit Connection and Data Acquisition

4. Update Profile

Activities to perform to update any profile.

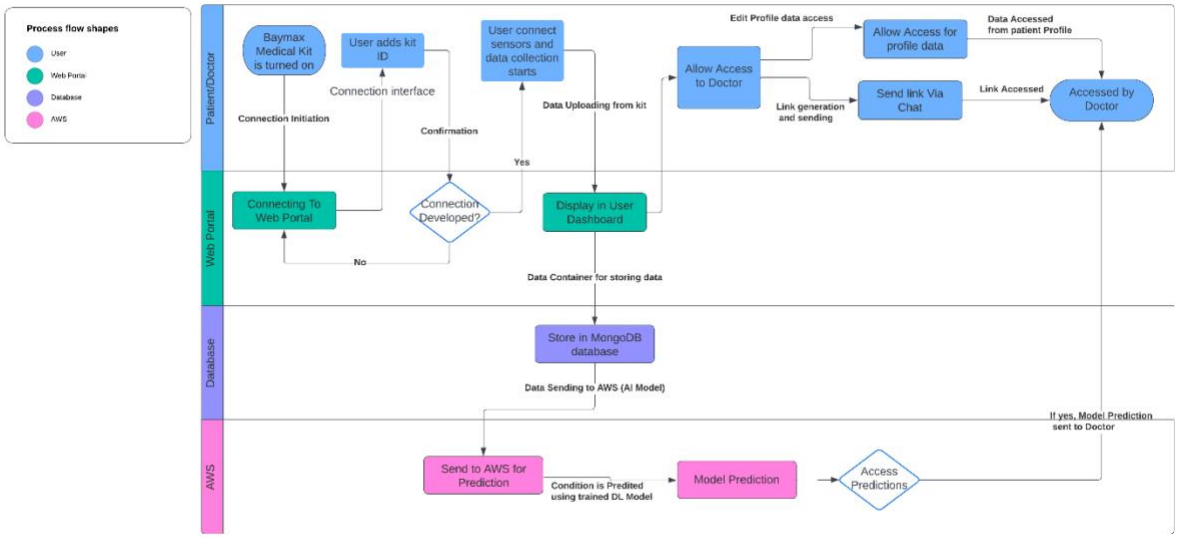


Activity Diagram 4 Update Profile

5.5 Data Flow Diagrams

1. Sending of Data from Patient to Doctor

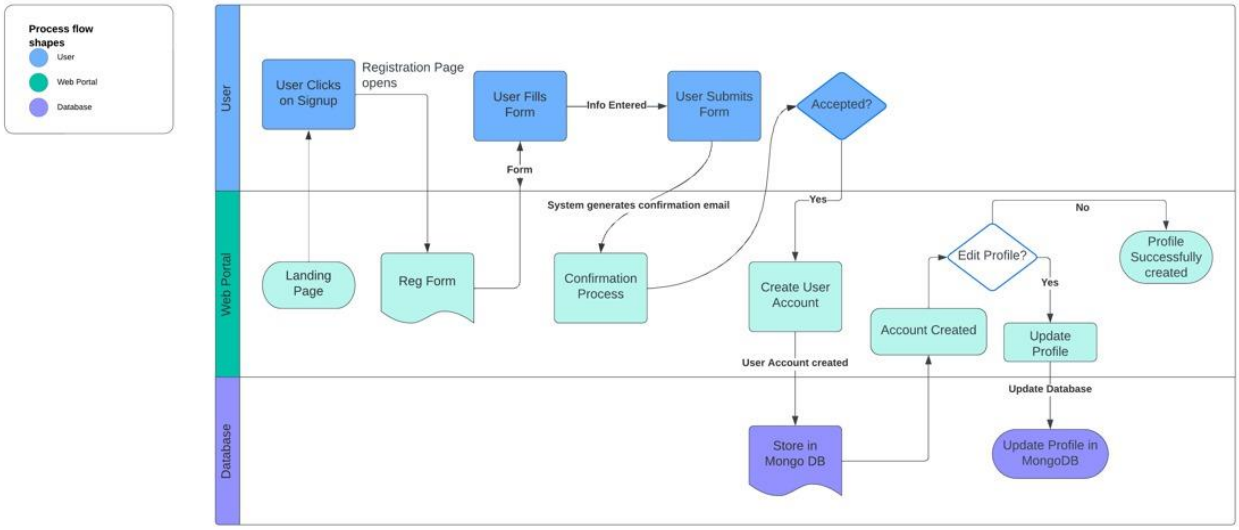
It describes the flow of processes to send a patient data to the relevant doctor.



Data Flow Diagram 1 Sending of Data from Patient to Doctor

2. Registration/Sign Up

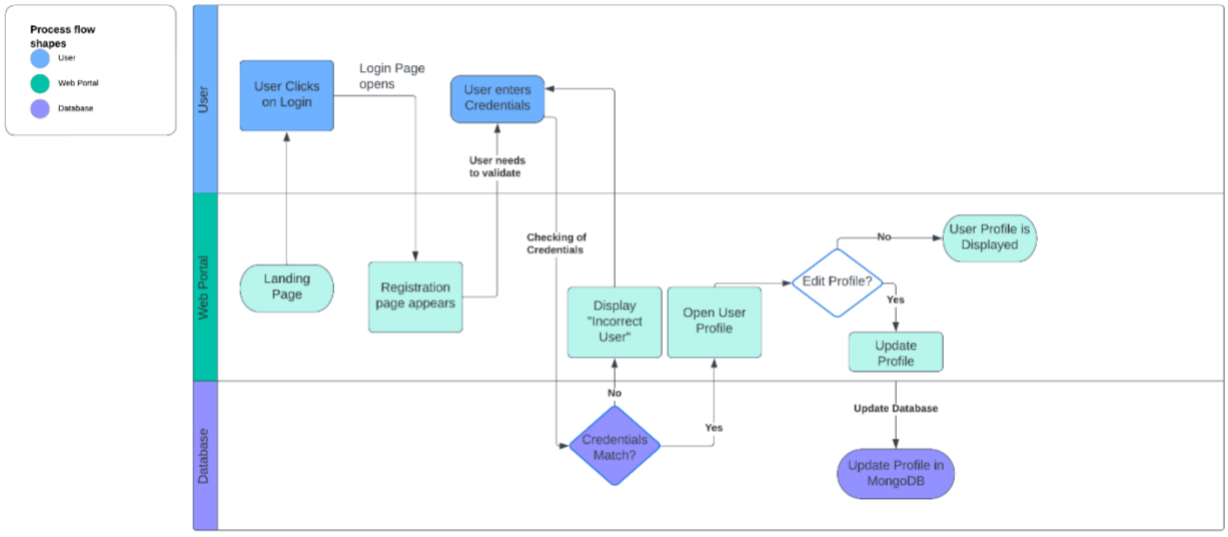
It describes the Registration/Sign Up process through flow of process.



Data Flow Diagram 2 Registration/Sign Up

3. Login

This diagram describes the flow of events/processes/activities to login to a profile page.



Data Flow Diagram 3 Login

6. Data Design

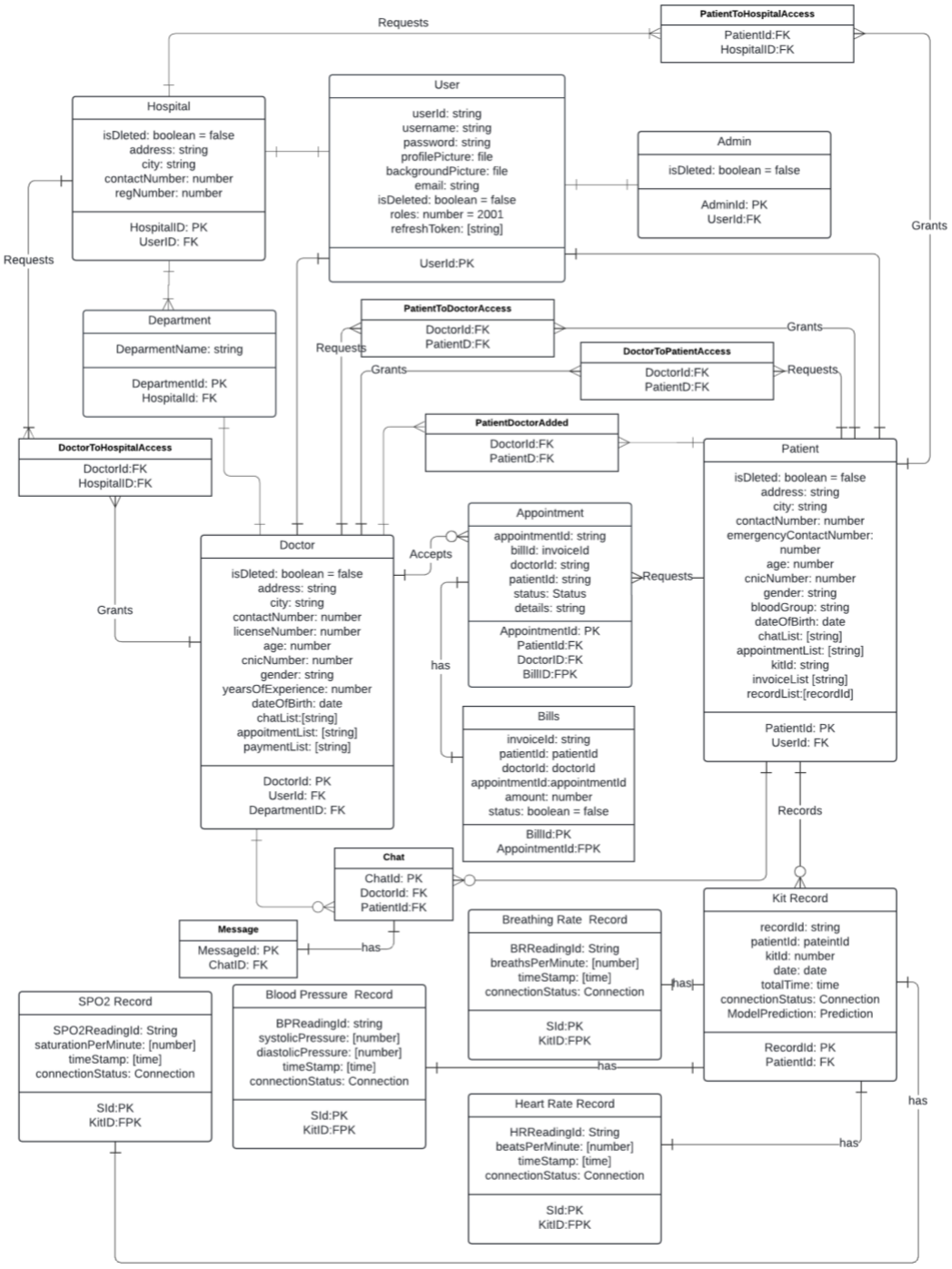
In the context of MongoDB and an AWS ML model predicting on sensor data, the information domain is transformed into data structures as follows:

1. Entities and Relationships:
    a. Entities: Include patients, doctors, hospitals, appointments, sensor readings, and predictions etc.
    b. Relationships: There are relationships between entities, such as a patient having multiple sensor readings, a doctor having multiple patients, etc.
2. MongoDB Data Structures:
    a. MongoDB is a NoSQL database that stores data in BSON (Binary JSON) format.
    b. Each entity is represented as a collection, and each instance of the entity is represented as a document within that collection.
    c. Relationships between entities are handled through references or embedded documents.
3. AWS ML Integration:
    a. WS ML models integrated to process sensor data and make predictions.
    b. AWS services like S3 used to store large volumes of sensor data before feeding it to the ML model.
    c. The ML model can then generate predictions based on the input data.
4. Storage and Retrieval:
    a. MongoDB's flexible schema allows easy storage and retrieval of diverse data types.
    b. Queries are optimized based on the nature of the data, allowing efficient retrieval.
5. Indexes and Performance:
    a. MongoDB supports indexing, which can be used to enhance query performance, especially when searching large datasets.
    b. Indexes on fields commonly used in queries (e.g., patientId, date) can speed up data retrieval.
6. Scaling:
    a. MongoDB's horizontal scaling capabilities make it suitable for handling large amounts of data. You can distribute your data across multiple servers or clusters.
7. Security:

a. MongoDB provides security features, allowing you to control access to the database and encrypt sensitive data.

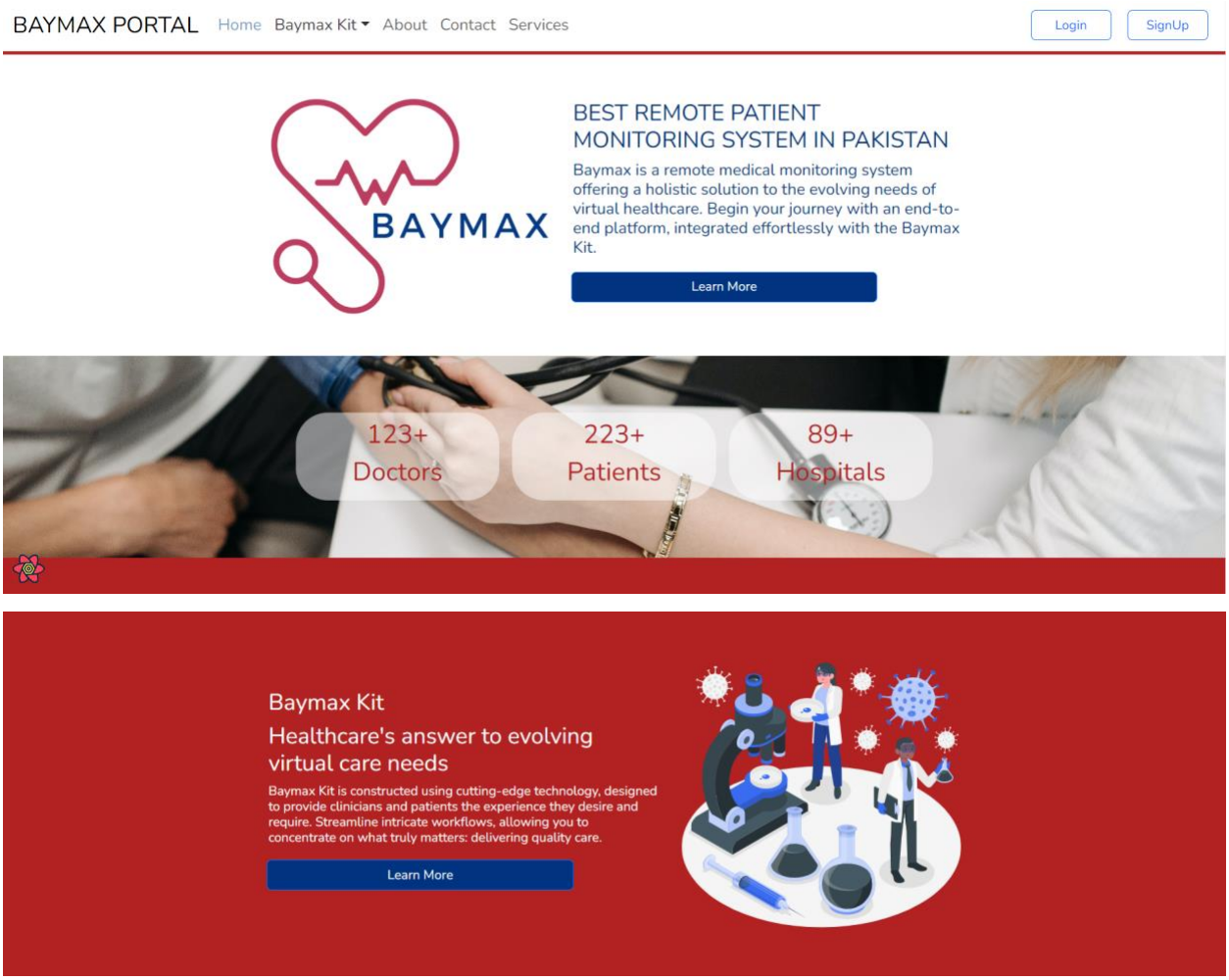
6.1 Entity Relationship (ER) Diagram

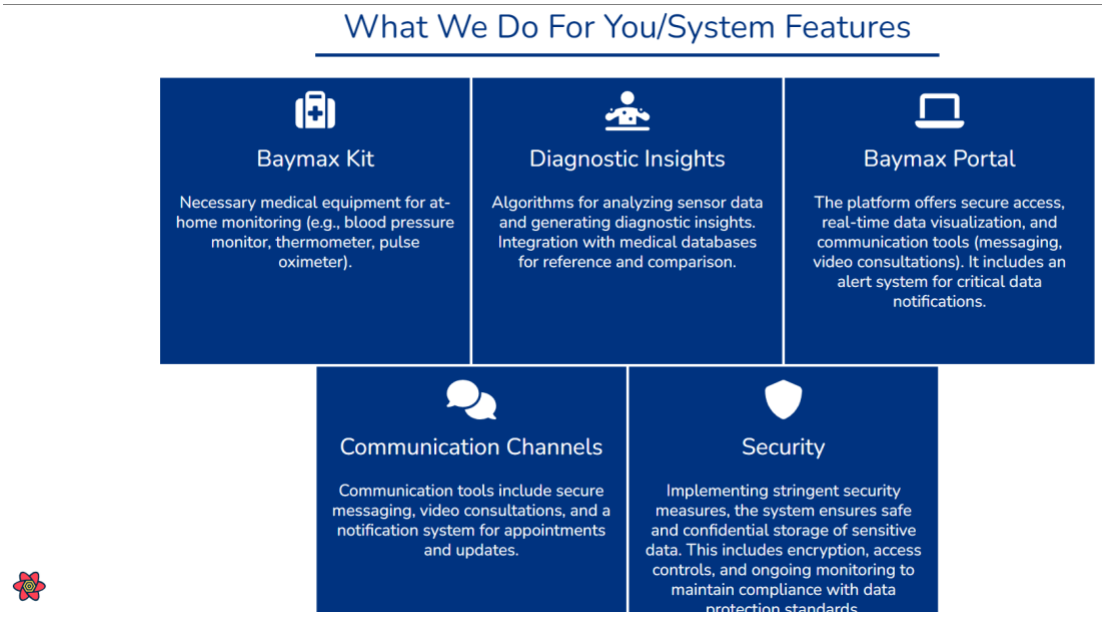
This diagram represents the schema of our relational database.

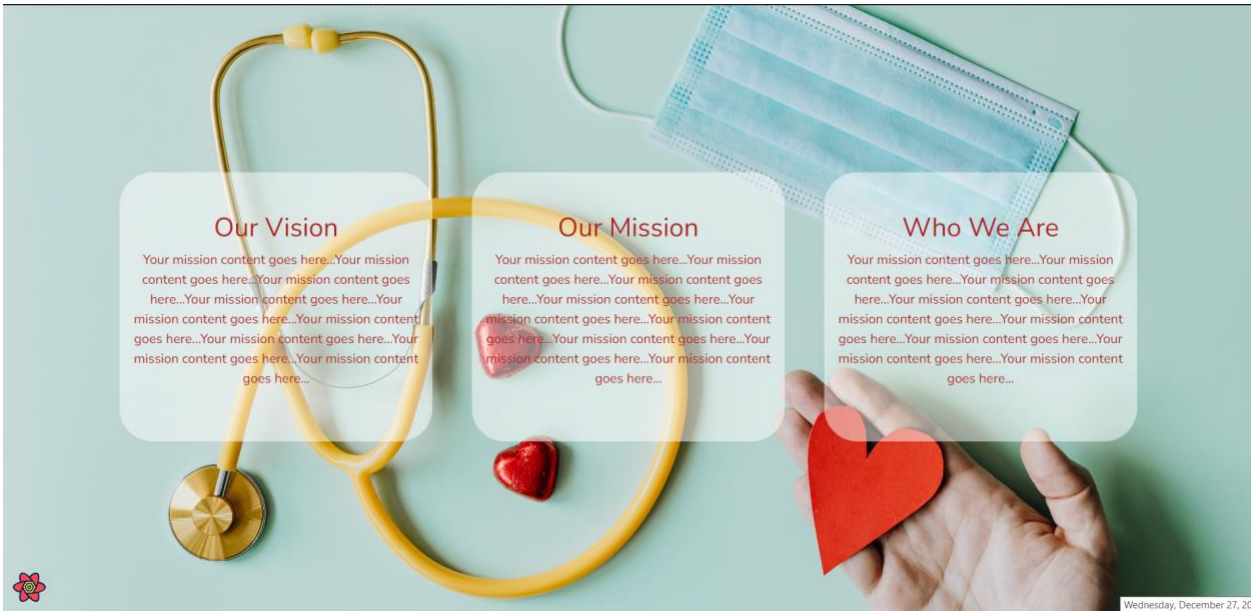


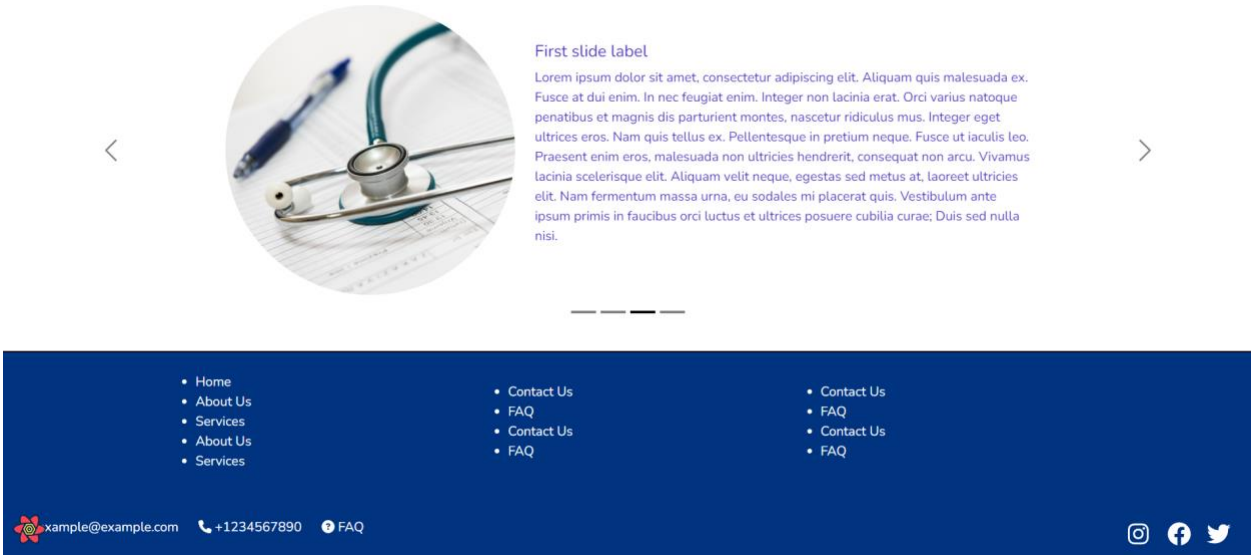
6.2 Screen Images

6.2.1 Landing Page

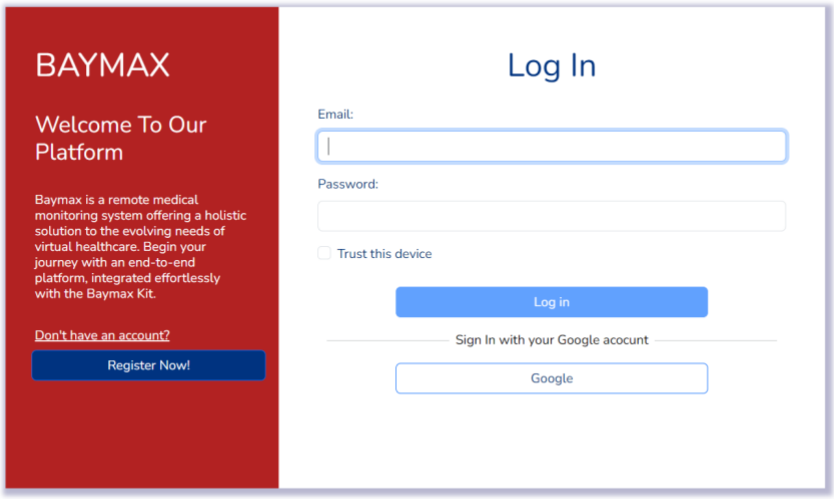



What We Do For You/System Features
Baymax Kit
Necessary medical equipment for at-home monitoring (e.g., blood pressure monitor, thermometer, pulse oximeter).
Diagnostic Insights
Algorithms for analyzing sensor data and generating diagnostic insights. Integration with medical databases for reference and comparison.
Baymax Portal
The platform offers secure access, real-time data visualization, and communication tools (messaging, video consultations). It includes an alert system for critical data notifications.
Communication Channels
Communication tools include secure messaging, video consultations, and a notification system for appointments and updates.
Security
Implementing stringent security measures, the system ensures safe and confidential storage of sensitive data. This includes encryption, access controls, and ongoing monitoring to maintain compliance with data

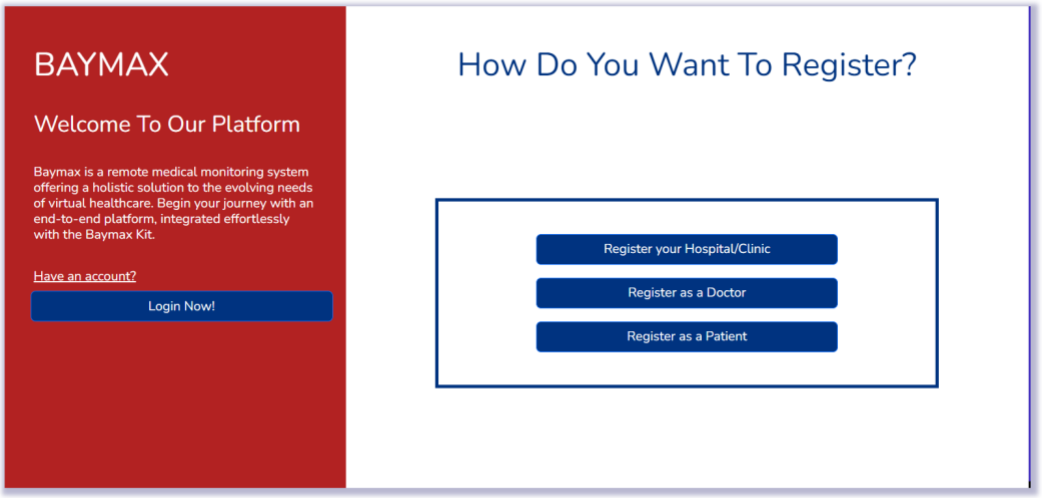

Our Vision
Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...
Our Mission
Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...
Who We Are
Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...Your mission content goes here...
Wednesday, December 27, 2


First slide label
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam quis malesuada ex. Fusce at dui enim. In nec feugiat enim. Integer non lacinia erat. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer eget ultrices eros. Nam quis tellus ex. Pellentesque in pretium neque. Fusce ut iaculis leo. Praesent enim eros, malesuada non ultricies hendrerit, consequat non arcu. Vivamus lacinia scelerisque elit. Aliquam velit neque, egestas sed metus at, laoreet ultricies elit. Nam fermentum massa urna, eu sodales mi placerat quis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Duis sed nulla nisi.
Home
About Us
Services
About Us
Services
Contact Us
FAQ
Contact Us
FAQ
Contact Us
FAQ
Contact Us
FAQ
xample@example.com
+1234567890
FAQ

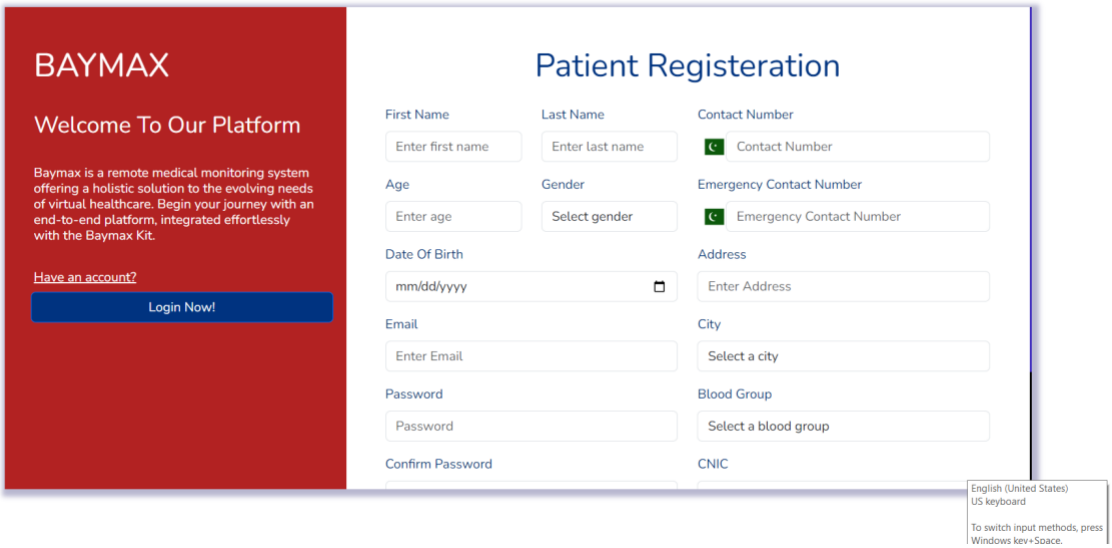
6.2.2 Login


BAYMAX
Welcome To Our Platform
Baymax is a remote medical monitoring system offering a holistic solution to the evolving needs of virtual healthcare. Begin your journey with an end-to-end platform, integrated effortlessly with the Baymax Kit.
Don't have an account?
Register Now!
Log In
Email:
Password:
Trust this device
Log in
Sign In with your Google acocunt
Google

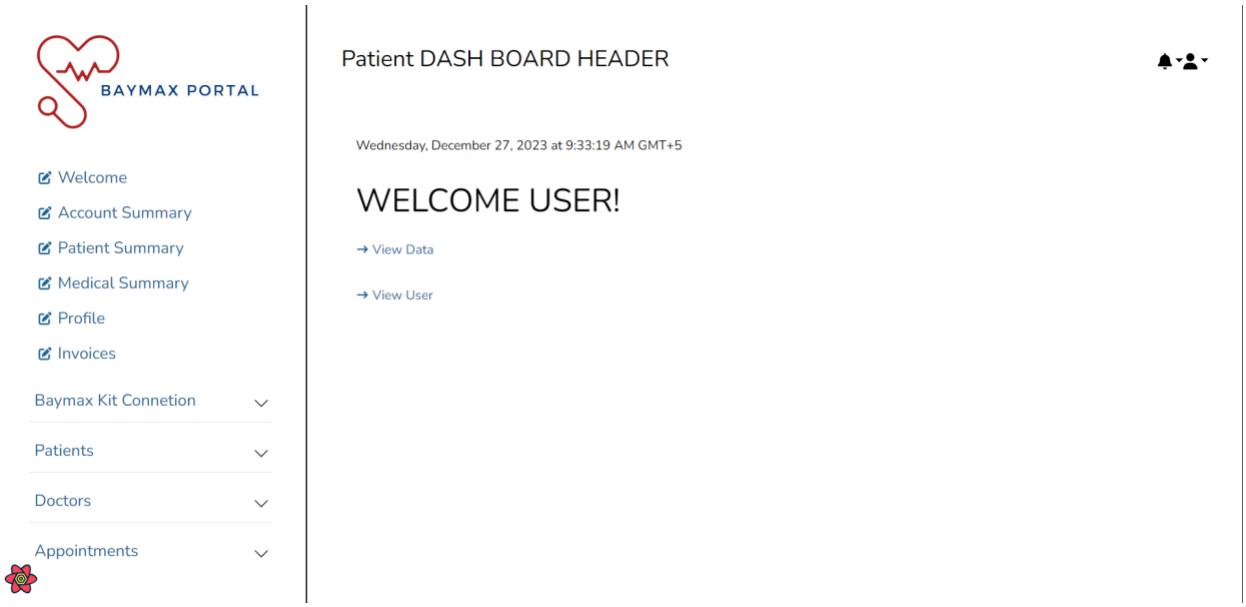
6.2.3 Select Registration



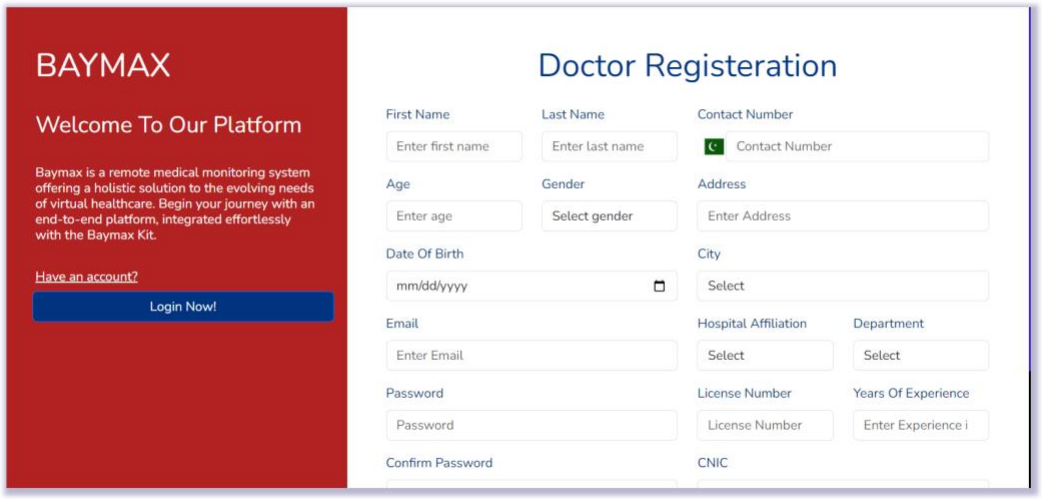
6.2.4 Patient Registration



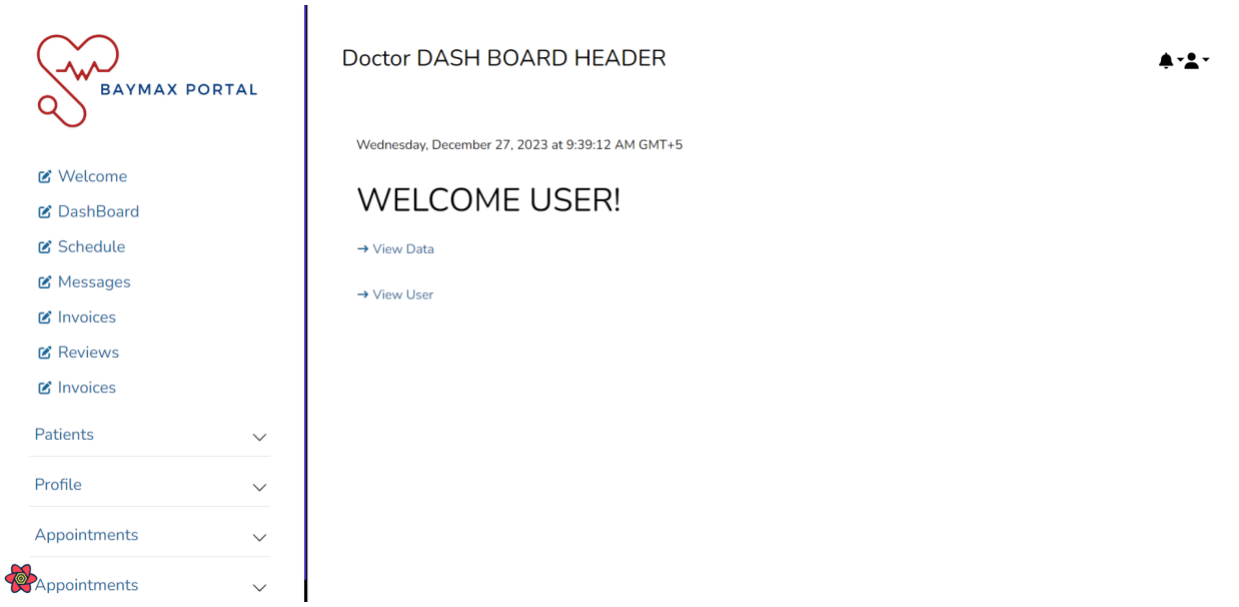
6.2.5 Patient Dashboard



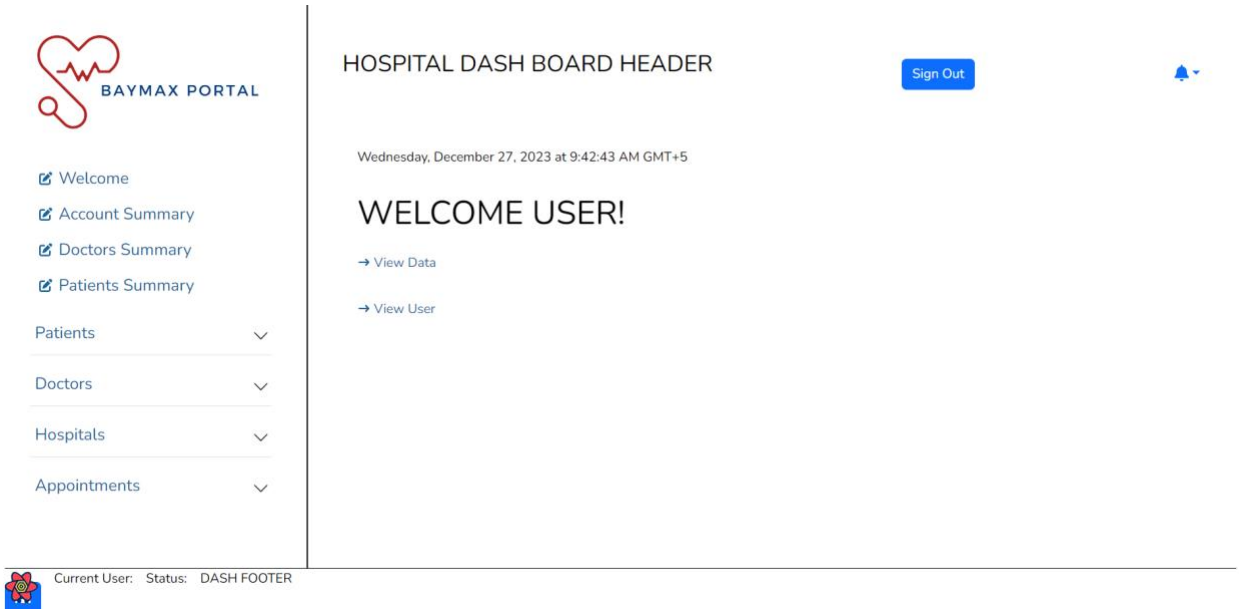
6.2.6 Doctor Registration



6.2.7 Doctor Dashboard

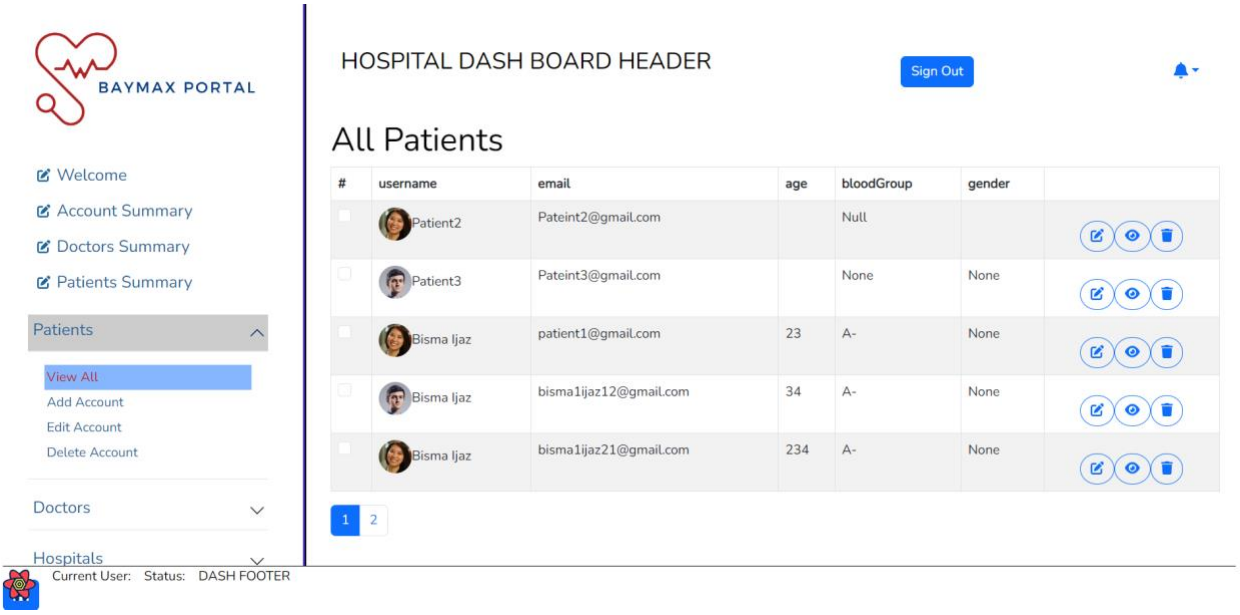


6.2.8 Hospital DashBoard



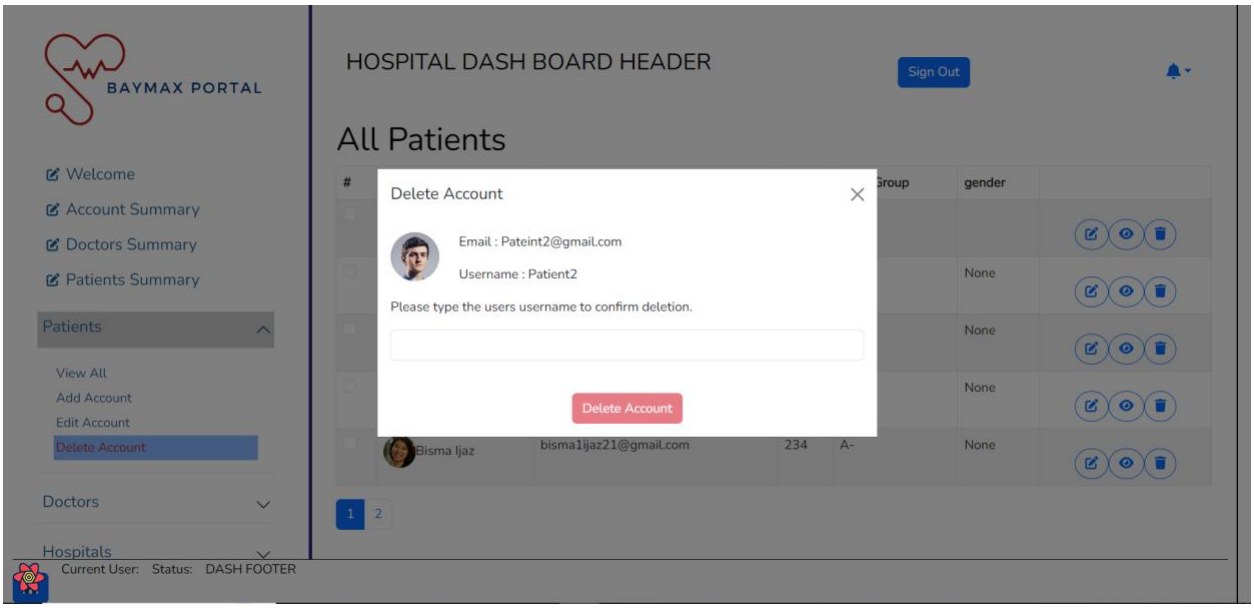
6.3 Screen Objects and Actions

6.3.1 View All



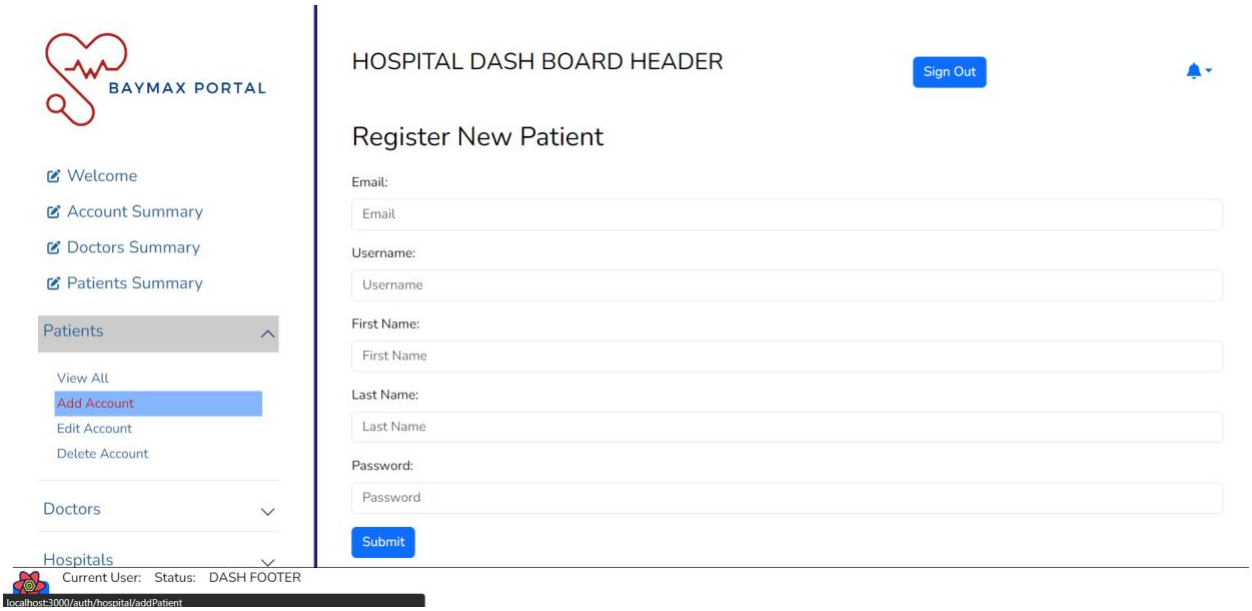
View all gives access to list of profiles of patients/doctors/hospitals according to allowed access

6.3.2 Delete



Allows deletion of objects from list

6.3.3 Add



Allows Adding new accounts or affiliations

The rest of the functionalities are also very similar.

7. Appendix

1. JSON Schema

```
UserSchema
 {
  username: {
   type: String,
   required: true,
  },
  password: {
   type: String,
   required: true,
  },
  profilePicture: {
   type: String,
   required: false,
  },
  backgroundPicture: {
   type: String,
   required: false,
  },
  email: {
   type: String,
   unique: true,
   required: true,
  },
  isDeleted: {
   type: Boolean,
   required: true,
   default: false,
  },
  roles: {
   User: {
    type: Number,
    default: 2001,
   },
   Admin: Number,
   Doctor: Number,
   Patient: Number,
   Hospital: Number,
  },
  profileId: {
   type: mongoose.Schema.Types.ObjectId,
   refPath: 'rolesRef',
  },
  rolesRef: {
   type: String,
   enum: ['Admin', 'Doctor', 'Patient', 'Hospital'],
  },
  refreshToken: [String],
 },
 { timestamps: true }
);

AdminSchema
AdminProfileSchema = new mongoose.Schema({
 userId: {
  type: mongoose.Schema.Types.ObjectId,
  required: true,
  ref: "User",
 },
 isDeleted: {
  type: Boolean,
  required: true,
  default: false,
 },
});

DoctorSchema
```

```
{
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
    ref: "User",
  },
  doctorId: {
    type: String,
    required: true,
  },
  isDeleted: {
    type: Boolean,
    default: false,
  },
  address: {
    type: String,
    required: true,
  },
  city: {
    type: String,
    required: true,
  },
  contactNumber: {
    type: Number,
    required: true,
  },
  licenseNumber: {
    type: Number,
    required: true,
  },
  age: {
    type: Number,
    required: true,
  },
  cnicNumber: {
    type: Number,
    required: true,
  },
  gender: {
    type: String,
    required: true,
  },
  yearsOfExperience: {
    type: Number,
    required: true,
  },
  dateOfBirth: {
    type: Date,
    required: true,
  },
  chatList: {
    type: [String],
    default: [],
  },
  appointmentList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Appointment",
    default: [],
  },
  paymentList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Payment",
    default: [],
  },
  addedPatients: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Patient",
    default: [],
  },
  selectedHospitalList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Hospital",
```

```
    default: [],
  },
  selectedPatientList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Patient",
    default: [],
  },
 }
```

PatientSchema

```
{
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
    ref: "User",
  },
  patientId: {
    type: String,
    required: true,
  },
  isDeleted: {
    type: Boolean,
    default: false,
  },
  address: {
    type: String,
    required: true,
  },
  city: {
    type: String,
    required: true,
  },
  contactNumber: {
    type: Number,
    required: true,
  },
emergencyContactNumber: {
  type: Number,
  required: true,
},
age: {
  type: Number,
  required: true,
},
cnicNumber: {
  type: Number,
  required: true,
},
gender: {
  type: String,
  required: true,
},
bloodGroup: {
  type: String,
  required: true,
},
dateOfBirth: {
  type: Date,
  required: true,
},
chatList: {
  type: [mongoose.Schema.Types.ObjectId],
  ref: "Chat",
  default: [],
},
appointmentList: {
  type: [mongoose.Schema.Types.ObjectId],
  ref: "Appointment",
  default: [],
},
kitId: {
  type: mongoose.Schema.Types.ObjectId,
```

```
    ref: "Kit",
    required: true,
  },
  invoiceList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Invoice",
    default: [],
  },
  addedDoctorsList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Doctor",
    default: [],
  },
  selectedDoctorList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Doctor",
    default: [],
  },
  selectedHospitalList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Hospital",
    default: [],
  },
  recordList: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Record",
    default: [],
  },
 }
AppointmentSchema
{
  appointmentId: {
    type: String,
    required: true,
  },
  billId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Invoice",
  },
  doctorId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Doctor",
    required: true,
  },
  patientId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Patient",
    required: true,
  },
  status: {
    type: String,
    required: true,
  },
  details: {
    type: String,
  },
 }
BillSchema  {
  invoiceId: {
    type: String,
    required: true,
  },
  patientId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Patient",
    required: true,
  },
  doctorId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Doctor",
    required: true,
  },
```

```
  appointmentId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Appointment",
    required: true,
  },
  amount: {
    type: Number,
    required: true,
  },
  status: {
    type: Boolean,
    default: false,
  },
 }
MessageSchema
{
  messageId: {
    type: String,
    required: true,
  },
  content: {
    type: String,
    required: true,
  },
  time: {
    type: Date,
    required: true,
  },
  senderId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true,
  },
 }
ChatSchema
 {
  chatId: {
    type: String,
    required: true,
  },
  patientId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Patient",
    required: true,
  },
  doctorId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Doctor",
    required: true,
  },
  messageList: {
    type: [String],
    default: [],
  },
 }
KitRecordSchema
 {
  recordId: { type: String, required: true },
  patientId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Patient",
    required: true,
  },
  kitId: { type: Number, required: true },
  date: { type: Date, required: true },
  totalTime: { type: String, required: true },
  connectionStatus: { type: String, required: true },
  BPReadingsIds: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "BPReading",
    default: [],
  },
```

```
  HRReadingsIds: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "HRReading",
    default: [],
  },
  SPO2ReadingsIds: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "SPO2Reading",
    default: [],
  },
  BRReadingsIds: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "BRReading",
    default: [],
  },
  ModelPrediction: { type: String },
 }
BPRecordSchema
{
 BPReadingId: { type: String, required: true },
 systolicPressure: { type: [Number], required: true },
 diastolicPressure: { type: [Number], required: true },
 timeStamp: { type: [Date], required: true },
 connectionStatus: { type: String, required: true },
}
HeartRateRecordSchema
{
 HRReadingId: { type: String, required: true },
 beatsPerMinute: { type: [Number], required: true },
 timeStamp: { type: [Date], required: true },
 connectionStatus: { type: String, required: true },
}

BreathingRateRecordSchema
{
 BRReadingId: { type: String, required: true },
 breathsPerMinute: { type: [Number], required: true },
 timeStamp: { type: [Date], required: true },
 connectionStatus: { type: String, required: true },
}
Spo2RecordSchema
{
 SPO2ReadingId: { type: String, required: true },
 saturationPerMinute: { type: [Number], required: true },
 timeStamp: { type: [Date], required: true },
 connectionStatus: { type: String, required: true },
}
```