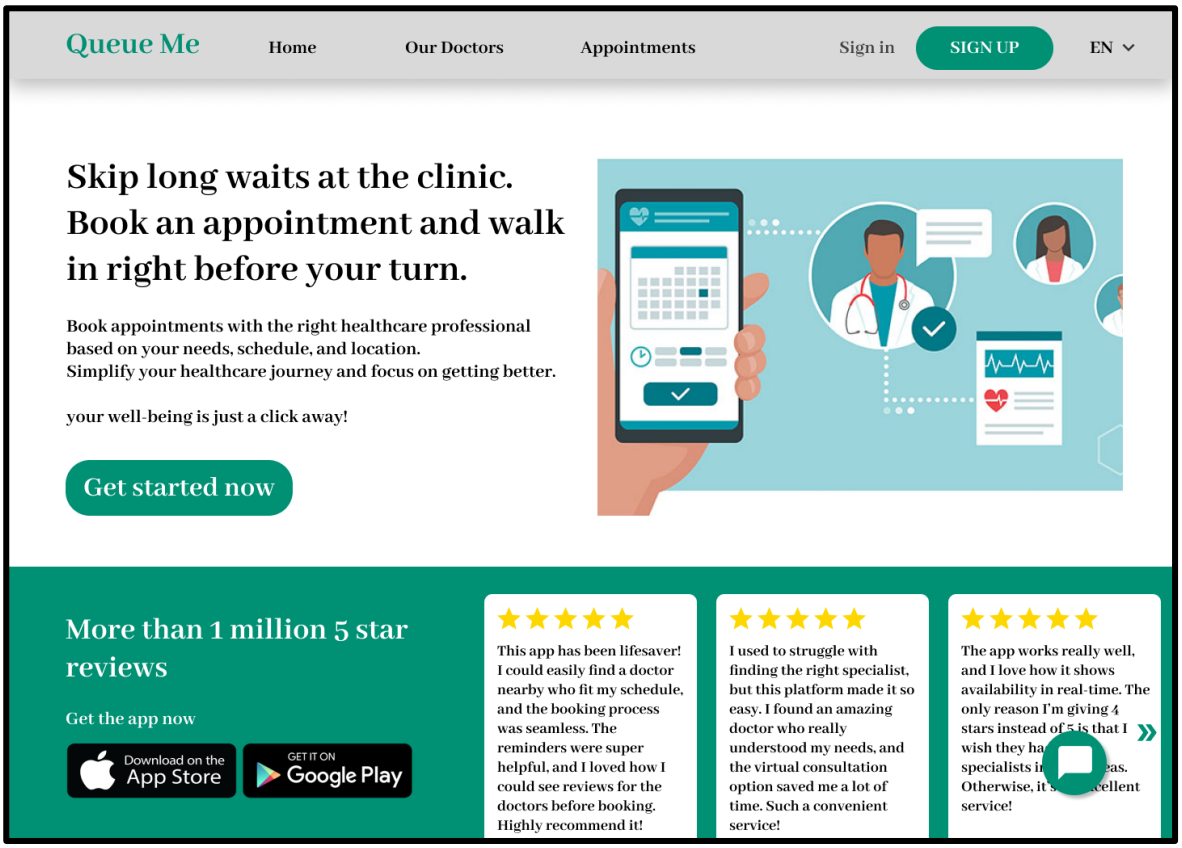
Software Requirements

Specification

for Queue.Me



Prepared by Team 5 (COMP246-007):
Al Bouchi, Ahmad
Jongsubcharoen, Jaturaput
Lee, Jin Hoi
Menil, Lorenzo Jr.
Quazi, Mohammed
Salunga, Bianca
Tiquio, Angelo

Table of Contents

Revision History

Name	Date	Reason For Changes	Author	Version
Part A	09/05/24	First version's SRS Part A is defined	BS	v1.0
Part A	09/12/24	SRS template is updated, System Context Diagram is updated, and SRS Part A is revised	AT, BS	v1.1
Part A	09/19/24	SRS Part A is revised, and added use cases	LM, AT, JJ, MQ, BS	v1.2
Part A	09/24/24	Updated SRS Part A and use cases as per current requirements, added ERD, user stories, and acceptance criteria. Also, removed sections from previous semester	AT, JJ, MQ, BS,	v1.3
Part A	09/30/24	Added Sequence and State Diagrams	JJ, MQ	v1.4
Part A	10/17/24	Update on SRS Part A Overall Feedback	AT, JJ, BS, MQ	v1.5
Part B	10/28/24	SRS Part B is defined	BS	v2.0
Part B	11/04/24	Update on SRS Part A additional Feedback	AT, JJ	v2.1
Part C	11/09/24	SRS Part C is defined	BS	v3.0
Part B	11/15/24	Update on SRS Part B Overall Feedback	AT, JJ, BS, MQ	v3.1
Part C	11/13/24	Added Section 1 and 2 Software Design Patterns	AT, BS, LM	v3.1
Part C	11/18/24	Added Section 3 UI/UX Design (mobile/web) and section 5 Gantt Chart	JJ, MQ, BS	v3.1
Part C	11/23/24	Added and revised Section 4 High-Level Component/Deployment Diagram	JL, BS	v3.2
Part C	12/02/24	Added navigation pages for Section 3 UI/UX Design in web	MQ	v3.2
Part C	12/04/24	Updated Section 4 High-Level Component/Deployment Diagram based on feedback	BS	v3.3

Part A: Project Scope and Requirements

Section 1: Problem Statement

1.1.a Problem Domain and Need

The healthcare landscape is brimming with appointment booking solutions. While existing products and applications offer basic functionalities, Queue.Me takes a revolutionary leap forward by delivering a more comprehensive and patient-centric platform that simplifies appointment booking and enhances communication between patients and healthcare providers.

With AI-driven scheduling, centralized medical records, and virtual consultation capabilities, Queue.Me empowers patients to manage their care while streamlining administrative tasks for healthcare professionals. This comprehensive solution improves access to healthcare, reduces wait times, and optimizes resource management, ultimately contributing to better patient outcomes and satisfaction.

1.1.b List of Capabilities and Benefits

To improve the experiences of patients and providers in the healthcare system, Queue.Me provides a unique combination of cutting-edge features and useful advantages. The list of essential capabilities and benefits that go along with them is as follows:

Capabilities:

1. User and Account Management - For both patients and physicians, secure account creation, login, profile maintenance, and password recovery are essential.
2. AI-Powered Appointment and Queue Management - To cut down on wait times and boost productivity, intelligent scheduling and rescheduling via chatbots with automatic reminders and real-time updates is implemented.

3. Virtual Consultation and Medical Records Access - Patients can safely access or share their medical records with their doctors and consult them from a distance to improve care coordination.
4. Integrated Payment and Feedback System - Patients can evaluate and review doctors after appointments, and payments are processed securely with payment history accessible.

Benefits:

1. Improved User Experience - A patient-friendly interface and easy user account management make it easier for patients and providers to communicate.
2. Enhanced Access to Healthcare - A wider range of people may now access healthcare more easily thanks to AI-driven appointment scheduling, online consultations, and current medical information.
3. Efficient Care and Resource Management - Wait times are decreased and healthcare operations are improved through AI-powered queue management, real-time notifications, and resource allocation optimization.
4. Seamless Medical Record Access - During consultations, informed, consistent care is guaranteed by secure sharing and access to medical history.
5. Streamlined Payments and Patient Satisfaction - Increased patient satisfaction and service trust are the results of transparent payment processing and the option to rate and review physicians.

1.2. Identify the Stakeholders & their Roles

There are various types of users that will interact with Queue.Me.

- Patients - will use the platform to book appointments, access medical reports & prescriptions, and review doctors.
- Doctors - will use software to manage their schedules, view appointments, access medical records, communicate with patients, provide prescriptions digitally and consult with patients virtually.

- Medical receptionist / clinical staff - will use software to manage appointments, handle inquiries & process payments. They have limited access to doctor's account.
- Hospital and Clinics – organizations where doctors and medical staff work. They benefit from streamlined appointment scheduling and patient management.
- Pharmacy Services – Providers who will fill prescriptions generated by doctors within the system.
- System Administrator – The System Administrator's job is to ensure the software runs smoothly and troubleshoots any problems.
- Investors – The investors will fund the development and expansion of the system.
- IT and Development Team – These people are responsible for building, maintaining, and upgrading of the system.

1.3. Identify the Sub-systems of your Application

There are various sub-systems for the Queue.Me application.

- User Management Subsystem
- Appointment Management Subsystem
- Medical Records Management Subsystem
- Rating and Review Subsystem
- AI Integration Subsystem
- Administrative Subsystem
- Payment Management Subsystem

1.4. Intended Users or Audience and Reading Suggestions

This document for the Queue.Me system is intended for a variety of stakeholders involved in the project.

- Software Developers (Devs) - Devs should focus on the functional and non-functional requirements section, and other technical details included in this document.

- Project Managers (PM) - PMs should review sections 1 (Introduction), 2 (Overall Description), and skim sections 3 (Requirements) and 4 (System Features) for a high-level understanding.

- Business Analysts/Product Owners (BA/PO) - BAs and POs should focus on Sections 1 (Introduction), 2 (Overall Description), and 3 (Requirements).

- Software Quality Assurance (QA) Engineers - QAs should focus on Section 3 (Requirements) and 4 (System Features) to understand expected system behavior and functionalities to be included during testing.

- Technical Writers (Optional) - Technical writers should review Section 1 (Introduction), 2 (Overall Description), and 3 (Requirements) to understand user interactions and system functionalities.

- End Users - End Users such as doctors and patients can refer to section 3 (Requirements) and 4 (System Features) for the list of requirements. Additionally, they should focus on section 1 (Introduction) and 2 (Overall Description) and refer to section 2.6 (User Documentation) for tutorials and support information of the system.

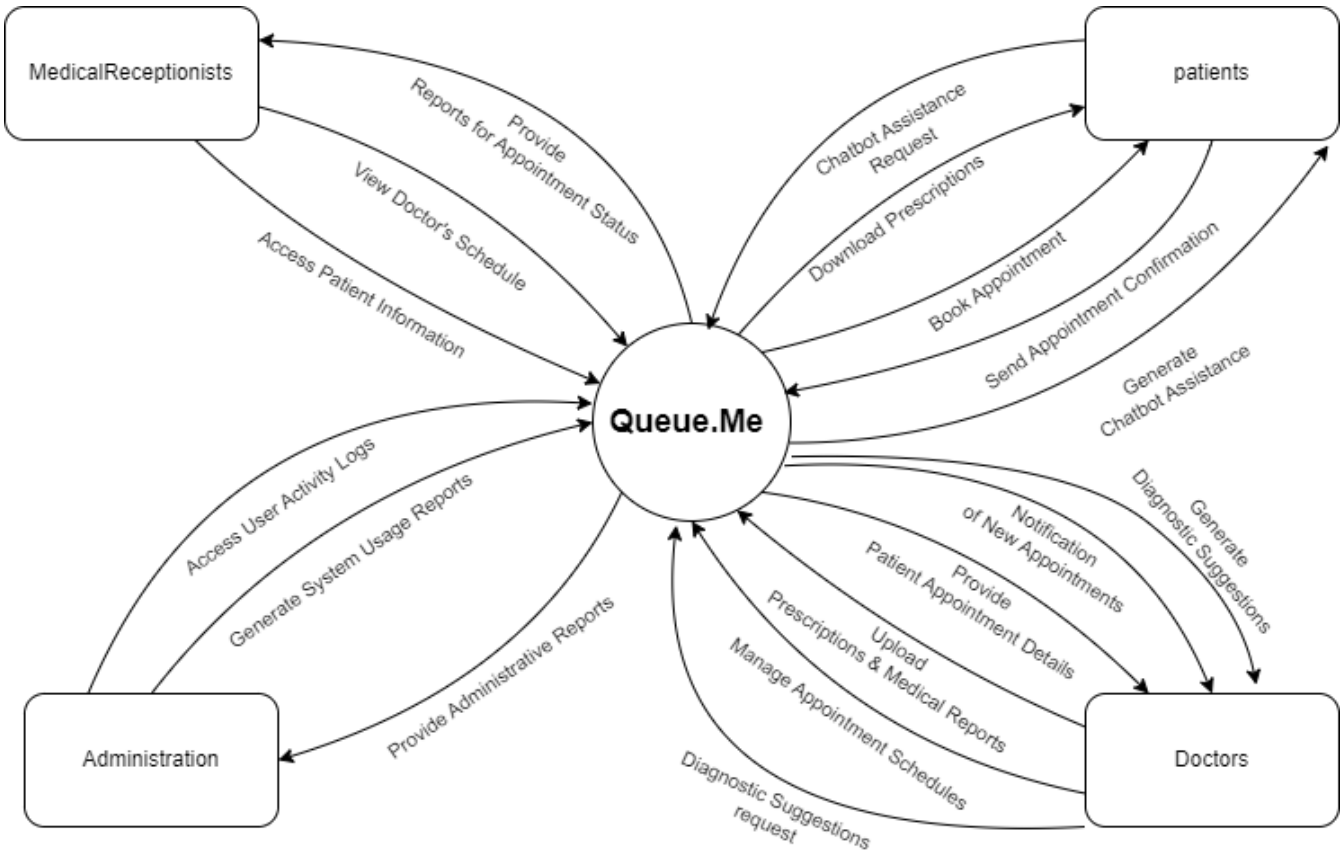
It is recommended to read this document sequentially. Readers with a strong technical background can skim Introduction and Overall Description and delve deeper into technical details in Requirements, System Features, Functional and Non-Functional Requirements. While non-technical individuals can focus on Introduction, Overall Description and Requirements for a general understanding of the system’s purpose and functionalities.

Section 2: Context Flow Diagram Structured Overview

2.1. Context Flow Diagram (CFD)

This section of the SRS outlines the context flow diagram of Queue.Me. These interfaces define how the system interacts with the users.

The users will access the software using a mobile app will support both iOS and Android platforms, while the web interface will support modern web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. The backend will utilize a JSON Node.js server for efficient and secure data management.



System Context Diagram

Section 3: Functional Requirements

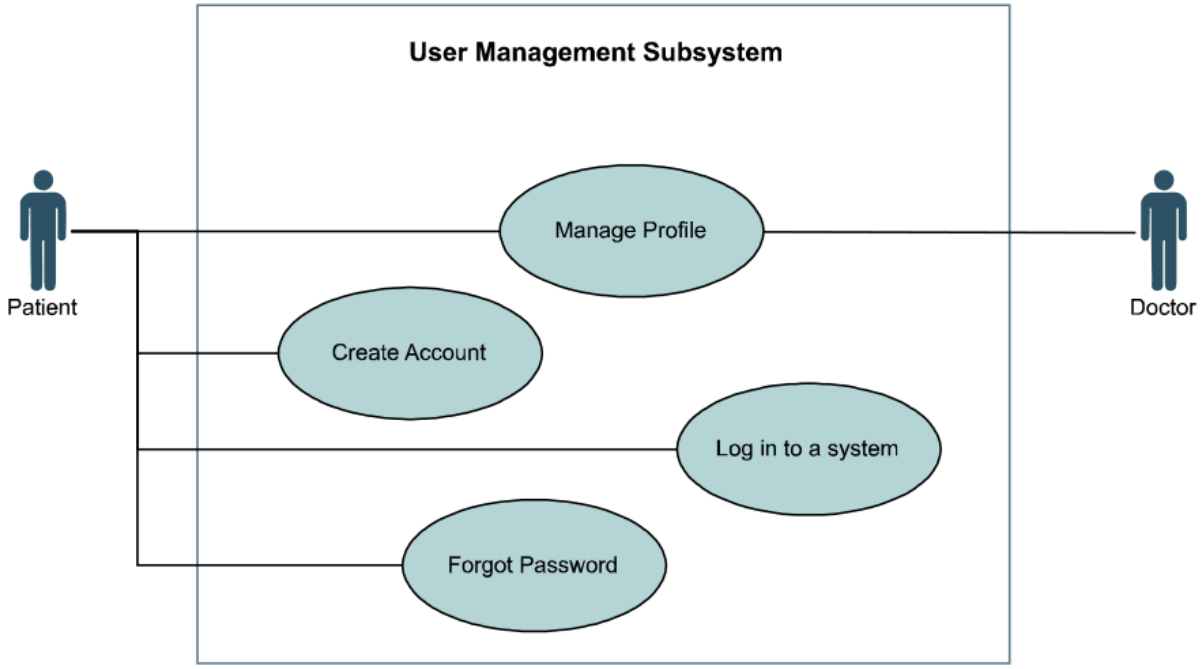
3.1. Goal Use Cases List, Use Case Diagrams (3.2) and User Stories (3.3) for each Sub-systems

This document details the **goal use cases list (3.1)**, **use case diagrams (3.2)** and **user stories (3.3)** for each sub-systems for the Queue.Me application with their role players and descriptions.

3.1.1. User Management Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Create Account	Patient, Doctor	Patients and doctors click the sign-up button, fill out a registration form, and create an account in the system.
FR02	Edit Profile	Doctor	Doctor edits their profile to add or update personal information.
FR03	Login User	Patient, Doctor	Patients and doctors enter their credentials (username and password) to access their accounts in the system.
FR04	Request Password Reset	Patient, Doctor	Patients and doctors who forget their passwords can request a password reset. They will receive instructions via email to reset their password securely.
FR05	Authenticate User	Patient, Doctor	Patients and doctors authenticate their identity through various methods (login, password recovery) to ensure secure access to the system.

Use case Diagram:



User Story:

1. **Create Account**

   As a Doctor/Patient,

   I want to create an account

   so that I can access the system to manage appointments and my medical profile.

Acceptance Criteria:

- The user can click the sign-up button.
- The registration form must include fields for necessary information.
- Upon submission, the account is created successfully.

2. **Edit Profile**

   As a Patient/Doctor,

   I want to edit my profile

   so that I can update my professional information.

Acceptance Criteria:

- The patient/doctor can access and edit the profile page.
- Changes are saved and reflected in the profile right away.

3. Login User

As a Patient/Doctor,

I want to login into my account

so that I can access my personal information and manage my medical appointments.

Acceptance Criteria:

- The patient/doctor can navigate to the login page from the main application interface.
- The login page must include fields for entering a username and password, along with a "Login" button.
- Passwords are encrypted during input and not displayed in plain text.
- Upon entering valid credentials, the user is navigated to their dashboard.
- If invalid credentials are entered, an error message is displayed indicating that the username or password is incorrect.
- A "Forgot Password" link is available on the login page, redirecting the user to a password recovery page.
- A logout button is accessible in the user interface once logged in.

4. Request Password Reset

As a Patient/Doctor,

I want to reset my password if I forgot it

so that I can regain access to my account securely.

Acceptance Criteria:

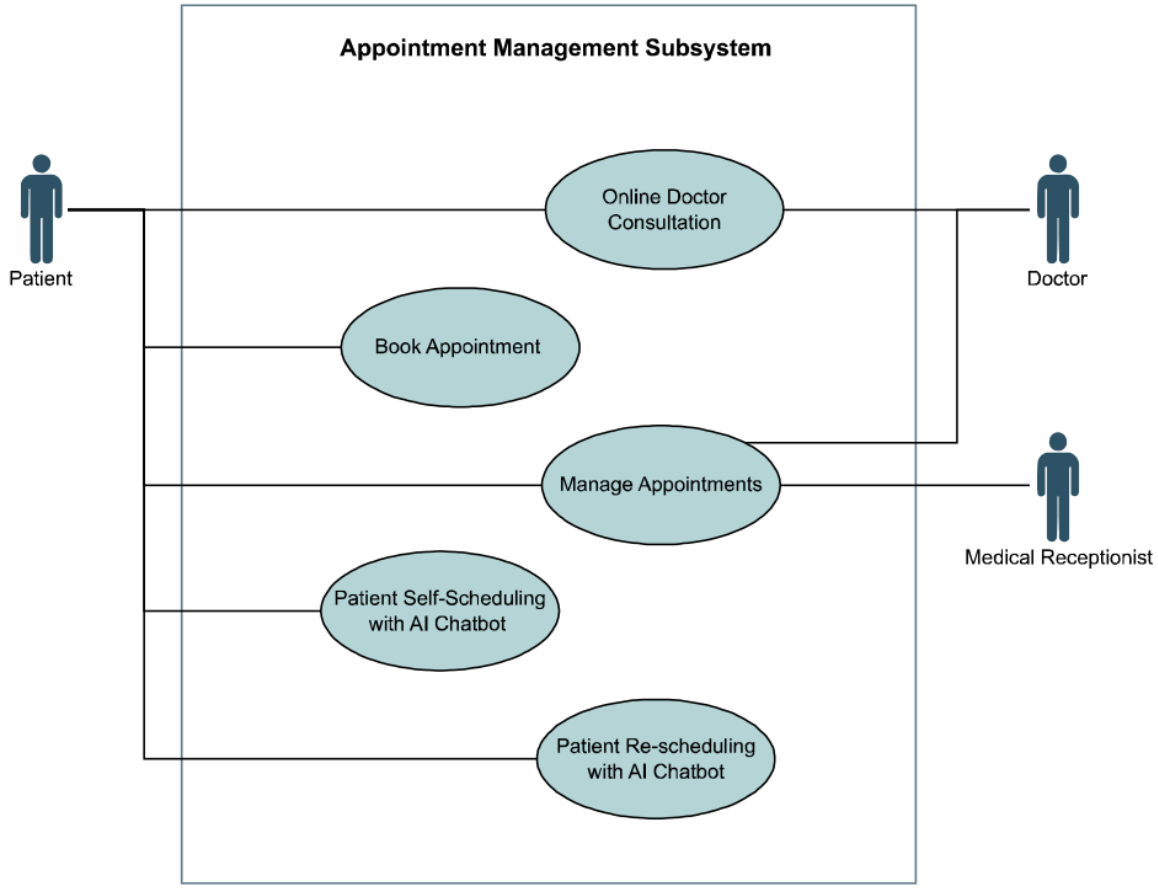
- Clicking the "Forgot Password" link from the login page redirects the user to a password recovery page.
- The password recovery page must include a field for entering the registered email address.

- Upon submission of the email address, the system sends a password reset link to the user's registered email.
- The user can set a new password using the link provided in the email.

3.1.2. Appointment Management Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Book Appointment	Patient, Medical Receptionist	Patient searches for doctors by name, specialty or location, views profiles, and selects available time slots. The system ensures real-time availability by cross-referencing the doctor's schedule, which is managed and updated by the medical receptionist to avoid conflicts.
FR02	Conduct Online Consultation	Patient, Doctor	Patients can opt for online consultations, facilitating remote medical advice and reducing the need for physical visits.
FR03	Manage Appointments	Doctor, Patient, Medical Receptionist	Doctor, patient, and medical receptionist view, cancel, or reschedule appointments, receiving notifications for any changes made.
FR04	Schedule with AI	Patient, AI	Patients interacts with an AI chatbot to schedule a medical appointment based on preferences and availability.
FR05	Re-Schedule with AI	Patient, AI	Patient uses the AI chatbot to reschedule an existing appointment, receiving updated notifications.

Use case Diagram:



User Story:

1. **Book Appointment**

   As a Patient,

   I want to book an appointment with a doctor

   so that I can receive the medical attention I need at a convenient time.

Acceptance Criteria:

- The patient can search for doctors by name, specialty, or location.
- The system displays available time slots based on the doctor's schedule, which is managed and updated by the medical receptionist thru the system.
- The patient can select an available time slot that is verified to be available, accounting for cancellations or rescheduling managed by the medical receptionist.

- The system prevents double-booking by cross-referencing the doctor's real-time appointment data.
- Upon booking, the appointment is confirmed and added to both the patient's and doctor's calendars.

2. Conduct Online Consultation

As a Patient,

I want to receive consultations online

so that I can get advice without needing to visit the clinic in person.

Acceptance Criteria:

- The patient can choose to have an online consultation when booking an appointment.
- The system provides a secure video or chat interface for the consultation.
- The patient receives confirmation of the online consultation with all necessary details (date, time, access link).
- Any prescribed medication is noted and prepared for pickup or delivery.

3. Manage Appointments

As a Doctor/Patient/Medical Receptionist,

I want to manage my appointments

so that I can keep track of my schedule and make necessary changes if needed.

Acceptance Criteria:

- Users can view all appointments in a list.
- Users can cancel or reschedule appointments easily.
- Notifications are sent for any changes made to appointments.

4. Schedule with AI

As a Patient,

I want to schedule an appointment using an AI chatbot

so that I can quickly find and book a suitable time without hassle.

Acceptance Criteria:

- The chatbot asks for the patient's preferences.
- The chatbot displays available slots and books the appointment.
- The patient receives a confirmation message.

5. Reschedule with AI

As a Patient,

I want to reschedule my appointment using the AI chatbot

so that I can easily find a new time that works for me without hassle.

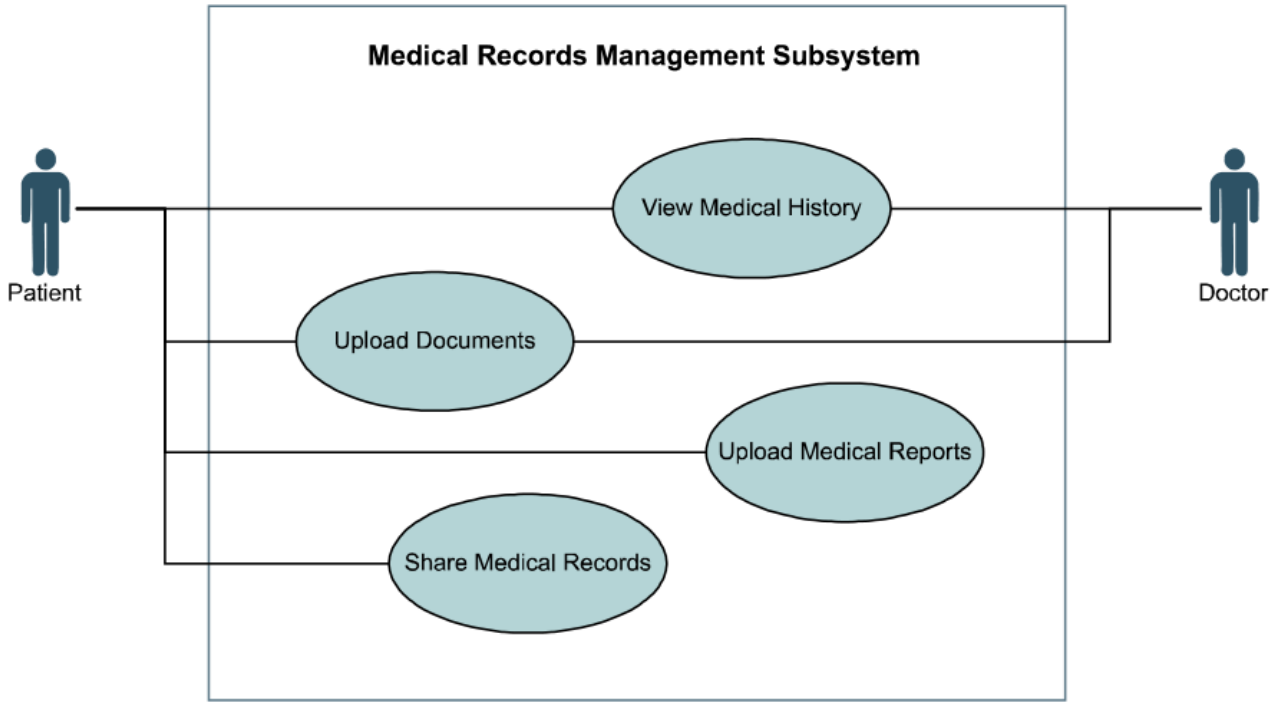
Acceptance Criteria:

- The chatbot retrieves the patient's current appointment.
- The chatbot suggests alternative time slots.
- The rescheduled appointment is confirmed and updated in the system.

3.1.3. Medical Records Management Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Upload Documents	Patient, Doctor, Medical Staff	Doctors and Medical staff can upload prescriptions and reports; patients can view them and receive copies thru email.
FR02	View Medical History	Patient, Doctor	Both patient and doctor can access the complete medical history, including previous visits and prescriptions.
FR03	Share Medical Records	Patient	Patients share their medical records with other healthcare providers thru various methods.
FR04	Upload Medical Records	Patient	Patients can upload medical reports for doctor review prior to their consultations.

Use case Diagram:



User Story:

1. **Upload Documents**

   As a Doctor/Medical Staff,

   I want to upload medical documents to the system

   so that they can be easily accessed by the patient for future reference.

Acceptance Criteria:

- The doctor or medical staff can select and upload medical documents to the patient's record.
- The uploaded documents are securely stored in the patient's medical records and made available for the patient to access.
- The patient receives a confirmation email.

2. **View Medical History**

   As a Patient,

   I want to view my medical history

   so that I can keep track of past visits and treatments.

Acceptance Criteria:

- The user can access a complete medical history page.
- The history includes previous visits, treatments, and prescriptions.
- The information is displayed in an easily accessible and readable format.

3. Share Medical Records

As a Patient,

I want the system to securely share my medical records via email

so that I can facilitate my care with other healthcare providers while ensuring privacy and security.

Acceptance Criteria:

- The system automatically sends the selected records via a secure, encrypted email.
- Recipients can access the shared records only through an encrypted link in the email, with proper authentication for security.
- Requires the recipient to authenticate their identity (e.g., using a multi-factor authentication process) and has a time-limited expiration to restrict access after a set period.

4. Upload Medical Reports

As a Patient,

I want to upload my medical reports

so that my doctor can review them before my consultation.

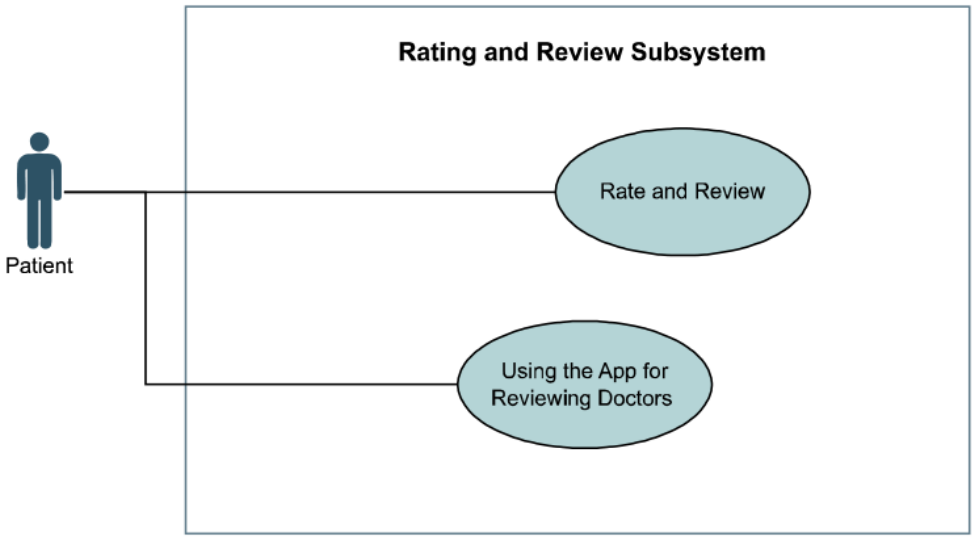
Acceptance Criteria:

- The patient can easily select and upload files.
- The doctor can access the uploaded reports in their system.
- The patient receives confirmation of the upload.

3.1.4. Rating and Review Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Rate and Review	Patient	After an appointment, patients can rate the service and leave reviews on the doctor's profile page.
FR02	Review Doctors	Patient	Patients can use the app to view doctor ratings and reviews before scheduling.

Use case Diagram:



User Story:

1. **Rate and Review**

   As a Patient,

   I want to rate and review my doctor

   so that I can share my experience and help others make informed decisions.

Acceptance Criteria:

- The patient can leave a rating and write a review after an appointment.
- The review is visible on the doctor's profile page.
- The patient receives confirmation that their review has been submitted.

2. Review Doctors

As a Patient,

I want to see ratings and reviews of doctors

so that I can choose the best healthcare provider for my medical needs.

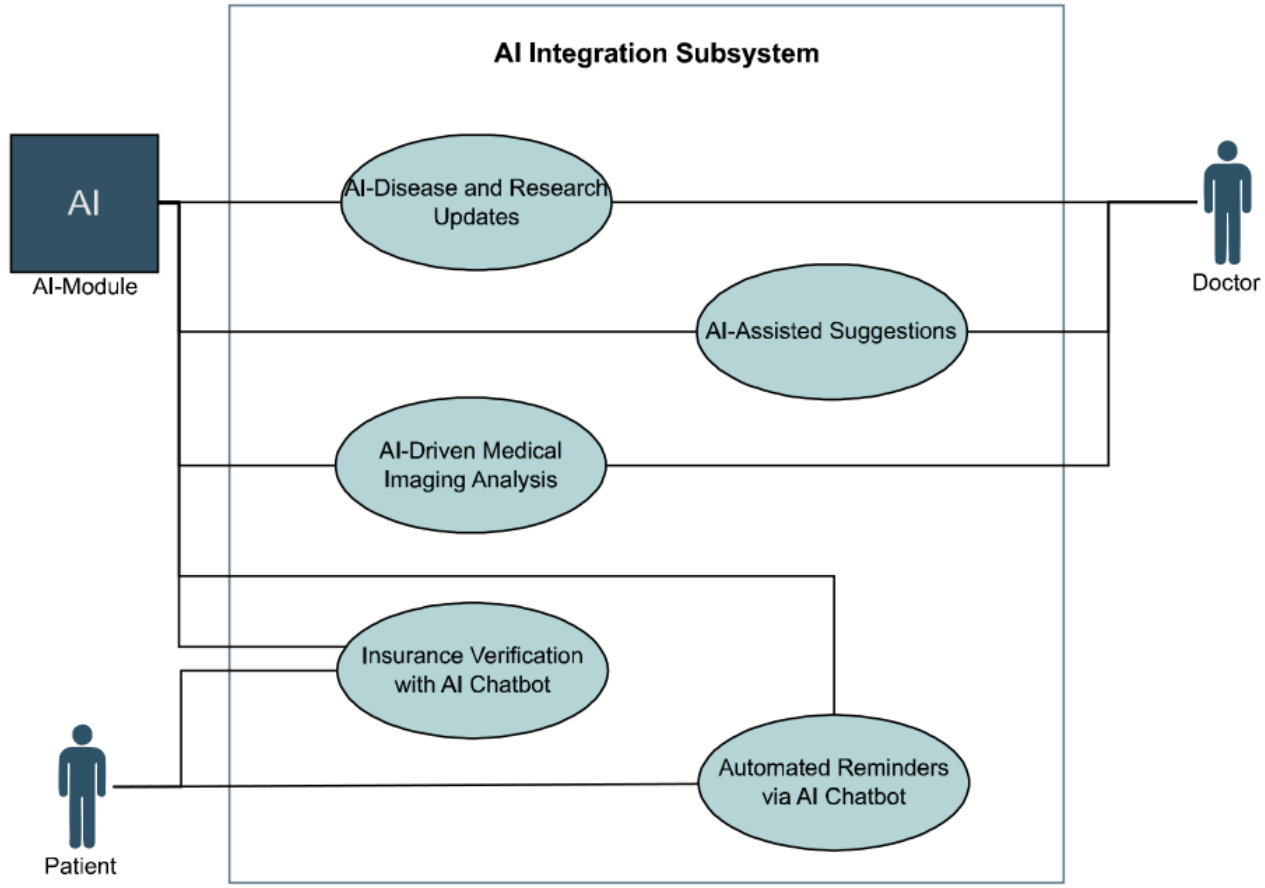
Acceptance Criteria:

- The patient can access a list of doctors with their ratings and reviews.
- Reviews include details and feedback from other patients.
- The information is updated in real-time.

3.1.5. AI Integration Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Update Medical Information	Doctor, AI	The AI provides up-to-date information on diseases and research findings to support doctors.
FR02	Suggest Treatments	Doctor, AI	The AI suggests treatments based on patient data and current research trends.
FR03	Analyze Medical Imaging	Doctor, AI	The AI analyzes medical images (e.g., X-rays, MRIs) to provide diagnostic support to doctors.
FR04	Send Automated Reminders	Patient, AI	The AI chatbot sends automated appointment reminders to patients via their preferred communication channel.
FR05	Verify Insurance Coverage	Patient, AI	Patients use the AI chatbot to verify their insurance coverage and understand co-pays.

Use case Diagram:



User Story:

1. **Update Disease and Research Information**

   As a Doctor,

   I want to receive updates on diseases and research

   so that I can stay informed and provide the best care.

Acceptance Criteria:

- The AI system delivers timely updates on relevant medical research.
- Doctors can access and review this information easily.
- Notifications are sent for significant updates.

2. Suggest Treatments

As a Doctor,

I want to receive AI-assisted treatment suggestions

so that I can make informed decisions based on the latest trends.

Acceptance Criteria:

- The AI provides suggestions based on the patient data.
- The doctor can review and accept or modify the suggestions.
- Suggestions are documented in the patient's record.

3. Analyze Medical Imaging

As a Doctor,

I want to receive AI analysis of medical images

so that I can improve diagnostic accuracy and speed.

Acceptance Criteria:

- The AI analyzes images and provides a report.
- The doctor can access the analysis alongside the images.
- Feedback on the analysis is available for the doctor.

4. Send Automated Reminders

As a Patient,

I want to receive automated appointment reminders

so that I don't forget my upcoming appointments.

Acceptance Criteria:

- The AI chatbot sends reminders via the patient's preferred communication method.
- Reminders include date, time, and location of the appointment.
- The patient can confirm or reschedule directly from the reminder.

5. Verify Insurance Coverage

As a Patient,

I want to verify my insurance coverage through the AI chatbot

so that I can understand my costs before booking an appointment.

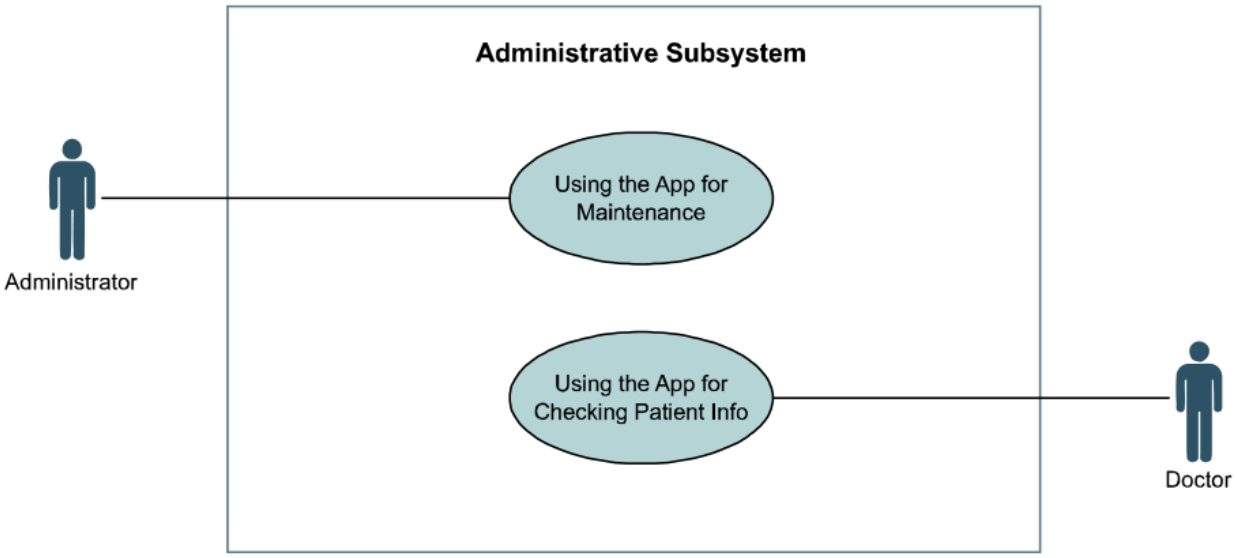
Acceptance Criteria:

- The chatbot accesses the patient's insurance information.
- The patient receives a clear breakdown of coverage and costs.
- The information is provided in real-time.

3.1.6. Administrative Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Maintain System	Administrator	Administrators use the app to manage system maintenance and monitor performance.
FR02	Check Patient Information	Doctor	Doctors check patient information frequently to ensure accurate diagnoses and treatment plans.
FR03	Manage Notifications	Patient, Doctor, Medical Receptionist, Administrator	The system sends notifications to users regarding appointment confirmations, reminders, and important updates.

Use case Diagram:



User Story:

1. **Maintain System**

   As an Administrator,

   I want to manage the Queue.Me's maintenance

   so that I can ensure smooth operation and address any issues.

Acceptance Criteria:

- The administrator can access maintenance tools.
- Reports on system performance are available.
- Notifications for issues are sent to the administrator.

2. **Check Patient Information**

   As a Doctor,

   I want to check patient information easily

   so that I can provide timely and accurate care.

Acceptance Criteria:

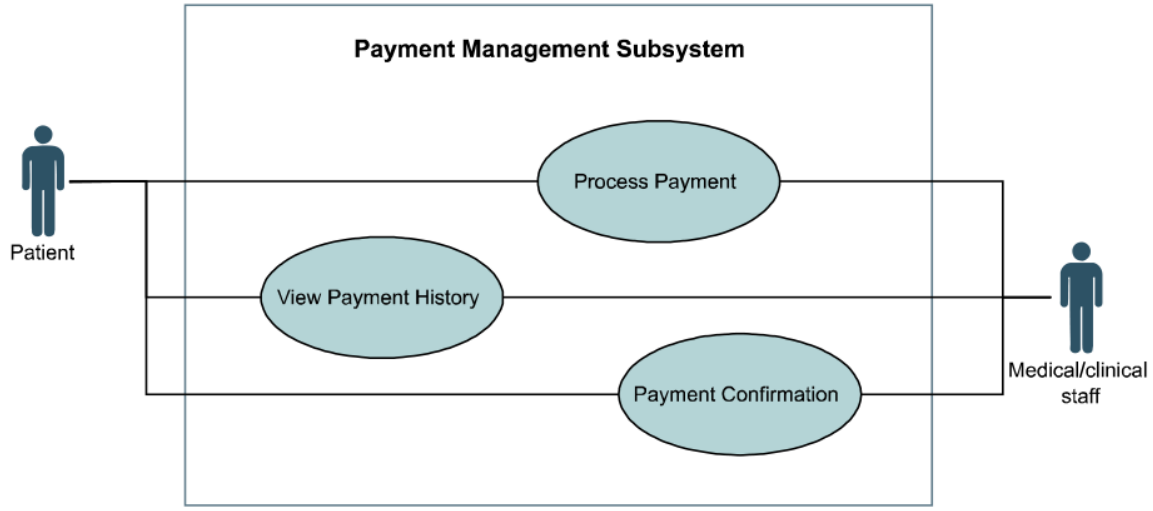
- The doctor can access a patient's complete profile quickly.

- The information includes medical history, current treatments, and prescriptions.
- The system allows for easy navigation between different patient profiles.

3.1.7. Payment Management Subsystem

FR #	Goal Use Case	Role Player	Description
FR01	Process Payment	Patient, Medical/clinical Staff	Patients can enter payment information to complete the transaction for their appointments. While medical/clinical staff manage and verify the payment process.
FR02	View Payment History	Patient, Medical/clinical Staff	Patients and medical/clinical staff can view a history of past payments for services rendered.
FR03	Confirm Payment	Patient, Medical/clinical Staff	After a successful transaction, Queue.Me sends a confirmation message or email to the patient, and clinical staff monitor and confirm payment records.
FR04	Verify Payment	Patient	The system verifies the validity of the payment method (e.g., credit card) before processing the payment to reduce transaction errors.

Use case Diagram:



User Story:

1. **Process Payment**

   As a Patient,

   I want to securely process my payment for appointments

   so that I can confirm my booking.

Acceptance Criteria:

- The payment page must be accessible after booking an appointment.
- Users must be able to select a saved payment method or enter a new payment information.
- Upon submitting payment, the system securely processes the transaction and displays a confirmation message.
- Medical/clinical staff can access the details to verify and manage the transaction.
- Users receive an email receipt confirming the payment and appointment details.

2. **View Payment History**

   As a Patient,

   I want to view my payment history

   so that I can keep track of my expenses related to appointments.

Acceptance Criteria:

- The payment history page must be accessible from the user's profile.
- The page must display a list of past transactions, including date, amount, and appointment details.
- Medical/clinical staff can view patient payment history for record-keeping.
- Users must be able to filter the payment history by date range or appointment type.
- Users can click on a transaction to view detailed information.

3. Confirm Payment

As a Patient,

I want to receive a confirmation of my payment

so that I can have a record of the transaction.

Acceptance Criteria:

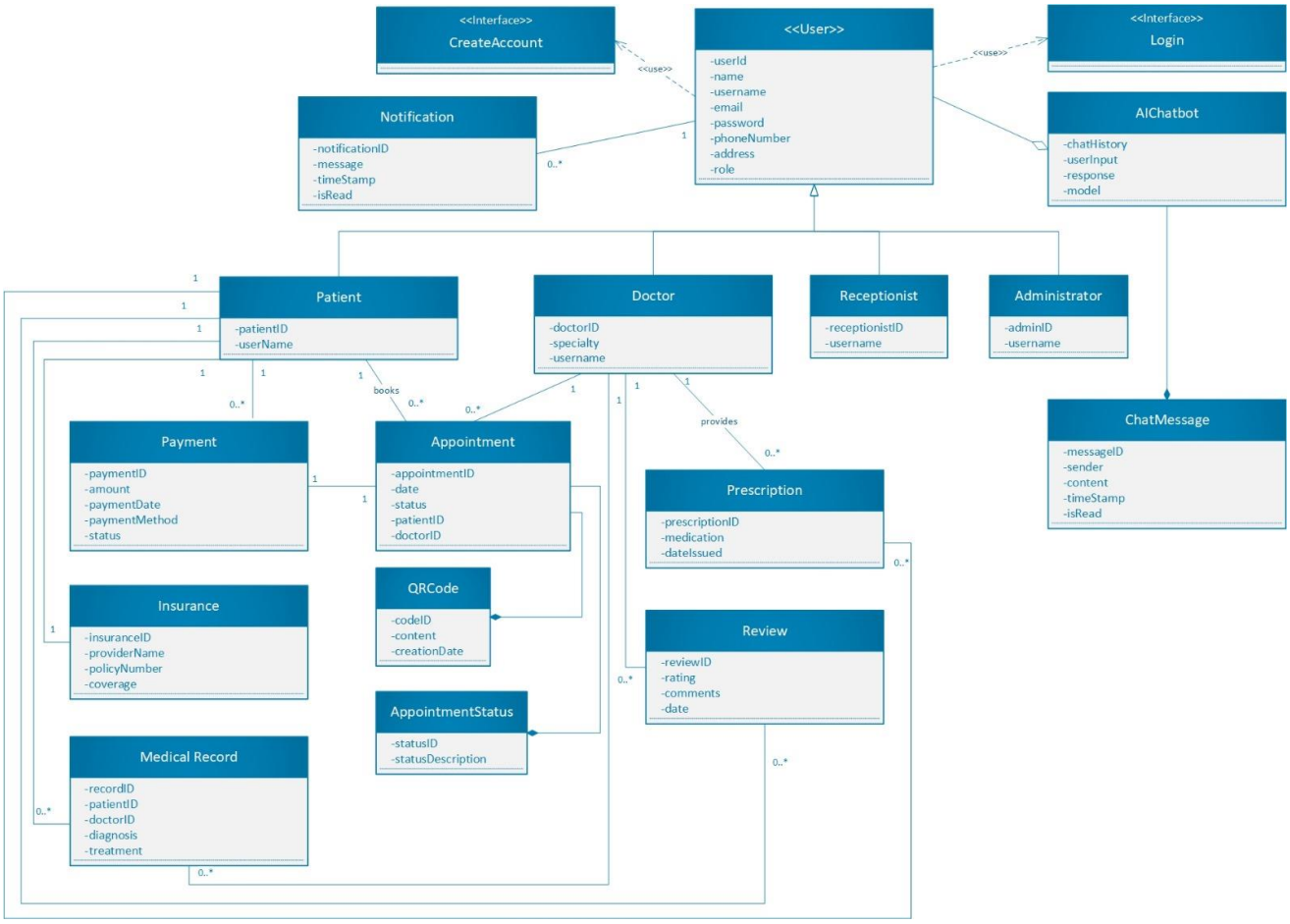
- Upon successful payment processing, Queue.Me displays a confirmation message on the screen indicating that the payment was successful.
- An email receipt must be sent to the patient's registered email address, detailing the payment amount, appointment date and time, and any other relevant information.
- Medical/clinical staff can confirm payment details.
- The payment confirmation details must be accessible in the patient's account under the payment history section.
- The confirmation message must be displayed immediately after payment processing, and the email should be sent within a few minutes.
- If there is an issue sending the email, the system should display a warning message and log the error for administrative review.

Section 4: UML Domain Class Diagram

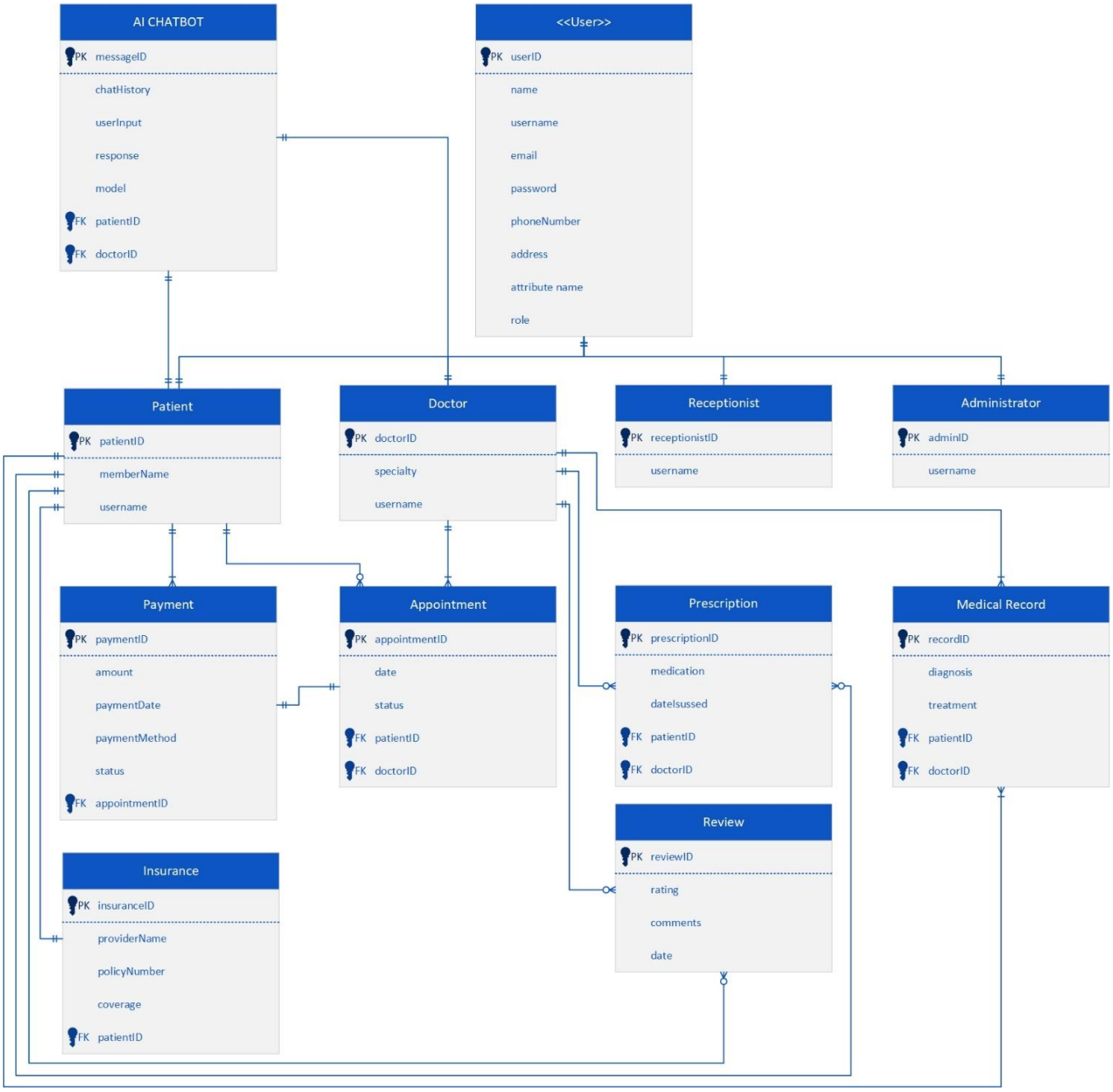
4.1. List of Classes

• User • Patient • Doctor • Receptionist • Appointment • MedicalRecord • Prescription • Medication	• Review • Payment • QRCode • Notification • Virtual Consultation • AppointmentStatus

4.2. Domain Class Diagram

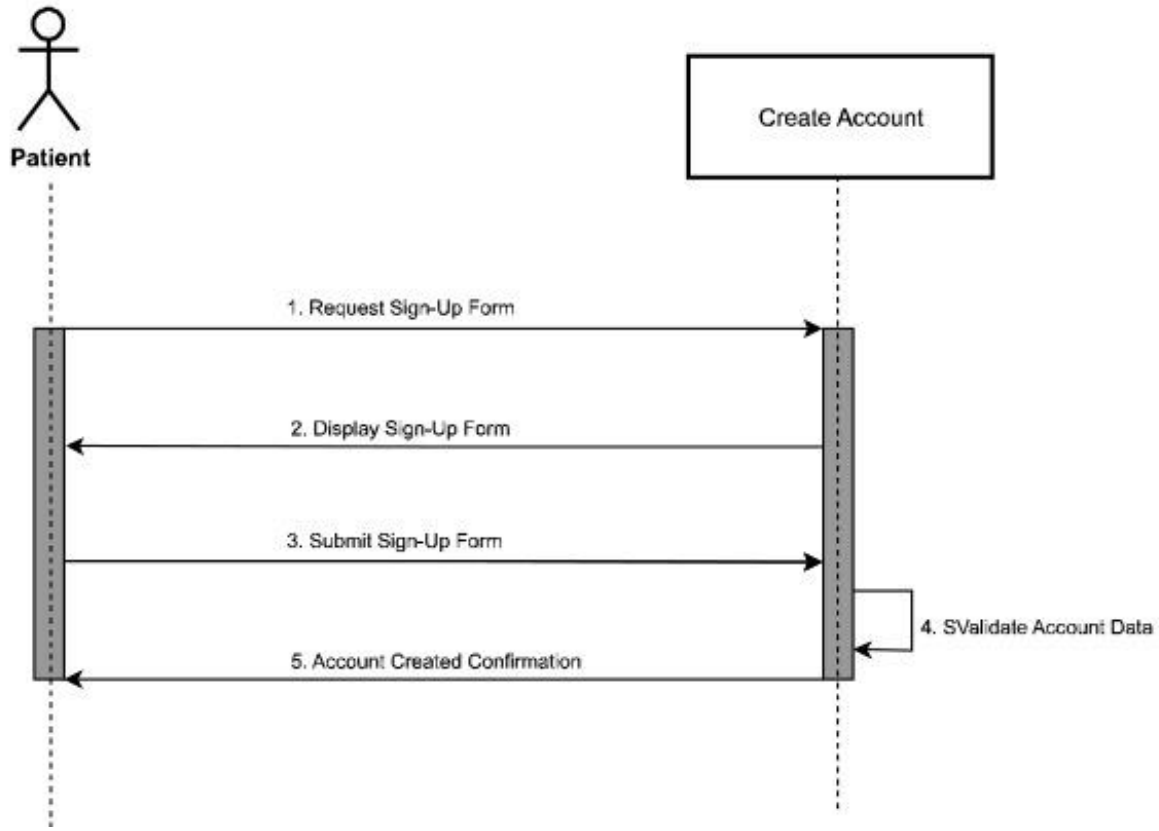


Section 5: Entity Relationship Diagram (ERD)

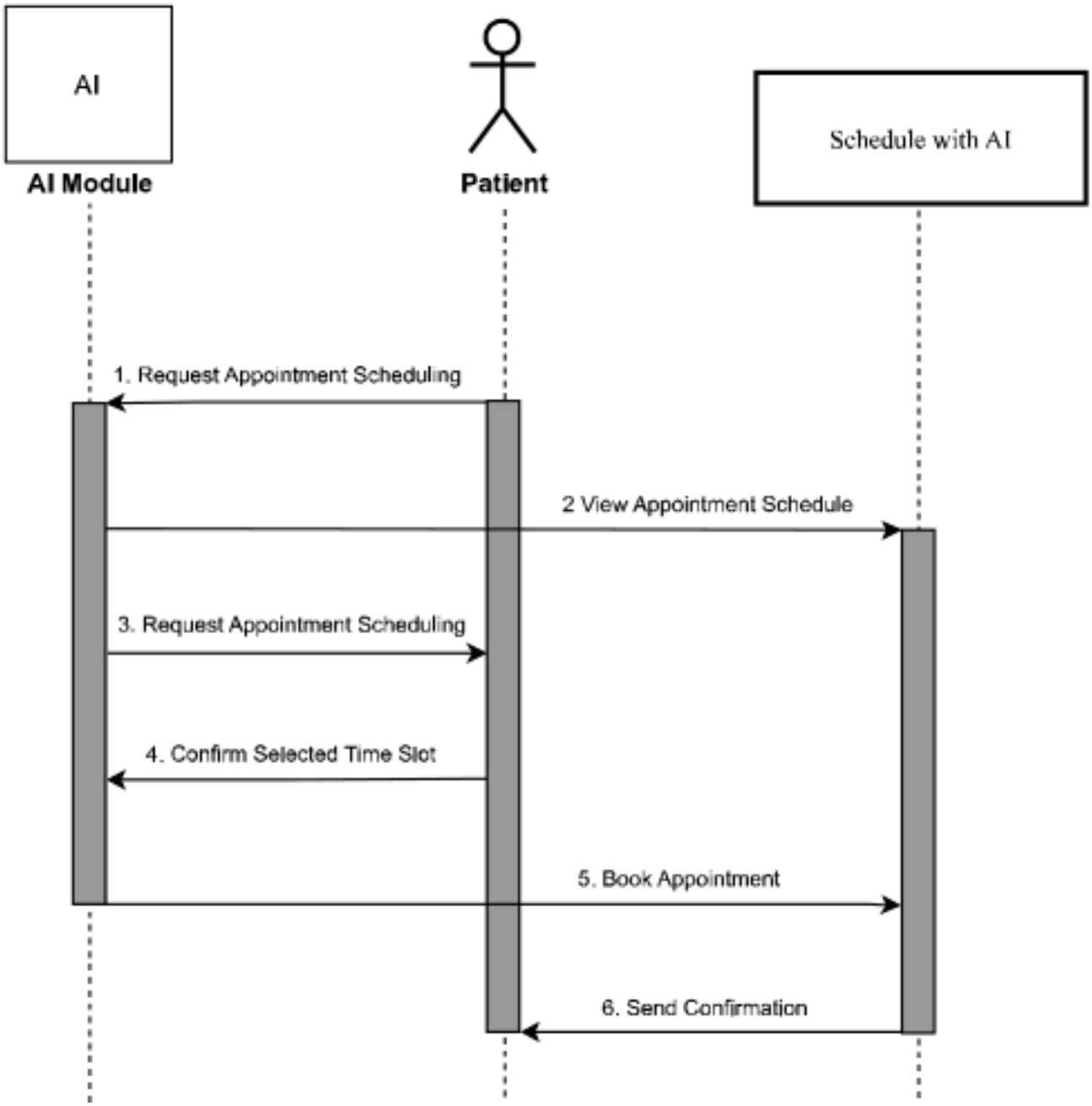


Section 6: UML Systems Sequence Diagrams

6.1. Sequence Diagram: Create Account Use Case (User Management Subsystem)



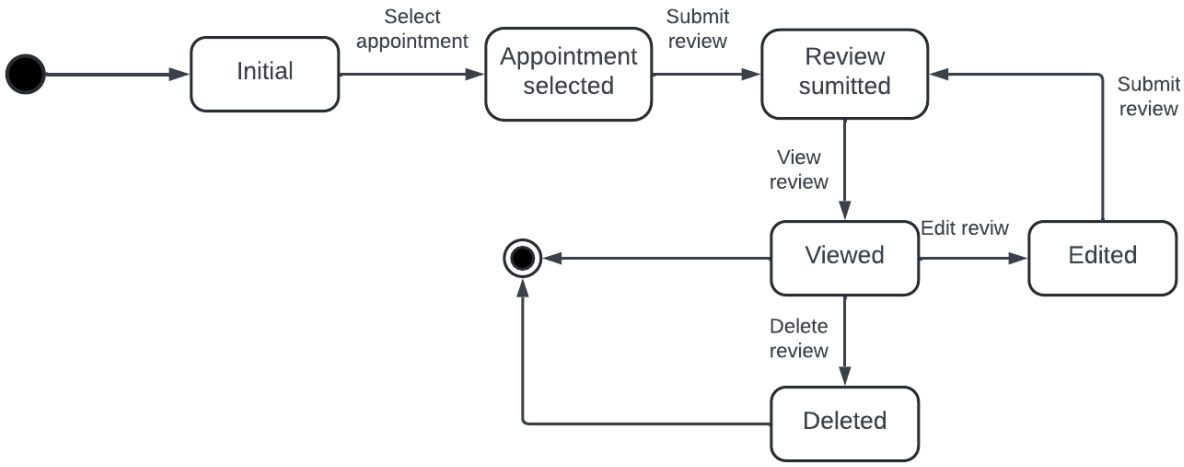
6.2. Sequence Diagram: Scheduling with AI Use Case (Appointment Management Subsystem)



Section 7: Object State Machine Diagrams

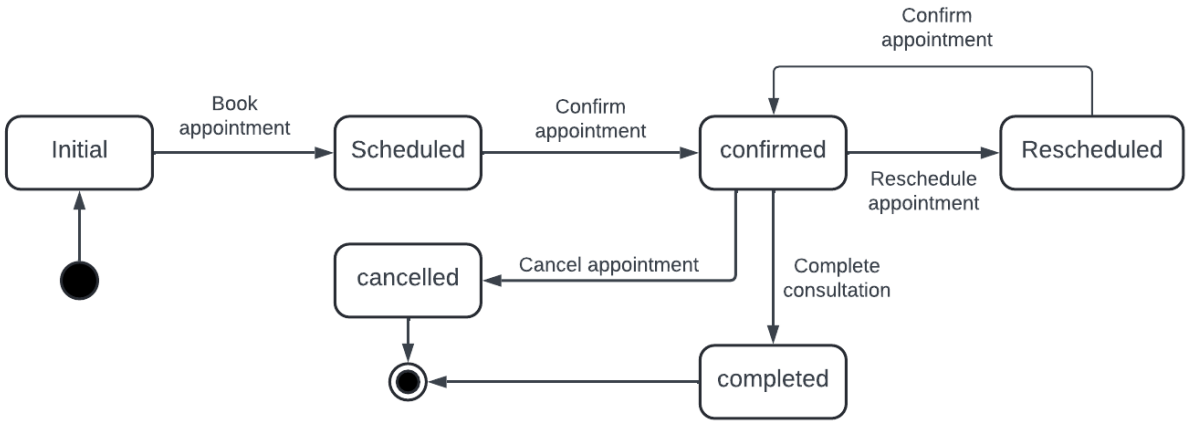
7.1 Review State Diagram

Class: Review



7.2 Appointment State Diagram

Class: Appointment



Section 8: Technologies

The development of the **Queue.Me healthcare system** involves building both mobile and web applications, utilizing a range of tools and technologies at various layers of the application stack. Below is a combined overview of the development approach, detailing the tools used for both types of applications:

Mobile Application

- **Client Side (Front-end – GUI):**

  React Native: Using a single codebase, developers can create native mobile apps for iOS and Android with this JavaScript framework. It lowers development time and costs and guarantees a smooth user experience on both platforms.

- **Business Logic (Middle Layer – Class Methods):**

  Node.js with Express.js: This stack handles request management, back-end communication, and app logic. Large datasets can be handled by Node.js, which makes it perfect for real-time applications in the healthcare industry where instantaneous data access is crucial.
  Tools/Resources: Node.js for Mobile App Backend

- **Back End (Database):**

  Oracle DB: This robust database system is used to manage patient data, medical histories, appointments, and doctor information, ensuring scalability and reliability.
  Tools/Resources: Oracle Database Documentation

Web Application

- **Client Side (Front-end – GUI):**

  React.js: This JavaScript library is ideal for creating responsive and dynamic user interfaces, particularly for single-page applications. Fast, seamless, and effective

interaction is guaranteed by React.js, which is essential for web-based healthcare management systems.

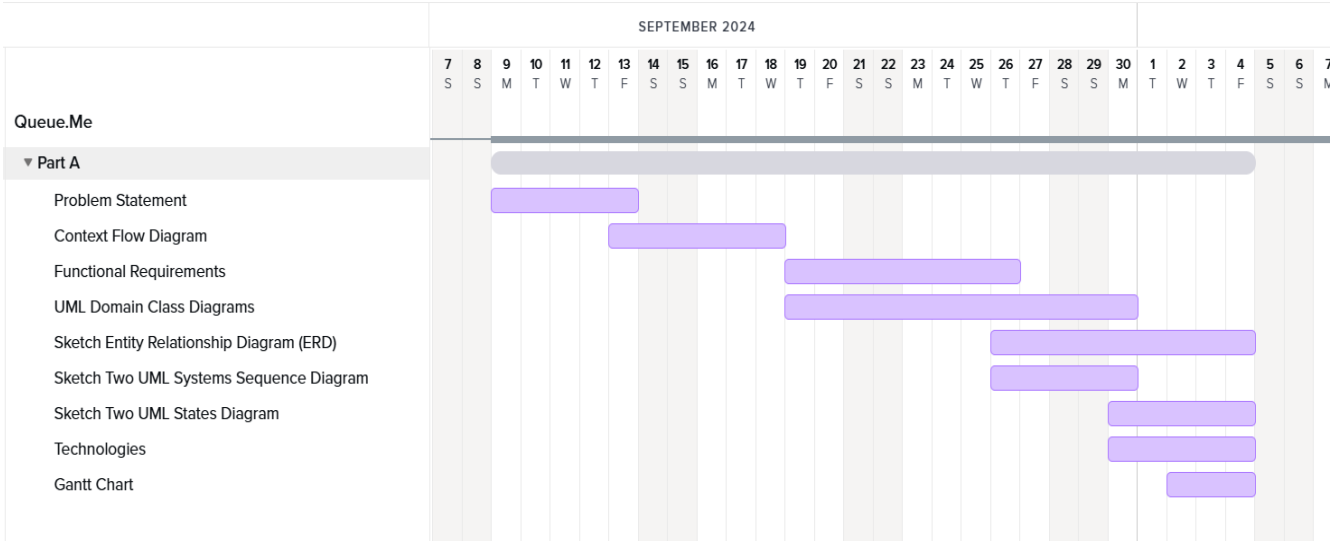
- **Business Logic (Middle Layer – Class Methods):**

  Node.js with Express.js: Node.js handles queries from the web client and handles business logic and database interactions, same as in the mobile version.
  Tools/Resources: Express.js for Web Applications

- **Back End (Database):**

  Oracle DB: The web application, like its mobile counterpart, stores and manages vital information like medications, appointment details, and patient records using Oracle Database.
  Tools/Resources: Oracle for Web Applications

Section 9: Project Management



Part B: Software Design Architecture

Section 1: Requirements to Edits to Part A

1.1. Correction on Section 1.1b – List of capabilities and benefits

- Added the list of capabilities and benefits.

1.2. Correction on Section 2 – Context Flow Diagram

- Updated the Context Flow Diagram by merging the AI module with Queue.Me.

1.3. Correction on Section 3.1 – Goal Use Cases List

- Updated the goal use cases list to use the "Verb-Noun" format for nomenclature in use cases.

1.4. Correction on Section 3 – Goal Use Cases List (3.1)

- Added Abstract Uses Cases. Also, the Login, create account and forgot password has been added in the User Management subsystem previously.

1.5. Correction on Section 3 - Use Case Diagrams (3.2) Payment Management Subsystem

- Updated the Use Case Diagram on the Payment Management Subsystem.
- Rather than the doctor, it is the medical staff/receptionist who deals with the processing of payments.

1.6. Correction on Section 3.3 – User Stories for each Sub-system

- Show the use case goal list in this section as per feedback.
- Updated use case 1.2. Manage Profile, to also mention Patient along with the Doctor.

1.7. Correction on Section 3 – User Stories (3.3) Appointment Management Subsystem

- Updated the Book Appointment use case and user story to have a connection with Manage

Appointments/Medical Receptionist to show the empty slots for the particular doctor.

1.8. Correction on User Story 3.3. of Share Medical Records

- Updated the user story for the Share Medical Records

1.9. Correction on Domain Class Diagram

- Updated the Domain Class Diagram

1.10. Correction on Section 5 – Entity Relationship Diagram No PK shown for AI chatbot table

- Added PK for AI Chatbot table
- Updated Entity Relationship Diagram

1.11. Correction on Section 6.0 – System Sequence Diagram

- Updated the two System Sequence Diagram and show the goal use cases instead of the subsystems.

1.12. Correction on Section 7.0 – Sketch Two UML State Diagrams

- Updated the two UML State Diagrams and show two classes from the domain class diagram.

1.13. Correction on Section 9 – Project Management

- Updated the Gantt Chart to more professional as per feedback.

Section 2: Overview Model

2.1. Intended users of the SDD document

This document for the Queue.Me system is intended for a variety of users involved in the project.

- Software Developers - Developers are the primary users of the SDD for implementation guidance. They use it as a technical reference to understand the system's architecture, design specifications, and component interactions.

  Tasks: They focus on functional and non-functional requirements, ensuring that each module is developed as intended. The SDD provides detailed diagrams, class structures, and interface descriptions, which developers use to write code that aligns with the documented design.

- Project Managers (PM) - Project managers review the SDD to gain an understanding of the system's architecture and complexity, which helps them oversee the project lifecycle and resource allocation.

  Tasks: They use the introductory sections and high-level descriptions to identify deliverables, set timelines, and allocate tasks based on the workload each module represents. They also track project milestones and ensure each phase meets the documented requirements.

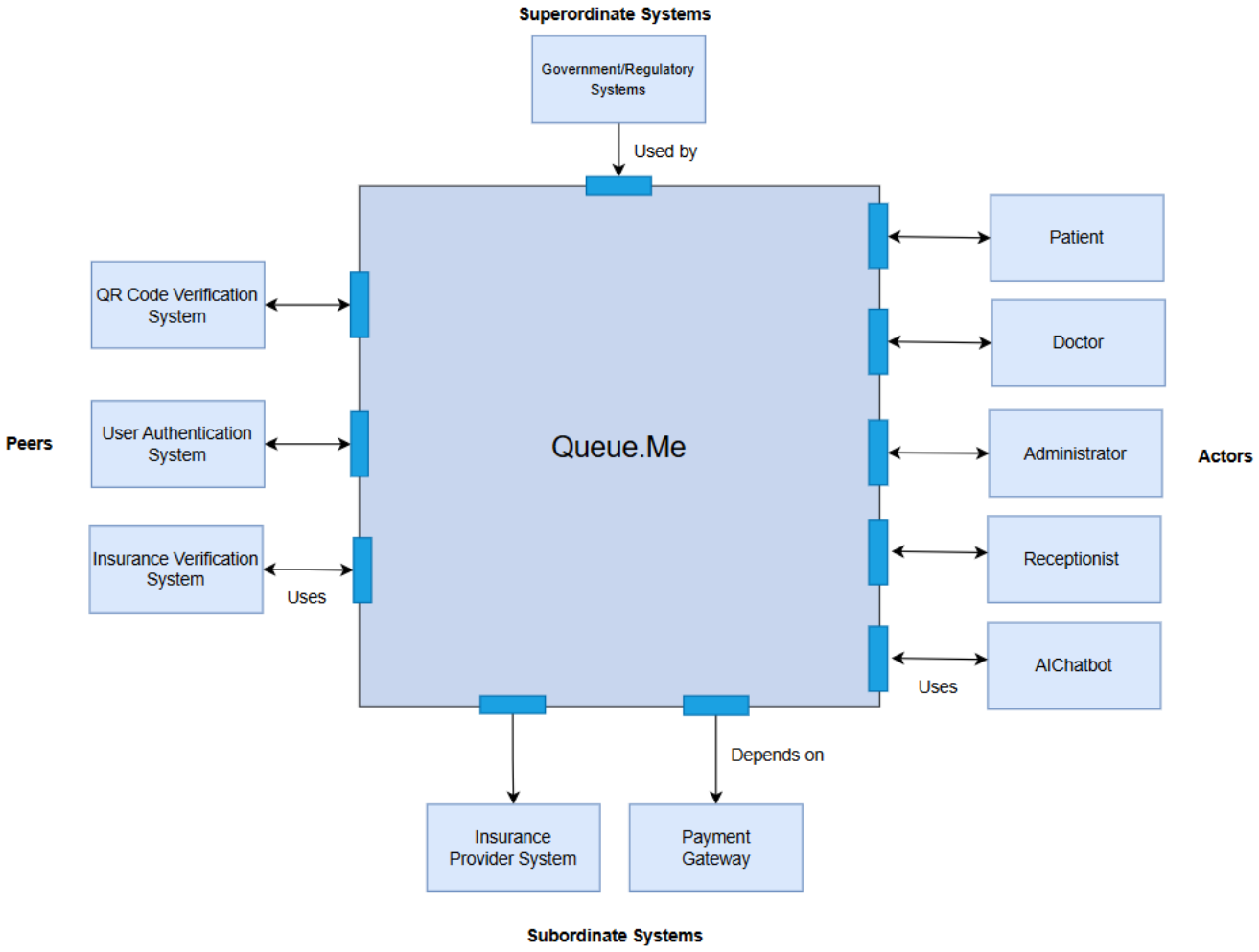
- Business Analysts/Product Owners (BA/PO) - Business analysts and product owners reference the SDD to verify that the design aligns with the product vision and stakeholder requirements.

  Tasks: BAs and POs concentrate on the requirements sections, making sure that business needs are accurately represented and that the design will deliver value to end-users. They communicate feedback from stakeholders and may suggest design adjustments if requirements evolve.

- Software Quality Assurance (QA) Engineers - QA engineers use the SDD to develop test plans and validation criteria, ensuring that the system meets specified requirements.

  Tasks: They focus on functional requirements and system features to understand expected behaviors and edge cases. Using this, they create test cases that cover all critical paths and potential issues, thereby ensuring the software is robust and free of defects.

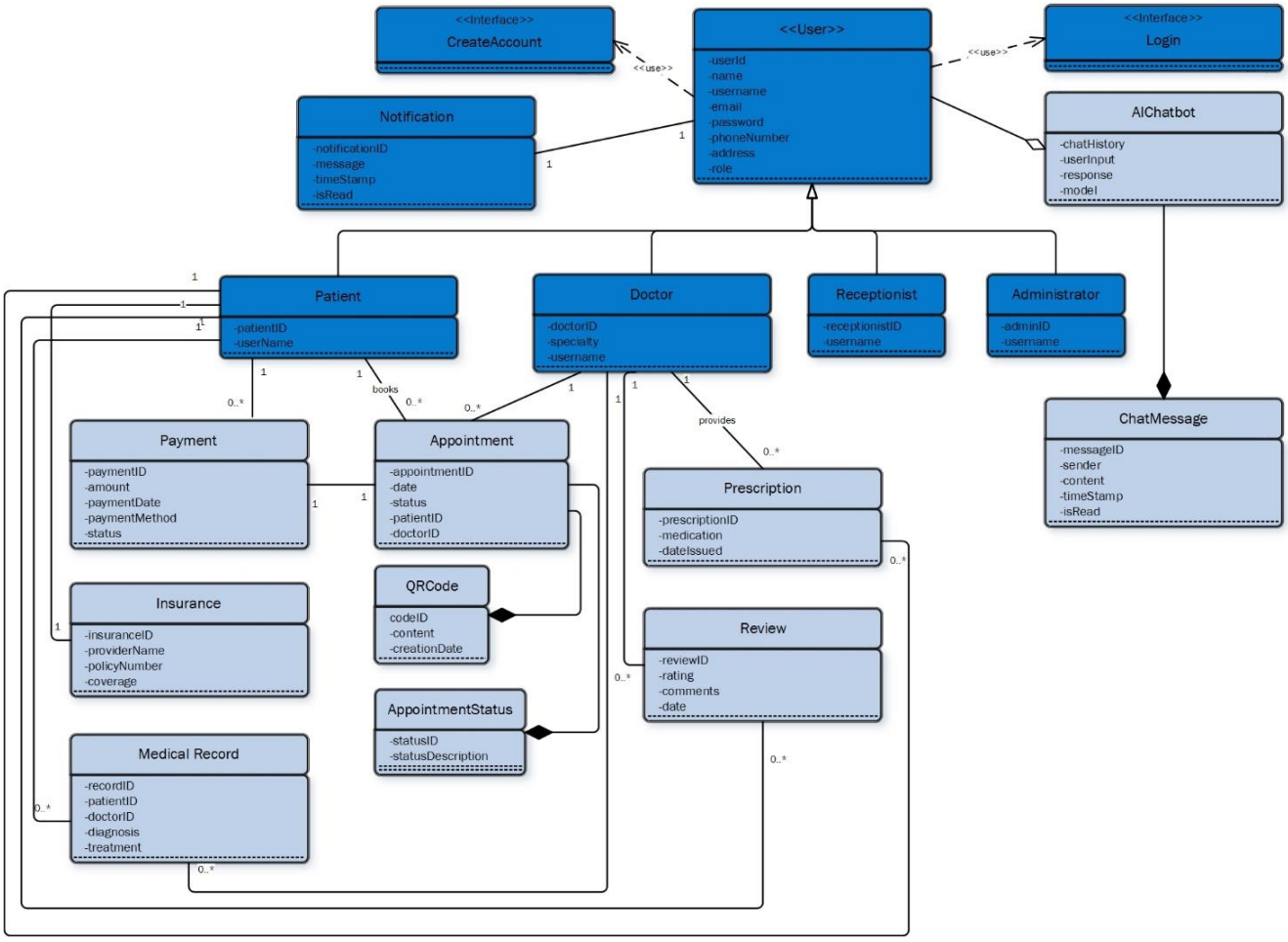
2.2. Architectural Context Diagram (ACD)



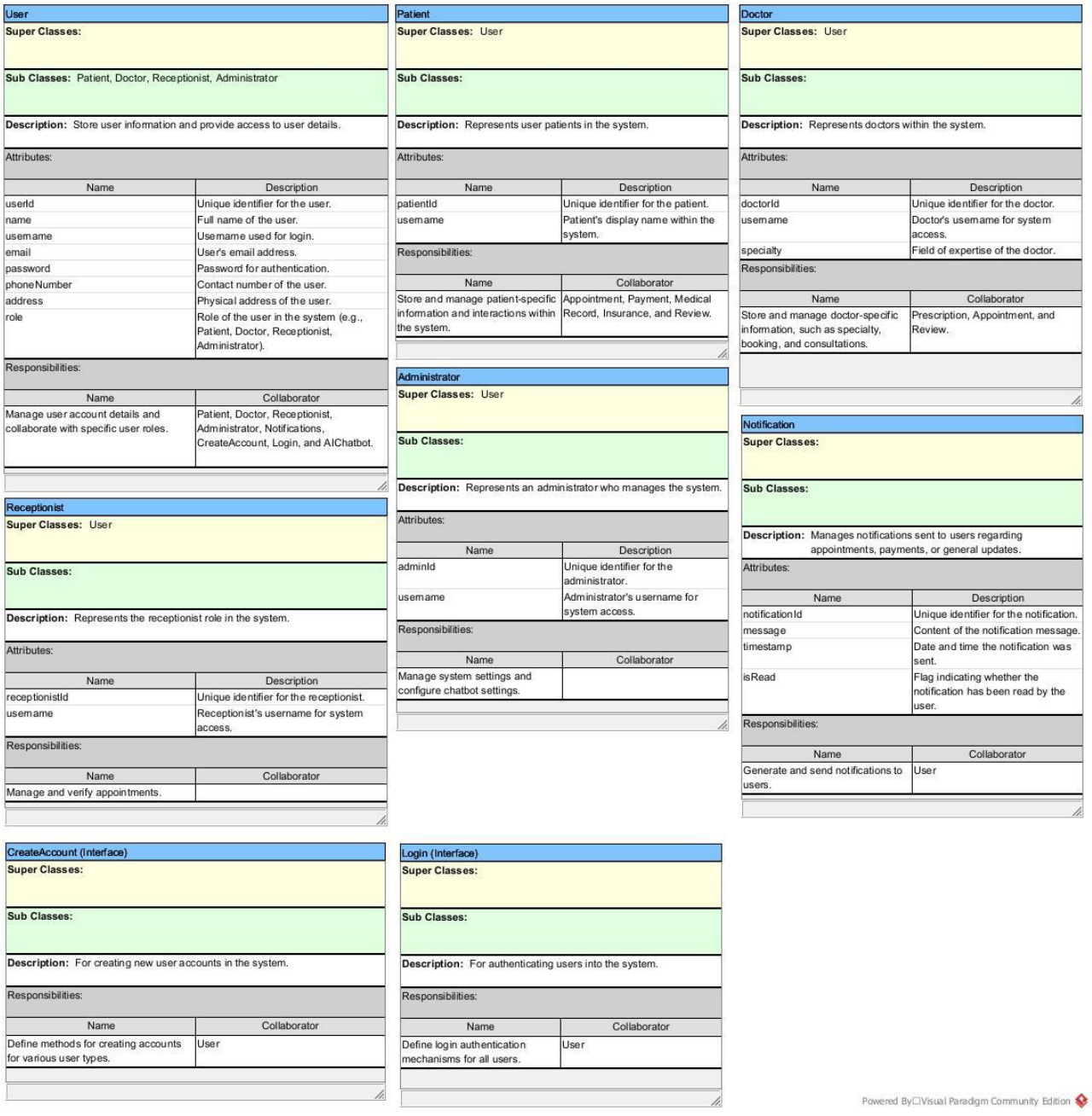
Section 3: Modularization

3.1. User Management Subsystem

3.1.1. Partition of Analysis Model



3.1.2. Class Responsibility Collaboration Cards (CRC cards)



User

Super Classes:

Sub Classes: Patient, Doctor, Receptionist, Administrator

Description: Store user information and provide access to user details.

Attributes:

Name	Description
userId	Unique identifier for the user.
name	Full name of the user.
username	Username used for login.
email	User's email address.
password	Password for authentication.
phoneNumber	Contact number of the user.
address	Physical address of the user.
role	Role of the user in the system (e.g., Patient, Doctor, Receptionist, Administrator).

Responsibilities:

Name	Collaborator
Manage user account details and collaborate with specific user roles.	Patient, Doctor, Receptionist, Administrator, Notifications, CreateAccount, Login, and AIChatbot.

Patient

Super Classes: User

Sub Classes:

Description: Represents user patients in the system.

Attributes:

Name	Description
patientId	Unique identifier for the patient.
username	Patient's display name within the system.

Responsibilities:

Name	Collaborator
Store and manage patient-specific information and interactions within the system.	Appointment, Payment, Medical Record, Insurance, and Review.

Doctor

Super Classes: User

Sub Classes:

Description: Represents doctors within the system.

Attributes:

Name	Description
doctorId	Unique identifier for the doctor.
username	Doctor's username for system access.
specialty	Field of expertise of the doctor.

Responsibilities:

Name	Collaborator
Store and manage doctor-specific information, such as specialty, booking, and consultations.	Prescription, Appointment, and Review.

Receptionist

Super Classes: User

Sub Classes:

Description: Represents the receptionist role in the system.

Attributes:

Name	Description
receptionistId	Unique identifier for the receptionist.
username	Receptionist's username for system access.

Responsibilities:

Name	Collaborator
Manage and verify appointments.	

Administrator

Super Classes: User

Sub Classes:

Description: Represents an administrator who manages the system.

Attributes:

Name	Description
adminId	Unique identifier for the administrator.
username	Administrator's username for system access.

Responsibilities:

Name	Collaborator
Manage system settings and configure chatbot settings.	

Notification

Super Classes:

Sub Classes:

Description: Manages notifications sent to users regarding appointments, payments, or general updates.

Attributes:

Name	Description
notificationId	Unique identifier for the notification.
message	Content of the notification message.
timestamp	Date and time the notification was sent.
isRead	Flag indicating whether the notification has been read by the user.

Responsibilities:

Name	Collaborator
Generate and send notifications to users.	User

CreateAccount (Interface)

Super Classes:

Sub Classes:

Description: For creating new user accounts in the system.

Responsibilities:

Name	Collaborator
Define methods for creating accounts for various user types.	User

Login (Interface)

Super Classes:

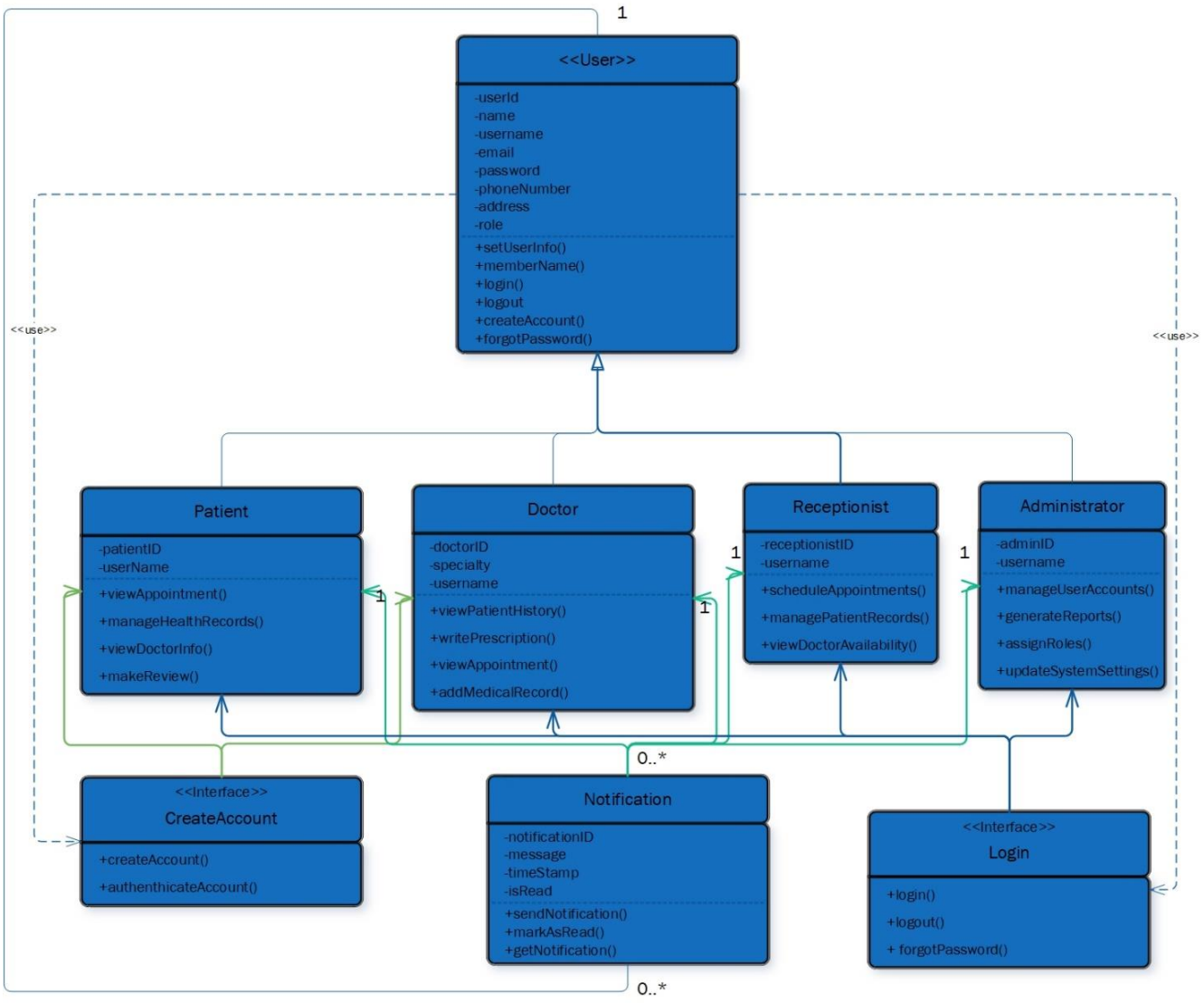
Sub Classes:

Description: For authenticating users into the system.

Responsibilities:

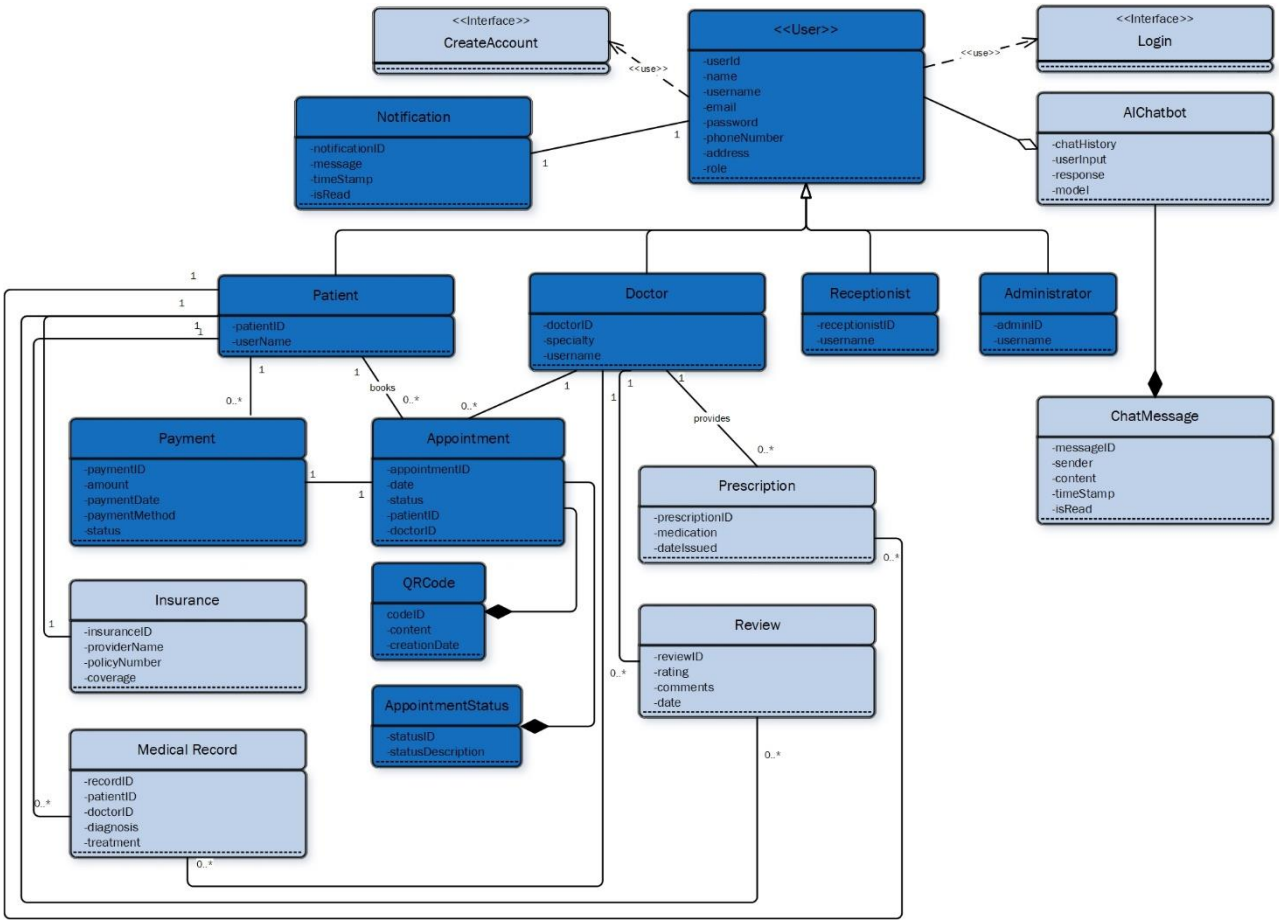
Name	Collaborator
Define login authentication mechanisms for all users.	User

3.1.3. Design Class Diagram

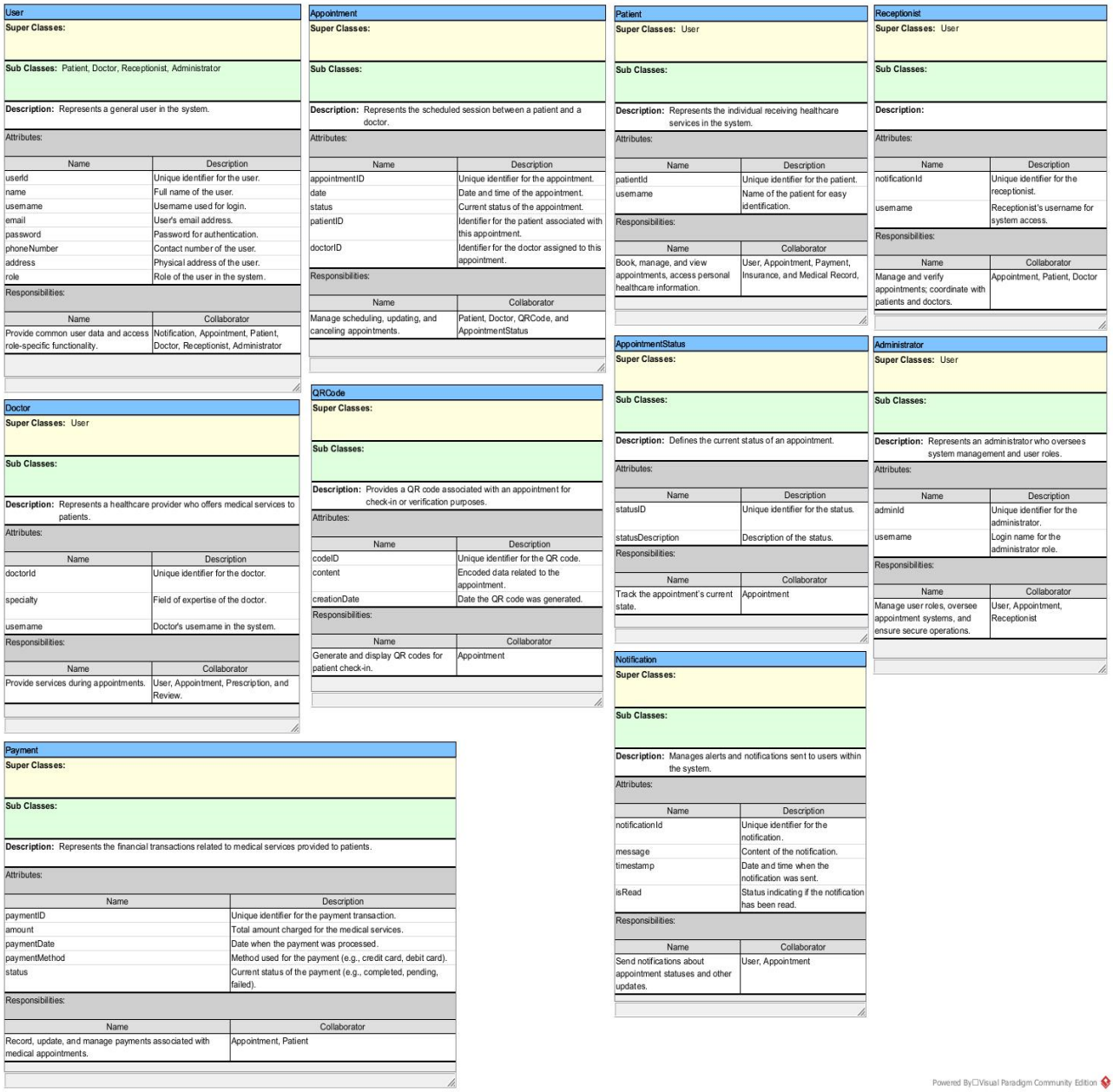


3.2. Appointment Management Subsystem

3.2.1. Partition of Analysis Model



3.2.2. Class Responsibility Collaboration Cards (CRC cards)



User

Super Classes:

Sub Classes: Patient, Doctor, Receptionist, Administrator

Description: Represents a general user in the system.

Attributes:

Name	Description
userId	Unique identifier for the user.
name	Full name of the user.
username	Username used for login.
email	User's email address.
password	Password for authentication.
phoneNumber	Contact number of the user.
address	Physical address of the user.
role	Role of the user in the system.

Responsibilities:

Name	Collaborator
Provide common user data and access role-specific functionality.	Notification, Appointment, Patient, Doctor, Receptionist, Administrator

Appointment

Super Classes:

Sub Classes:

Description: Represents the scheduled session between a patient and a doctor.

Attributes:

Name	Description
appointmentID	Unique identifier for the appointment.
date	Date and time of the appointment.
status	Current status of the appointment.
patientID	Identifier for the patient associated with this appointment.
doctorID	Identifier for the doctor assigned to this appointment.

Responsibilities:

Name	Collaborator
Manage scheduling, updating, and canceling appointments.	Patient, Doctor, QRCode, and AppointmentStatus

Patient

Super Classes: User

Sub Classes:

Description: Represents the individual receiving healthcare services in the system.

Attributes:

Name	Description
patientId	Unique identifier for the patient.
username	Name of the patient for easy identfication.

Responsibilities:

Name	Collaborator
Book, manage, and view appointments, access personal healthcare information.	User, Appointment, Payment, Insurance, and Medical Record,

Receptionist

Super Classes: User

Sub Classes:

Description:

Attributes:

Name	Description
notificationId	Unique identifier for the receptionist.
username	Receptionist's username for system access.

Responsibilities:

Name	Collaborator
Manage and verify appointments; coordinate with patients and doctors.	Appointment, Patient, Doctor

Doctor

Super Classes: User

Sub Classes:

Description: Represents a healthcare provider who offers medical services to patients.

Attributes:

Name	Description
doctorId	Unique identifier for the doctor.
specialty	Field of expertise of the doctor.
username	Doctor's username in the system.

Responsibilities:

Name	Collaborator
Provide services during appointments.	User, Appointment, Prescription, and Review.

QRCode

Super Classes:

Sub Classes:

Description: Provides a QR code associated with an appointment for check-in or verification purposes.

Attributes:

Name	Description
codeID	Unique identifier for the QR code.
content	Encoded data related to the appointment.
creationDate	Date the QR code was generated.

Responsibilities:

Name	Collaborator
Generate and display QR codes for patient check-in.	Appointment

AppointmentStatus

Super Classes:

Sub Classes:

Description: Defines the current status of an appointment.

Attributes:

Name	Description
statusID	Unique identifier for the status.
statusDescription	Description of the status.

Responsibilities:

Name	Collaborator
Track the appointment's current state.	Appointment

Administrator

Super Classes: User

Sub Classes:

Description: Represents an administrator who oversees system management and user roles.

Attributes:

Name	Description
adminId	Unique identifier for the administrator.
username	Login name for the administrator role.

Responsibilities:

Name	Collaborator
Manage user roles, oversee appointment systems, and ensure secure operations.	User, Appointment, Receptionist

Payment

Super Classes:

Sub Classes:

Description: Represents the financial transactions related to medical services provided to patients.

Attributes:

Name	Description
paymentID	Unique identifier for the payment transaction.
amount	Total amount charged for the medical services.
paymentDate	Date when the payment was processed.
paymentMethod	Method used for the payment (e.g., credit card, debit card).
status	Current status of the payment (e.g., completed, pending, failed).

Responsibilities:

Name	Collaborator
Record, update, and manage payments associated with medical appointments.	Appointment, Patient

Notification

Super Classes:

Sub Classes:

Description: Manages alerts and notifications sent to users within the system.

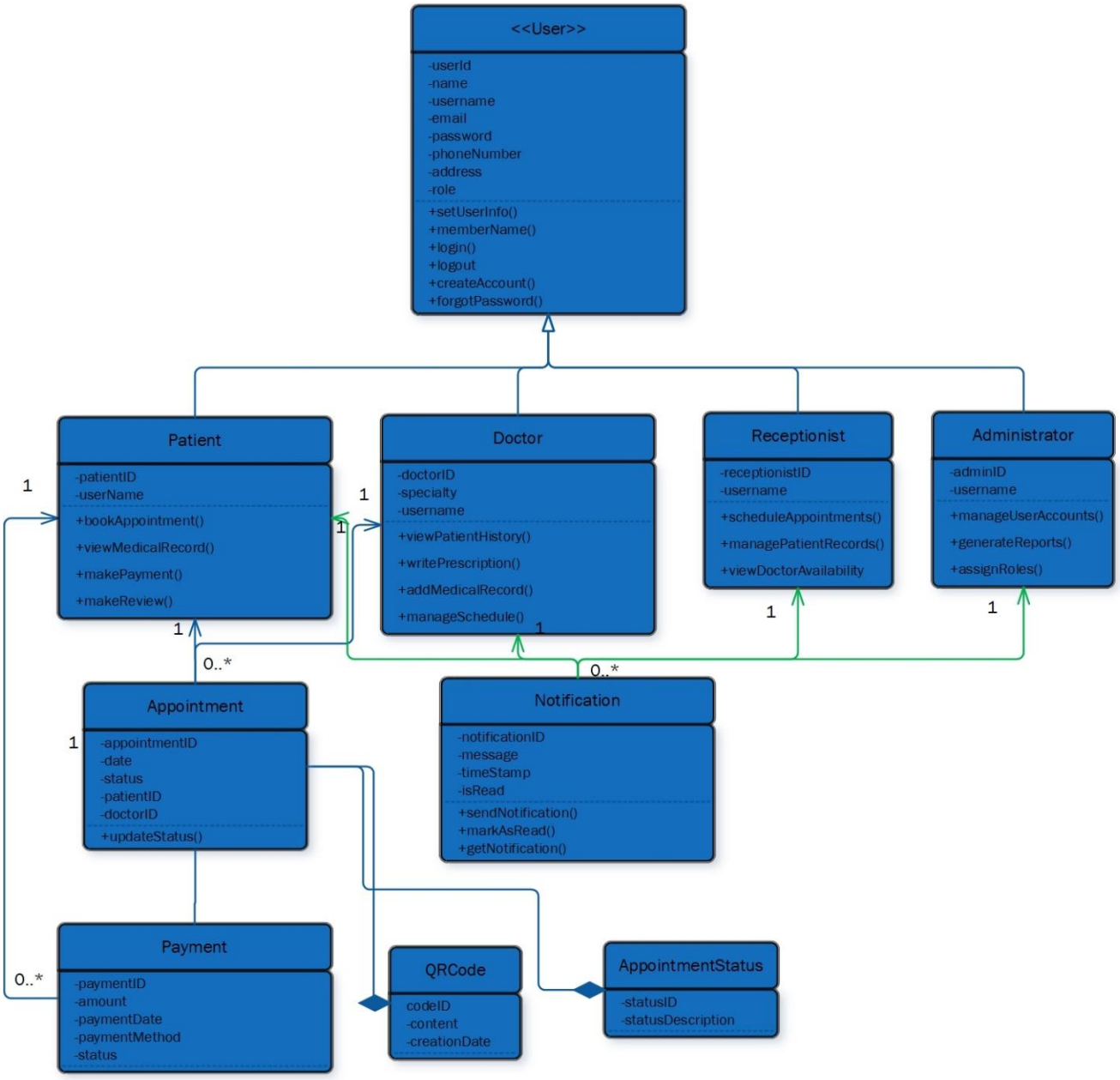
Attributes:

Name	Description
notificationId	Unique identifier for the notification.
message	Content of the notification.
timestamp	Date and time when the notification was sent.
isRead	Status indicating if the notification has been read.

Responsibilities:

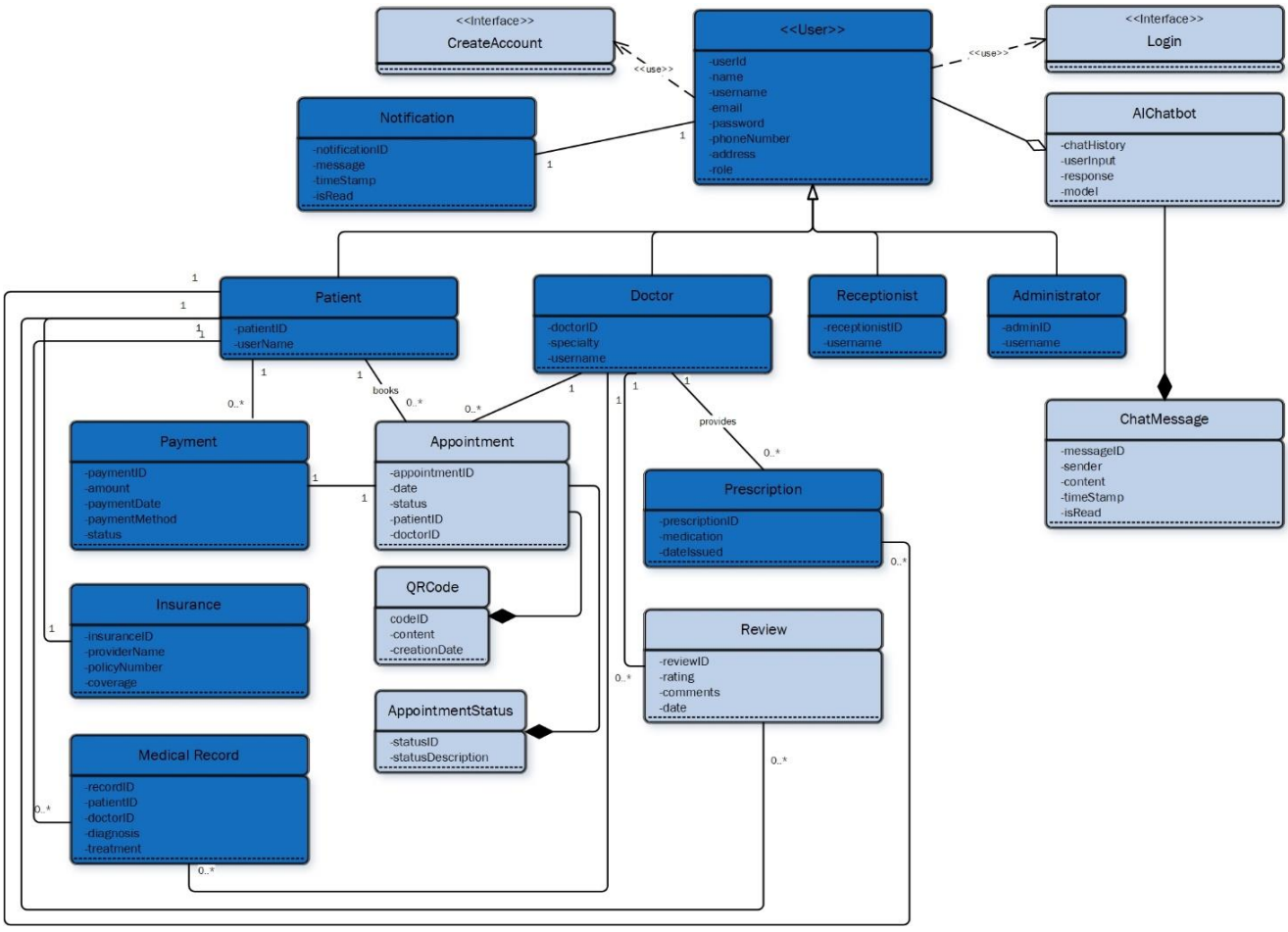
Name	Collaborator
Send notifications about appointment statuses and other updates.	User, Appointment

3.2.3. Design Class Diagram



3.3. Medical Records Subsystem

3.3.1. Partition of Analysis Model



3.3.2. Class Responsibility Collaboration Cards (CRC cards)

User

Super Classes:

Sub Classes: Doctor, Patient, Receptionist, Administrator

Description: Represents a general user in the system.

Attributes:

Name	Description
userId	Unique identifier for the user.
name	Full name of the user.
username	Username used for login.
email	User's email address.
password	Password for authentication.
phoneNumber	Contact number of the user.
address	Physical address of the user.
role	Role of the user in the system

Responsibilities:

Name	Collaborator
Provide common user data and access role-specific functionality.	Notification, Patient, Doctor, Receptionist, Administrator, Medical Record, Appointment

Notification

Super Classes:

Sub Classes:

Description: Sends notifications to users within the system about appointment updates, prescriptions, and other medical record alerts.

Attributes:

Name	Description
notificationId	Unique identifier for the notification.
message	Content of the notification.
timeStamp	Date and time when the notification was sent.
isRead	Status indicating if the notification has been read.

Responsibilities:

Name	Collaborator
Inform users about important updates related to their medical records.	User, Appointment, Prescription

Doctor

Super Classes: User

Sub Classes:

Description: Represents a doctor who provides medical services to patients.

Attributes:

Name	Description
doctorId	Unique identifier for the doctor.
specialty	Field of expertise of the doctor.
username	Doctor's username of the system.

Responsibilities:

Name	Collaborator
Provide medical consultations, issue prescriptions, and update medical records.	Medical Record, Prescription, Patient, Review

Patient

Super Classes: User

Sub Classes:

Description: Represents a patient using the medical record in the system.

Attributes:

Name	Description
patientId	Unique identifier for the patient.
username	

Responsibilities:

Name	Collaborator
Access medical records, manage appointments, and view prescriptions and payments.	Medical Record, Prescription, Insurance, Appointment, Payment, Review

Receptionist

Super Classes: User

Sub Classes:

Description: Manages administrative tasks, including medical records and services, and handling patient inquiries.

Attributes:

Name	Description
receptionistId	Unique identifier for the receptionist.
username	Username for receptionist login.

Responsibilities:

Name	Collaborator
Coordinate medical services with patients and doctors.	Appointment, Notification

Administrator

Super Classes: User

Sub Classes:

Description: Handles administrative settings and oversees system management functions.

Attributes:

Name	Description
adminId	Unique identifier for the administrator.
username	Username for administrator login.

Responsibilities:

Name	Collaborator
Manage user roles, oversee appointment systems, and ensure secure operations.	Notification, User Management

Medical Record

Super Classes:

Sub Classes:

Description: Stores patient diagnosis and treatment history for medical reference.

Attributes:

Name	Description
recordID	Unique identifier for the medical record.
patientID	Identifier for the associated patient.
doctorID	Identifier for the doctor who created or modified the record.
diagnosis	Medical diagnosis details.
treatment	Treatment information provided to the patient.

Responsibilities:

Name	Collaborator
Record, store, and manage patient's medical history.	Patient, Doctor, Prescription, Insurance

Payment

Super Classes:

Sub Classes:

Description: Manages payments made by patients for their medical services.

Attributes:

Name	Description
paymentID	Unique identifier for the payment record.
amount	Amount of the payment.
paymentDate	Date the payment was made.
paymentMethod	Method of payment
status	Payment status

Responsibilities:

Name	Collaborator
Record and process payments for medical services.	Patient, Insurance

Insurance

Super Classes:

Sub Classes:

Description: Manages patient insurance information, including provider details and coverage.

Attributes:

Name	Description
insuranceID	Unique identifier for the insurance record.
providerName	Name of the insurance provider.
policyNumber	Patient's insurance policy
coverage	Coverage details offered by the insurance.

Responsibilities:

Name	Collaborator
Store and provide access to patient insurance details.	Patient, Payment, Medical Record

Prescription

Super Classes:

Sub Classes:

Description: Manages prescriptions issued by doctors for patient treatment.

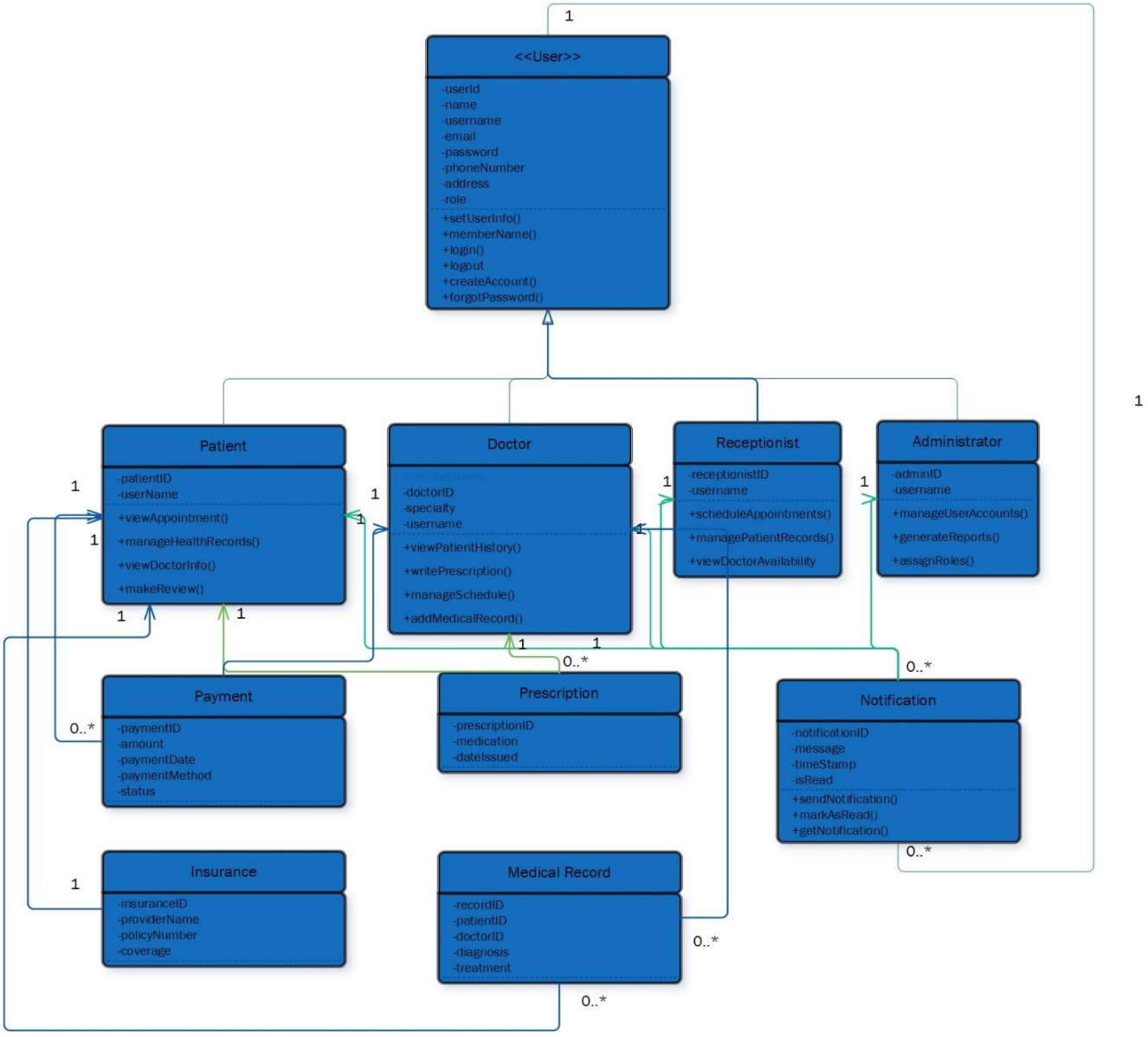
Attributes:

Name	Description
prescriptionID	Unique identifier for the prescription.
medication	Medication details prescribed to the patient.
dateIssued	Date the prescription was created.

Responsibilities:

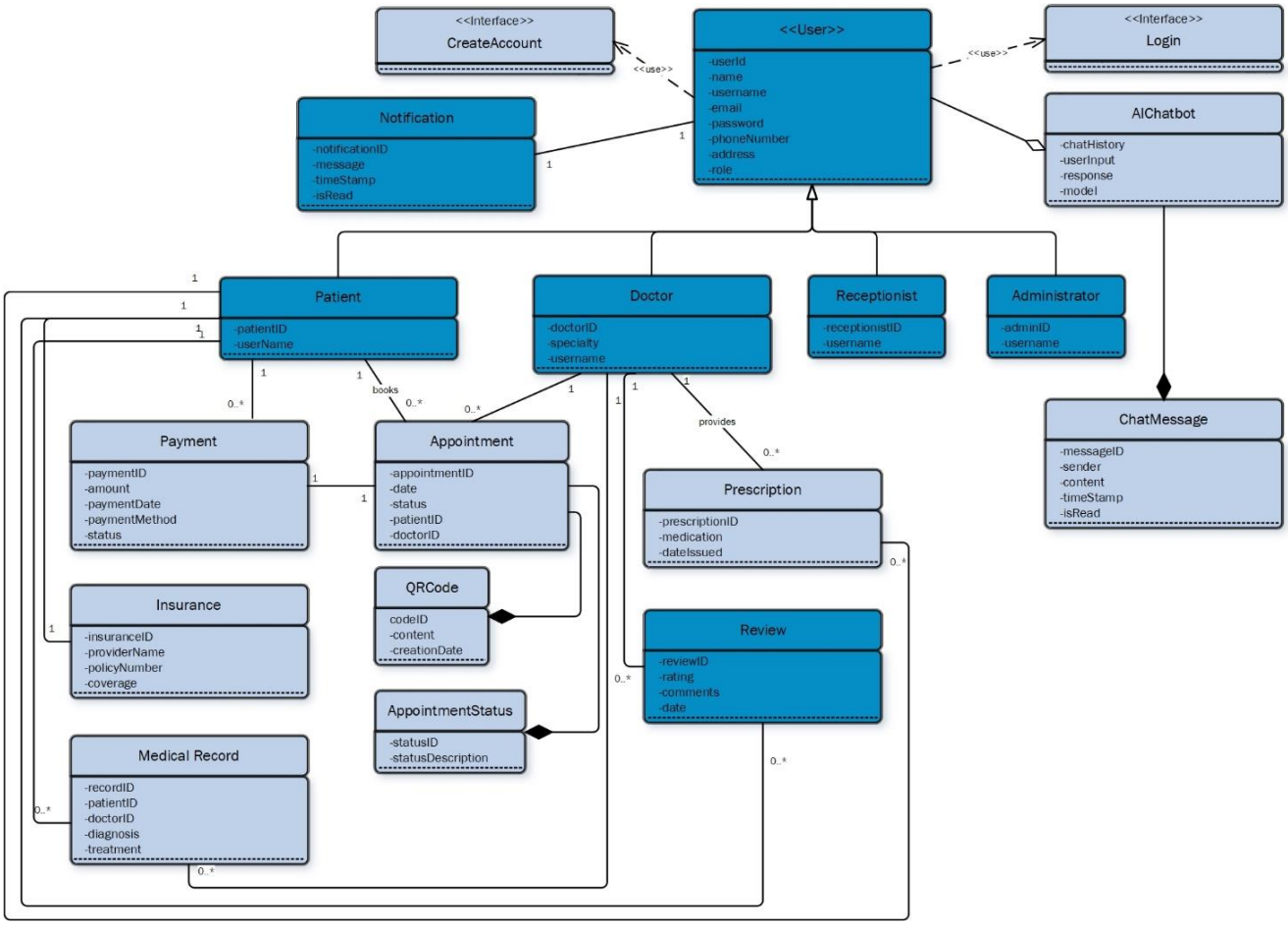
Name	Collaborator
Record prescription information and associate it with patients and their medical records.	Doctor, Patient, Medical Record

3.3.3. Design Class Diagram



3.4. Rating and Review Subsystem

3.4.1. Partition of Analysis Model



3.4.2. Class Responsibility Collaboration Cards (CRC cards)

User

Super Classes:

Sub Classes: Patient, Doctor, Receptionist, Administrator

Description: Represents a general user in the system.

Attributes:

Name	Description
userId	Unique identifier for the user.
name	Full name of the user.
username	Username used for login.
email	User's email address.
password	Password for authentication.
phoneNumber	Contact number of the user.
address	Physical address of the user.
role	Role of the user in the system

Responsibilities:

Name	Collaborator
Manage account information and view and access relevant notifications	Notification, Review

Patient

Super Classes: User

Sub Classes:

Description: Represents a patient who can provide doctor reviews.

Attributes:

Name	Description
patientId	Unique identifier for the patient
username	Patient's display name within the system.

Responsibilities:

Name	Collaborator
Create and submit reviews for doctors.	Review, Doctor, Notification

Doctor

Super Classes: User

Sub Classes:

Description: Represents a doctor who can receive reviews.

Attributes:

Name	Description
doctorId	Unique identifier for the doctor.
specialty	Field of expertise of the doctor.
username	Doctor's username of the system.

Responsibilities:

Name	Collaborator
Access and review patient feedback	Patient, Review, Notification

Review

Super Classes:

Sub Classes:

Description: Represents feedback given by patients regarding doctors or services.

Attributes:

Name	Description
reviewID	Unique identifier for the review.
rating	Numeric rating given by the patient.
comments	Additional feedback comments.
date	Date the review was submitted.

Responsibilities:

Name	Collaborator
Create reviews after medical appointments/consultation.	Patient, Doctor

Notification

Super Classes:

Sub Classes:

Description: Sends notifications related to reviews and ratings.

Attributes:

Name	Description
notificationId	Unique identifier for the notification.
message	Content of the notification.
timeStamp	Date and time when the notification was sent.
isRead	Status indicating if the notification has been read.

Responsibilities:

Name	Collaborator
Inform users about important updates related to their medical records.	User

Receptionist

Super Classes: User

Sub Classes:

Description: Manages patient interactions and coordinates reviews if needed.

Attributes:

Name	Description
receptionistId	Unique identifier for the receptionist.
username	Username for receptionist login.

Responsibilities:

Name	Collaborator
Oversee patient-doctor interactions. Also, Notify doctors and administrators of notable patient feedback	Doctor, Administrator, Notification

Administrator

Super Classes: User

Sub Classes:

Description: Oversees review moderation and user management.

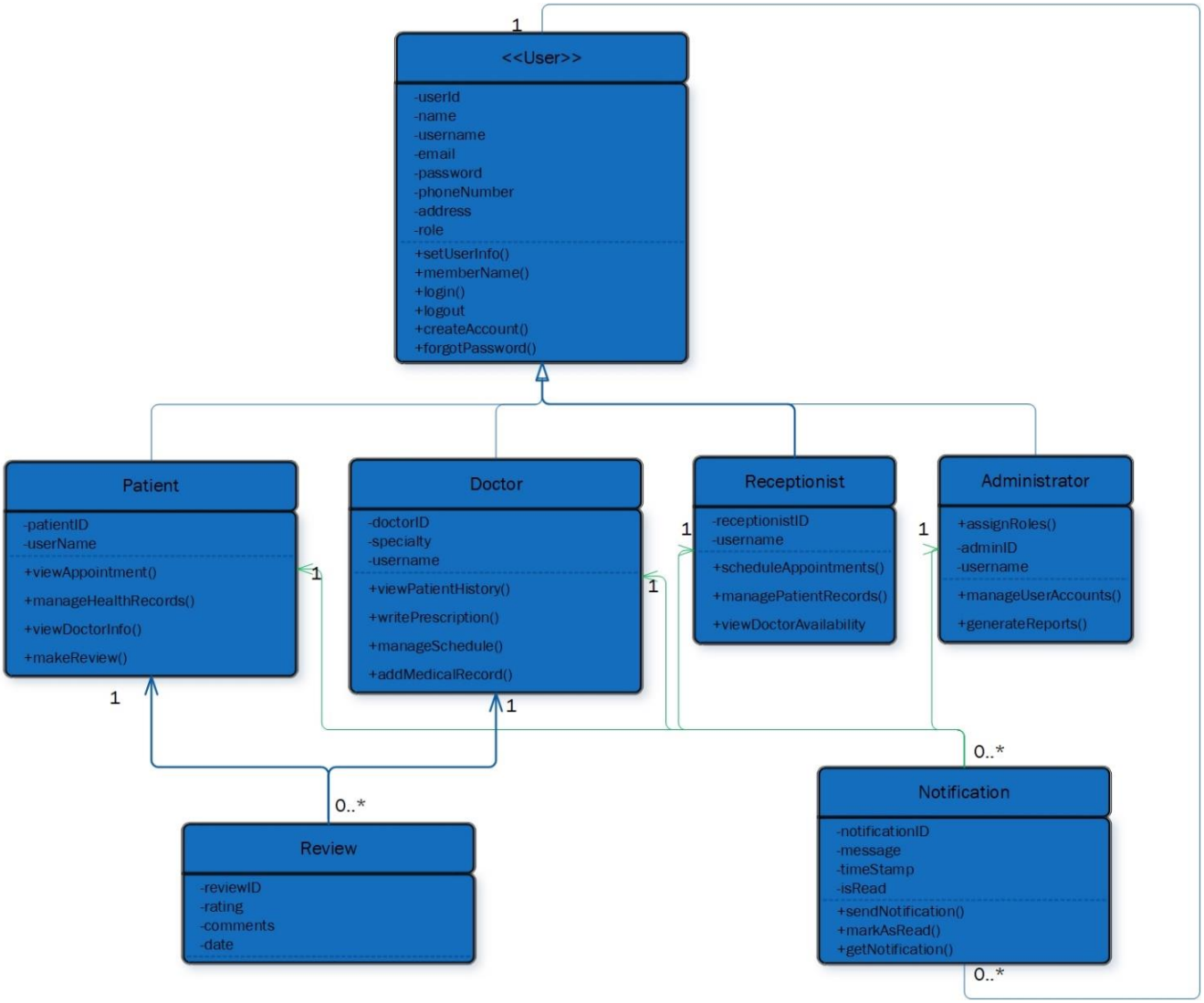
Attributes:

Name	Description
adminId	Unique identifier for the administrator.
username	Username for administrator login.

Responsibilities:

Name	Collaborator
Monitor and manage review activity	Review, Receptionist
Approve or flag inappropriate reviews	

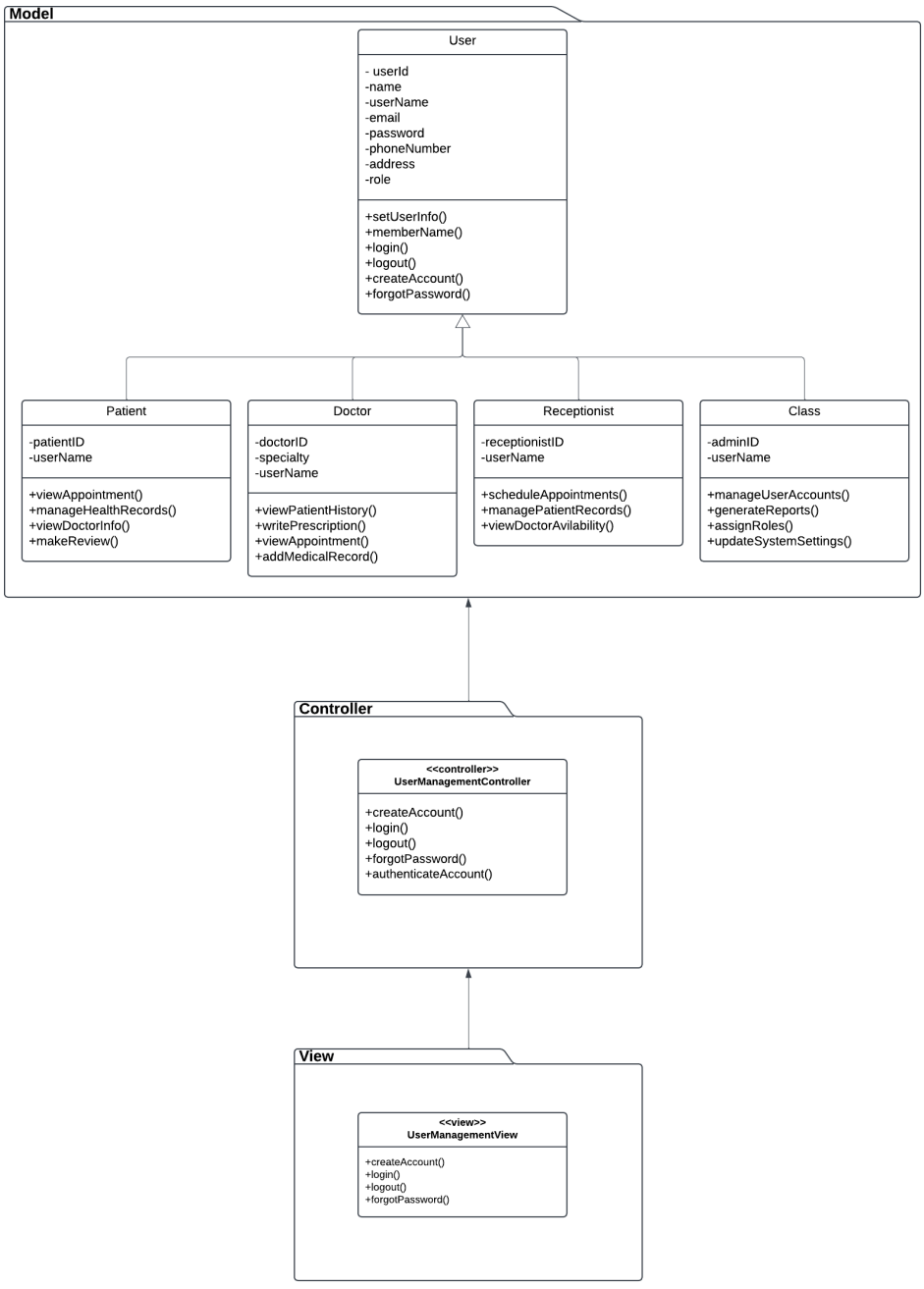
3.4.3. Design Class Model



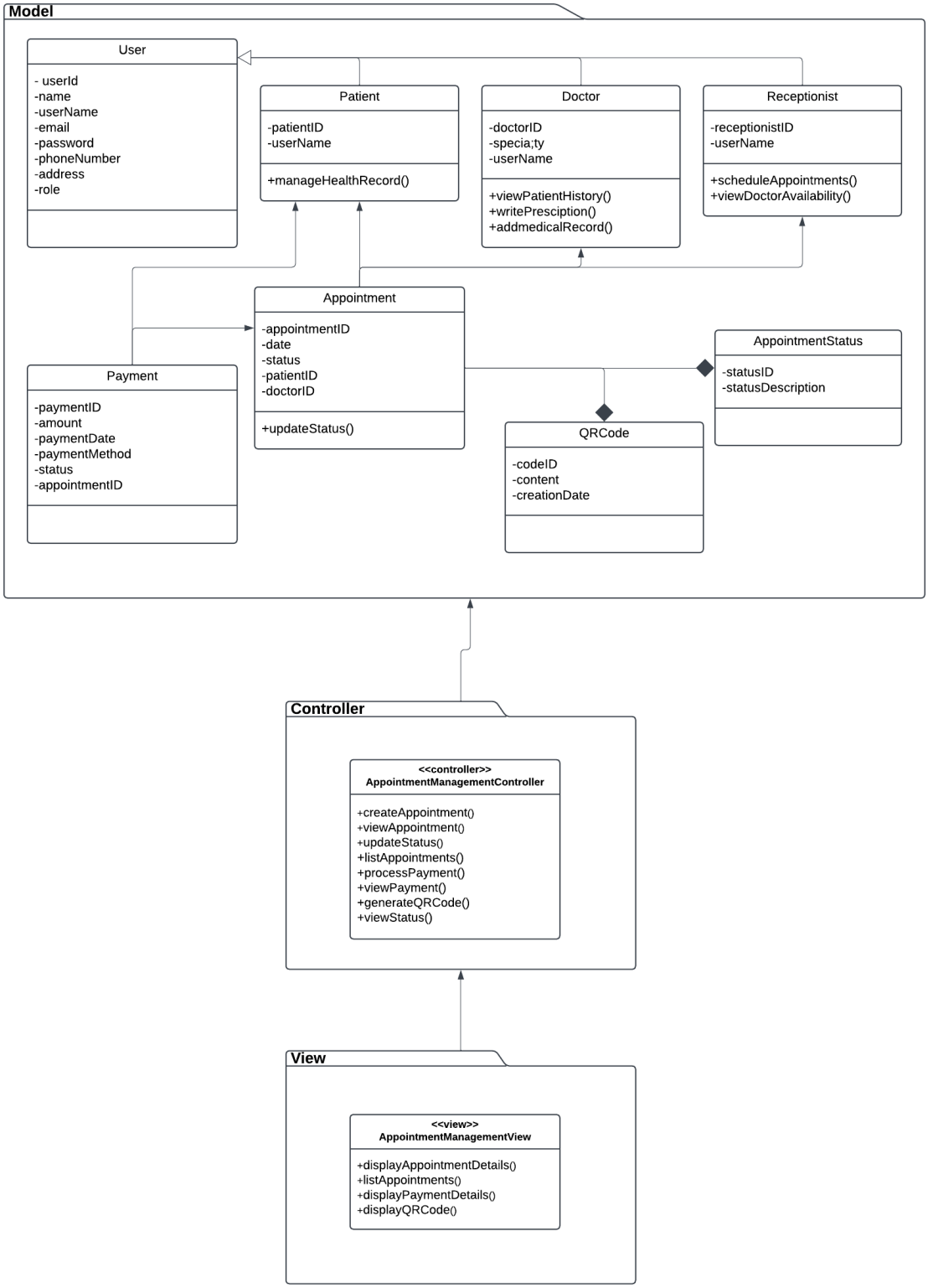
Section 4: Framework M(odel) V(iew) C(ontroller)

4.1. MVC pattern

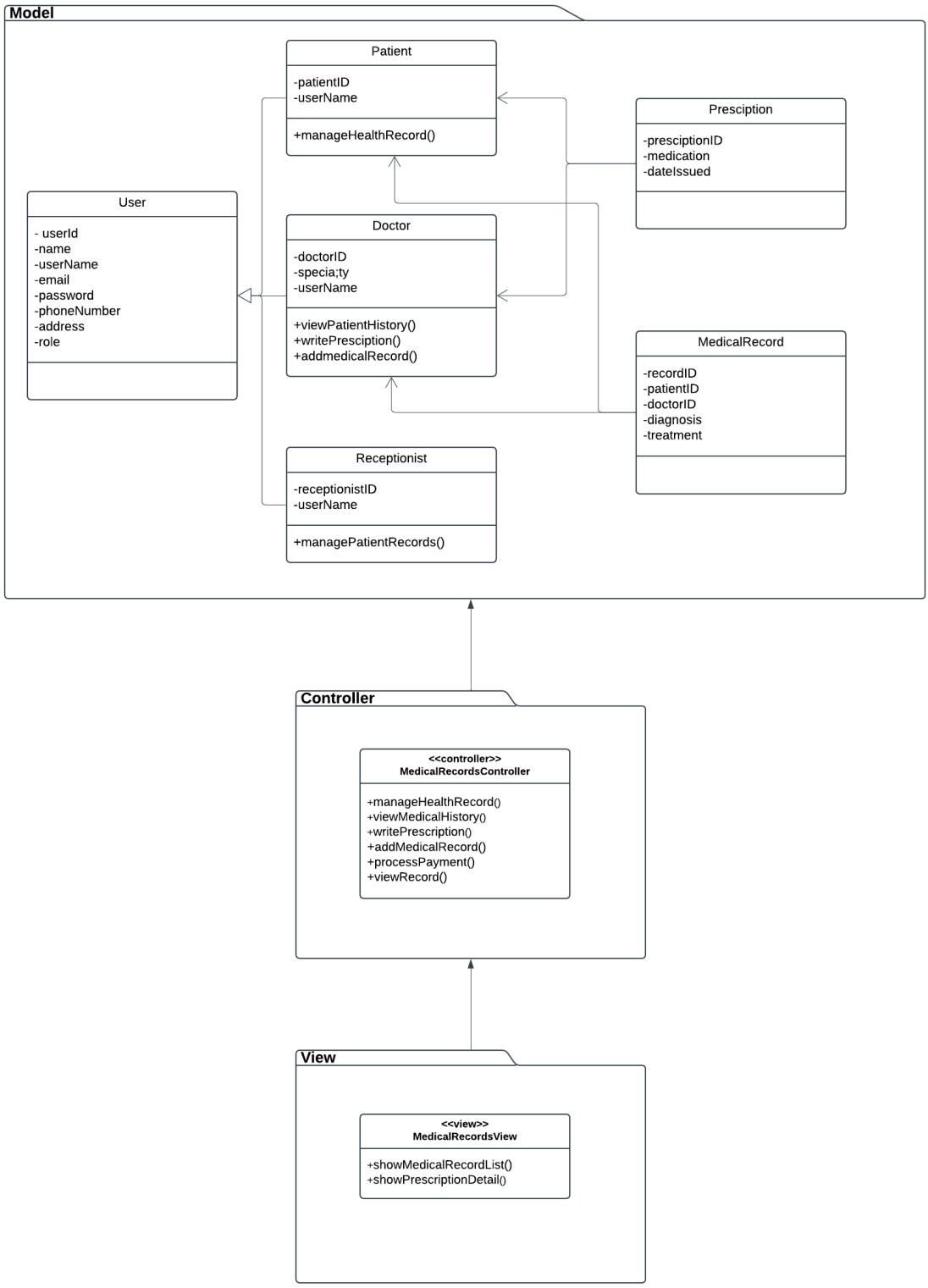
4.1.1. User Management Subsystem



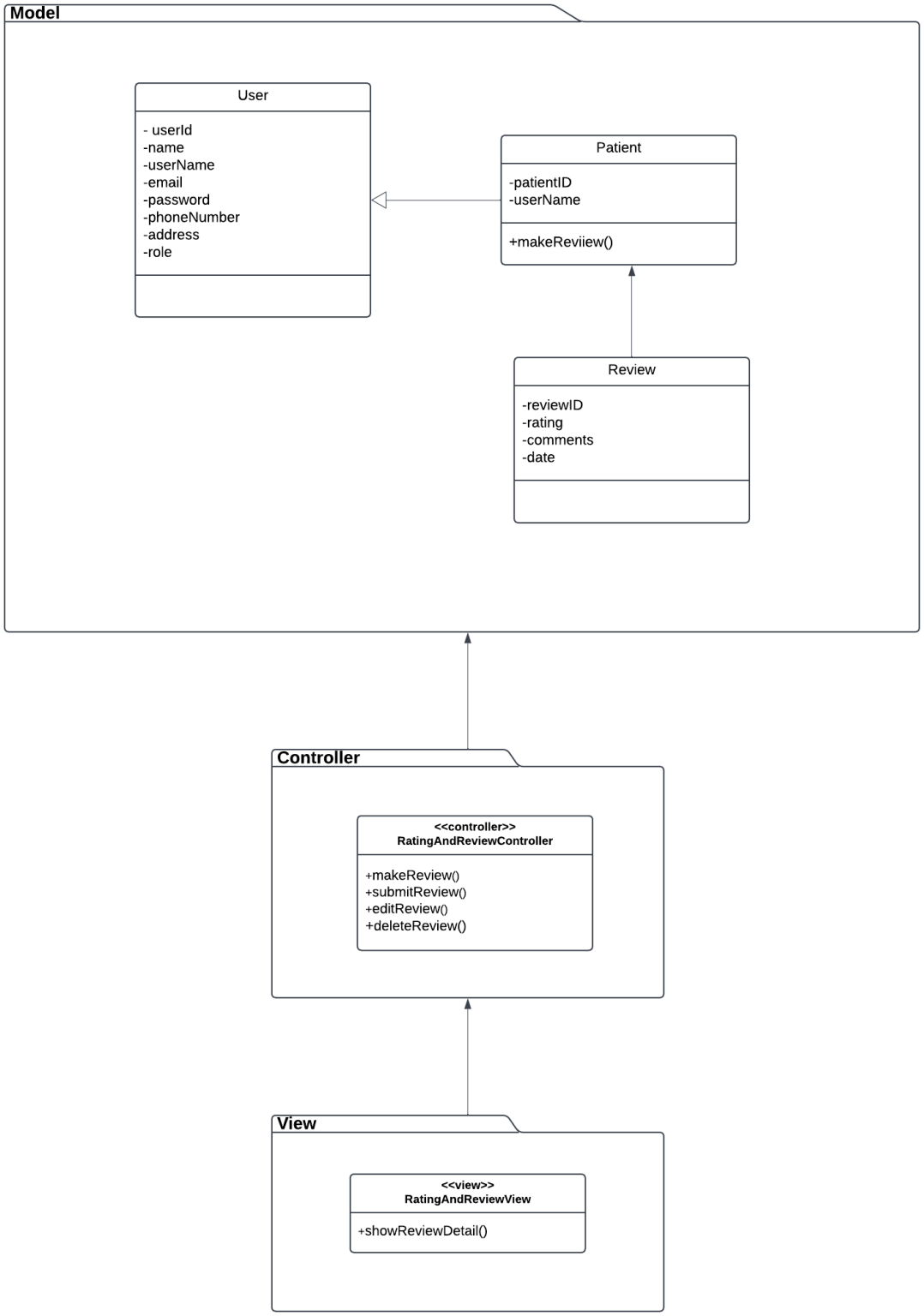
4.1.2. Appointment Management Subsystem



4.1.3. Medical Records Subsystem

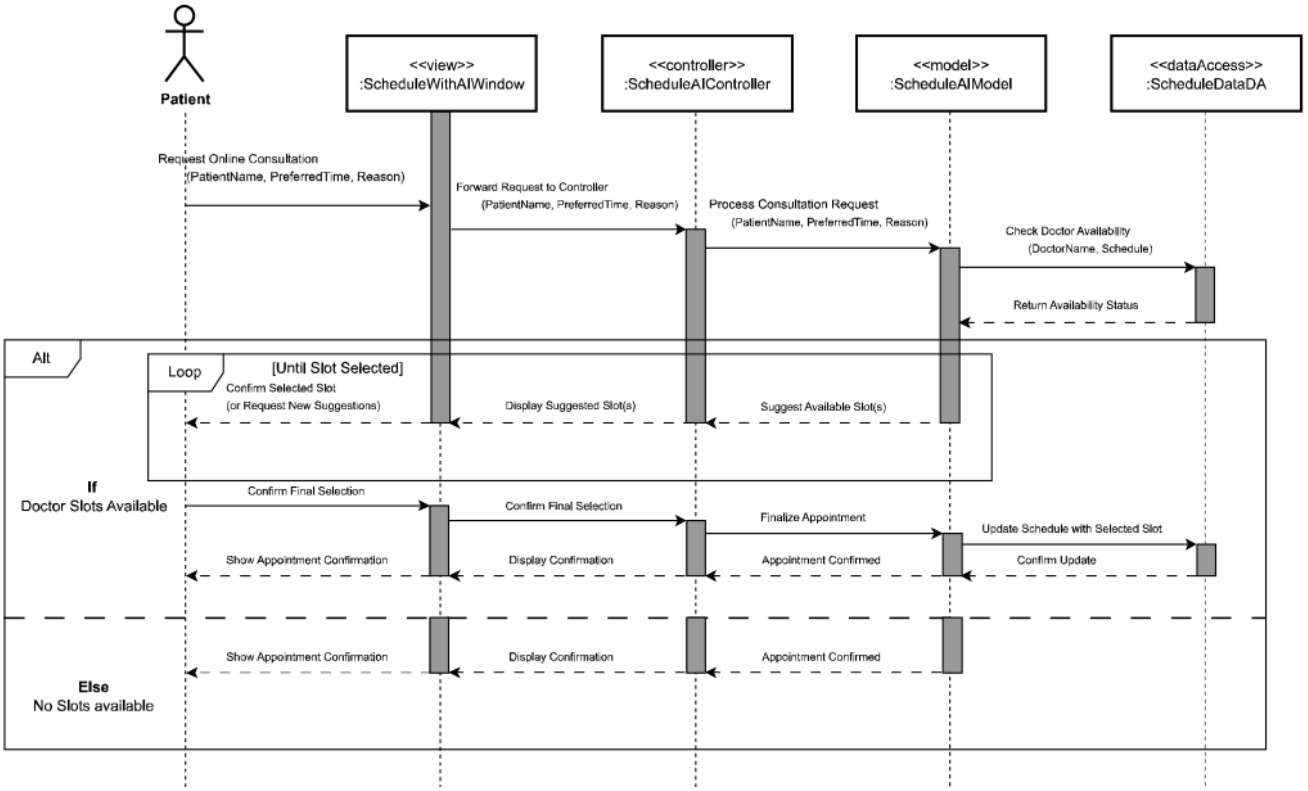


4.1.3. Rating and Review Subsystem

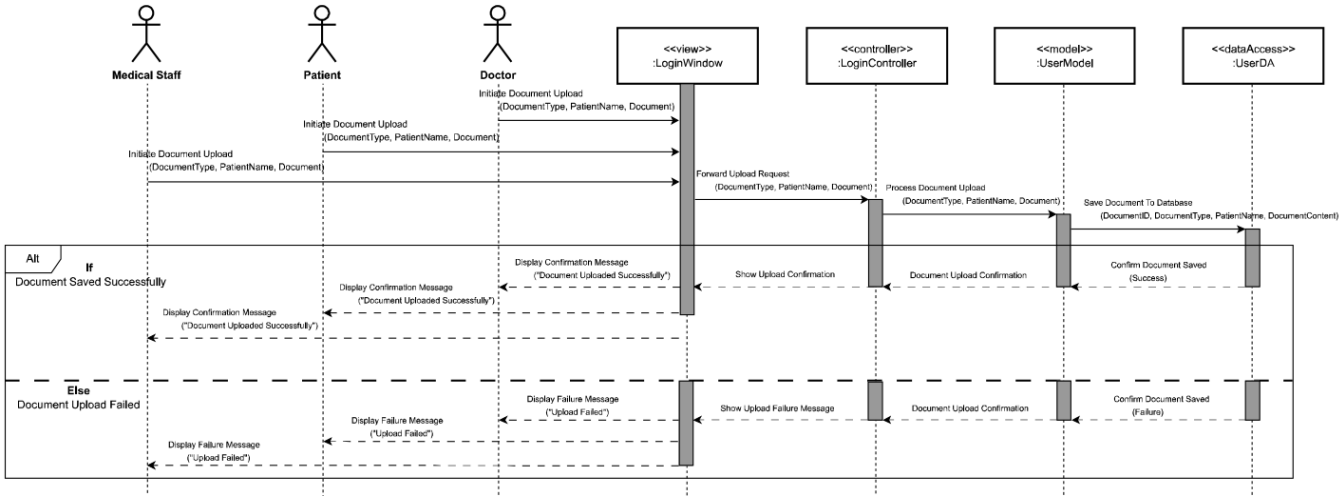


4.2. Full Sequence Diagrams

Use Case: Schedule with AI (in the Appointment Management Subsystem)

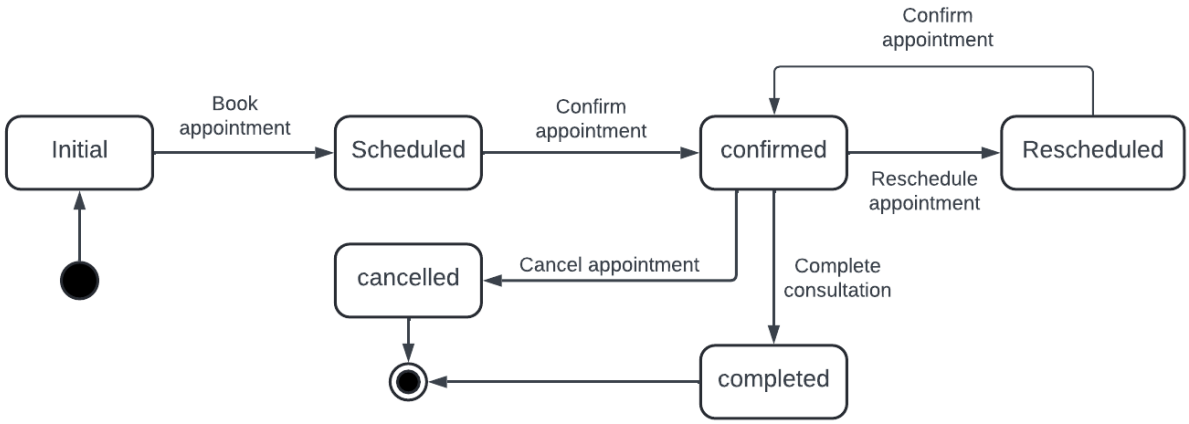


Use Case: Upload Documents (in the Medical Records Management Subsystem)

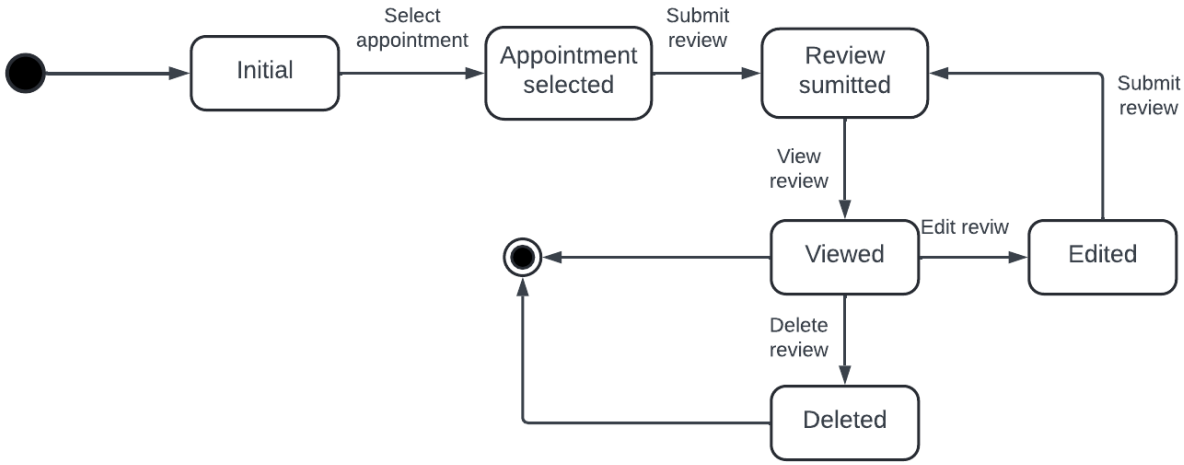


4.3. State Machine Diagrams

Class: Appointment

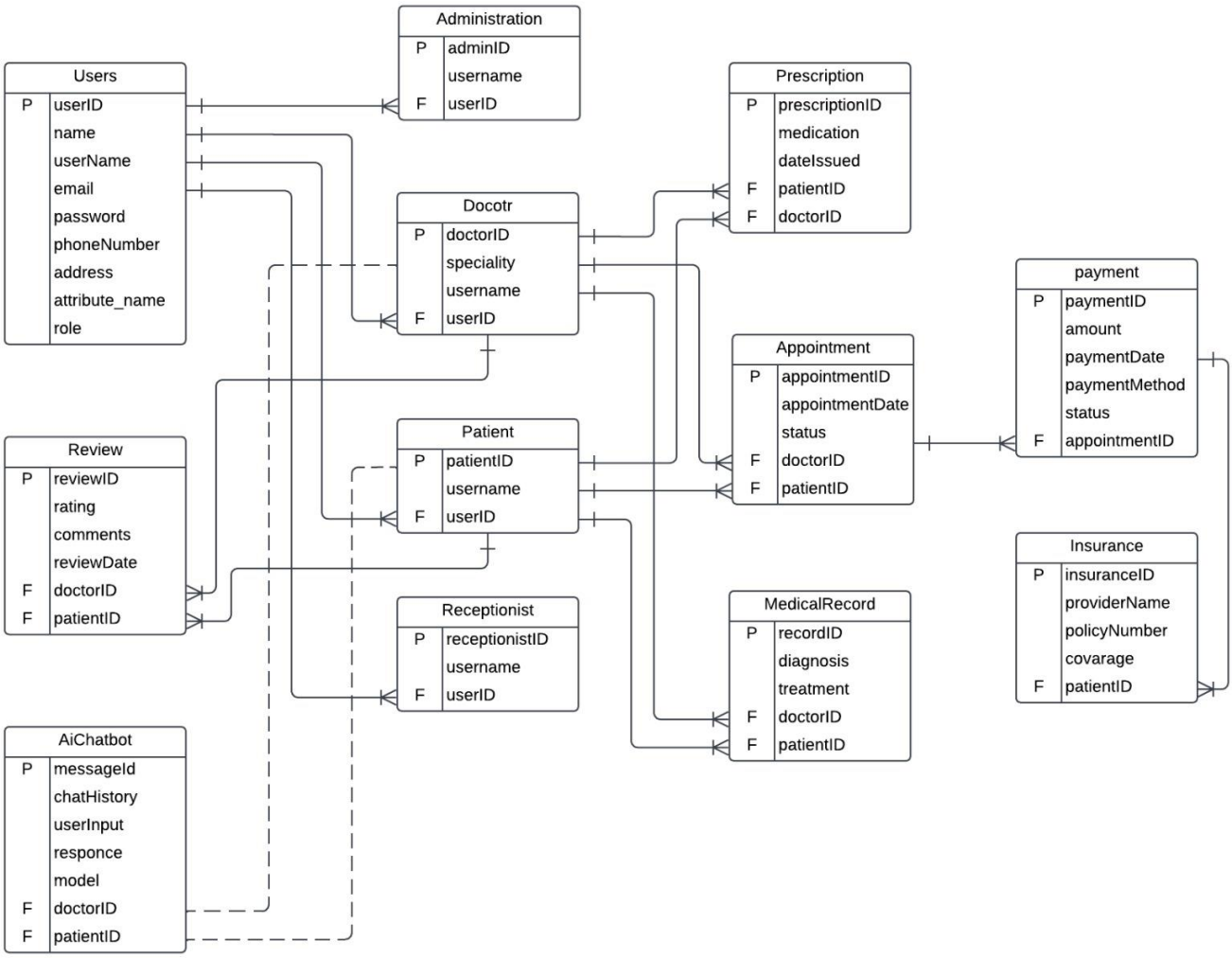


Class: Review



Section 5: Data Layer

5.1. Show a database schema with attributes, data types and sizes for each of the tables



5.2. Technology List Update

Additional Technologies

AI and Machine Learning:

- TensorFlow/PyTorch: For building and integrating AI models (e.g., AI chatbots for scheduling, medical image analysis).
- Natural Language Processing (NLP) Toolkit: To support the chatbot and automated reminder systems, enhancing patient interaction.

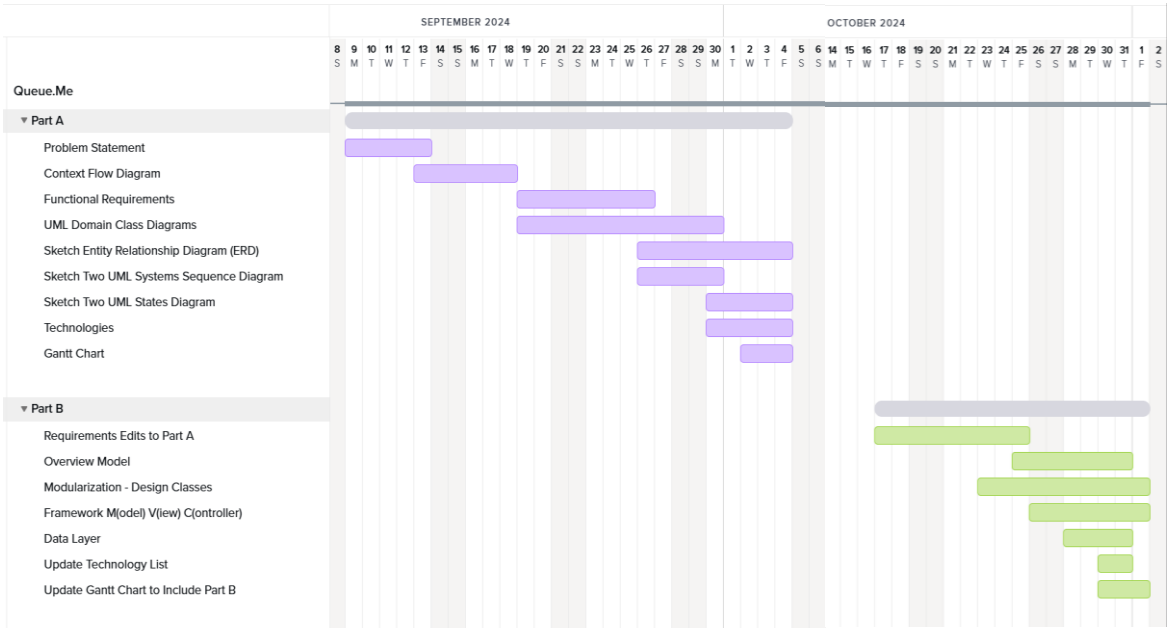
DevOps and Infrastructure:

- Docker: Containerization for consistent deployment across environments.
- Kubernetes: For orchestrating containerized applications, offering scalability and resilience.
- AWS / Google Cloud: Cloud services for hosting, database management, and scalable infrastructure.
- Jenkins / GitLab CI/CD: For continuous integration and continuous deployment, automating build and deployment pipelines.

Security and Compliance:

- JWT (JSON Web Tokens): For secure, stateless authentication.
- OAuth 2.0: To manage secure third-party access to user data.
- SSL/TLS Encryption: To ensure secure data transmission between clients and servers.
- HIPAA Compliance Tools: Data encryption and access logging tools to comply with healthcare regulations.
- This expanded list offers Queue.Me robust performance, security, and scalability while catering to healthcare-specific needs such as secure data handling and real-time communication.

Section 6: Project Management



Part C: System Design Documents

Section 1.0: Corrections to Part B

1.1. Correction on Section 2.1

- Removed the end users in the Intended users for SDD.

1.2. Correction on Section 2.2 ACD

- Add a superordinated System in the ACD.

1.3. Correction on Section 3

- Updated the format on Section 3: 1) Partition Analysis Model 2) CRC Diagram 3) Design Class Diagrams.
- Updated the colors of the diagrams instead of keeping it as plain white.

1.4. Correction on Section 4.2 Diagram

- Updated the sequence diagram if no slots available, and the timeline to show the differences.

1.5. Correction on Section 5.1

- Updated the Data Layer: Database Schema to use bi-directional relationships with other tables.

1.6. Correction on Section 6: Gantt Chart

- Added another row with days of the week (MTWTFSS).
- Updated the Gantt Chart for Part A-B-C that provides a customized header.

1.7.Correction on Revision History

- Added Part A/Part B/Part C under the Name column for the Revision History table.
- Added Author column

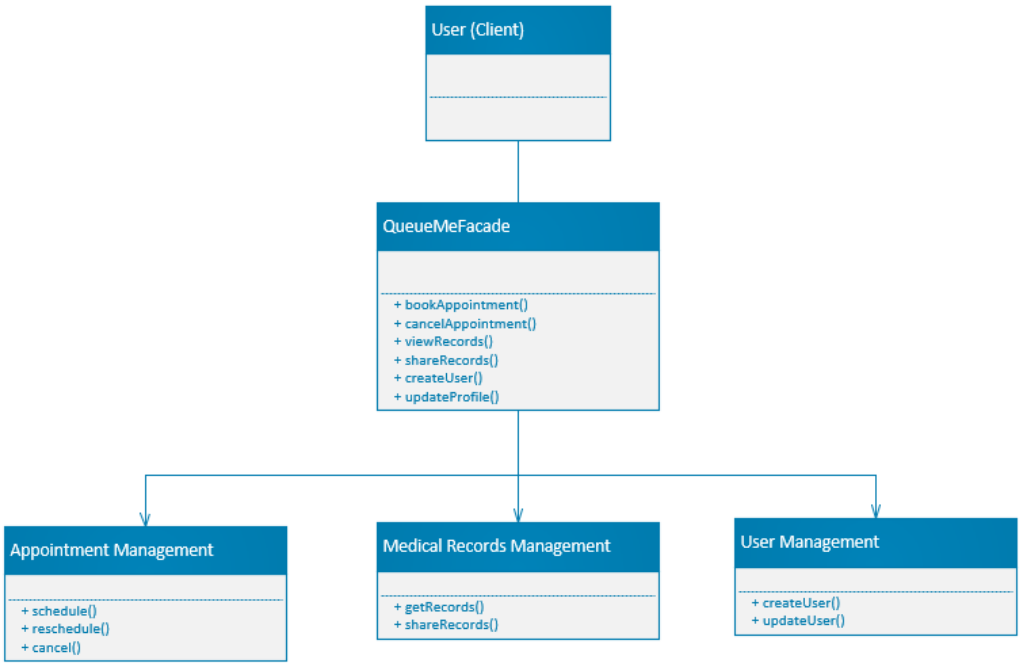
Section 1: Software Design Patterns

- Façade Pattern
- Observer Pattern
- Singleton Pattern

Section 2: Using Common Software Design Patterns

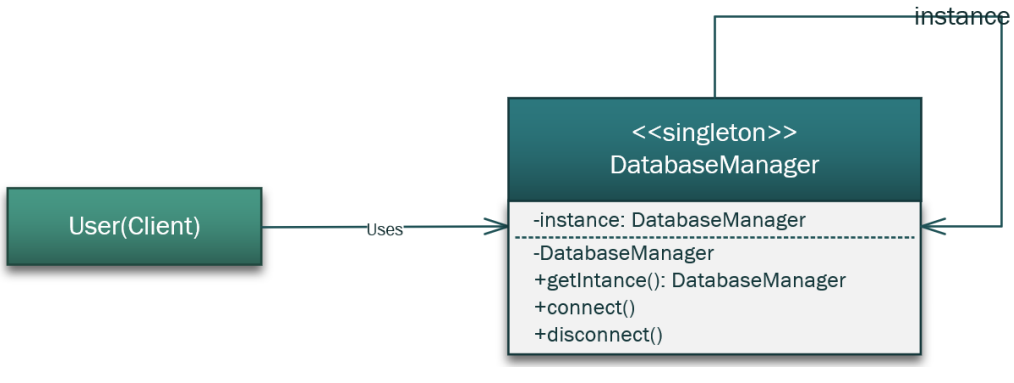
2.1 Façade Pattern

Name:	Façade Pattern
Problem:	User management, appointment management, and medical records management are some of the subsystems that make up the Queue.Me system. Although each subsystem has unique features, clients (users) who are attempting to access and administer these services face challenges. Users would have to understand and navigate the complexities of each subsystem to interact directly, which would result in a poor user experience and increased maintenance costs.
Solution:	The Façade Pattern provides a unified interface (QueueMeFacade) that wraps complex subsystems, offering simplified access to their functionalities. Instead of each client directly interacting with subsystems, they interact solely with the QueueMeFacade. This facade handles operations by coordinating between subsystems, allowing for streamlined operations, easier code maintenance, and a more user-friendly client experience. This allows the client to interact with the system through a single entry point without needing to understand the underlying architecture.

Graph:	
Benefits and consequences:	**Benefits:** 1. Simplified Interaction - To reduce complexity, clients communicate with a single entry point rather than several subsystem interfaces. 2. Decoupled Subsystems: The codebase is easier to maintain and expand when subsystems are weakly coupled. 3. Enhanced Security: Consistent security checks are made possible by centralized access via the façade, shielding subsystem specifics from unwanted access. 4. Flexibility: Subsystems can evolve independently without affecting the client. 5. Easier Testing and Maintenance: Since clients only depend on the stable interface of the façade, developers can alter subsystem internals without impacting the client. **Consequences:** 1. Overhead in Façade Implementation: In order to effectively coordinate interactions among subsystems, the facade layer needs extra coding and maintenance. 2. Risk of Over-Simplification: If the facade is not properly constructed, it may

	oversimplify certain functions, which would prevent clients from taking full advantage of subsystem capabilities. 3. Possible Bottleneck: It may become a bottleneck in performance if it is a frequently used entry point, especially if a large number of interactions are routed through it. 4. Over-reliance on the Façade: If the façade becomes too complex, it may turn into a bottleneck for extending or maintaining the system.

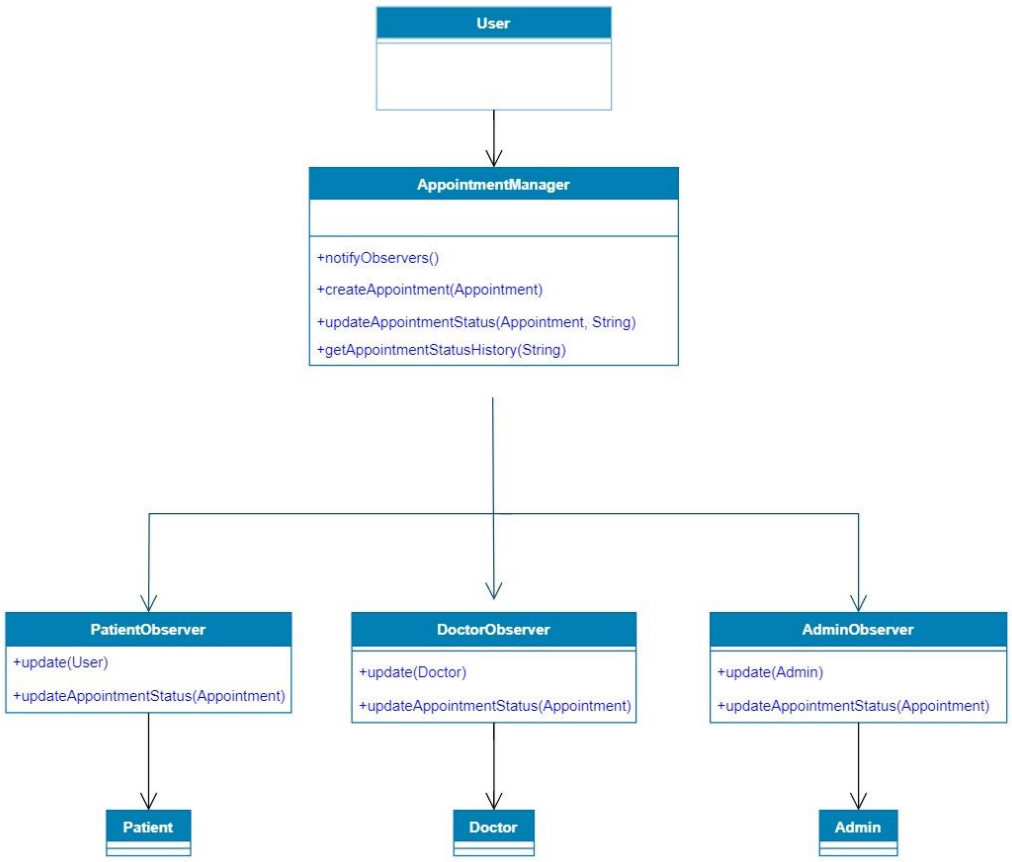
2.2 Singleton Pattern

Name:	Singleton Pattern
Problem:	The Queue.Me application requires certain components to have only one instance throughout the system's lifecycle. These components include: 1. Database connection manager 2. Configuration settings manager 3. Logging service Having multiple instances of these components could lead to inconsistencies, resource wastage, and potential conflicts in the system.
Solution:	The Singleton pattern ensures that a class has only one instance and provides a global point of access to that instance. In Queue.Me, we can implement Singleton classes for the database connection manager, configuration settings manager, and logging service.
Graph:	

Benefits and consequences:	**Benefits:** 1. Controlled Access: Ensures only one instance of critical components exists, preventing inconsistencies and conflicts 2. Global State: Provides a global point of access to the instance, simplifying system-wide operations 3. Lazy Initialization: The instance is created only when it's first requested, conserving system resources 4. Reduced Memory Footprint: Only one instance is created, saving memory compared to multiple instances of resource-heavy objects **Consequences:** 1. Global State Risks: The global state can make the system harder to test and potentially introduce unexpected side effects 2. Tight Coupling: Classes using the Singleton become tightly coupled to it, which can make the code less flexible and harder to modify 3. Concurrency Challenges: In a multi-threaded environment, special care must be taken to ensure thread-safety during instance creation 4. Violation of Single Responsibility Principle: The Singleton class takes on the responsibility of managing its own instance, potentially violating the Single Responsibility Principle

2.3 Observer Pattern

Name:	Observer Pattern
Problem:	In the Queue.Me system, multiple components or users need to be notified of specific changes in the system in real-time. For instance, when an appointment is booked, canceled, or rescheduled, all relevant parties (patients, doctors, and perhaps administrative staff) need instant updates. Managing such notifications manually can lead to complex code, tight coupling between components, and difficulty in adding new observers (such as new user types) in the future.

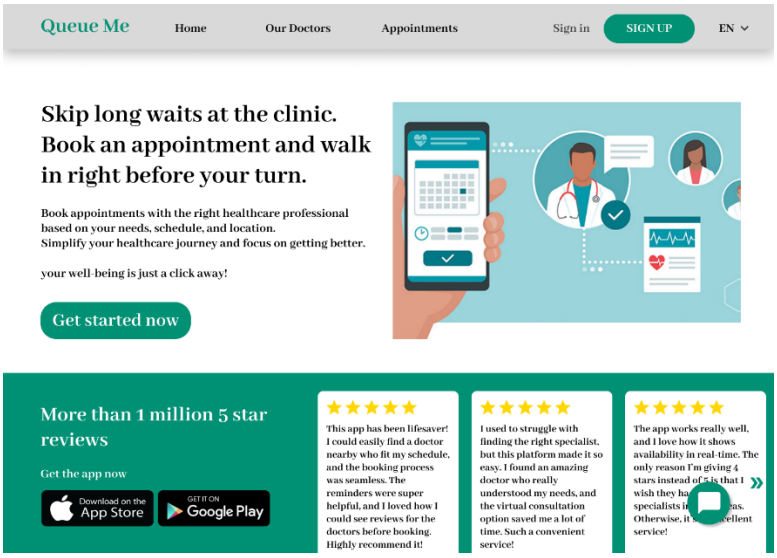
Solution:	The Observer Pattern allows a "subject" object to maintain a list of "observer" objects that need to be notified when a particular event or change occurs. In Queue.Me, each component or user interested in appointment updates can register as an observer. When an appointment's status changes, the subject (e.g., an Appointment Manager component) will notify all registered observers automatically. This decouples the objects, making the notification process more flexible and manageable.
Graph:	
Benefits and consequences :	**Benefits:** 1. Real-time Notifications: Observers receive instant updates as changes happen, improving user engagement and satisfaction. 2. Decoupling of Components: Observers and subjects are loosely coupled, allowing easier maintenance and future expansion. 3. Easier to Add New Observers: New components or users can easily subscribe

	to notifications without altering the core subject. 4. Centralized Event Management: The subject manages the notification process, reducing redundancy and potential errors in communication logic. **Consequences:** 1. Potential Performance Overhead: With many observers, real-time notifications could slow down the system, especially if each update triggers multiple complex actions. 2. Unintended Dependencies: Over time, the system could accumulate unintended dependencies if too many components rely on each other's updates, complicating the system's behavior. 3. Increased Complexity: Managing many observers and their registration or removal can add complexity, especially if not properly managed or documented.

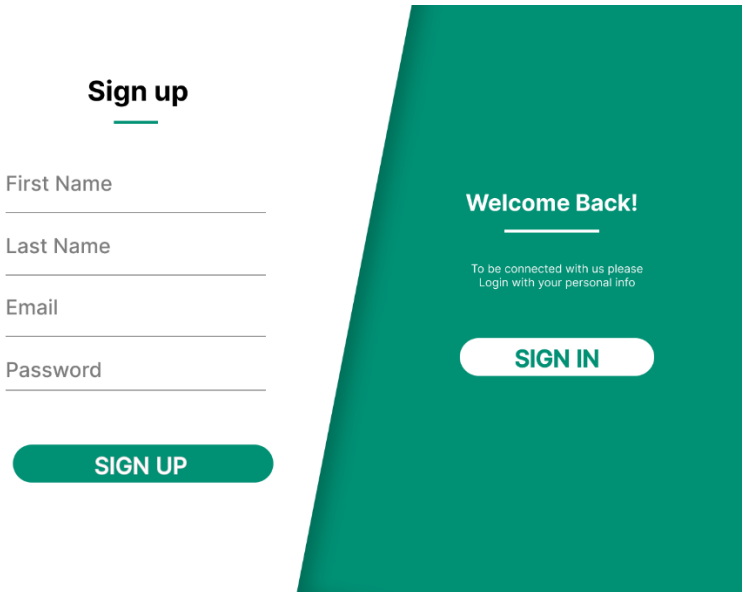
Section 3: UI/UX Design

3.1 Web

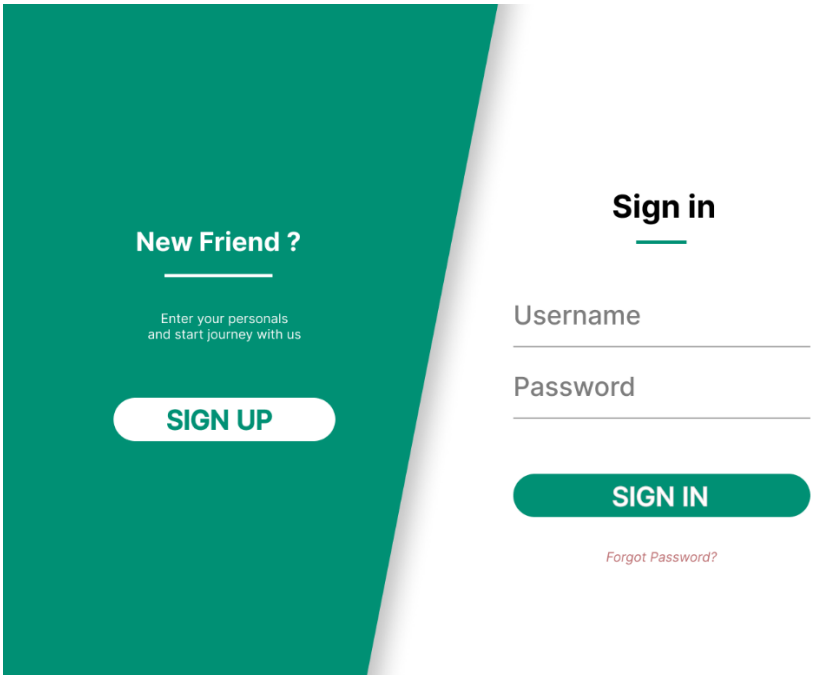
3.1.1 Home page



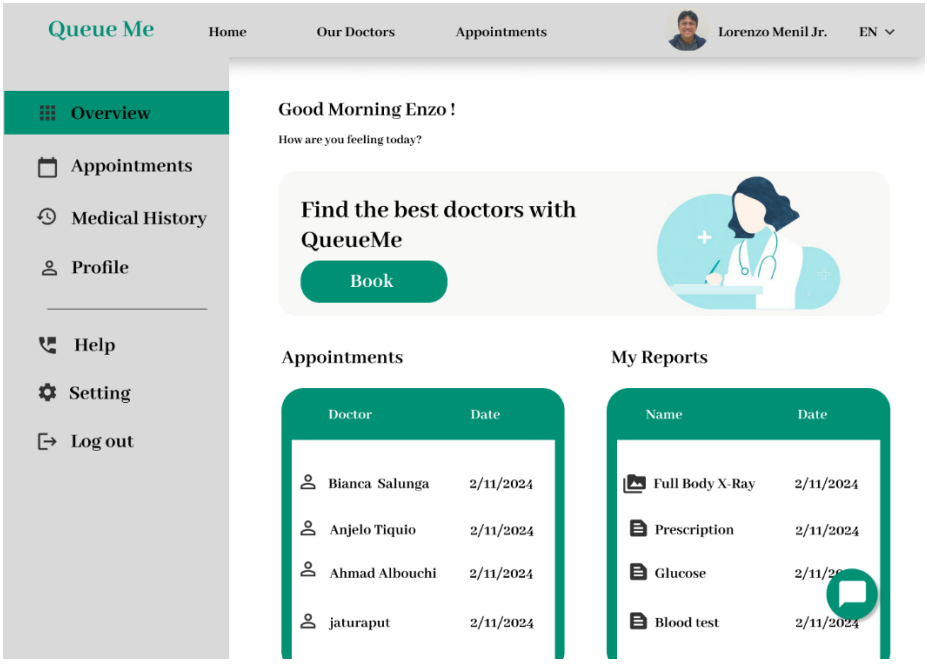
3.1.2 Sign up page



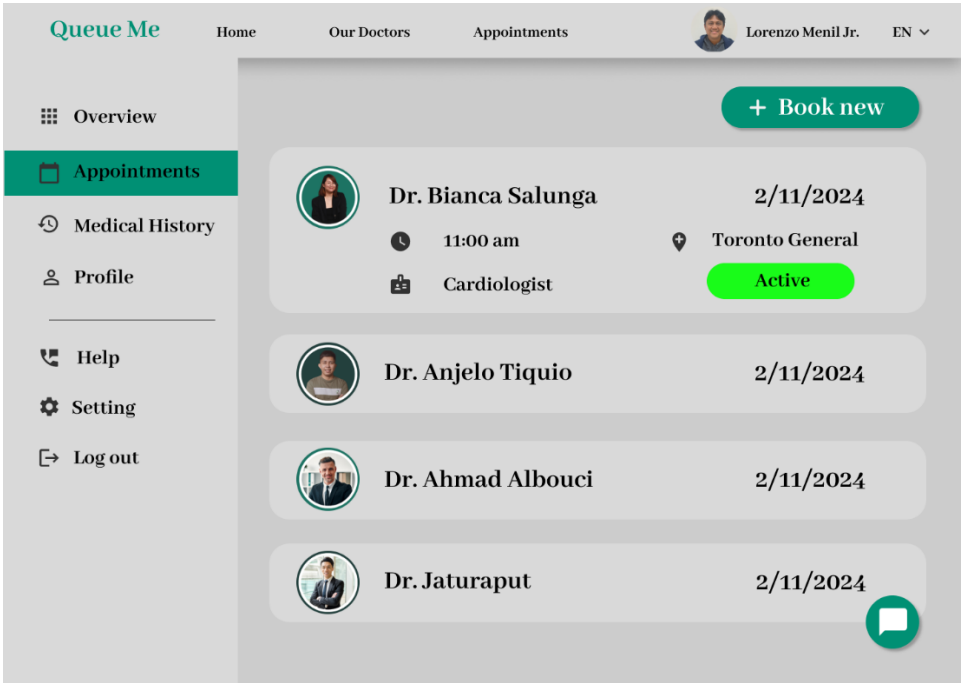
3.1.3 Sign in page



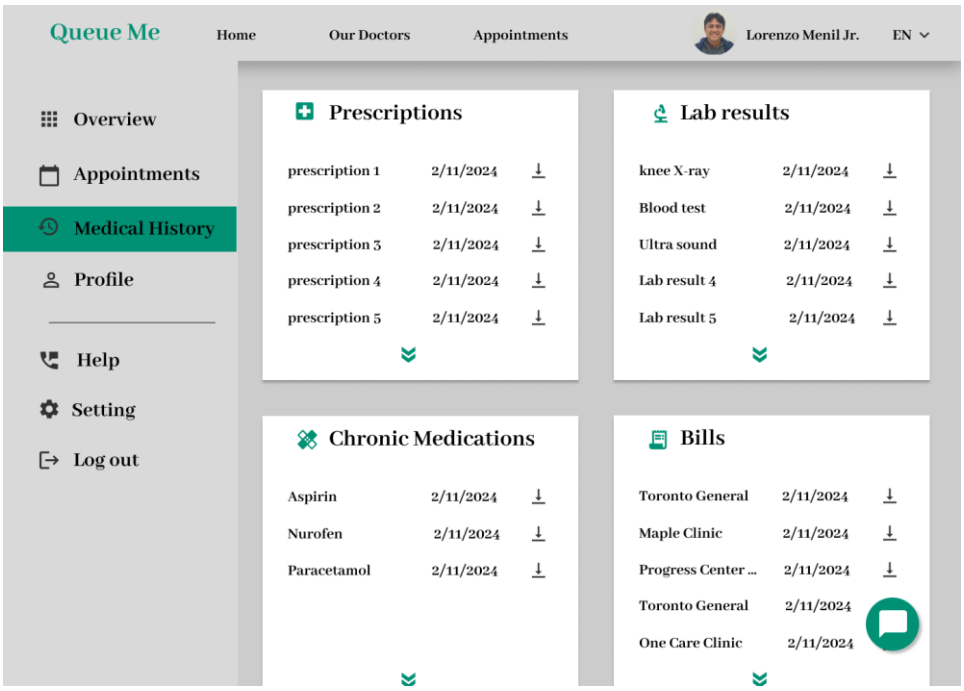
3.1.4 Dashboard page



3.1.5 Appointment page

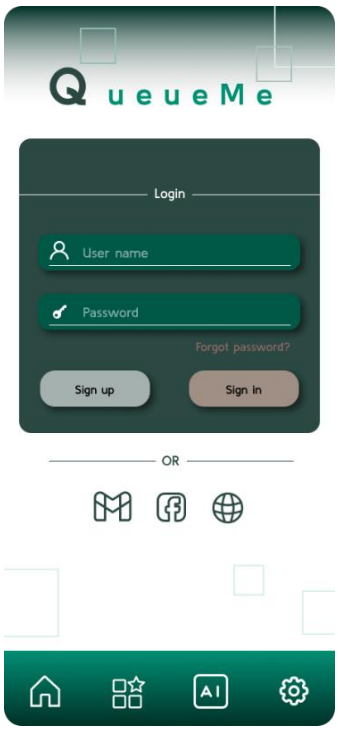


3.1.6 Medical Record/History page

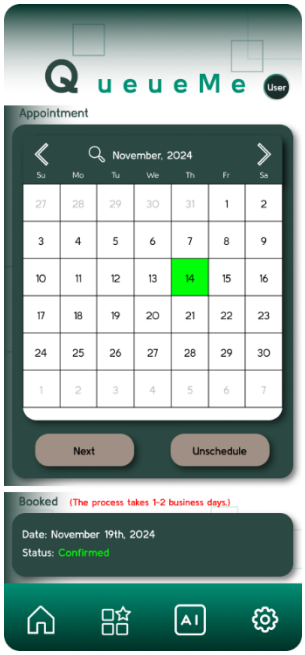


3.2 Mobile

3.2.1 Home page, Login and Loading page

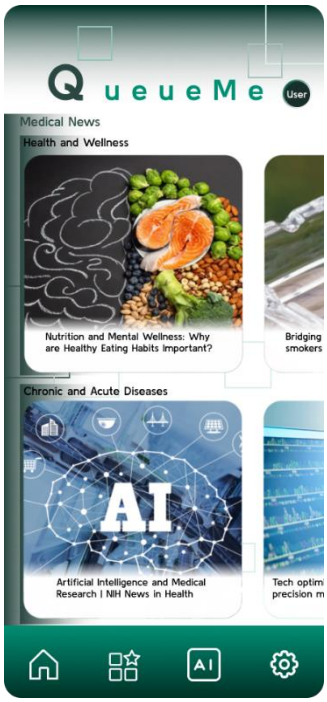


3.2.2 Appointment page

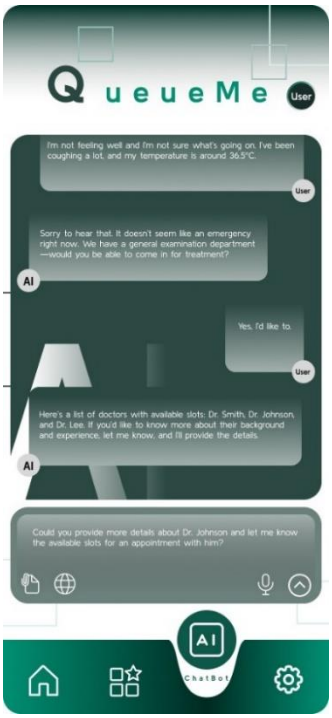


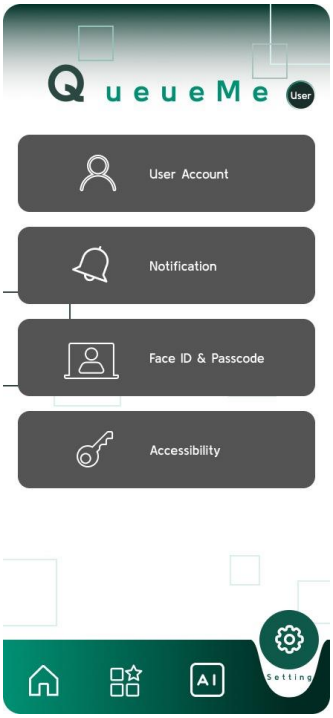
3.2.3 Prescription page

3.2.4 Medical news page and Department page

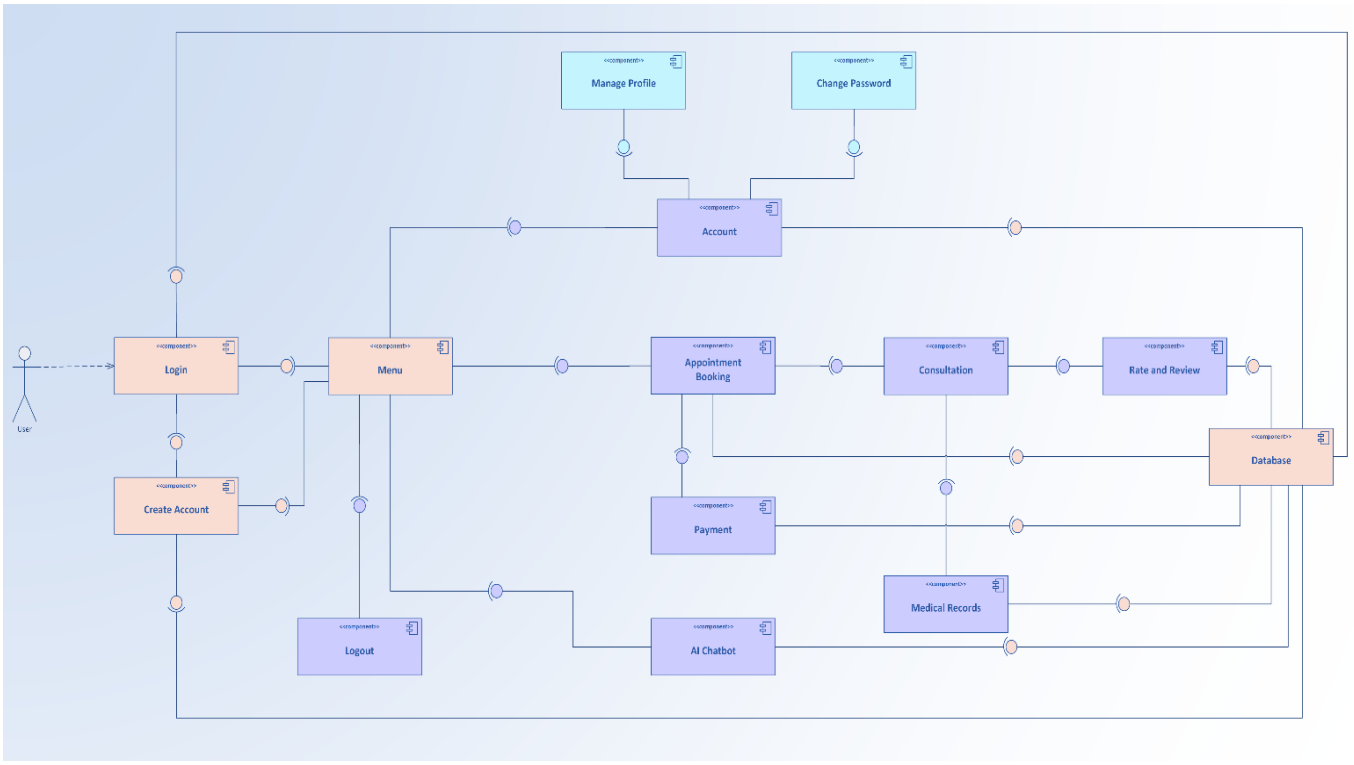


3.2.5 AI Chatbot page and User Management page





Section 4: High Level Component/Deployment Diagram



Section 5: Project Management