IT314 - SOFTWARE ENGINEERING

PROF. SAURABH TIWARI

Software Requirements Specification

Group 31: Swasthya Sarathi

Team Members :

SR.NO	Student ID	Name
1	202201463	Harsh Popatiya
2	202201464	Mayank Parmar
3	202201474	Jeet Desai
4	202201481	Anuj Valambhiya
5	202201484	Jaikrit Sanandiya
6	202201487	Nandini Mandaviya
7	202201493	Harsh Lad
8	202201511	Aryan Patel
9	202201521	Jemini Chaudhari
10	202201522	Arnold Mochahari

Table of Contents

Chapter 1: Introduction

1.1 Purpose of the Document

This document outlines the Software Requirements Specification (SRS) for the Patient Management System. Its purpose is to provide a comprehensive understanding of the system's functionality, objectives, and target audience, ensuring effective communication among stakeholders, designers, and developers.

1.2 Overview of the Hospital Management System

The Patient Management System is designed to streamline the management of hospital operations. It connects patients, doctors, and hospital administrators, facilitating efficient appointment scheduling, patient record maintenance, and resource management. The platform allows patients to book appointments, view their medical history, and manage their health records. Doctors can record diagnoses and prescriptions, while hospital admins oversee operations such as managing doctors and approving appointments.

1.3 Scope

This project involves the development of a web-based system to manage hospital-related processes. Key functionalities include user registration, appointment scheduling, medical history management, and administrative oversight. The system will also provide tools for efficient resource allocation, record-keeping, and communication among stakeholders. This document outlines both functional and non-functional requirements, serving as a foundation for the design, development, testing, and deployment of the system.

Chapter 2: Requirements

2.1 Functional Requirements

1) User Login and Registration

- The system will provide login and Registration functionality for all users, including patients, doctors, and hospital admins.
- Each user will have role-based access control to ensure appropriate permissions.

2) Doctor Profile Add/Delete

- Hospital admins can create and delete doctor profiles in the system.
- This ensures accurate doctor information and availability for patients.

3) Doctor Search

- Patients can search for doctors based on their name, department, or the hospital they are associated with.
- This functionality helps patients find the right doctor quickly and efficiently, tailored to their specific needs.

4) Appointment Booking and Scheduling

- Patients can book appointments with doctors, while hospital admins can manage and schedule appointments as required.

5) Medical History Access

- Patients will have access to their medical records, including previous diagnoses, prescriptions, and test results.

6) Prescription and Diagnosis Management

- Doctors can add prescriptions and record diagnoses for patients during consultations.
- These updates will be automatically reflected in the patient's medical history for future reference.

2.2 : Non Functional Requirement

1. Data Redundancy and Integrity

- The system will ensure that all medical records, appointments, and user data are stored without unnecessary duplication.

2. Availability and Reliability

- The system must be accessible 24/7 to support patients, doctors, and admins, ensuring minimal downtime.

3. Compatibility

- The platform will be compatible across multiple devices,

4. User-Friendliness

- The interface will be intuitive and easy to navigate for all types of users (patients, doctors, and admins).

5. Privacy and Security

- Multi-factor authentication and regular security updates will be implemented to safeguard against unauthorized access and cyber threats.

6. Scalability

- The system will be designed to handle an increasing number of users, appointments, and records without performance degradation.

Chapter 3: User Stories

	Front of the card	Back of the card
	As a doctor/patient/staff member, I want to register by providing my personal details when I visit the portal for the first time so that my information is stored within the portal and I can log in whenever I wish.	1. The registration form must collect required details such as name, email, phone number, password, gender, DOB, etc. 2. Upon submission and validation of all the fields, the information is stored in the database. 3. The registration process should prevent duplicate registrations for the same email address.
2	As a doctor/patient/staff member, I want to log in through Email and Password so that I can access the features of the portal.	1. The login page must allow users to log in using Email/Password. 2. For Email/Password login, validate credentials against stored data in the database. 3. Incorrect Email/Password combination must result in an error message. 4. Successful login should direct users to their respective role-based dashboard.
3	As a doctor/patient/staff member, I want a password recovery option so that I can reset my password securely through email if I forget it.	1. The password recovery page must ask the user for their registered email address. 2. Upon submission, an email with a password reset link is sent to the user. 3. The reset link should expire after a predefined time (e.g., 24 hours). 4. Clicking on the reset link should direct the user to a secure page where they can set a new password and use it to login.
4	As a doctor/patient/staff member, I want to select my role from the available options (Patient, Doctor, Admin) so that I can access features specific to my role.	1. The registration form must include three options: Patient, Doctor, and Admin. 2. Upon login, the user is directed to the features associated with their selected role. 3. The system must store the selected role in the database and ensure

		appropriate access control based on the role.
5	As a doctor/patient/staff member, I want my password to be stored securely through encryption so that my privacy is protected.	1. Passwords must never be stored in plain text, they must be hashed and salted before being stored in the database. 2. The system must use secure cryptographic methods to store and retrieve passwords.
6	As a doctor, I want to be able to update or edit my personal and professional details, such as contact information and qualifications, so that my profile remains accurate and up-to-date.	1. The system should allow the doctor to modify fields such as contact information and qualifications after registration. 2. The doctor should receive a confirmation notification once the profile is successfully updated.
7	As a patient, I want to register on the app by providing my name, age, gender, date of birth, contact number and email so that I can have a verified profile in the system.	1. The system must allow the patient to input all required details (name, age, gender, date of birth, contact number, and email) during registration. 2. Upon successful registration, their profile should be marked as "verified" after successful verification.
8	As a hospital admin, I want to register my hospital on the app by providing hospital name, mail, password, Regestration number, contact number, date of foundation and hospital type so that I can have a verified profile in the system.	1. The system must allow the hospital admin to input all required information (name, email, registration number, date of foundation, contact number, and hospital type) during the hospital registration process. 2. Upon successful submission, the hospital registration number should be verified, and the hospital's profile should display a "verified" badge once verification is completed.
9	As a hospital admin/patient/doctor, I want secure storage of all records so that my privacy is maintained.	1. All personal and medical records must be encrypted during transmission and while stored in the system to ensure data privacy and protection from unauthorized access. 2. Access to records should be role-based, ensuring that only authorized users (e.g., a patient, their assigned doctor, or the hospital admin) can view or modify relevant information.
10	As a doctor, I want to view the complete medical history of a patient, including past diagnoses, treatments, and medications, so that I can	1. The system must display the patient's full medical history, including past diagnoses, treatments, and medications, in an organized manner when the doctor accesses their profile. 2. The medical history should be available to the doctor only if they

	make informed decisions during their current visit.	have been assigned the patient for a consultation.
11	As a doctor, I want to generate and provide medical certificates directly from the app, so that patients can receive official documentation for their health conditions or treatment without additional paperwork	1. The app must enable the doctor to create a medical certificate with necessary details, such as patient name and diagnosis. 2. The patient should be able to download or receive the medical certificate in a PDF format from their profile.
12	As a doctor, I want to view a patient's medical history sorted by a specific timeline to have better analysis.	1. Upon accessing a patient's profile, an option should be available to view the patient's medical history. 2. There should be an option to sort and view this history in timeline sequence.
13	As a patient, I want my medical reports to be accessible only to the appointed doctor and authorized staff.	1. The system should ensure that your medical reports are accessible only to your designated doctor and authorized staff.
14	As a patient, I want a search bar so that I can easily find information about my desired doctor.	1. The system should provide a search bar, allowing patients to easily find information about their desired doctor. 2. The search bar should provide relevant information about the desired doctor.
15	As a patient, I want a navigation bar in the system for quick access to my records, appointments, and other features so I can easily manage my healthcare needs.	1. The system’s home page should include a navigation bar. 2. This bar must incorporate essential functions such as records, appointments, and other key features of the system.
16	As a doctor/patient/hospital admin, I want the ability to log in from multiple devices so that I can access and manage my account from different devices conveniently.	1. The system should support logging in from multiple devices.
17	As a patient/admin/doctor, I want a logout option in the system so that I can securely end my session.	1. The system should provide a logout option for users to securely end their session. 2. After logging out, users must re-enter their credentials to access the system again.

18	As a Hospital admin, I want to be able to update/add/delete records, so that I can efficiently maintain them.	1. The system must provide the Hospital admin with a user interface to securely add, update, or delete records, with appropriate validation to ensure data integrity and accuracy. 2. Changes to records (addition, modification, or deletion) should be logged with timestamps and user details to ensure traceability and accountability.
19	As a doctor, I want the patient to provide their Patient ID so that I can view their medical history accurately and efficiently.	1. The system should provide a full medical history of the patient when patient ID is entered. 2. The system should provide the search functionality for viewing patient's history only to the doctors.
20	As a doctor, I want to add a patient's prescription so that the patient receives the correct medication and dosages in their medical record.	1. The system provides the 'Add prescription' option so that doctors can add prescriptions in the patient record.
21	As a hospital admin, I should view, manage and print the patient's medical invoice from the respective hospital in the system.	1. The hospital admin can view, manage, and print an invoice from the hospital data in the system.
22	As a patient, I want the prescription format to be in a comprehensible manner so that I can understand the diagnosis more simply.	1. The system should provide a description in a certain format such that it would also be easy for the doctors to type inside it. 2. The format should also include Patient ID, Patient Name, Disease, name of prescription, dosage times, test reports, test recommendations, remarks, etc.
23	As a patient, I want to search for doctors so that it would be easy for me to browse doctor details and book appointments as per my choice.	1. The system provides search functionality to users for selecting Doctors as per their convenience. 2. The user can also search doctors as per chronological names, hospital names, department, etc.
24	As a patient, I want to browse doctors through a speciality filter to see which doctors are relevant to my treatment.	1. The search should only show doctors that provide treatment of the particular speciality. 2. If no such doctors are found, it should show a relevant message on the screen.

25	As a patient/doctor/hospital admin, I want the website to be mobile responsive so I can open the website on different devices.	1. The website interface should be scaled and responsive across all devices. 2. The website should be able to support both portrait and landscape mode.
26	As a patient, I want a feature where I can book appointments so I do not have to go through the hassle of calling and rescheduling appointments all the time.	1. The patient must be able to book appointment and select their preferred time and date without requiring assistance.
27	As a hospital admin, I want a feature that allows me to see appointments requests so I Can arrange the schedule.	1. The hospital admin should be able to view a list of all upcoming appointments, sorted by date and time. 2. The appointment details should include the patient's name and patient id.

Chapter 4A: List of Use Cases

For Patient :

1. Book Appointment
2. Cancel Appointment
3. View Medical History
4. Find Doctor
    - Search by Doctor's Name
    - Search by Department
    - Search by Hospital Name
5. View Pending Appointments
6. Download Medical History

For Doctor :

1. View Past Appointments

2.View Pending Appointments

3.Record Patient Details(Add prescription/diagnosis)

For Hospital Admin :

1. Register
2. Add Doctors and Their Profiles
3. View Appointments
4. Delete Doctor Profile
5. Approve Appointments
6. Reject Appointments

Chapter 4B: Use case Description (For specific set of Use Cases)

1) Use Case: Login/Register (Admin/Patient)

1. Name: Login/Register

2. Actor(s):

- Primary Actor: Hospital Admin, Patient
- Supporting Actor: System

3. Goal: Allow hospital admins and patients to log in to their accounts or register for a new account.

4. Precondition(s):

- For login: The user must already have a registered account.
- For registration: Required user details must be provided (e.g., name, email, password).

5. Postcondition(s):

- For login: The user gains access to the system based on their role (admin/patient).
- For registration: A new user account is created, and the user can log in afterward.

6. Trigger:

- The user opens the system and selects the option to log in or register.

---

7. Main Flow (Basic Path for Login):

1. The user selects the "Login" option on the system interface.
2. System prompts the user to enter their credentials (email/username and password).
3. The user enters the credentials and submits the form.
4. The system validates the credentials.
5. If valid, the user is granted access to their dashboard based on their role (admin or patient).

---

8. Main Flow (Basic Path for Register):

1. The user selects the "Register" option on the system interface.
2. System prompts the user to enter the required details, such as:
    - Name
    - Email
    - Password
    - Contact information
    - (Optional) Additional details (e.g., role selection for admins or patients).
3. The user submits the registration form.

4. The system validates the information and creates a new account.
5. The user receives a confirmation message or email and is redirected to the login page.

---

9. Alternative Flows:

9.1. Invalid Login Credentials

- If the credentials are invalid, the system displays an error message: *"Invalid username or password. Please try again."*

9.2. Duplicate Registration

- If the email or username is already registered, the system displays a message: *"Account with this email already exists. Please log in or use a different email."*

9.3. System Error

- If the system fails to process the login or registration, the system notifies the user and logs the error for resolution.

---

10. Extensions:

- **Forgot Password**: Users can reset their password if they forget it by providing their registered email.
- **Two-Factor Authentication**: The system may prompt for a second layer of authentication (e.g., OTP) for added security.

2) Use Case: Book Appointment

1. Name: Book Appointment

2. Actor(s):

- Primary Actor: Patient
- Supporting Actor: System

3. Goal: Allow the patient to book an appointment with a doctor.

4. Precondition(s):

- Patient must be logged into the system.
- Doctor profiles must exist in the system.
- Available time slots must be defined.

5. Postcondition(s):

- Appointment is successfully booked and confirmed.
- Patient receives a notification of the confirmed appointment.

6. Trigger: Patient initiates the booking process from the system interface.

---

7. Main Flow (Basic Path):

1. Patient logs in and navigates to the "Book Appointment" section.
2. Patient searches for a doctor by name, department, or hospital.
3. System displays a list of doctors matching the search criteria.
4. Patient selects a doctor from the list.
5. System displays available time slots for the selected doctor.
6. Patient selects a preferred time slot.
7. System confirms the appointment and updates the database.
8. Patient receives a confirmation notification with appointment details.

---

8. Alternative Flows:

8.1. Doctor Not Found

- If no doctor matches the search criteria, the system notifies the patient and prompts to refine the search.

8.2. Time Slot Unavailable

- If the selected time slot is unavailable, the system suggests alternative slots.

8.3. System Error

- If the booking fails due to a system error, the system notifies the patient and logs the issue for resolution.

3) Use Case: View Medical History

1. Name: View Medical History

2. Actor(s):

- Primary Actor: Patient
- Supporting Actor: System

3. Goal: Allow the patient to view their past medical records, including prescriptions, diagnoses, and reports.

4. Precondition(s):

- Patient must be logged into the system.
- Medical history must exist in the database for the patient.

5. Postcondition(s):

- Patient views their complete medical history.
- No changes are made to the medical history data.

6. Trigger: Patient initiates the request to view medical history from the system interface.

---

7. Main Flow (Basic Path):

1. Patient logs in and navigates to the "Past appointments" section.
2. System fetches the patient's medical records from the database.
3. System displays the medical history, including:
    - Past appointments.
    - Prescriptions.
    - Diagnoses.
    - Reports.
4. Patient reviews the displayed medical history.

---

8. Alternative Flows:

8.1. No Medical History Available

- If no records exist, then the page will be blank.

8.2. System Error

- If an error occurs while fetching records, the system notifies the patient and logs the issue for resolution.

. Extensions:

- **Download Medical History**: Patient can download their medical records in a printable format (PDF).

4) Use Case: Find Doctor

1. Name: Find Doctor

2. Actor(s):

- Primary Actor: Patient
- Supporting Actor: System

3. Goal: Allow the patient to search for a doctor based on specific criteria.

4. Precondition(s):

- Patient must be logged into the system.
- Doctor profiles must exist in the system.

5. Postcondition(s):

- Patient receives a list of doctors matching the search criteria.

6. Trigger: Patient initiates a search for a doctor using the search feature.

---

7. Main Flow (Basic Path):

1. Patient logs in and navigates to the "Find Doctor" section.
2. Patient enters search criteria, such as:
    - Doctor’s name.
    - Department.

   - Hospital name.
3. System searches the database for doctors matching the input criteria.
4. System displays a list of doctors matching the search results.

---

8. Alternative Flows:

8.1. No Doctor Found

- If no doctor matches the search criteria, the system will with nothing to show.

8.2. System Error

- If the search fails due to a system issue, the system notifies the patient and logs the issue for resolution.

---

9. Extensions:

- **Refine Search Results**: Patient can refine the search by applying additional filters (e.g., availability, ratings).
- **Save Doctor Details**: Patient can save a doctor's details for future reference.

5) Use Case: View pending appointments

Use Case: View Pending Appointments

1. Name: View Pending Appointments

2. Actor(s):

- Primary Actor: Doctor, Patient
- Supporting Actor: System

3. Goal: Allow the doctor or patient to view a list of pending appointments.

4. Precondition(s):

- The actor (doctor or patient) must be logged into the system.
- Pending appointments must exist in the database.

5. Postcondition(s):

- The actor successfully views a list of pending appointments.

6. Trigger: The actor selects the "View Pending Appointments" option from the interface.

---

7. Main Flow (Basic Path):

1. The actor logs in and navigates to the "View Pending Appointments" section.
2. The system fetches a list of pending appointments from the database.
3. The system displays the pending appointments, including details such as:
    - Appointment date and time.
    - Patient's name (for doctors).
    - Doctor's name and specialization (for patients).
    - Appointment status (e.g., pending approval).

---

8. Alternative Flows:

8.1. No Pending Appointments

- If no pending appointments exist, the system will show blank page.

8.2. System Error

- If the system encounters an error while retrieving data, it notifies the actor and logs the issue for resolution.

6) Use Case: Record Patient Details

1. Name: Record Patient Details (Add Prescription/Diagnosis)

2. Actor(s):

- Primary Actor: Doctor
- Supporting Actor: System

3. Goal: Allow the doctor to record and update patient details, including adding prescriptions, diagnoses, and reports.

4. Precondition(s):

- Doctor must be logged into the system.
- The patient must have an existing appointment record.

5. Postcondition(s):

- Patient details are updated with the prescription, diagnosis, and/or reports.
- The system saves the updated details in the patient's medical history.

6. Trigger: Doctor opens a patient's appointment record and selects the option to record details.

---

7. Main Flow (Basic Path):

1. Doctor logs in and navigates to the patient's appointment record.
2. Doctor selects the "Record Patient Details" option.
3. System displays the patient’s details, including prior prescriptions, diagnoses, and reports.
4. Doctor adds new information:
   - Prescription details.
   - Diagnosis information.
   - Relevant reports.
5. Doctor saves the entered details.
6. System updates the patient's medical history with the new information.
7. The doctor receives a confirmation that the details have been saved.

---

8. Alternative Flows:

8.1. Missing Patient Record

- If no patient record is found, the system prompts the doctor to check the patient's ID or creates a new record.

8.2. System Error

- If there is a failure while saving the data, the system notifies the doctor and logs the error for resolution.

7) Use Case: Add Doctor

Use Case: Add Doctor (Hospital Admin)

1. Name: Add Doctor

2. Actor(s):

- Primary Actor: Hospital Admin
- Supporting Actor: System

3. Goal: Allow the hospital admin to add a new doctor and their profile into the system.

4. Precondition(s):

- Hospital admin must be logged into the system.
- Doctor profile information (name, specialization, availability, etc.) must be available.

5. Postcondition(s):

- A new doctor profile is successfully added to the system.
- The doctor's profile becomes available for patient searches.

6. Trigger: Hospital admin initiates the process of adding a new doctor through the system interface.

---

7. Main Flow (Basic Path):

1. Hospital admin logs into the system.
2. Admin navigates to the "Add Doctor" section.
3. System prompts the admin to enter the doctor's details:
    - Doctor's name.

   - Specialization/department.
   - Contact details (phone number, email).
   - Work schedule/availability.
   - Other necessary information (e.g., qualifications, certifications).
4. Admin fills in the required fields and submits the form.
5. System validates the entered information and confirms that the data is correct.
6. Admin confirms the addition of the doctor's profile.
7. System stores the doctor's details in the database and updates the doctor list.
8. The doctor profile is now available for patient searches.

---

8. Alternative Flows:

8.1. Missing Information

- If any required information is missing, the system prompts the admin to fill in the missing fields.

8.2. Invalid Data

- If the data entered is invalid (e.g., incorrect format for contact information), the system notifies the admin of the error and prompts for correction.

---

9. Extensions:

- **Delete Doctor Profile**: Admin can delete a doctor's profile if needed, removing it from the system and patient searches.

8) Use Case: Approve/Reject Appointment

Use Case: Approve/Reject Appointment (Hospital Admin)

1. Name: Approve/Reject Appointment

2. Actor(s):

- Primary Actor: Hospital Admin
- Supporting Actor: System

3. Goal: Allow the hospital admin to approve or reject appointment requests made by patients.

4. Precondition(s):

- Hospital admin must be logged into the system.
- Pending appointment requests must exist in the system.

5. Postcondition(s):

- Approved appointments are confirmed and scheduled.
- Rejected appointments are marked as declined, and patients are notified.

6. Trigger: Hospital admin accesses the list of pending appointments.

---

7. Main Flow (Basic Path):

1. Hospital admin logs into the system.
2. Admin navigates to the "Pending Appointments" section.
3. System displays a list of pending appointment requests, including:
    - Patient details (name, contact info).
    - Doctor details (name, specialization).
    - Requested date and time.
4. Admin reviews the appointment details.
5. Admin selects an appointment to approve or reject.
6. If approved, the system updates the appointment status to "Approved" and notifies the patient and doctor.

7. If rejected, the system updates the appointment status to "Rejected" and notifies the patient with the reason for rejection (if provided).

---

8. Alternative Flows:

8.1. No Pending Appointments

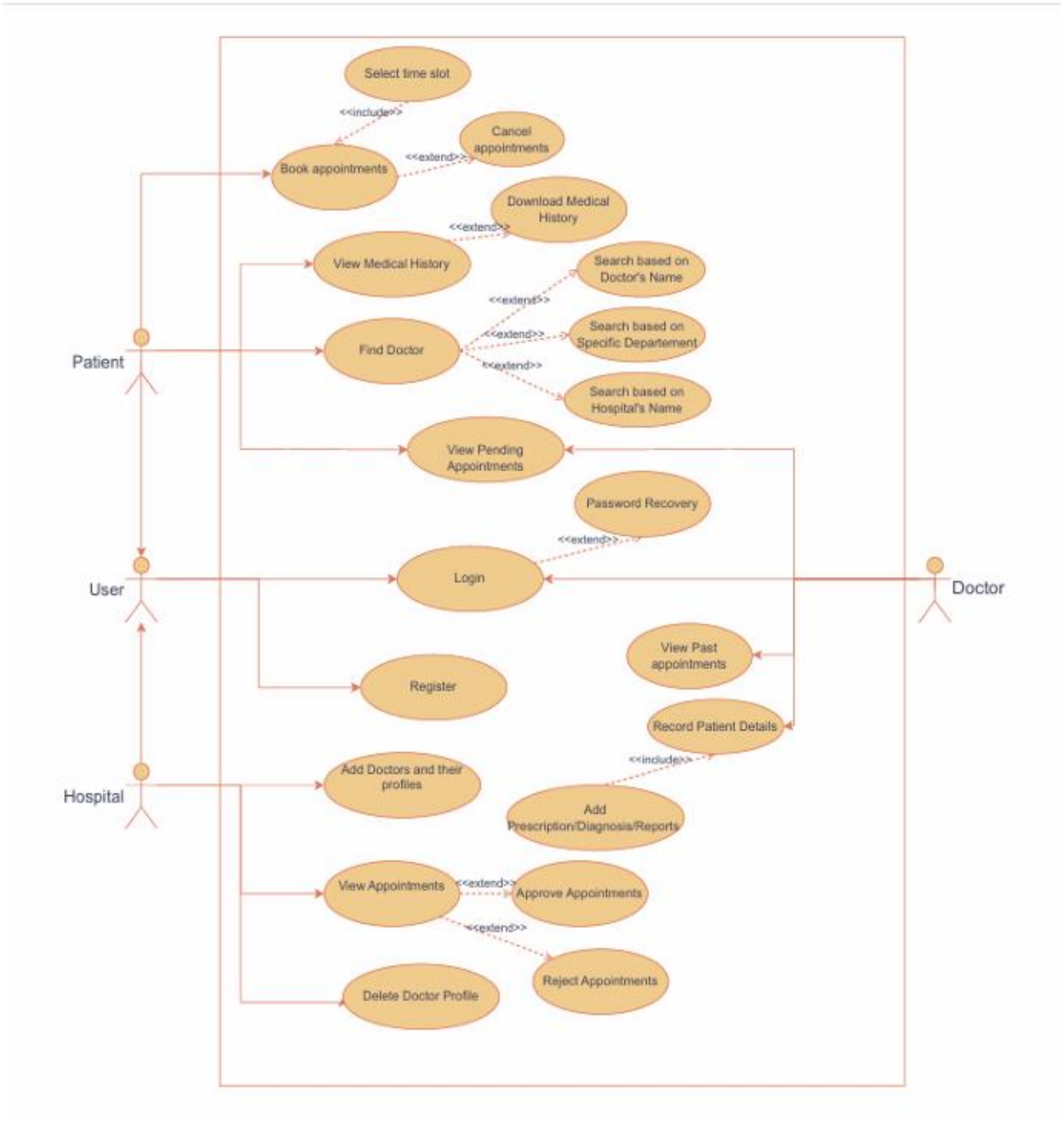
- If no pending appointments exist, the system will show blank page.

8.2. Invalid Request

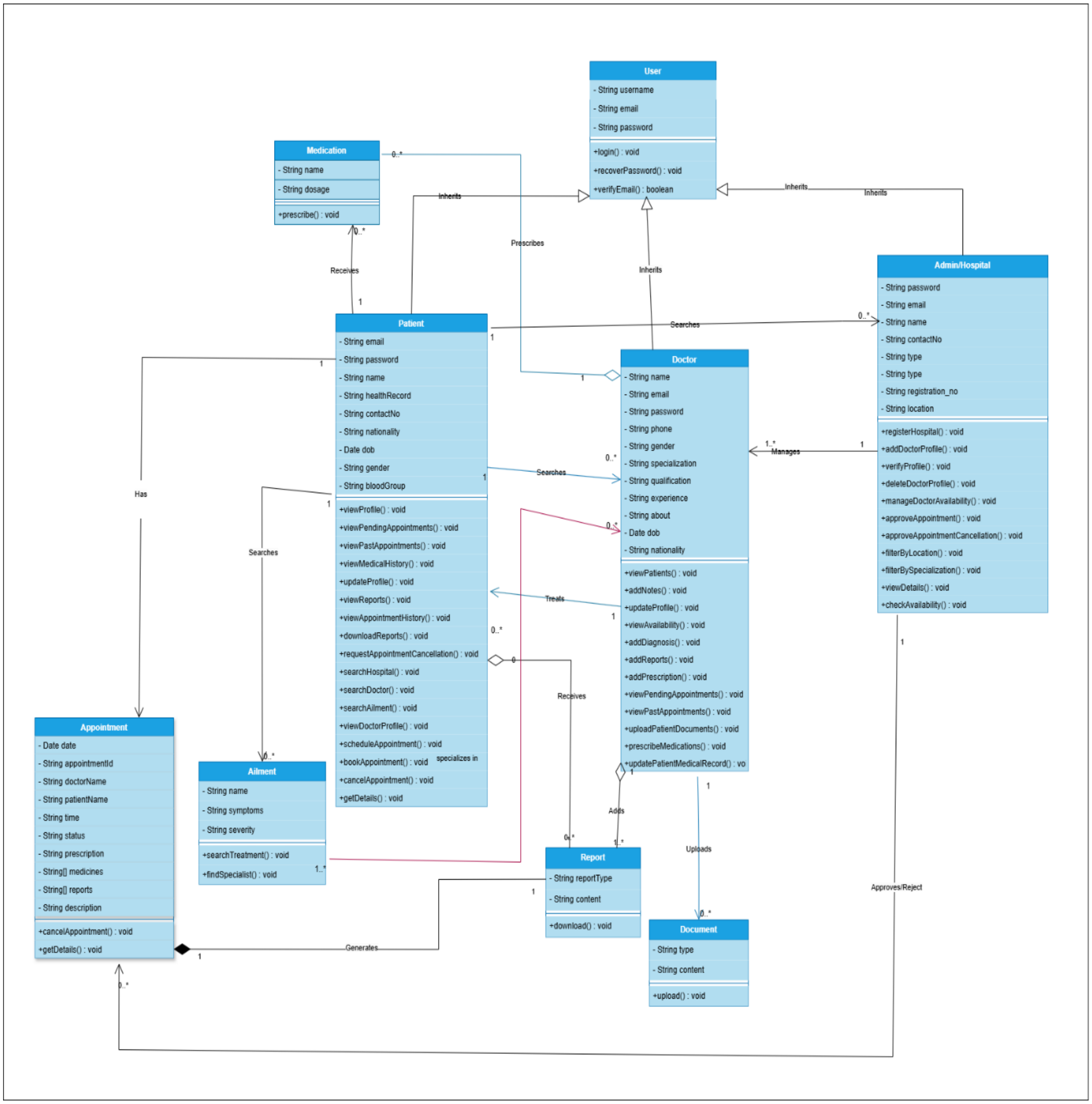
- If the appointment request has already been processed or canceled, the system notifies the admin.

Chapter 5:System Behaviour

5.1 General Use Case Diagram



5.2 General Class Diagram



5.3 :Individual Sprints & their

UseCase,Sequence and Class Diagrams:

Updated Sprints:

Sprint 1 (Patient/Admin):

Login/Register: Enables patients and admins to securely create accounts or log in to the platform for personalized access.
Forgot Password: Allows users to recover their account credentials via email or other recovery methods.

Sprint 2 (Patient Side):

View Medical History: Provides patients with access to their medical records, including past diagnoses, prescriptions, and treatments.
View Pending Appointments: Displays a list of upcoming appointments for better scheduling and planning.

Sprint 3 (Admin Side):

Add Doctors and Their Profiles: Enables admins to onboard new doctors, inputting their qualifications, specializations, and availability.
Approve Appointments: Allows admins to manage and approve patient appointment requests for efficient scheduling.

Sprint 4 (Doctor Side):

Add Diagnosis, Reports, Prescriptions: Facilitates doctors in recording diagnoses and uploading relevant reports and prescriptions for patients.

View Pending Appointments: Lists all upcoming appointments for better time management.
View Past Appointments: Provides a summary of previous consultations to aid in continuity of care.

Sprint 5 (Patient Side):

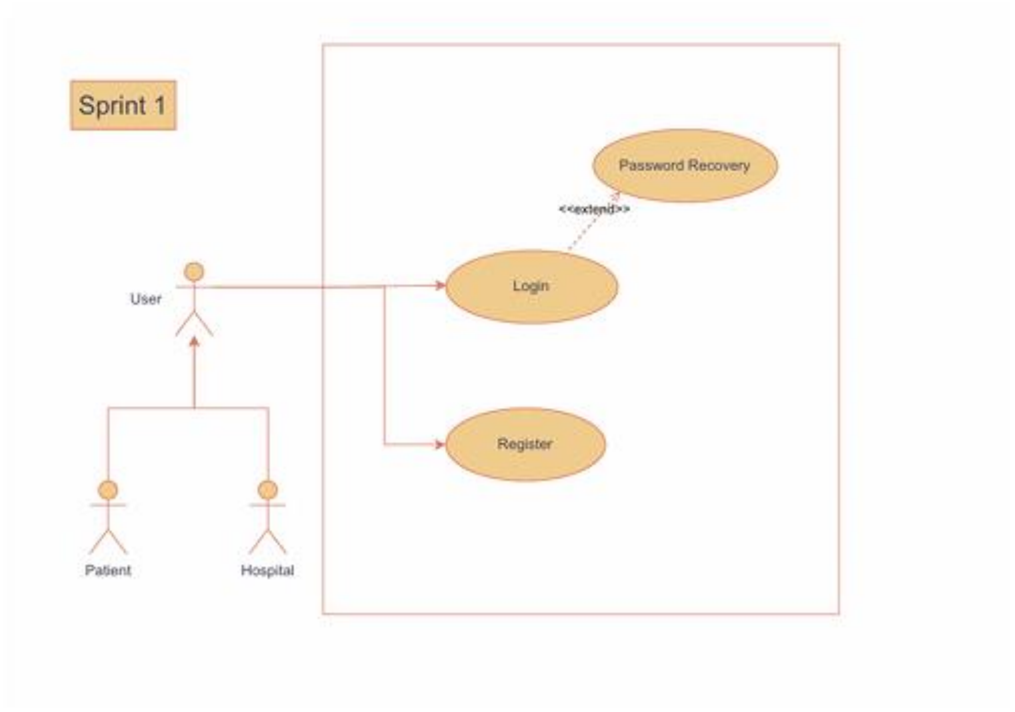
Search by Hospital: Helps patients locate healthcare providers within specific hospitals.
Search by Doctor: Enables filtering of doctors based on name, specialization, or availability.
Search by Ailment: Allows patients to find relevant doctors or services based on their medical conditions.

Sprint 6 (Patient Side):

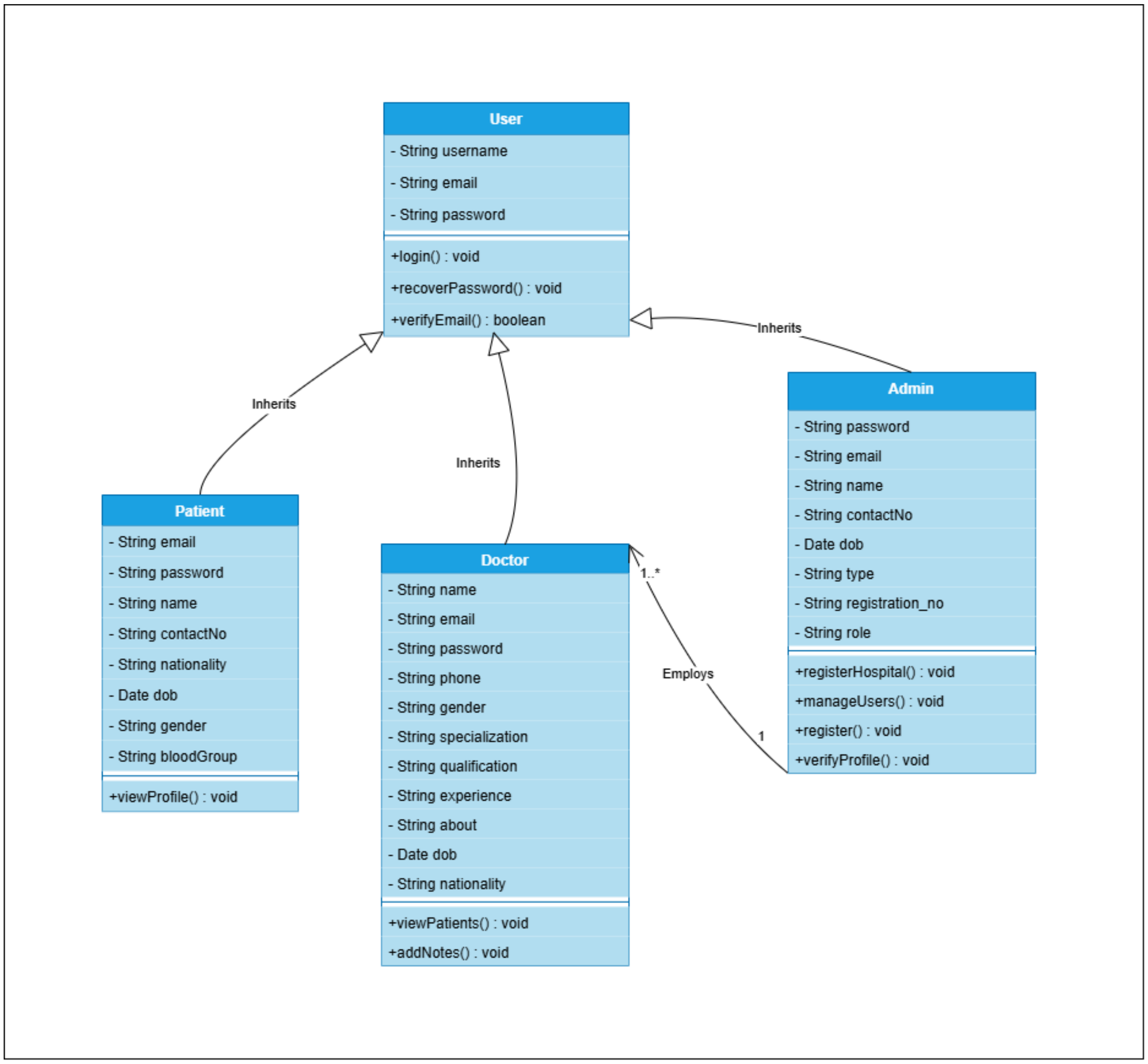
Book Appointment: Lets patients schedule appointments with preferred doctors, ensuring a streamlined booking process.
Select Time Slot: Offers the flexibility to choose from available time slots based on the patient's and doctor's schedules.

1) Sprint -1

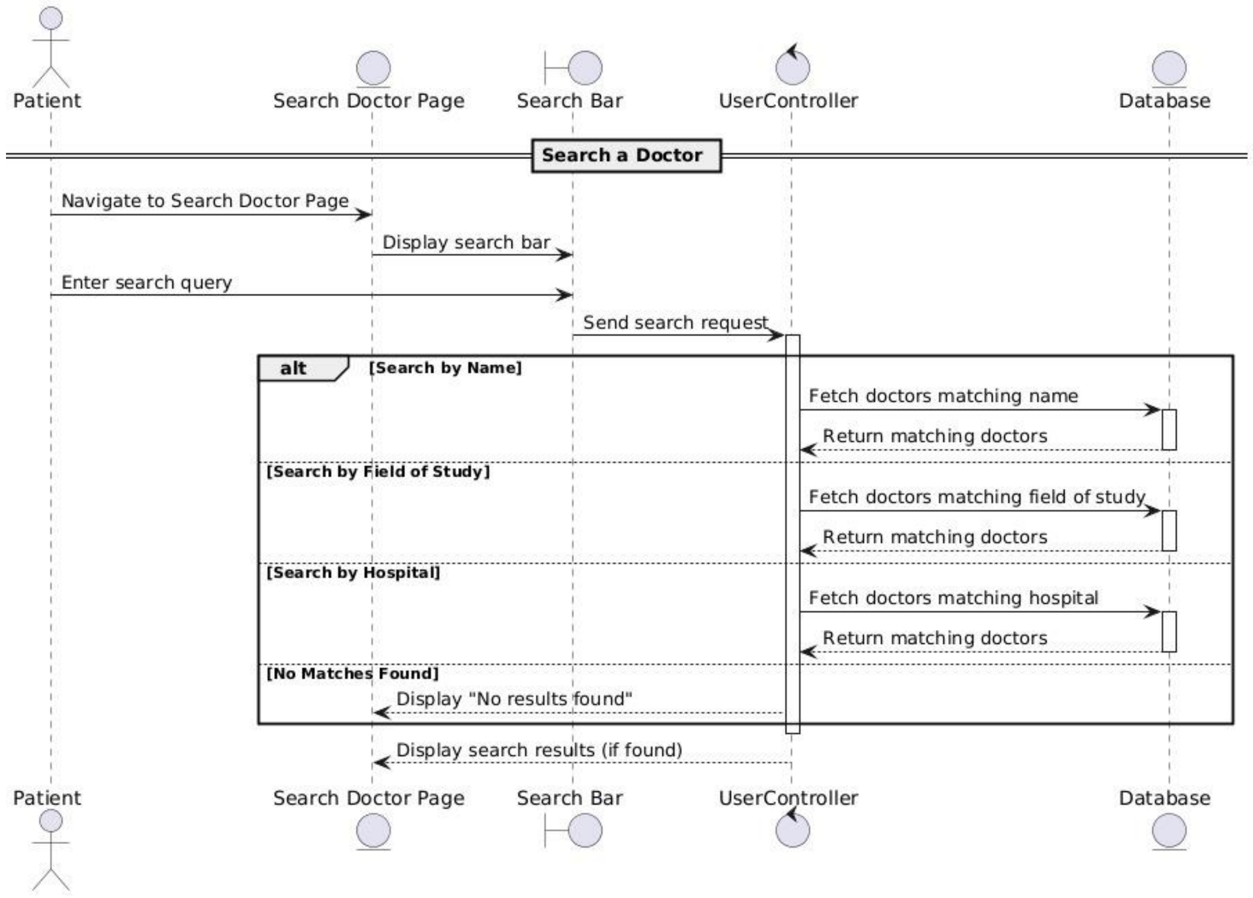
UseCase Diagram



Class Diagram

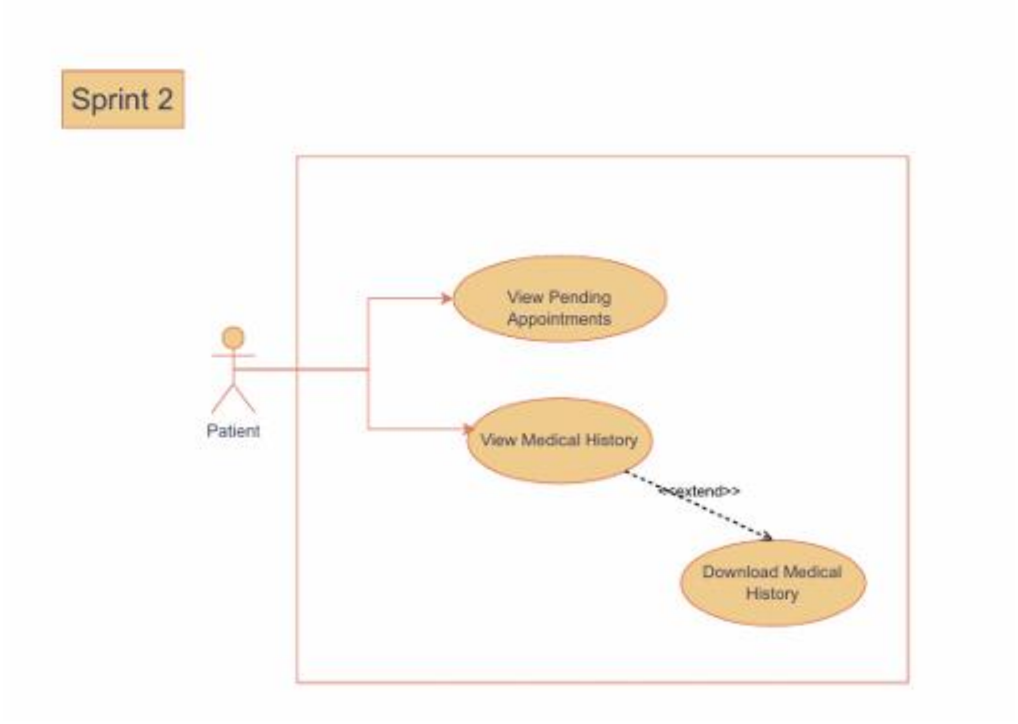


Sequence Diagram

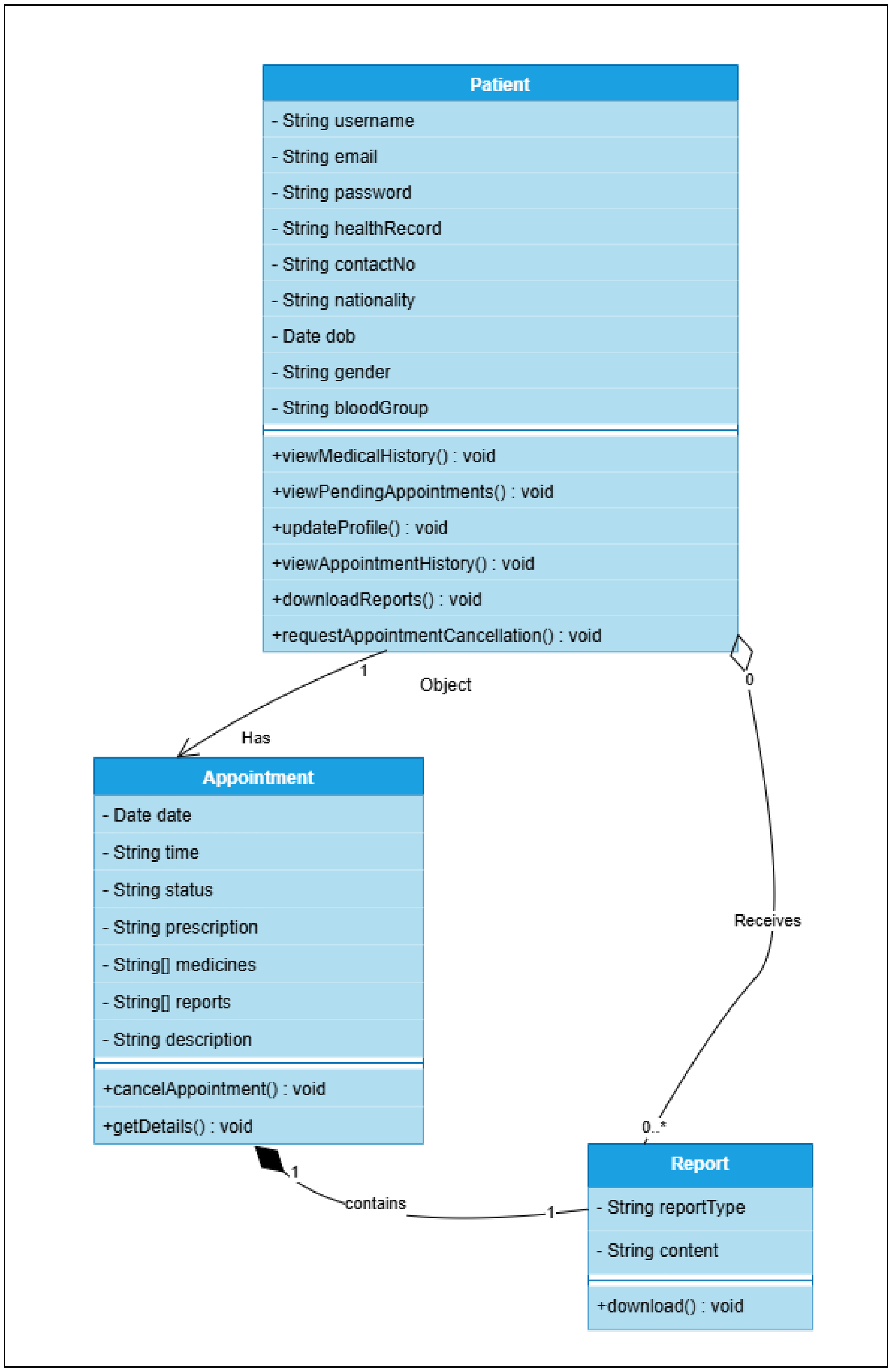


2) Sprint -2

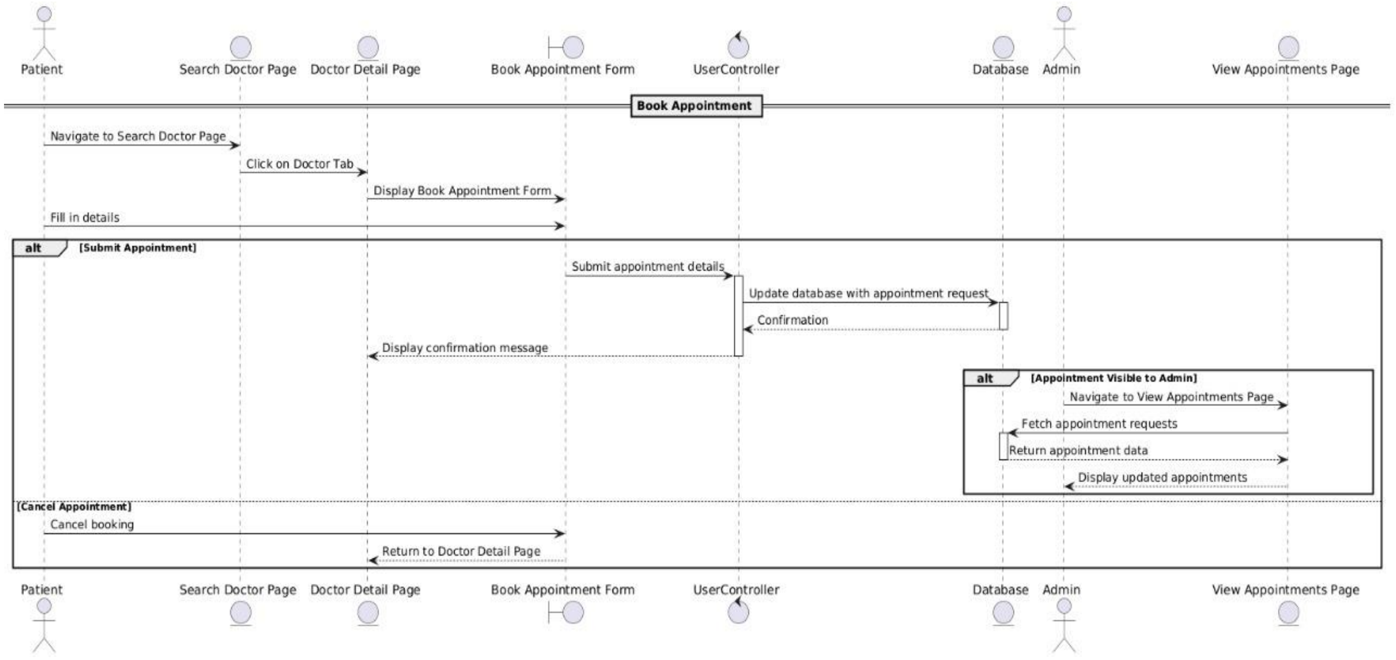
UseCase Diagram



Class Diagram

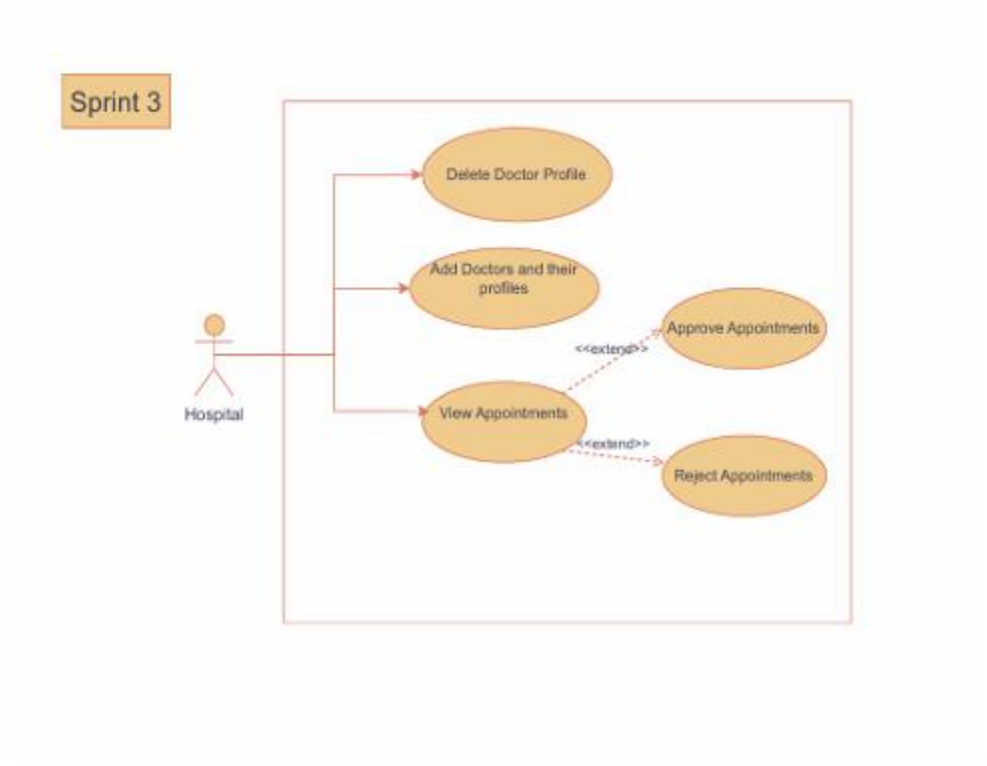


Sequence Diagram

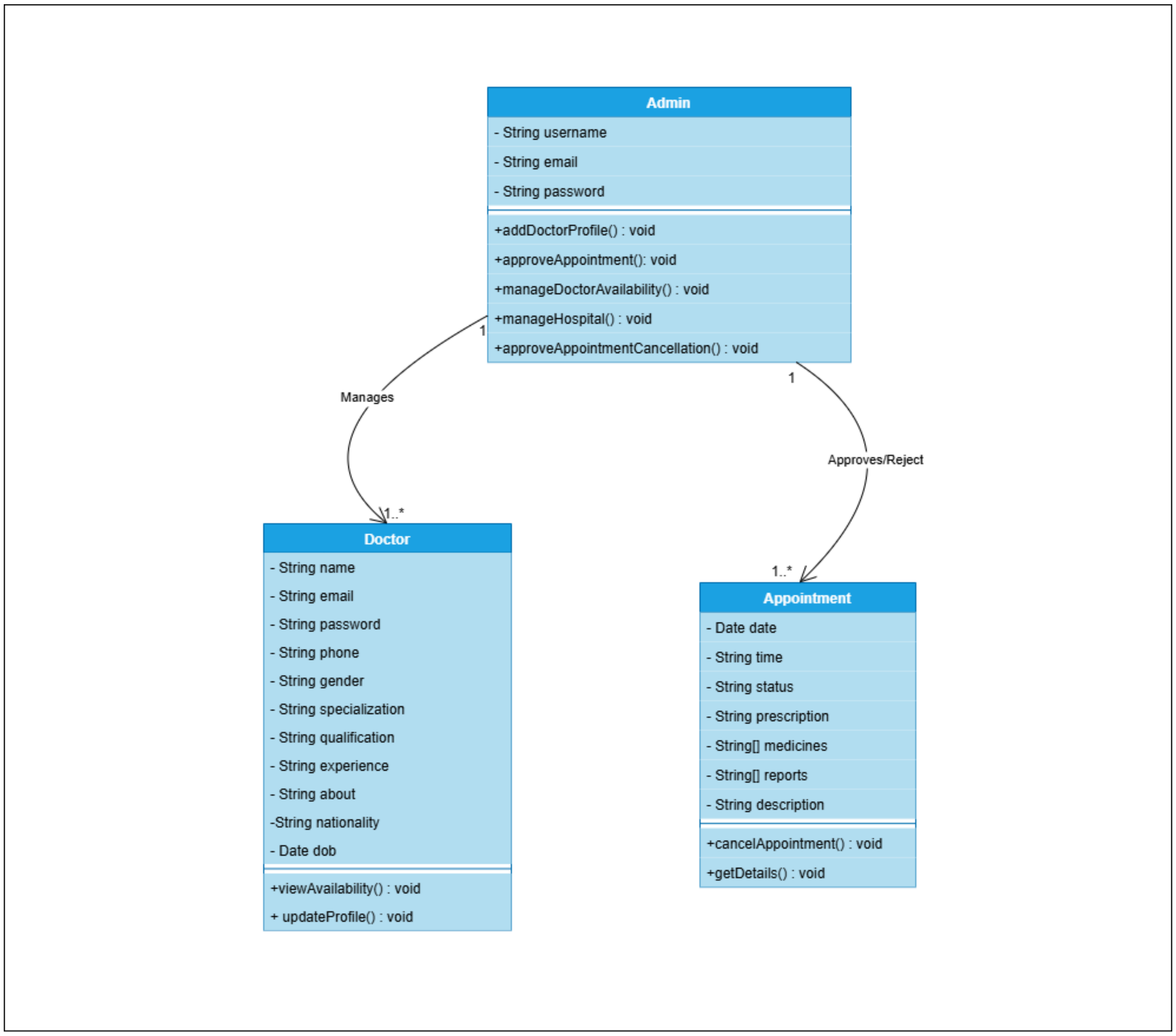

Patient
Search Doctor Page
Doctor Detail Page
Book Appointment Form
UserController
Database
Admin
View Appointments Page
Book Appointment
Navigate to Search Doctor Page
Click on Doctor Tab
Display Book Appointment Form
Fill in details
alt [Submit Appointment]
Submit appointment details
Update database with appointment request
Confirmation
Display confirmation message
alt [Appointment Visible to Admin]
Navigate to View Appointments Page
Fetch appointment requests
Return appointment data
Display updated appointments
[Cancel Appointment]
Cancel booking
Return to Doctor Detail Page

3) Sprint -3

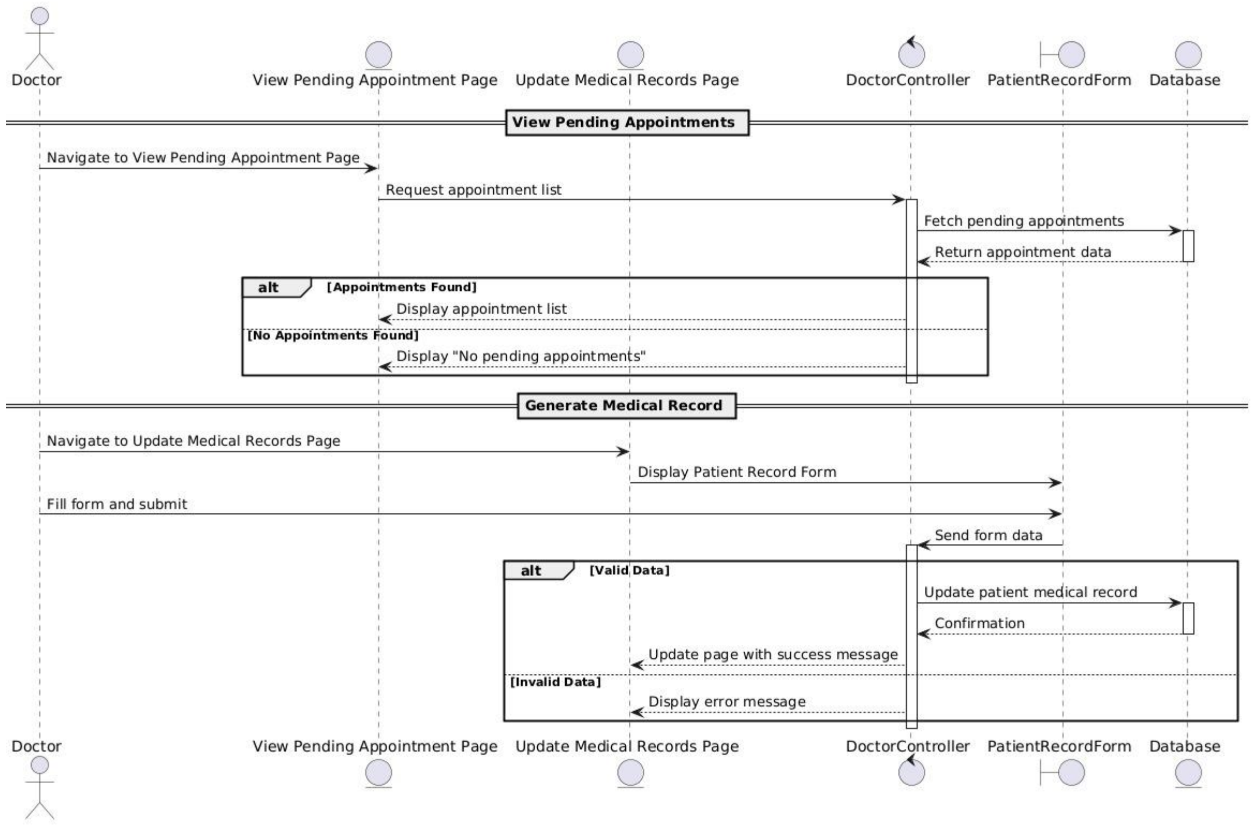
UseCase Diagram



Class Diagram



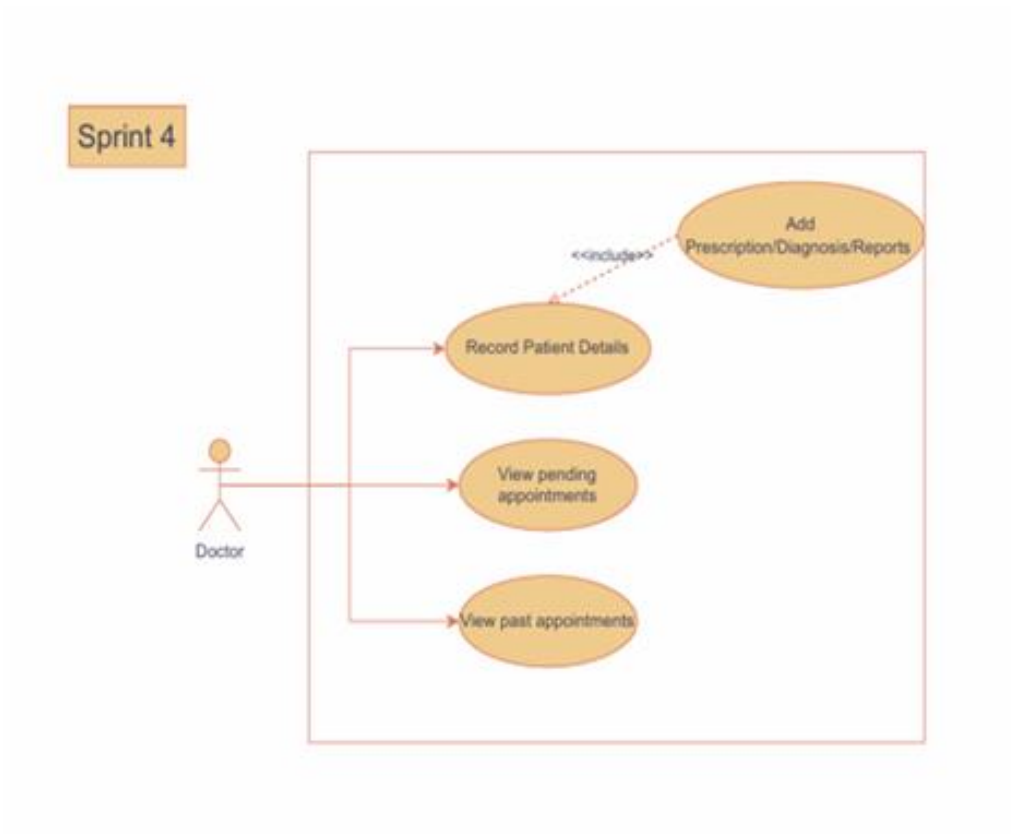
Sequence Diagram



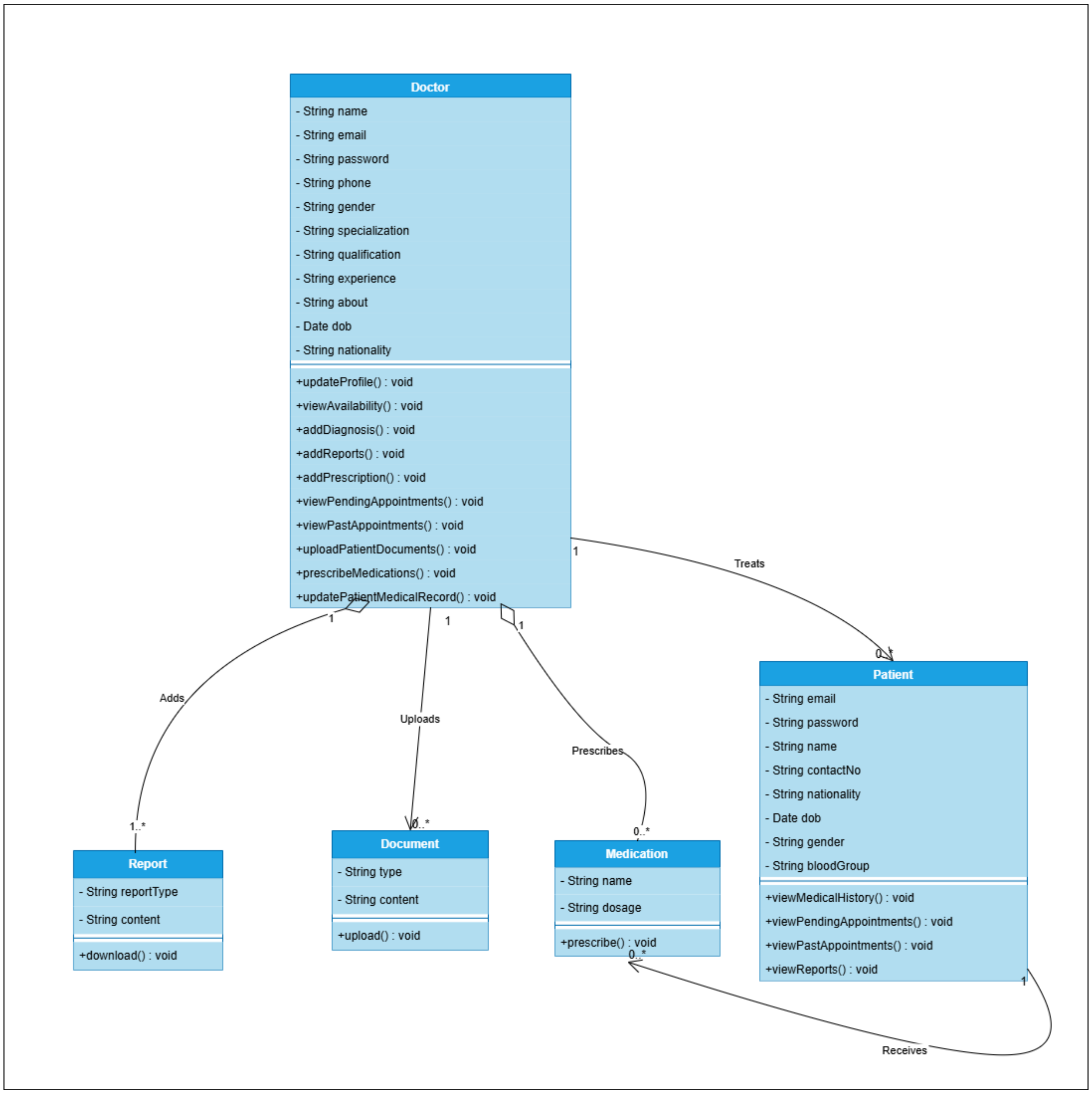
Doctor
View Pending Appointment Page
Update Medical Records Page
DoctorController
PatientRecordForm
Database
View Pending Appointments
Navigate to View Pending Appointment Page
Request appointment list
Fetch pending appointments
Return appointment data
alt
[Appointments Found]
Display appointment list
[No Appointments Found]
Display "No pending appointments"
Generate Medical Record
Navigate to Update Medical Records Page
Display Patient Record Form
Fill form and submit
Send form data
alt
[Valid Data]
Update patient medical record
Confirmation
Update page with success message
[Invalid Data]
Display error message

4) Sprint - 4

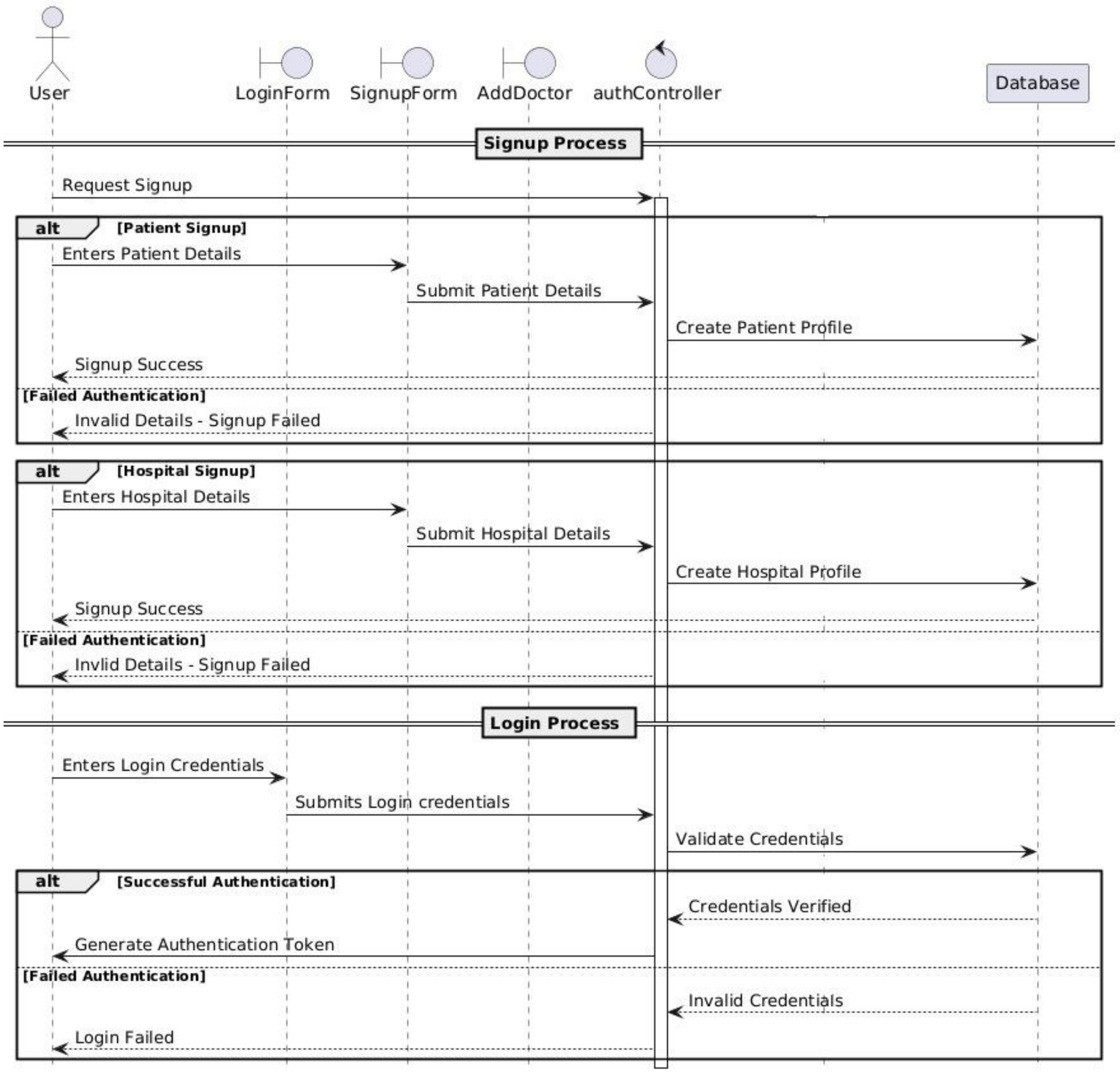
UseCase Diagram



Class Diagram

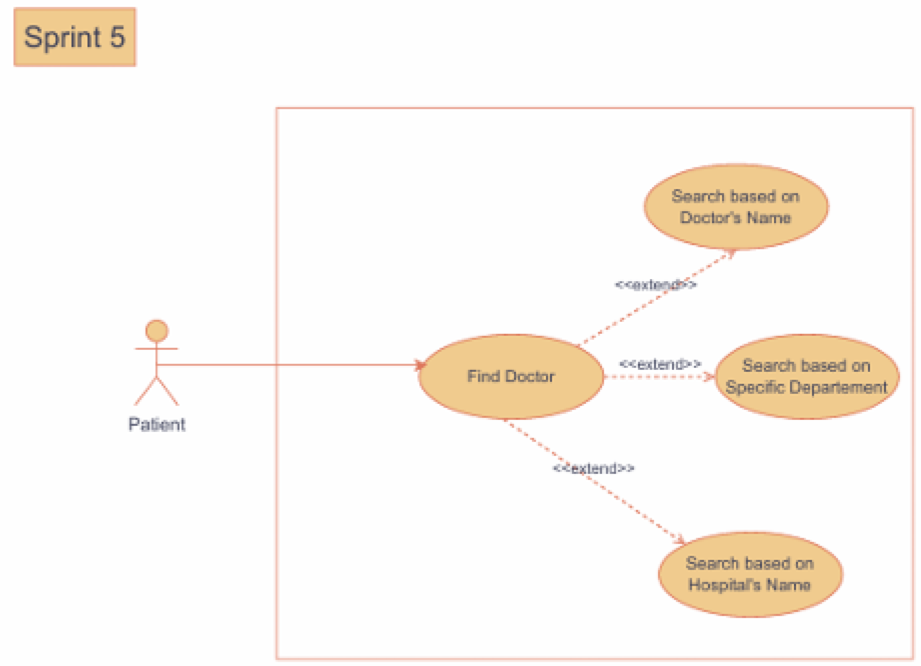


Sequence Diagram

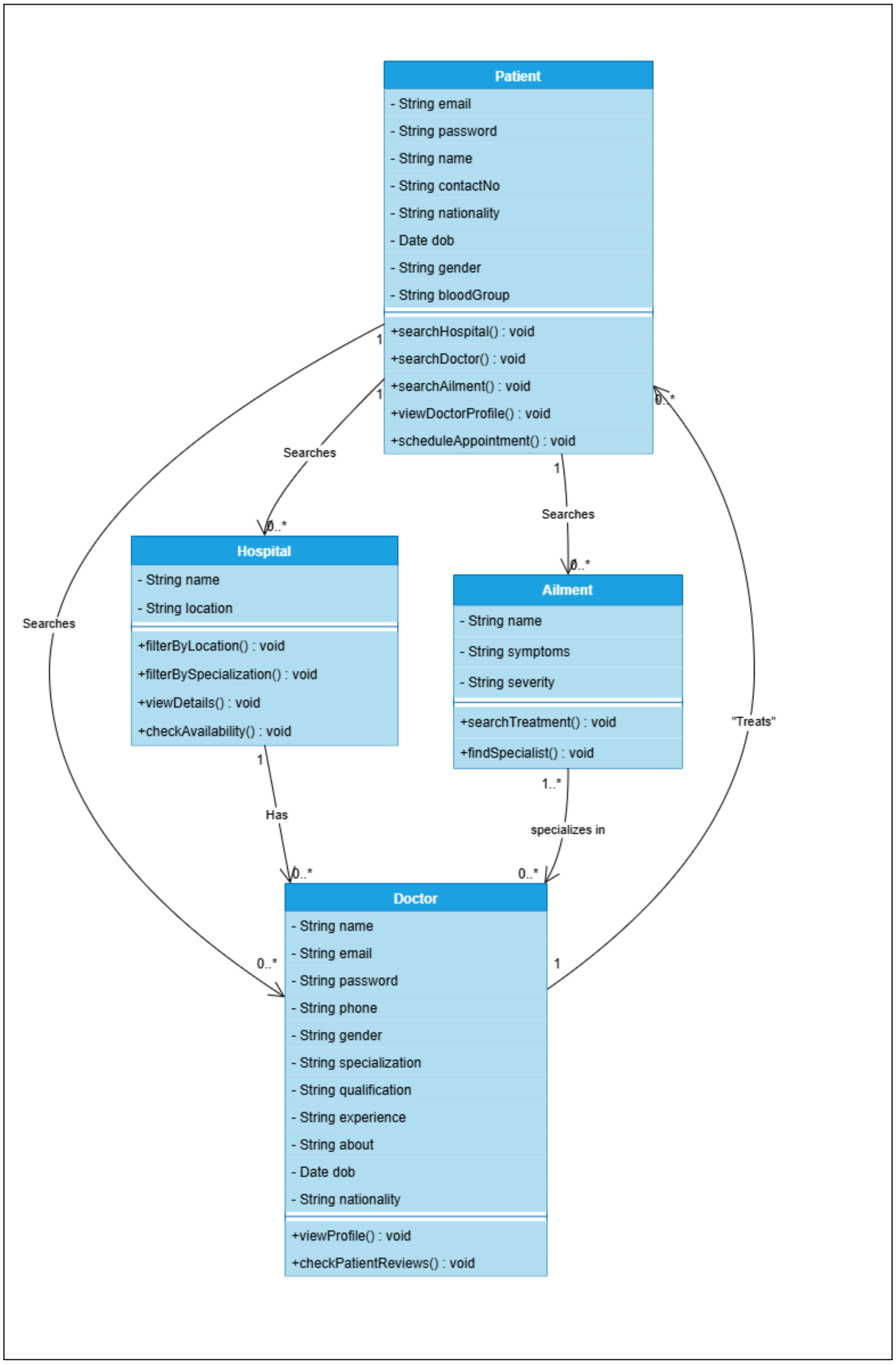


5) Sprint - 5

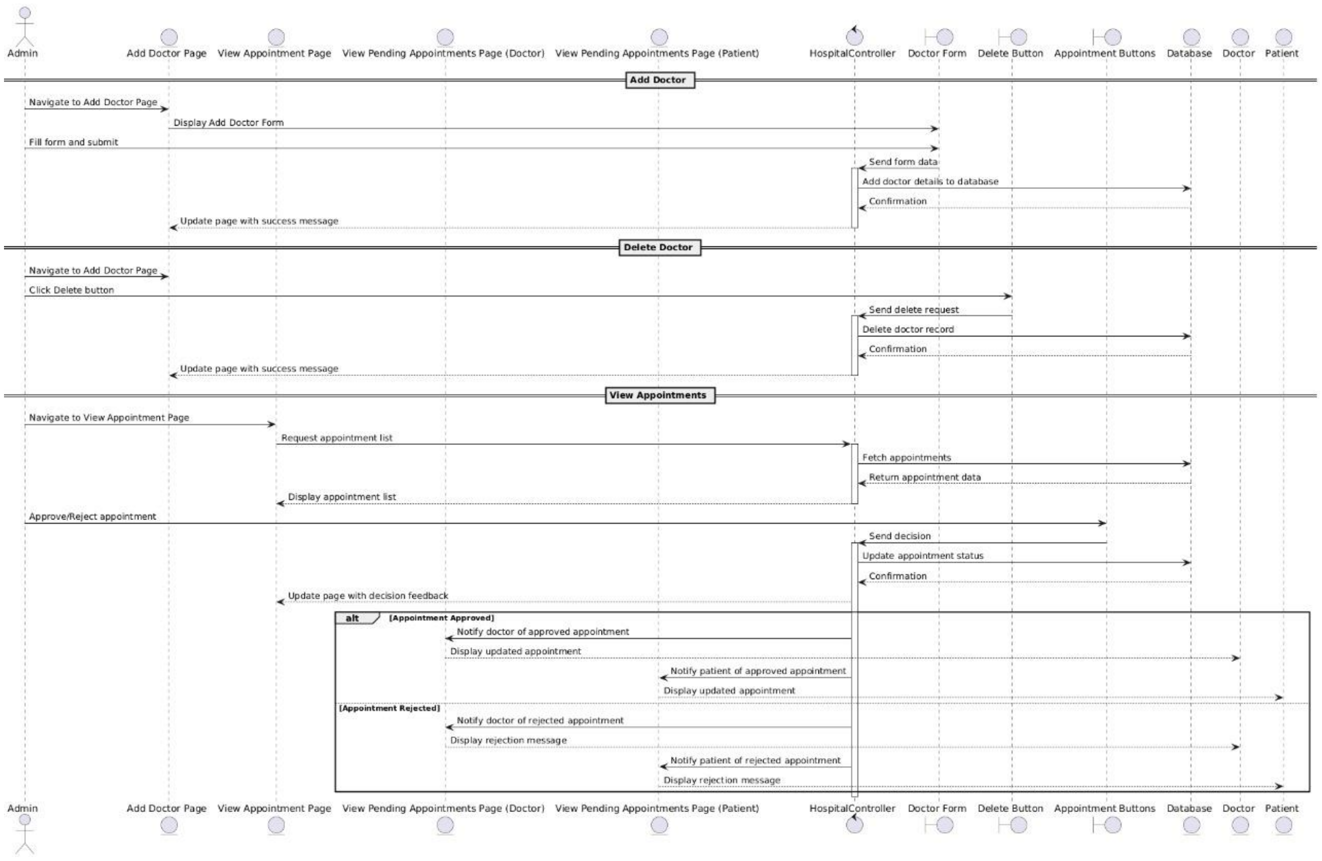
UseCase Diagram


Sprint 5
Patient
Find Doctor
<<extend>>
Search based on Doctor's Name
<<extend>>
Search based on Specific Departement
<<extend>>
Search based on Hospital's Name

Class Diagram



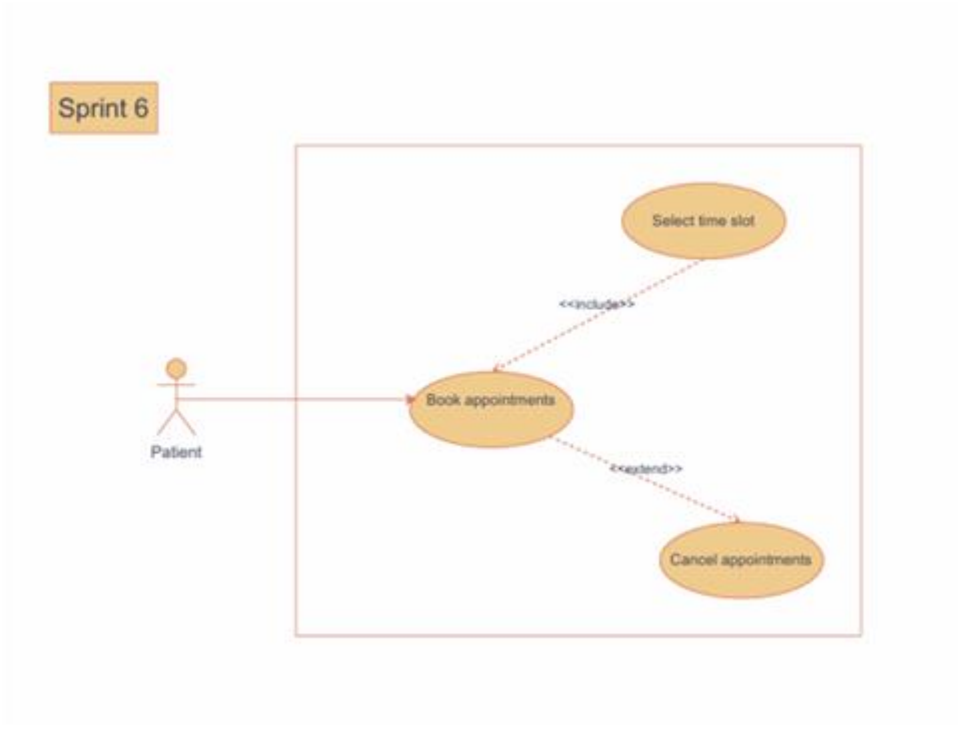
Sequence Diagram



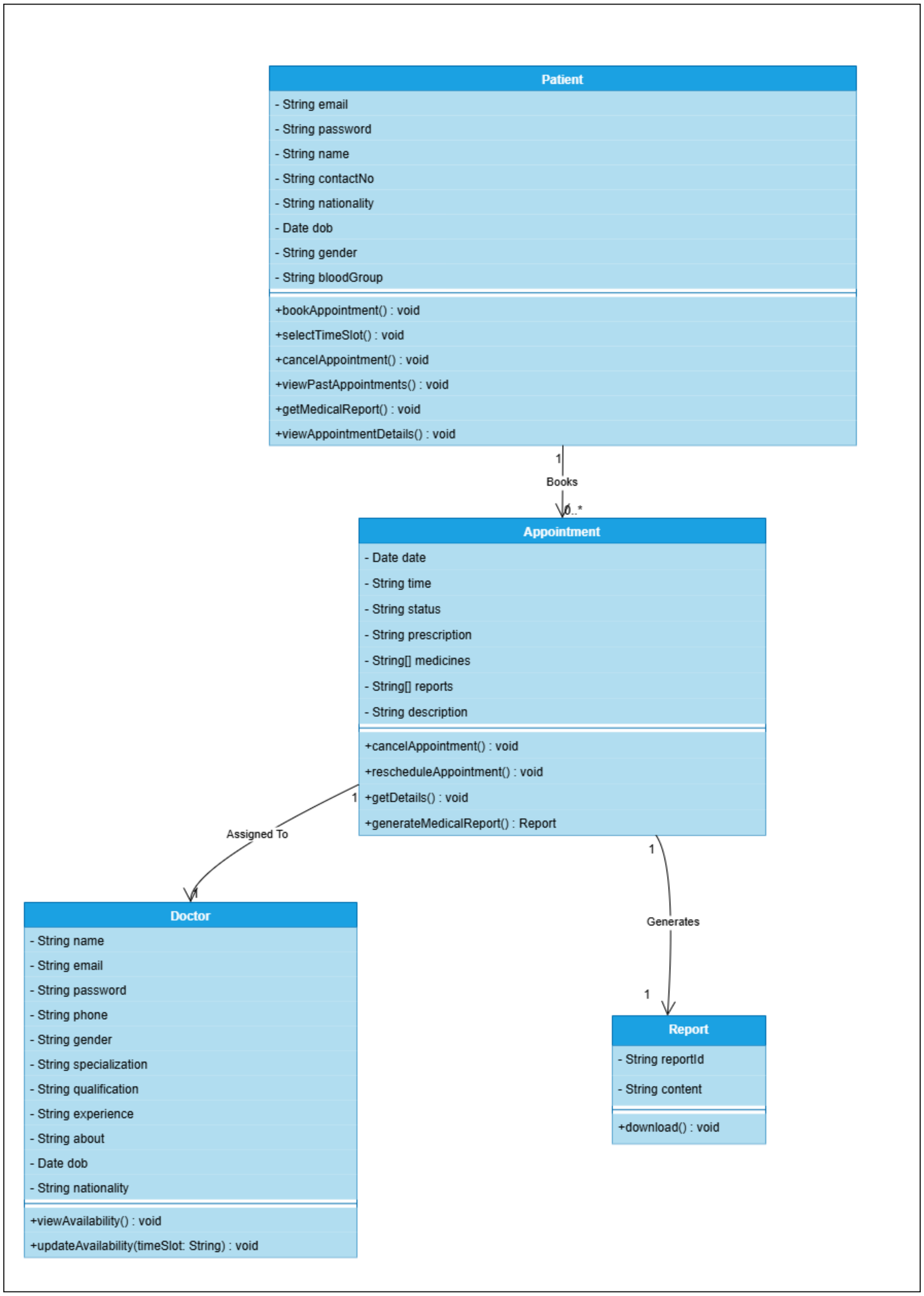
Admin
Add Doctor Page
View Appointment Page
View Pending Appointments Page (Doctor)
View Pending Appointments Page (Patient)
HospitalController
Doctor Form
Delete Button
Appointment Buttons
Database
Doctor
Patient
Add Doctor
Navigate to Add Doctor Page
Display Add Doctor Form
Fill form and submit
Send form data
Add doctor details to database
Confirmation
Update page with success message
Delete Doctor
Navigate to Add Doctor Page
Click Delete button
Send delete request
Delete doctor record
Confirmation
Update page with success message
View Appointments
Navigate to View Appointment Page
Request appointment list
Fetch appointments
Return appointment data
Display appointment list
Approve/Reject appointment
Send decision
Update appointment status
Confirmation
Update page with decision feedback
alt
[Appointment Approved]
Notify doctor of approved appointment
Display updated appointment
Notify patient of approved appointment
Display updated appointment
[Appointment Rejected]
Notify doctor of rejected appointment
Display rejection message
Notify patient of rejected appointment
Display rejection message

5) Sprint - 6

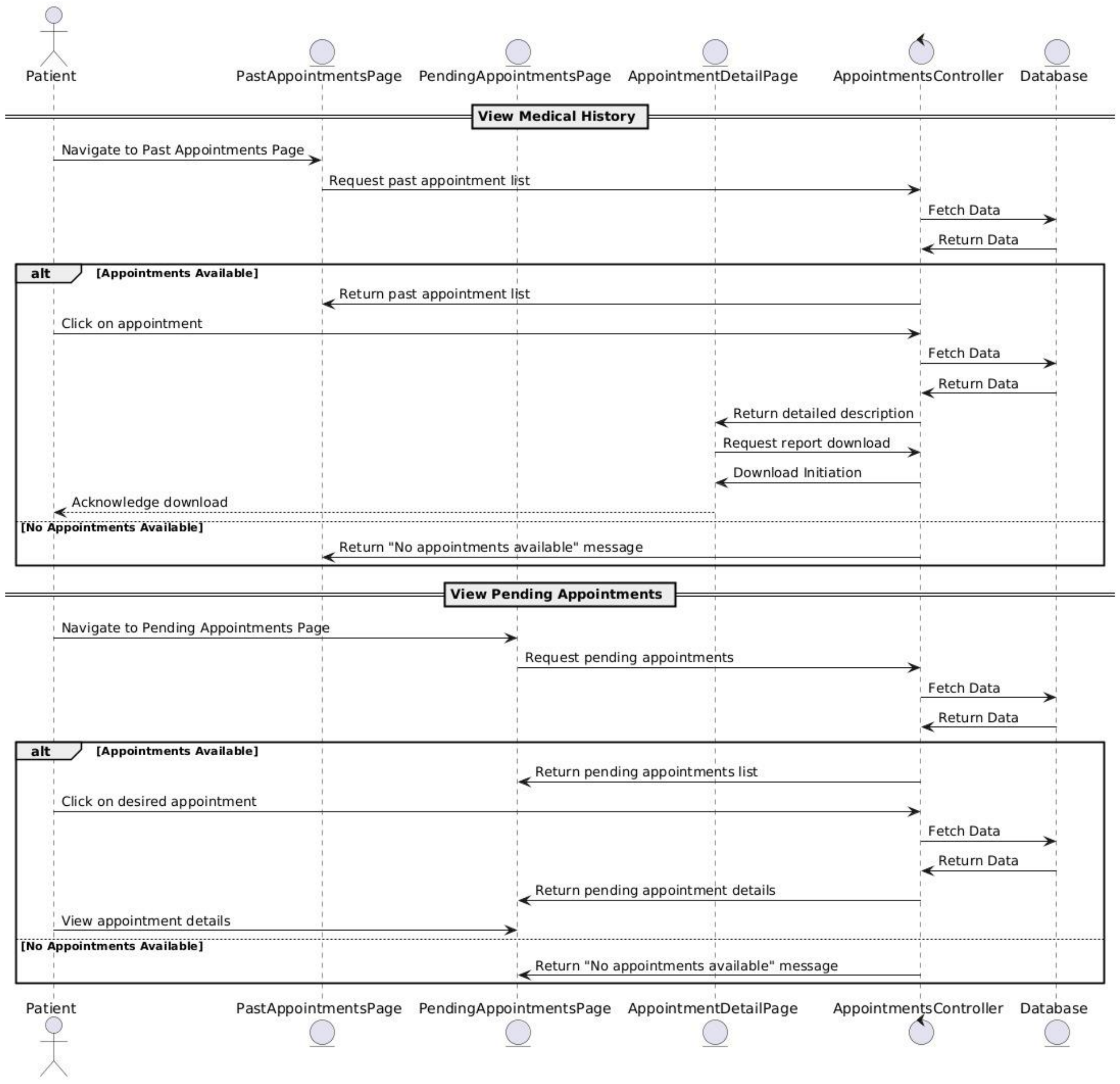
UseCase Diagram



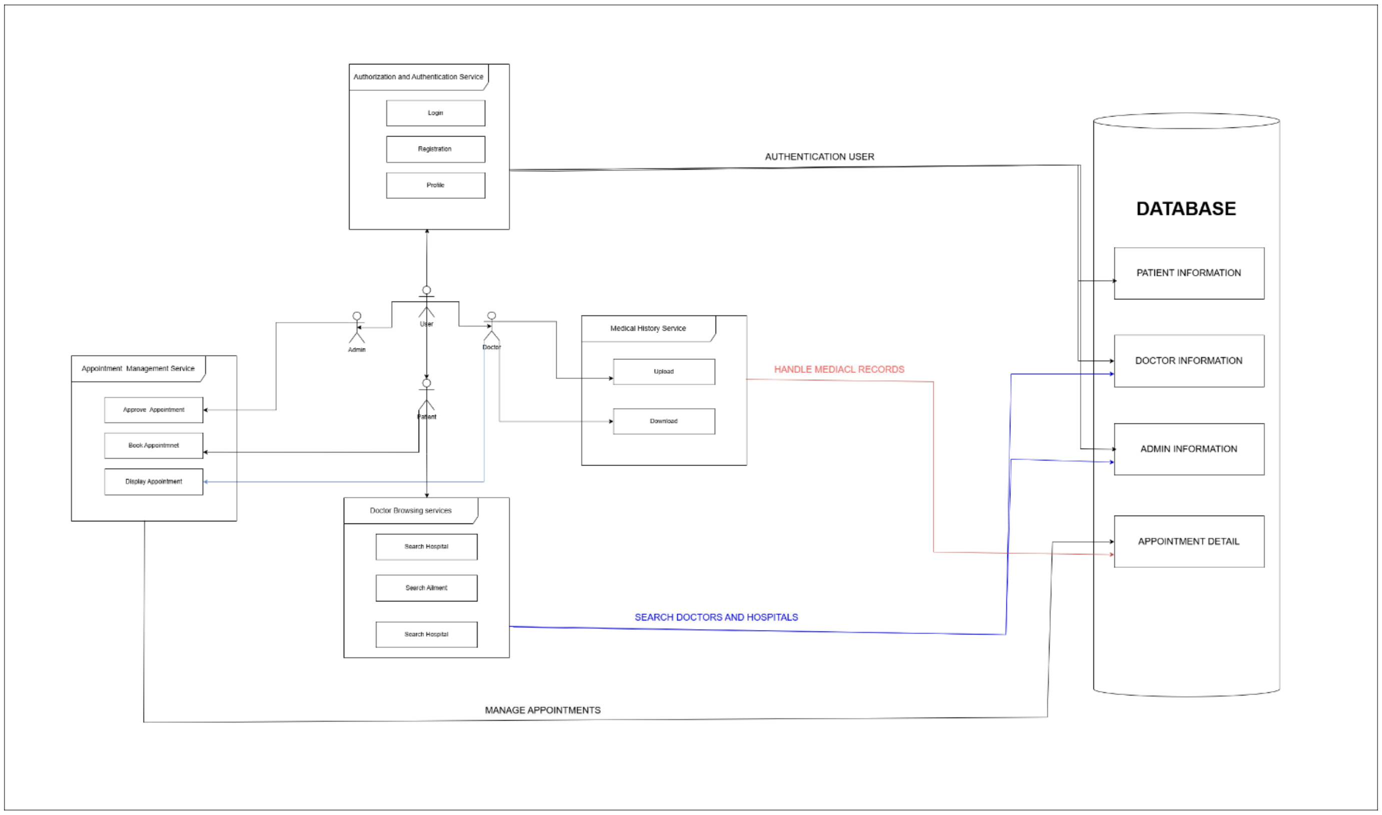
Class Diagram



Sequence Diagram



Chapter 6: System Design



Chapter 7:Proposed Model for Development

7.1 Chosen Development Model

For the development of the **Patient Management System**, the **Agile methodology** has been selected due to its flexibility and iterative approach. This model facilitates continuous feedback and incremental refinement, ensuring that the system evolves in alignment with the dynamic needs of all stakeholders. By embracing Agile, the team can consistently enhance key functionalities—such as appointment scheduling, medical record management, and doctor-patient interactions—through regular sprints. This adaptive process allows the system to remain responsive to user requirements and ensures timely delivery of high-quality features.

7.2 Requirements Gathering

Objective: To engage with stakeholders—patients, doctors, and hospital administrators—to thoroughly understand the needs and essential features of the Patient Management System.

Activities:

- Develop initial prototypes to visualize core system functionalities, such as user registration, appointment management, and medical history tracking, for early-stage validation.
- Facilitate user testing sessions with the prototypes to refine requirements and ensure alignment with user expectations, focusing on usability and functionality.
  Output: A comprehensive, user-centric set of requirements encompassing essential features such as registration, authentication, appointment scheduling, medical history management, and real-time notifications.

7.3 Sprint Planning

Objective: To translate the gathered requirements into actionable tasks, prioritize them based on business value and dependencies, and establish a clear roadmap for development.

Activities:

- Decompose requirements into manageable tasks and group them into well-defined user stories (e.g., patient registration, doctor scheduling, appointment booking).
- Prioritize tasks for each sprint, ensuring that the work is distributed efficiently across teams and that each sprint contributes significantly to the project's objectives.
- Define specific sprint goals, targeting the delivery of key functionalities (e.g., building the doctor search API, implementing patient appointment management).
  Output: A meticulously organized sprint backlog with clearly prioritized tasks, each contributing to the timely delivery of functional milestones.

7.4 Development and Testing

Objective: To systematically develop and rigorously test each feature, ensuring optimal performance, security, and seamless integration across the system.

Activities:

- Develop the designated features for each sprint, including core backend functionalities such as registration, authentication, appointment management, and medical history updates.

- Implement unit tests to verify that each individual component performs as expected.
- Perform integration testing to ensure that newly developed features work cohesively with existing system components, fostering a unified user experience.
- Conduct comprehensive code reviews to maintain high coding standards, improve code quality, and promptly identify any potential issues.
  Output: Incrementally developed, thoroughly tested, and quality-assured features that are added to the system, enhancing its functionality at every stage.

7.5 Sprint Review

Objective: To present the progress made during each sprint to stakeholders, gather feedback, and make necessary refinements to ensure the system meets user expectations.

Activities:

- Demonstrate completed features—such as patient registration, doctor profile management, and appointment approval workflows—to stakeholders for review and validation.

- Solicit feedback from end-users to assess the usability, functionality, and overall satisfaction with the system.
- Analyze the feedback and identify potential improvements, adjusting the development approach as necessary to address any gaps or areas of concern. **Output:** Constructive feedback from stakeholders that drives continuous improvement and helps guide the refinement of the system in subsequent sprints.

7.6 Sprint Retrospective

Objective: To reflect on each sprint, identifying areas of excellence as well as opportunities for process enhancement to maximize future productivity.
Activities:

- Review the overall sprint process, assessing what worked well and where improvements can be made in both the technical execution and team collaboration.
- Highlight any challenges encountered and propose actionable solutions to improve efficiency in future sprints.

- Adjust tools, communication strategies, and workflow management practices as necessary to foster a more effective development environment.
  Output: A comprehensive list of improvements and actionable insights to enhance team productivity, communication, and overall sprint performance.

7.7 Iterative Progression

Objective: To leverage insights and feedback from each sprint to continuously refine the development process, delivering a high-quality and feature-complete Patient Management System.

Activities:

- Apply lessons learned from previous sprints to enhance feature development, prioritization, and testing strategies in subsequent cycles.
- Continue iterating and expanding functionality, ensuring that the system meets the needs of patients, doctors, and administrators while maintaining scalability and security.
  Output: Progressive improvements that culminate in a fully developed, robust Patient Management System that excels in user experience, security, and system performance.

Chapter 8 : Technology Used

The Hospital Management System is designed to streamline and optimize hospital operations, improving efficiency for patients, doctors, and administrators. Built with modern technologies, the frontend, backend, and database layers are developed with a focus on scalability, security, and performance, ensuring smooth operations and seamless interaction between users.

8.1 Frontend (Presentation Layer)

Technologies: HTML, CSS, ReactJS, Bootstrap

Features:

- Dynamic and responsive user interfaces for patients, doctors, and admins.
- User registration and login forms with role-based views.
- Appointment booking and scheduling forms.
- Patient medical history and profile views.
- Admin dashboards for doctor and appointment management.

8.2 Backend (Application Layer)

Technologies: Node.js, Express.js framework, Postman, Thunder Client

Features:

- Appointment approval and notification handling.
- Security measures like role-based access control (RBAC).
- Integration of medical record management and prescription updates.
- API testing and validation using Postman and Thunder Client to ensure reliability and consistency in backend operations.

8.3 Database (Data Layer)

Technologies: MongoDB

Features:

- Data models for patients, doctors, admins, appointments, and medical history.

- Secure storage for sensitive patient information.
- Query optimization for doctor searches, medical history access, and appointment filtering.

8.4 Development Tools

Technologies: Visual Studio Code (VSCode), GitHub Copilot

Features:

- VSCode for integrated development, debugging, and source control.
- GitHub for version control and collaborative development, aided by **GitHub Copilot** for intelligent code suggestions.

8.5 Deployment

Technology: Render

Features:

- Website deployment with scalable and reliable hosting using **Render**.