SOFTWARE ENGINEERING & CONCEPTS – LAB MANUAL

Software Concepts & Engineering - Lab Manual

JAYASHREE K	220701104
JAYEN SENTHILKUMAR	220701105
JOEL SUNDHARSINGH A	220701110
KALAISELVI S	220701116
KAVIN A	220701122
KEERTHIVASAN S	220701128

Software Concepts & Engineering - Lab Manual

Overview of the Project

Title : Patient Tracker Application

Objective : The objective of the project is to develop a patient tracker application that centralizes patient data and streamlines appointment scheduling. This will improve healthcare delivery by enhancing data accuracy and efficiency in managing patient information and treatment processes.

Problems Being Resolved

Fragmented Patient Information:

- Current Issue: Patient data is often scattered across different systems or paper records, making it difficult for doctors to access comprehensive information quickly.
- Resolution: The application consolidates patient data into a single, centralized system, ensuring that all relevant information is easily accessible.

Inconsistent Treatment Records:

- Current Issue: Treatment history may not be consistently recorded or updated, leading to gaps in information which can affect patient care.
- Resolution: The application ensures all treatment history is recorded systematically, providing a complete and up-to-date medical history for each patient.

Medication Management:

- Current Issue: Tracking patient medications manually can lead to errors, such as incorrect dosages or missed medications.
- Resolution: The application allows for accurate tracking of prescribed medications, dosages, and schedules, reducing the risk of medication errors.

Consultation and Cost Tracking:

- Current Issue: Managing consultation costs manually can result in billing errors and disputes.
- Resolution: The application provides a detailed record of consultations and associated costs, improving transparency and accuracy in billing.

Appointment Scheduling and Follow-Ups:

- Current Issue: Manual appointment scheduling can lead to double bookings, missed follow-ups, and inefficiencies.
- Resolution: The application offers an automated scheduling system that manages appointments and sends reminders for follow-ups, reducing the likelihood of missed appointments and improving patient adherence to treatment plans.

Data Management and Its Impact

- **Centralized Data Repository**: By consolidating patient data into a single system, doctors can access comprehensive patient records, including medical history, medications, treatment plans, and appointment schedules, from one place.

- **Real-time Data Access**: The application allows for real-time updates and access to patient information, ensuring that doctors have the most current data at their fingertips.
- **Data Accuracy and Consistency:** Automated data entry and management reduce the risk of human error, ensuring that patient information is accurate and up-to-date.
- **Secure Data Handling:** Implementing robust security measures ensures that patient data is protected from unauthorized access, maintaining patient confidentiality and complying with data protection regulations. **Benefits to Users**

For Doctors:

- **Improved Efficiency**: Quick access to patient data and automated administrative tasks free up more time for patient care.
- **Better Decision-Making:** Comprehensive and accurate patient information supports informed medical decisions.
- **Streamlined Workflow:** Automated scheduling and reminders reduce the administrative burden and improve practice management.

For Patients:

- **Enhanced Care Quality:** Accurate and up-to-date medical records lead to better treatment outcomes and personalized care.
- **Transparency:** Clear records of consultations and costs enhance trust and reduce billing disputes.

- **Improved Adherence:** Appointment reminders and follow-up management improve adherence to treatment plans and medication schedules.

For Administrative Staff:

- **Reduced Workload:** Automated processes for scheduling, billing, and record-keeping reduce the manual workload.
- **Minimized Errors**: Automation reduces the likelihood of errors in scheduling, billing, and data entry.

Implementation Impact

Implementing the patient tracker application will transform the way patient information is managed in a healthcare setting. It will lead to more organized, efficient, and accurate data management, ultimately improving the quality of care provided to patients. The application will enhance the productivity of healthcare providers, streamline administrative processes, and ensure a seamless and transparent experience for patients.

BUSINESSARCHITECTURE

Business architecture is essential in a patient tracker system to ensure that the technology solution aligns with the organization's goals, processes, and structures.
Here are some needs that business architecture addresses:

1. **Alignment with organizational strategy:** Ensures the patient tracker system supports the healthcare organization's overall mission, vision, and objectives.

2. **Process optimization:** Identifies and streamlines clinical and administrative processes, improving efficiency and patient care.

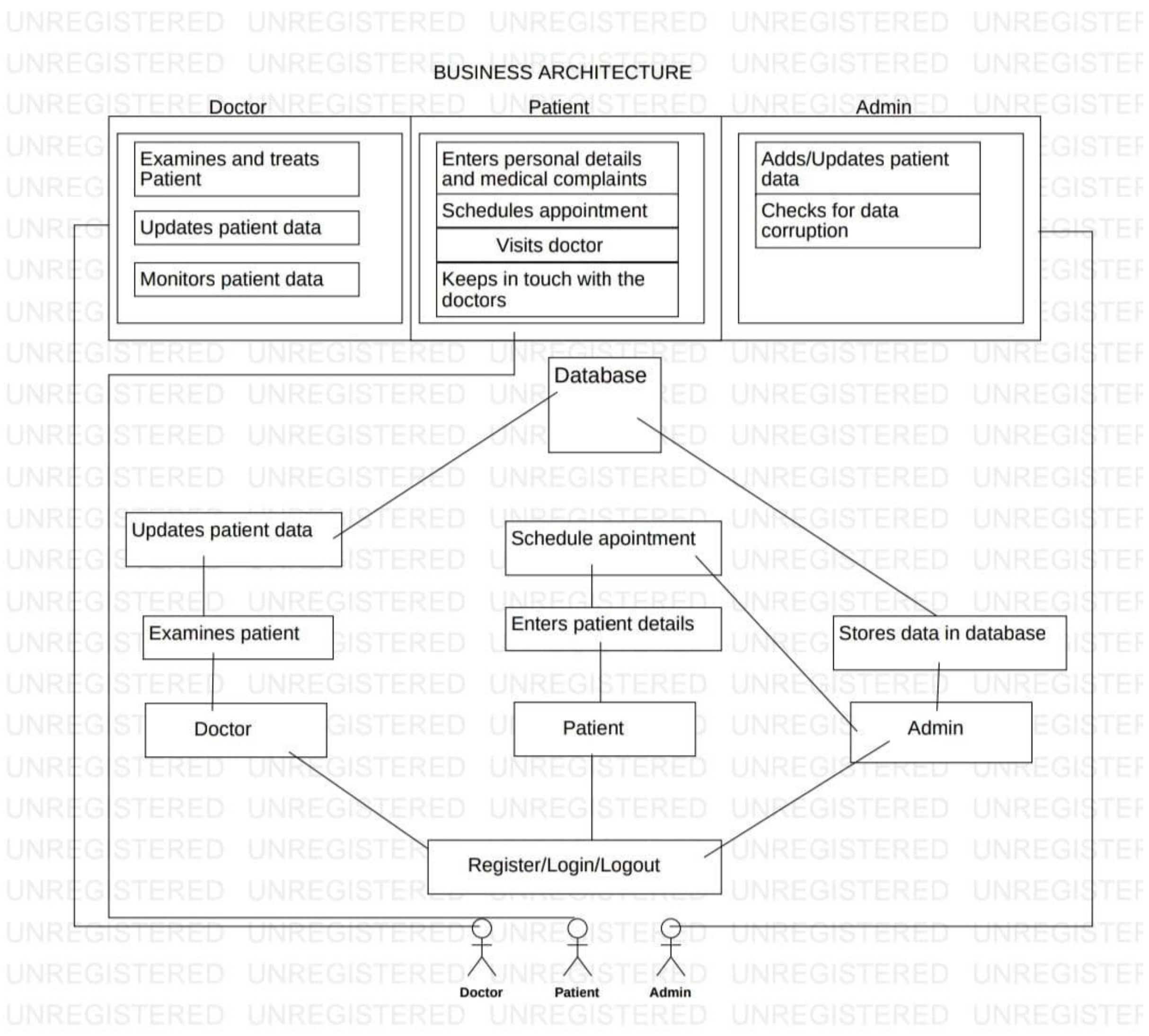
3. **Data management:** Defines how data is collected, stored, and used to inform patient care and business decisions.

4. **System integration:** Ensures seamless integration with existing systems, such as EHRs, billing systems, and scheduling software.
5. **Scalability and flexibility:** Enables the system to adapt to changing healthcare needs and technological advancements.

6. **Security and compliance:** Ensures the system meets regulatory requirements, such as HIPAA, and maintains patient data confidentiality and security.

7. **User experience:** Designs an intuitive and user-friendly interface for healthcare professionals, administrators, and patients.

By incorporating business architecture in the patient tracker system, healthcare organizations can enhance patient care, reduce costs, and improve overall efficiency.



Requirements as User Stories

User Story 1 : As a doctor, I want to be able to access and update patient information so that I can provide accurate and up-to-date medical care.

Acceptance Criteria :

- The doctor must have input fields to search for a patient using identifiers such as patient ID, name, or date of birth.

- Upon clicking the "Search" button, the system should retrieve and display the patient's demographic information, medical history, lab results, imaging reports, and current treatment plan.
- If the patient is found, the information should be displayed on the screen.
- If the patient is not found, an error message should be displayed.

User Story 2 : As a doctor, I want to schedule follow-up appointments for my patients so that they can continue their treatment without interruptions.

Acceptance Criteria :

- The doctor must have input fields to select the date and time for the follow-up appointment.
- The doctor should be able to select the type of appointment (e.g., in-person, telehealth, specialist consultation).

User Story 3 : As a doctor, I want to track the medications prescribed to patients so that I can ensure proper medication management and avoid prescription errors.

Acceptance Criteria :

- The doctor must be able to access the medication records for a patient from their profile.
- The system should display a comprehensive list of all medications currently prescribed to the patient.
- The doctor should be able to view detailed information about each prescribed medication, including dosage, frequency, start date, end date, and prescribing physician.
- The system should display any notes or instructions associated with each medication.

User Story 4 : As a doctor, I want to view a patient's past consultations so that I can better understand their medical history and ongoing treatment.

Acceptance Criteria :

- The doctor must be able to access the patient's past consultation records from their profile.
- The system should display a list of all past consultations for the patient.

User Story 5 : As a doctor, I want to generate comprehensive medical reports for my patients so that I can share detailed health information with them or other specialists.

Acceptance Criteria :

- The doctor must be able to access the report generation interface from the patient's profile.
- The interface should provide options to customize the content of the report.
- The doctor should be able to customize the date range for the report to include specific periods of the patient’s medical history.
- The system should provide formatting options to ensure the report is clear and professional.

User Story 6: As a doctor, I want to set reminders for upcoming patient appointments so that I can prepare adequately for each consultation.

Acceptance Criteria :

- The doctor must be able to access the reminder settings from the patient's appointment details or the main dashboard.
- The interface should provide options to configure reminders for upcoming appointments.
- The doctor should be able to set reminders for specific time intervals before the appointment (e.g., 1 hour, 1 day, 1 week).
- The system should allow the doctor to set multiple reminders for a single appointment if needed.

User Story 7: As a doctor, I want to send messages and updates to my patients through the application so that I can maintain effective communication and provide timely advice.

Acceptance Criteria :

- The doctor must be able to access the messaging interface from the patient's profile or main dashboard.
- The interface should provide options to compose and send new messages, as well as view message history.
- The doctor should have input fields to compose a message, including subject and body text.
- The system should allow the doctor to attach files or documents to the message if necessary.

User Story 8 : As a doctor, I want to view a patient's allergy information so that I can avoid prescribing medications that could cause adverse reactions.

Acceptance Criteria :

- The doctor must be able to access the patient's allergy information from their profile or the main dashboard.

- The allergy information should be prominently displayed and easily accessible.

User Story 9: As a doctor, I want to upload and attach diagnostic reports to a patient's record so that all relevant information is consolidated and easily accessible.

Acceptance Criteria :

- The doctor must be able to access the upload interface from the patient's profile or the main dashboard.
- The interface should provide options to upload new diagnostic reports.

User Story 10 : As a doctor, I want to generate and manage electronic prescriptions for patients, so that I can streamline the medication prescribing process.

Acceptance Criteria :

- The system should validate the entered prescription details to ensure accuracy and completeness.
- If any required information is missing or invalid, the system should prompt the doctor to correct the errors before proceeding.

Estimation of the User Stories using the Poker Planning Methodology:

User Story 1: Access and Update Patient Information

Estimation: 5 points (Moderate complexity and effort)

User Story 2: Schedule Follow-up Appointments

Estimation: 5 points (Moderate complexity and effort)

User Story 3: Track Medications Prescribed

Estimation: 5 points (Moderate complexity and effort)

User Story 4: View Patient Consultation History

Estimation: 3 points (Low complexity and effort)

User Story 5: Generate Medical Reports Estimation: 8 points (Moderate to high complexity and effort)

User Story 6: Set Appointment Reminders

Estimation: 3 points (Low complexity and effort)

User Story 7: Send Messages to Patients

Estimation: 5 points (Moderate complexity and effort)

User Story 8: View Patient Allergy Information

Estimation: 3 points (Low complexity and effort)

User Story 9: Upload Diagnostic Reports

Estimation: 5 points (Moderate complexity and effort)

User Story 10: Generate Electronic Prescriptions

Estimation: 5 points (Moderate complexity and effort)

Final Estimation:

3 Points (Low complexity and effort): User Stories **4, 6, 8**

5 Points (Moderate complexity and effort): User Stories **1, 2, 3, 7, 9, 10**

8 Points (Moderate to high complexity and effort): User Story **5**

Non Functional Requirements

Security:

- The system must ensure that only authorized personnel (e.g., doctors, nurses) can access and update patient information.
- Data encryption must be implemented both in transit and at rest to protect patient information.

Performance:

- The system should allow access to patient records within 2 seconds to avoid delays in medical care.
- Updates to patient information should be reflected in the system within 5 seconds.

Availability:

- The system should have an uptime of 99.9% to ensure that patient information is always accessible when needed.

Separation of Concerns:

Modularization: MVC separates the application into three interconnected components—Model, View, and Controller—each responsible for different aspects of the application, leading to a more organized codebase.

Maintenance: Makes the application easier to maintain and update because changes in one component have minimal impact on others.

2. Parallel Development:

Team Collaboration: Different developers can work on the Model, View, and Controller simultaneously, which speeds up the development process.

Specialization: Developers can specialize in one part of the application, improving productivity and code quality.

3. Reusability:

Code Reuse: Models and Views can be reused across different parts of the application or even in different projects.
Component Reuse: Views can be shared and reused, reducing duplication and inconsistencies.

4. Flexibility:

Adaptability: Changes in the business logic (Model) do not require changes in the user interface (View) and vice versa.

Testing: Components can be tested independently, making it easier to identify and fix bugs.

5. Scalability:

Component Scaling: MVC architecture allows individual components to be scaled independently based on their needs, improving performance and resource utilization.

Components in MVC Architecture for Patient Tracker Application

1. Model:

Definition: Represents the data and the business logic of the application.

Components:

- Patient: Handles data related to patients.
- Medication: Manages medication data.
- TreatmentHistory: Tracks treatment history and medical records.
- Appointment: Manages appointment scheduling and follow-up.
- ConsultationCost: Tracks consultation costs and billing information.

2. View:

Definition: Responsible for presenting data to the user.

Components:

- Patient Dashboard: Displays patient details and health records.
- Medication List: Shows medications prescribed to patients.
- Treatment History: Lists past treatments and medical history.
- Appointment Scheduler: Interface for scheduling and viewing appointments.
- Billing View: Visual representation of consultation costs and payment status.

3. Controller:

Definition: Acts as an intermediary between Model and View, handling user input and updating the Model.

Components:

- PatientController: Manages patient-related actions (create, update, delete).

- MedicationController: Handles medication management.
- TreatmentHistoryController: Manages treatment history records.
- AppointmentController: Oversees appointment scheduling and updates.
- BillingController: Manages billing and consultation cost tracking.

Design Principles Used and Their Importance

1. **Single Responsibility Principle (SRP)** Definition: A class should have only one reason to change, meaning it should have only one job or responsibility.

Usage in the Project:

Each class in the system, such as Patient,
Medication, TreatmentHistory, Appointment, and ConsultationCost, has a distinct responsibility. For example, the Patient class is responsible solely for managing patient-related data and operations.

Benefits:

- Maintainability: Makes the system easier to understand and maintain, as each class focuses on a specific aspect of the application.
- Scalability: Allows individual components to be modified or extended without affecting other parts of the system.

2. **Open/Closed Principle (OCP)**

Definition: Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.

Usage in the Project:
The design allows for adding new features (e.g., additional notification methods or new types of reports) without altering existing code. For instance, the NotificationService can be extended to include new notification channels. **Benefits:**

- Extensibility: Facilitates adding new functionalities with minimal impact on existing code.
- Reliability: Reduces the risk of introducing bugs when extending the system.

3. Liskov Substitution Principle (LSP)

Definition: Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.

Usage in the Project:
If the system has a base class User and derived classes like Doctor, Nurse, and Admin, these subclasses can be used interchangeably wherever a User object is expected.

Benefits:

- Interchangeability: Enhances the flexibility of the system, allowing for the use of more specific types while maintaining general functionality.
- Robustness: Ensures that substituting subclasses does not compromise system behavior.

4. Interface Segregation Principle (ISP)

Definition: Clients should not be forced to depend on interfaces they do not use.

Usage in the Project:
Different interfaces are designed for specific roles. For example, IDoctor might include methods specific to doctors, while INurse includes methods specific to nurses.

Benefits:

- Cohesion: Increases the cohesion of interfaces, making them more specific to the clients' needs.
- Reduced Complexity: Simplifies the implementation of interfaces and the classes that use them.

5. Dependency Inversion Principle (DIP)

Definition: High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

Usage in the Project:
High-level modules like AppointmentController depend on abstract services like INotificationService rather than concrete implementations.

Benefits:

- Decoupling: Reduces the coupling between high-level and low-level modules, promoting more flexible and maintainable code.
- Testability: Improves the testability of the system by allowing the use of mock implementations for testing purposes.

6. Don't Repeat Yourself (DRY)

Definition: Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

Usage in the Project:
Common functionality is abstracted into reusable methods or services, such as
AuthenticationService or NotificationService, to avoid duplication.

Benefits:

- Maintainability: Reduces redundancy, making the system easier to maintain and update.
- Consistency: Ensures consistency across the application by centralizing common functionality.

7. Separation of Concerns (SoC)

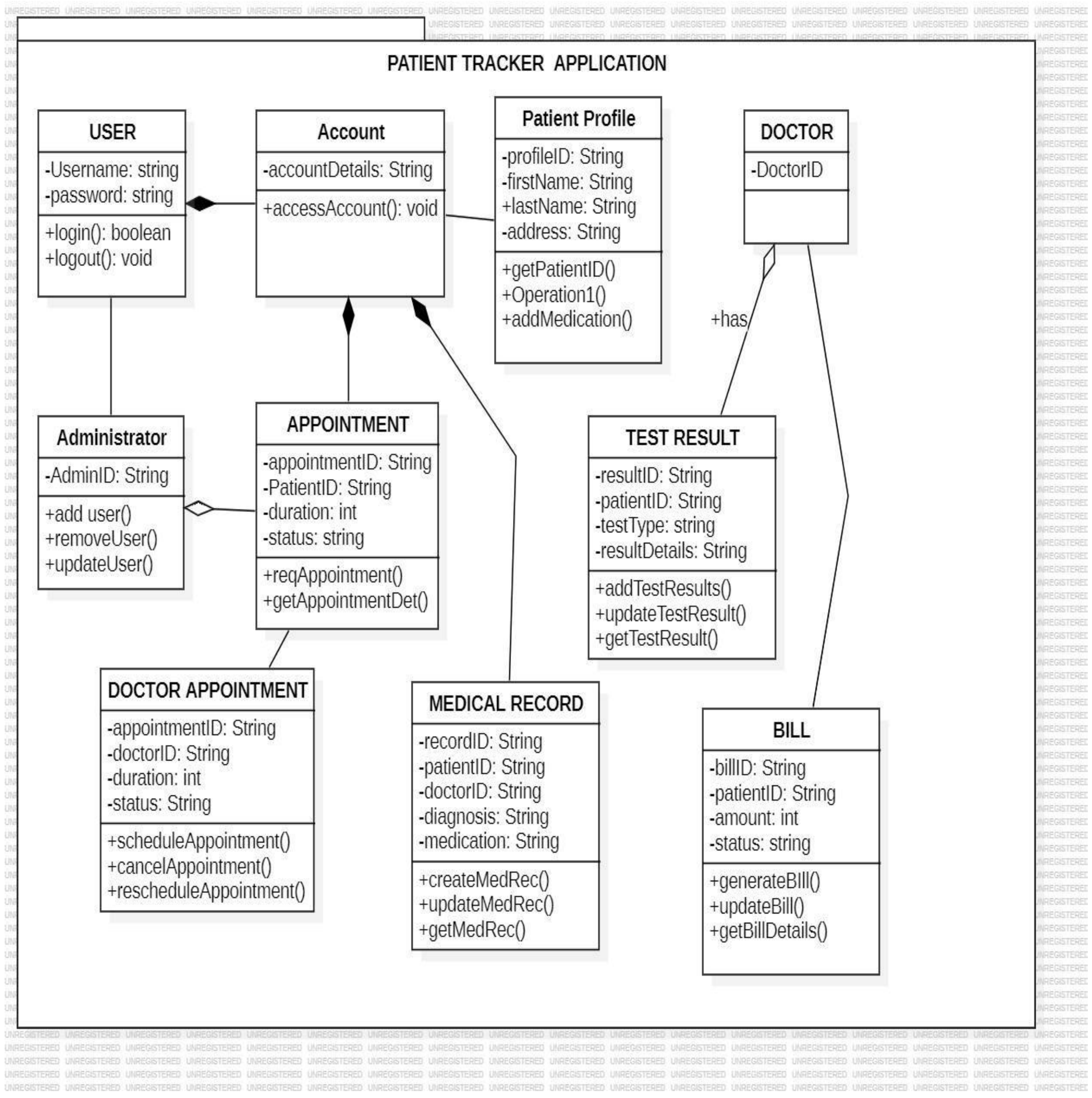
Definition: Different parts of a program should have distinct and separate responsibilities.

Usage in the Project:
The MVC architecture itself embodies SoC by separating the user interface (View), business logic (Controller), and data management (Model).

Benefits:

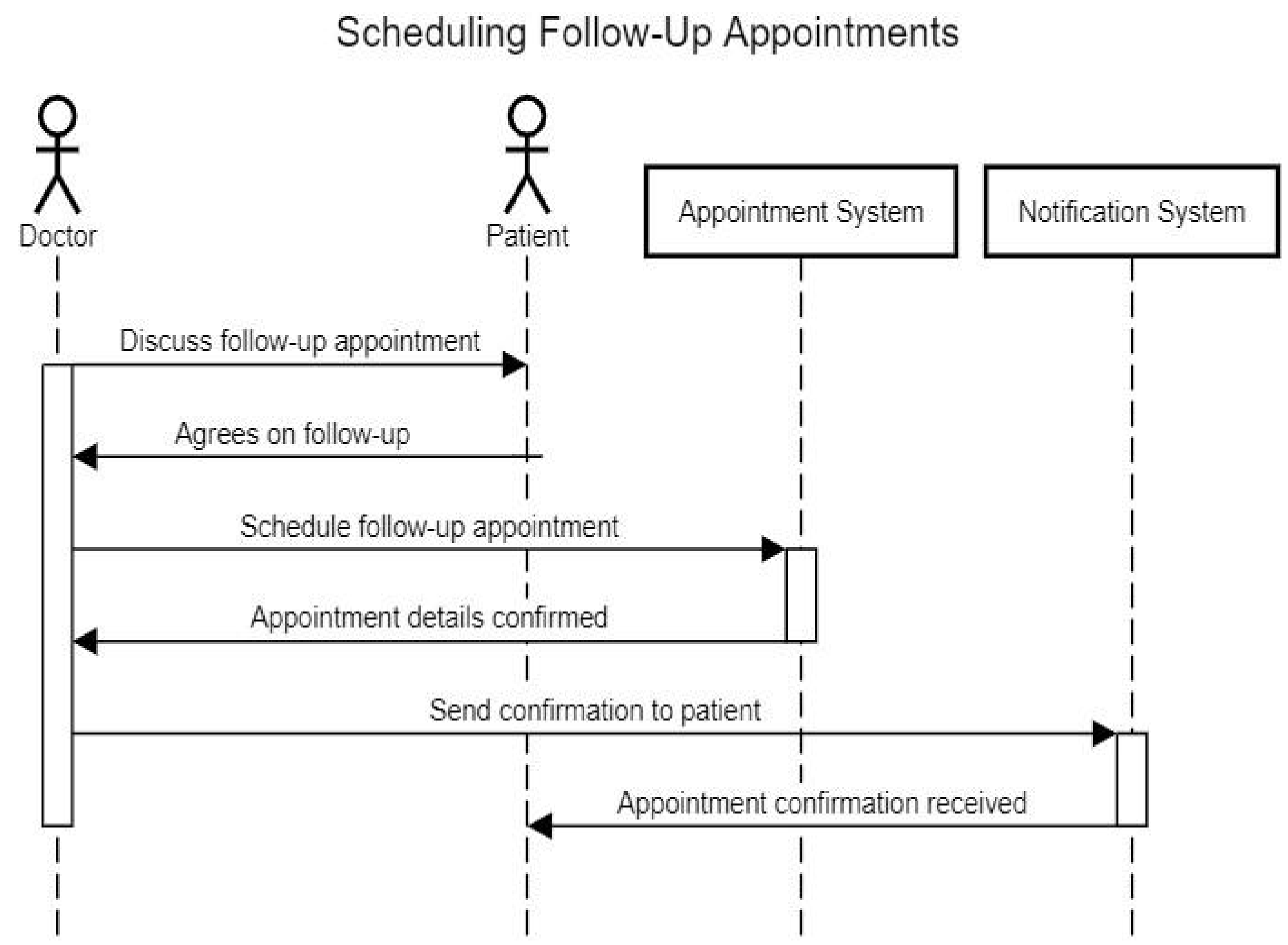
- Clarity: Enhances the clarity of the system by dividing it into distinct sections, each responsible for a specific aspect of the functionality.
- Modularity: Promotes modularity, allowing for easier maintenance and extension.

CLASS DIAGRAM

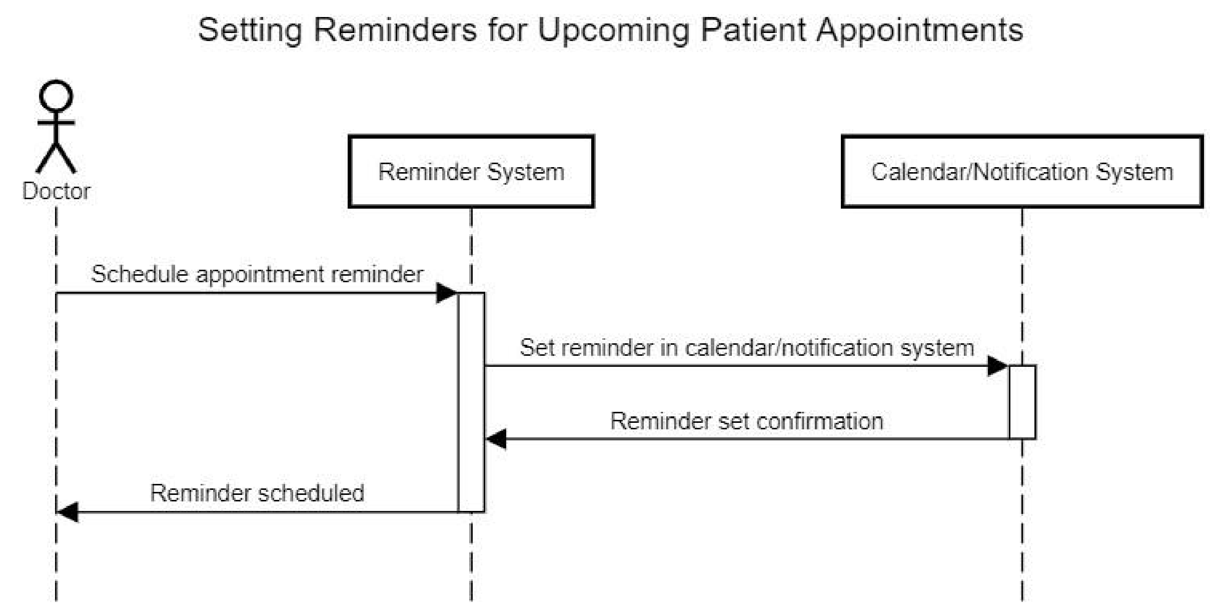


SEQUENCE DIAGRAM

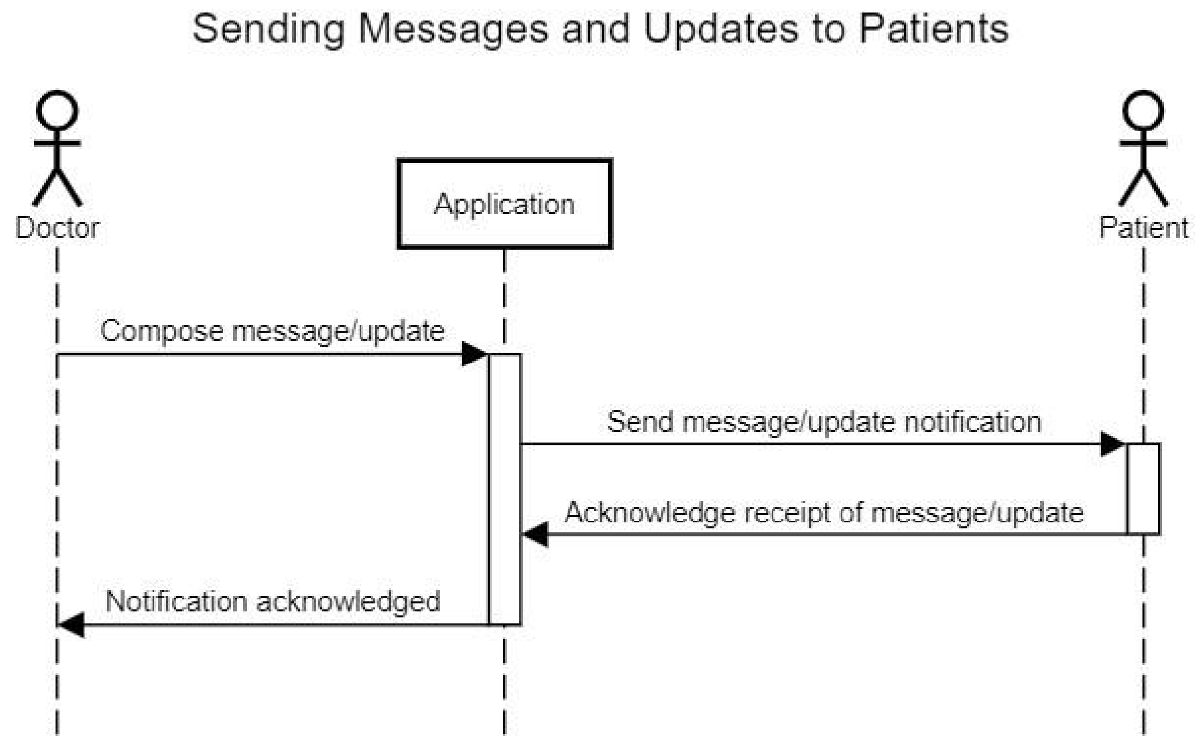
User Story :Schedule Appointment


Scheduling Follow-Up Appointments
Doctor
Patient
Appointment System
Notification System
Discuss follow-up appointment
Agrees on follow-up
Schedule follow-up appointment
Appointment details confirmed
Send confirmation to patient
Appointment confirmation received

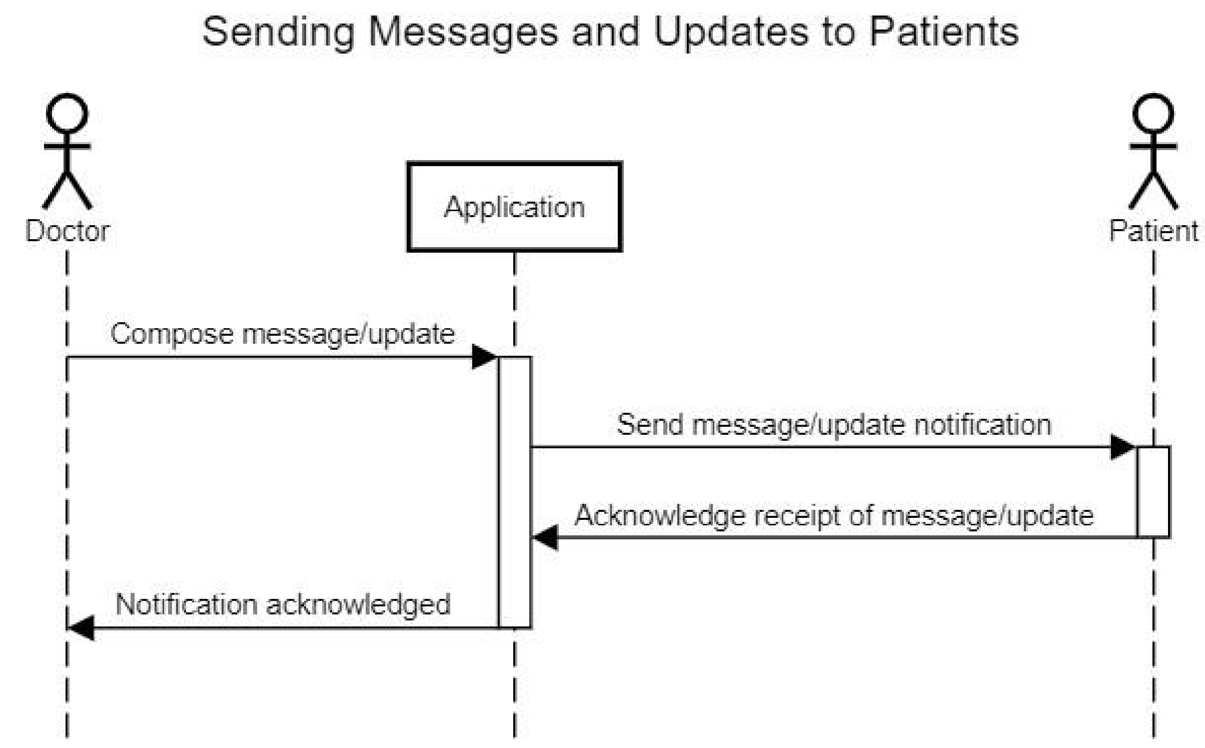
User Story : Setting Remainders


Setting Reminders for Upcoming Patient Appointments
Doctor
Reminder System
Calendar/Notification System
Schedule appointment reminder
Set reminder in calendar/notification system
Reminder set confirmation
Reminder scheduled

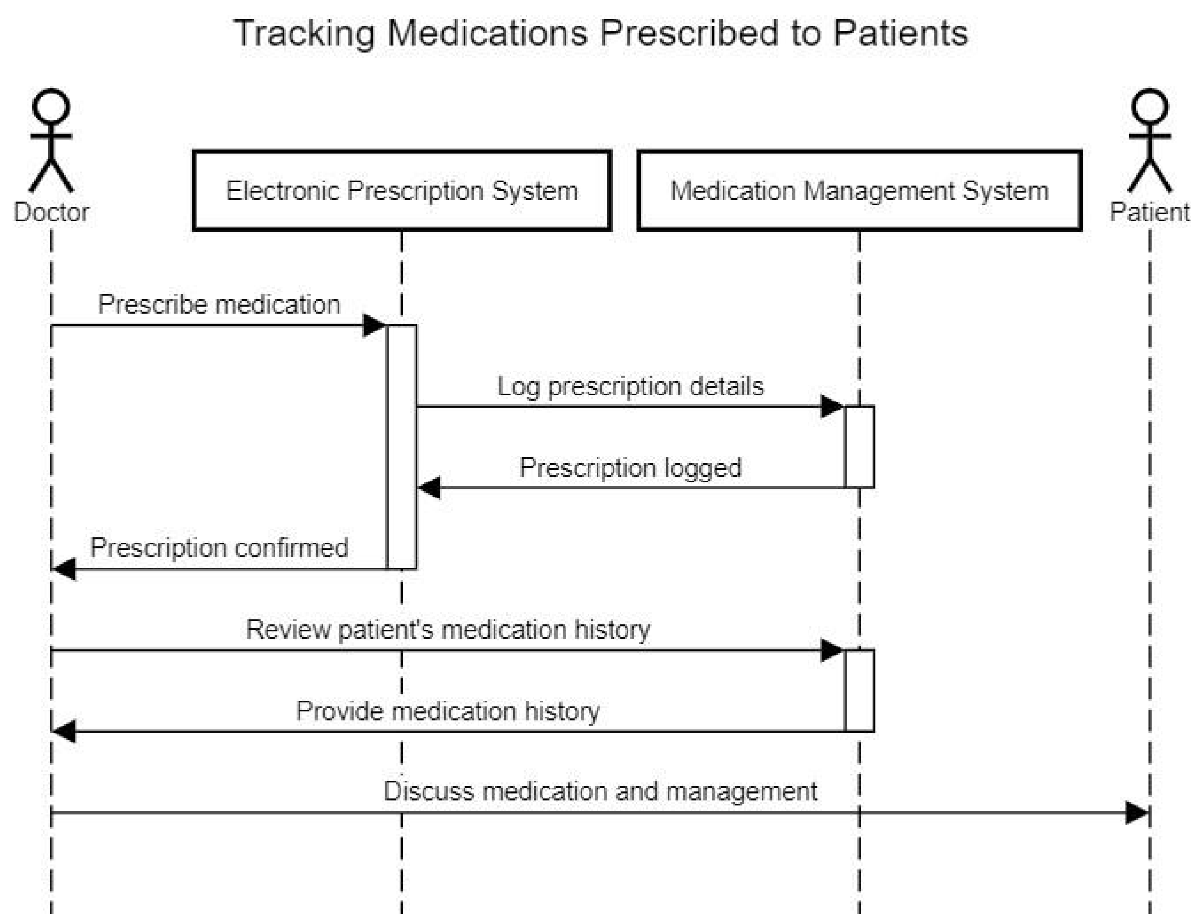
User Story : Sending Messages and Updates To Patients


Sending Messages and Updates to Patients
Doctor
Application
Patient
Compose message/update
Send message/update notification
Acknowledge receipt of message/update
Notification acknowledged

User Story :Generating And Managing Electronic Prescription


Sending Messages and Updates to Patients
Doctor
Application
Patient
Compose message/update
Send message/update notification
Acknowledge receipt of message/update
Notification acknowledged

User Story : Tracking Prescription


Tracking Medications Prescribed to Patients
Doctor
Electronic Prescription System
Medication Management System
Patient
Prescribe medication
Log prescription details
Prescription logged
Prescription confirmed
Review patient's medication history
Provide medication history
Discuss medication and management

Objectives ● Ensure the patient tracker application functions correctly and meets requirements.

- Verify data integrity and security.
- Validate the user interface and user experience.

- Ensure compliance with healthcare regulations.
- Identify and fix defects early in the development cycle.

Scope

Functional Testing: Validate core functionalities such as patient information management, medication tracking, treatment history, consultation costs, and appointment scheduling.

Security Testing: Ensure data protection and compliance with healthcare standards.

Usability Testing: Assess user interface and ease of use for healthcare providers.

Performance Testing: Evaluate the application's response time and performance under load.
Test Environments

Development Environment: Initial testing by developers.

QA Environment: Comprehensive testing by QA team.
Staging Environment: Pre-production testing to mimic the production environment. **Production Environment:** Post-deployment testing to ensure everything works as expected.
Tools

Test Management: JIRA, TestRail

Automation: Selenium, Appium

Performance: JMeter, LoadRunner **Security:** OWASP ZAP, Burp Suite

Test Cases for User Stories

User Story 1: Manage Patient Information

Happy Path

Test Case ID: TC01-H1

Description: Add a new patient record Preconditions: User logged into the application Steps:

Navigate to the "Add Patient" section.

Enter all required patient information (name, age, contact details, etc.).

"Save".

Expected Result: New patient record is saved successfully and displayed in the patient list. Error Scenario

Test Case ID: TC01-E1

Description: Add a patient record with missing mandatory fields Preconditions: User logged into the application Steps:

Navigate to the "Add Patient" section.

Click

Leave mandatory fields (e.g., name) blank.
Click "Save".
Expected Result: Application displays an error message indicating mandatory fields are missing.
User Story 2: Track Medications Happy Path

Test Case ID: TC02-H1
Description: Add a new medication for a patient Preconditions: Patient record exists Steps:
Open the patient record.
Navigate to the "Medications" section.
Click "Add Medication".
Enter medication details (name, dosage, frequency, duration).
"Save".
Expected Result: Medication details are saved and listed under the patient's medications. Error Scenario

Test Case ID: TC02-E1
Description: Add a medication with invalid dosage format
Preconditions: Patient record exists Steps:
Open the patient record.
Navigate to the "Medications" section.
Click "Add Medication".
Enter an invalid dosage format (e.g., "ten mg" instead of "10 mg").
Click "Save".
Expected Result: Application displays an error message indicating invalid dosage format.
User Story 3: Record Treatment History
Happy Path

Click

Test Case ID: TC03-H1

Description: Add a treatment record for a patient Preconditions:
Patient record exists Steps:
Open the patient record.
Navigate to the "Treatment History" section.
"Add Treatment".

Click

Enter treatment details (date, diagnosis, treatment description).
Click "Save".
Expected Result: Treatment record is saved and displayed under the patient's treatment history. Error Scenario

Test Case ID: TC03-E1
Description: Add a treatment record with past date Preconditions:
Patient record exists Steps:
Open the patient record.
Navigate to the "Treatment History" section.
Click "Add Treatment".
Enter a past date for the treatment (e.g., one year ago). Click "Save".
Expected Result: Application allows saving the record, but it should flag it as a past treatment.
User Story 4: Manage Consultation Costs
Happy Path

Test Case ID: TC04-H1

Description: Record a consultation cost Preconditions:
Patient record exists Steps:
Open the patient record.
Navigate to the "Consultation Costs" section.
Click "Add Cost".
Enter consultation cost details (date, amount, description).
Click "Save".
Expected Result: Consultation cost is saved and listed under the patient's consultation costs. Error Scenario

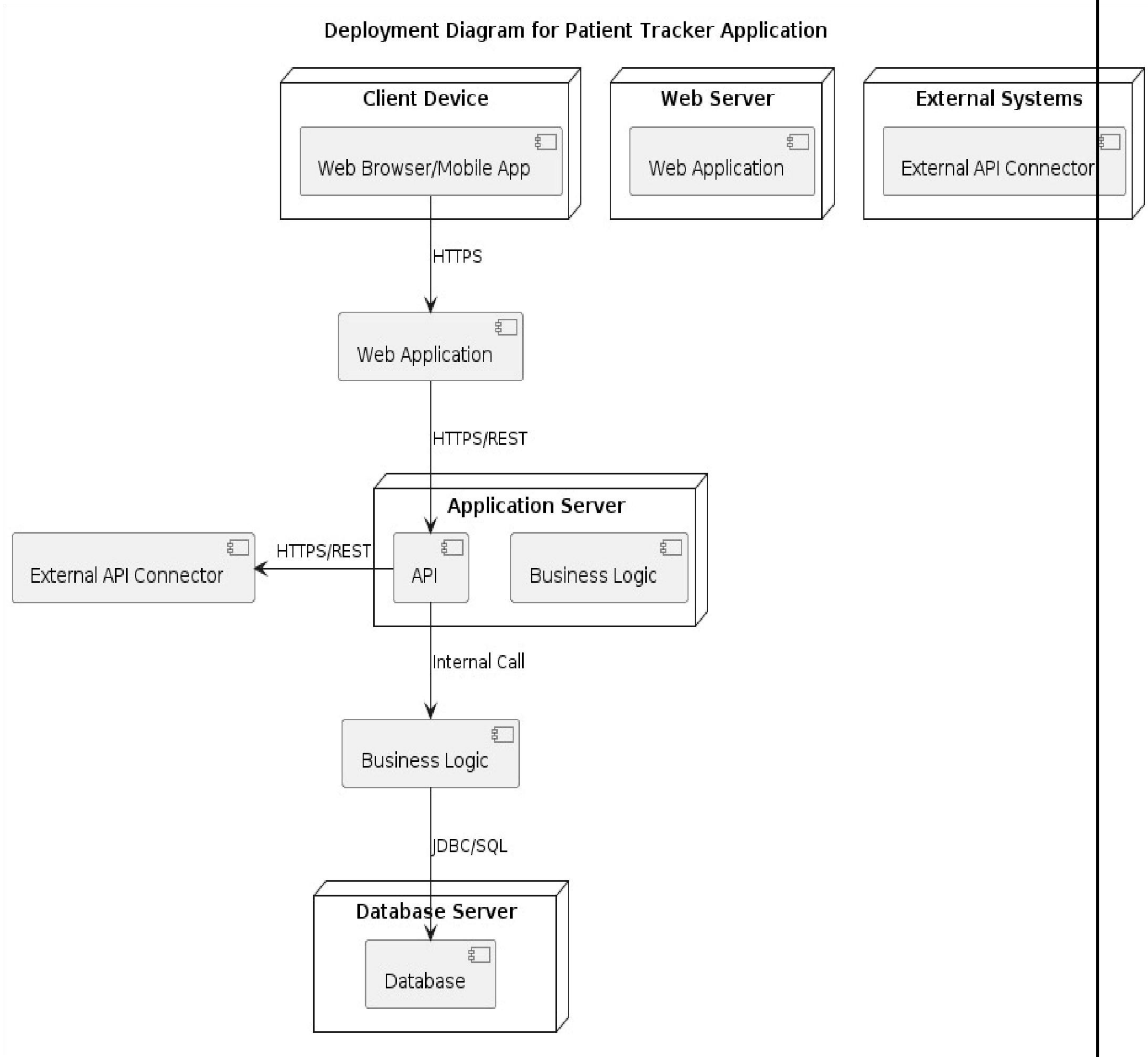
Test Case ID: TC04-E1

Description: Record a consultation cost with future date
Preconditions: Patient record exists Steps:
Open the patient record.
Navigate to the "Consultation Costs" section.
Click "Add Cost".
Enter a future date for the cost.
Click "Save".
Expected Result: Application displays an error message indicating the date cannot be in the future.
User Story 5: Schedule Follow-Up Appointments Happy Path

Test Case ID: TC05-H1
Description: Schedule a follow-up appointment Preconditions: Patient record exists Steps:
Open the patient record.
Navigate to the "Appointments" section.
Click "Schedule Appointment".
Enter appointment details (date, time, purpose).
Click "Save".
Expected Result: Appointment is scheduled and displayed under the patient's appointments. Error Scenario

Test Case ID: TC05-E1
Description: Schedule an appointment with a past date
Preconditions: Patient record exists Steps:
Open the patient record.
Navigate to the "Appointments" section.
Click "Schedule Appointment".
Enter a past date for the appointment.
Click "Save".

Expected Result: Application displays an error message indicating the appointment date cannot be in the past.


Deployment Diagram for Patient Tracker Application
Client Device
Web Browser/Mobile App
Web Server
Web Application
External Systems
External API Connector
HTTPS
Web Application
HTTPS/REST
Application Server
API
Business Logic
HTTPS/REST
External API Connector
Internal Call
Business Logic
JDBC/SQL
Database Server
Database