Software Requirements and Design Document

for

Centralized Hospital Management System

Prepared by

Fakhir Ali
Abdul Munhim Husain
Ayna Sulaiman
Muhammad Talha

CureTrack by Syntegrity

27/11/2024

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for **Cure Track**: A Centralized Hospital Management System. It provides a comprehensive outline of the system's functionalities and serves as a guide for its development and implementation.

1.2 Product Scope

Cure Track is a hospital management system aimed at streamlining hospital operations by centralizing resources, staff management, and scheduling. The system provides functionalities like online appointment booking, booking management, feedback, FAQs, video consultation, adding hospitals, integrated billing, equipment management and staff scheduling. It focuses on basic administrative functions, excluding advanced patient data management.

1.3 Title

Cure Track: A Centralized Hospital Management System for Streamlined Operations and Resource Optimization

1.4 Objectives

- Optimize hospital management with a streamlined dashboard to manage staff and resources efficiently.
- Provide patients with the convenience of online appointment booking.
- Provide doctors with the facility
- Enable easy billing with direct payment gateways and automatic systems for consultations, treatments, and room rent.
- Improve resource deployment for room, equipment, and staff scheduling.

- Enhance accessibility for hospital admins, doctors, and patients through a browser-based interface.
- Providing proper answers and questions between patients and doctors accordingly
- Handling emergency cases
- To provide admin the ability to add healthcare package, add Hospital, remove doctor, remove staff and remove patient.
- Enabling Online Video consultation for patients by doctors
- To provide patient the option to give proper feedback to specific doctors
- Digital Transformation of Hospitals

1.5 Problem Statement

The problem of inefficient and decentralized hospital management across a city affects hospital administrators, medical staff, and patients. The impact is increased operational delays, poor resource utilization, and patient dissatisfaction. A successful solution would be the implementation of a centralized hospital management system that integrates all operations, allowing administrators to efficiently manage resources, and patients to book appointments and make payments with ease. The system will also allow doctors to manage their appointments easily and inform patient about unavailability if any. The system will provide proper feedback and FAQ functionality for customer satisfaction. The resource and room allocation issues will be solved by special options that will be provided to admin. New hospitals and resources can be added and doctors, patients and staff can be removed by admin. This will make a digital transformation of hospitals worldwide.

2. Overall Description

2.1 Product Perspective

Cure Track is a self-contained, Java Application based system designed to streamline operations in hospitals by centralizing key administrative functions. It is not a replacement, but a new platform designed for centralized control of hospital operations across a city. It provides interfaces for admins, doctors, and patients, ensuring a seamless experience.

2.2 Product Functions

- User registration for admin, staff, doctors and patients.
- Online appointment booking
- Online appointment management.
- Video consultation
- Availability Management
- Patient Feedback for Doctors and hospitals
- Emergency Case management
- FAQs Asking Questions to specific Doctors
- FAQs answering
- Manage Inventory for resources
- Allocate Resources
- Add healthcare packages
- Remove doctors, Patients and Staff
- Add Hospitals
- Integrated Online payment system for consultation fees, treatments, and room rent.
- Browser-based interface for easy accessibility.

2.3 List of Use Cases

1. Book Appointment
2. Register Users
3. Compare Doctors
4. Access 24/7 Video Consultations
5. FAQs
6. Explore Healthcare Packages
7. View Patient Records
8. Manage Appointment
9. Allocate Resources in Emergency
10. Manage Inventory

11. Pay Bills
12. Submit and View Feedback
13. Generate Medical Report
14. Availability
15. Remove Doctors/Patient/Staff

2.4 Extended Use Cases

2.4.1 Use Case 1: Book Appointment (Abdul Munhim)

- **Use Case Name:** Book Appointment
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
    - **Patient:** Wants to book an appointment with a doctor.
    - **Doctor:** Wants to manage their schedule efficiently.
- **Preconditions:**
    - The patient is registered in the system.
    - The patient is logged in.
    - Doctors have available appointment slots.
- **Success Guarantee (Postcondition):**
    - The appointment is successfully booked, and confirmation is sent to both the patient and the doctor.

- **Main Success Scenario:**

1. The patient logs in to the system.
2. The patient selects the "Book Appointment" option from menu buttons of home page.
3. The system displays available hospitals.
4. The patient selects a hospital.
5. The system displays available specialty.
6. The patient selects a specialty.
7. The system displays available doctors.
8. The patient selects a doctor.
9. The system displays calendar.
10. The patient selects a date.
11. The system displays available appointment time slots.
12. The patient selects a time slot.
13. The patient selects the "Confirm" button.
14. The system confirms the booking and notifies the doctor

- **Extensions:**
  - **1a**: If login details are incorrect, the system asks patient to log in again

        - 13a. If any of the field is missing or invalid, then the patient is asked to refill all the fields.

- **Special Requirements:**
    - Text should be visible from 3 meters.
    - Data security for personal and medical information.
    - The date of birth must not be in the future.
    - Password must be at least 8 characters long and adhere to security standards.
    - The system should validate the format for fields like contact (11 digits) and username (alphanumeric).

- **Technology and Data Variations List:**
    - Input via computer, laptop or KIOSK app.
    - Appointment dates displayed in DD/MM/YYYY format.
    - Time slots follow AM and PM formats
    - Confirmation via a note.

2.4.2 Use Case 2: Register Users (Abdul Munhim)

- **Use Case Name:** Register Users
- **Scope:** CureTrack
- **Level:** Subfunction
- **Primary Actor:** User and System
- **Stakeholders and Interests:**
    - **Patient:** Wants to be registered in the system to receive medical care.
    - **Doctor:** Requires accurate patient information for treatment purposes.

  - **Staff:** Wants to manage his roles at Hospital

- **Preconditions:**
  - The system is operational and ready to accept new User registrations.

- **Success Guarantee (Postcondition):**
  - The User is successfully registered and provided with login credentials to access Hospital Management services through the system.

- **Main Success Scenario:**

1. The user opens the registration tab in CureTrack.
2. The system prompts the user to enter the patient's personal details (firstName, lastName, dob, contact number, address, username and password.).
3. The user enters the his/her details.
4. The user chooses the register option.
5. The system confirms that the user has been successfully registered and added to database, and the user can now log in.

- **Extensions:**
  - 4a. If personal details are incomplete or incorrect, the patient is asked to re-enter all the necessary details.

- **Special Requirements:**
  - Data validation for personal details (e.g., phone numbers, DOB, password).
  - Passwords must meet security standards (e.g., minimum length, alphanumeric, special characters).

- **Technology and Data Variations List:**
  - Data input through a KIOSK-based or desktop-based application.
  - Text should be visible from 3 meters.

2.4.3 Use Case 3: View Medical Report (Abdul Munhim)

- **Use Case Name:** Generate Medical Report
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
  - **Patient:** Wants to receive detailed and accessible medical reports.
- **Preconditions:**
  - The Patient is logged in to the system.
  - The patient has undergone Consulations.
- **Success Guarantee (Postcondition):**
  - The medical report is generated, saved in the system, and is accessible to the patient
- **Main Success Scenario:**

1. The Patient logs in to the system.

2. The Patient selects the view profile in the
Home Page.

3. The system displays the patient's medical history, including previous treatments and diagnoses.

- **Extensions:**
  - 1a: If login details are incorrect, the system asks patient to log in again
- **Special Requirements:**
  - Fast report generation with a large amount of data.
  - Text should be visible from 3 meters.
- **Technology and Data Variations List:**
  - Data input via computer.

2.4.4 Use Case 4: Compare Doctors (Abdul Munhim)

- **Use Case Name:** Compare Doctors
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
  - **Patient:** Wants to make an informed decision by comparing doctors based on specialty, hospital and average feedback ratings.
  - **Doctors:** Want their qualifications and reputation to be accurately reflected.
- **Preconditions:**
  - The patient is logged into the system.
  - Doctors' data available in the system.

- **Success Guarantee (Postcondition):**
  - The patient has successfully compared doctors and can make a decision.
- **Main Success Scenario:**

1. The patient logins into the system.
2. The patient selects "Compare" in CureTrack.
3. The system prompts the patient to choose a doctor1 and doctor2.
4. The patient selects specific doctors to compare and clicks “Proceed”.
5. The system retrieves and displays relevant data.
6. The patient reviews the comparison.

- **Extensions:**
  - 1a: If login details are incorrect, the system asks patient to log in again
  - 4a. if both doctors are not selected user is asked to select two doctors again.
- **Special Requirements:**
  - Real-time retrieval of doctors’ data in a table (e.g., specialty and ratings).
  - user-friendly user interface for easy comparison.
- **Technology and Data Variations List:**
  - Input via laptop.
  - Results displayed in a table format.

2.4.5 Use Case 5: Access 24/7 Video Consultations (Fakhir Ali)

- **Use Case Name:** Access 24/7 Video Consultations
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
  - **Patient:** Wants access to video consultation at any time without needing to visit the hospital.
  - **Doctor:** Provides timely video consultations.
- **Preconditions:**
  - The patient is registered and logged into the system.
- **Postconditions:**
  - The video consultation is completed.
  - The patient's medical record is updated with consultation details.
- **Main Success Scenario:**

1. The patient logs into the system.
2. The patient clicks "Book Appointment" tab
3. The system displays available hospitals.
4. The patient selects a hospital.
5. The system displays available specialty.
6. The patient selects a specialty.
7. The system displays available doctors.
8. The patient selects a doctor.

9. The system displays a list of specialists based on the search.

10. The patient presses the "video consultation" button.

11. Now, the doctor logs in and selects "video consultation" button from the menu and selects Patient Name.

12. The system shows a table having required fields and asks doctor to enter diagnosis and treatment.

13. The doctor writes the diagnosis and treatment and proceed.

14. The system updates the table and Medical Report of the Patient. Fixed Payment is generated and patient is notified

Extension:

- 1a: If login details are incorrect, the system asks patient to log in again
- 10a. If date and time fields are not empty, the system gives an alert that date and time field must be empty.
- 10b. If video Consultation request is pending then the patient is alerted that "Video Consultation is already Requested"
- 13a. If diagnosis and treatment fields are empty, the system gives an alert that diagnosis and treatment fields cannot be empty.

- **Special Requirements:**
  - The text should be visible from 3 meters.
  - User-friendly GUI.
- **Technology and Data Variations List:**
  - Different versions of windows should support the video consultation feature.

2.4.6 Use Case 6: FAQs (Fakhir Ali)

- **Use Case Name:** FAQs
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
  - **Patient:** Seeks advice and information from verified doctors.
  - **Doctor:** Provides professional answers to patient questions.
- **Preconditions:**
  - The patient is registered and logged into the system.
  - Verified doctors have profiles set up in the system to answer questions.
- **Postconditions:**
  - Verified doctors has responded to patient questions.
- **Main Success Scenario:**

1. The patient logs into the system.

2. The patient navigates to the "Faqs" tab.

3. The system displays a list of previously asked questions along with answers in the form of a table of that logged in patient and a drop-down menu of doctors to ask question to a specific doctor.

4. The patient asks a question and presses the “ask” button.

5. The system updates the table having questions and answers

5. The doctor logs in and selects the “FAQs” tab

6. The system shows the list of questions along with answers of that specific doctor.

7. The doctor selects the question and answers it.

8. The system updates the table with the answer

- **Extensions:**
  - 1a: If login details are incorrect, the system asks patient to log in again
  - 4a. If the question field is empty and the patient clicks “ask” button, the system gives an alert that it cannot be empty.
  - 5a: If login details are incorrect, the system asks Doctor to log in again
  - 7a. If a doctor does not select a question or the answer field is empty, the system gives an alert.

- **Special Requirements:**
  - The question can be answered only by the doctor specified by patient.
  - The text should be visible from 3 meters.
  - User-friendly GUI.
- **Technology and Data Variations List:**
  - A table with 2 columns is used to display previously asked questions.

2.4.7 Use Case 7: Explore Healthcare Packages (Fakhir Ali)

- **Use Case Name:** Explore Healthcare Packages
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
  - **Patient:** Wants to explore and subscribe to healthcare packages based on their needs.
  - **Hospital:** Offers tailored healthcare packages to increase patient engagement and service utilization.
- **Preconditions:**
  - The patient is registered and logged into the system.
  - The system has healthcare packages available.

- **Postconditions:**
  - The patient has explored available packages.
  - The patient has subscribed to a package if desired.
- **Main Success Scenario:**

1. The patient logs into the system.
2. The patient navigates to the "Healthcare Packages" section.
3. The system displays a list of available packages, including details (package name, hospital name, duration, price and description).
4. The patient browses the packages.
5. The patient selects and subscribes to the package if they find it suitable.
6. The system processes the subscription and updates the patient's account.

- **Extensions:**
  - 1a: If login details are incorrect, the system asks patient to log in again
  - 3a. If no packages are available, the system notifies the patient.
  - 5a. If no package is selected, the system gives an alert.
- **Special Requirements:**
  - The system must ensure that healthcare packages are updated in real-time to reflect accurate pricing and availability.
  - The text should be visible from 3 meters.

  - User-friendly GUI.

- **Technology and Data Variations List:**
  - Different versions of windows should support the video consultation feature.

2.4.8 Use Case 8: Update Patient Records (Fakhir Ali)

- **Use Case Name:** Update Patient Records
- **Scope:** CureTrack
- **Level:** User goal
- **Primary Actor:** Doctor
- **Stakeholders and Interests:**
  - **Doctor:** Requires complete accurate patient medical records to make informed decisions.
  - **Patient:** Wants the doctor to update all relevant medical information during consultations.
- **Preconditions:**
  - The doctor is logged into the system.
  - The patient has booked a Consultation.
- **Postconditions:**
  - The doctor has reviewed the patient's medical records.
  - The patient's record is updated with new information after the appointment.

- **Main Success Scenario:**

1. The doctor logs into the system and navigates to their Consultations.
2. The system displays a list of patients with scheduled Consultation.
3. The doctor selects a patient from the list.
4. The system retrieves and displays the patient's medical history, including consultations, Symptoms, diagnoses, Treatments.
5. The doctor reviews the information and updates Diagnosis and Treatment for the Consultation.
6. After the consultation, the system updates the patient's medical record with the latest information.

- **Extensions:**
  - **1a**: If login details are incorrect, the system asks doctor to log in again
- **Special Requirements:**
  - Real-time synchronization of patient data is required to ensure up-to-date information is available.

2.4.9 Use Case Name: Manage Appointment

- **Scope**: CureTrack
- **Level**: User Goal
- **Primary Actor:** Doctor
- **Stakeholders and Interests:**

  - **Doctor**: Wants to manage appointments efficiently by confirming.
  - **Patient**: Wants timely updates on appointment status changes.
- **Preconditions:**
  - The doctor is registered and logged into the system.
  - Appointments for the doctor are already stored in the system.

- **Success Guarantee (Postcondition):**
  - Appointment confirmations are reflected in the system.
  - The displayed appointment table updates dynamically.

- **Main Success Scenario:**

1. The doctor logs into the system and navigates to the **Manage Appointment** page.

2. The system fetches and displays all appointments in the **Table View**.

3. The doctor chooses an appointment from the table and selects "Confirm".

4. System notifies the Patient that your appointment has been confirmed.

- **Extensions:**
  - **1a**: If login details are incorrect, the system asks doctor to log in again
  - **3a.** If no appointment is selected for confirmation:
    - The system shows a warning prompting the doctor to select an appointment.

- **Special Requirements:**
  - The text should be visible from 3 meters.
  - User-friendly GUI.

2.4.10 Use Case Name: Allocate Resources in Emergency (Muhammad Talha)

- **Scope:** CureTrack
- **Level:** User Goal
- **Primary Actor:** Staff
- **Stakeholders and Interests:**
  - **Staff:** Needs to allocate resources efficiently in case of emergency.
  - **Patients:** Want immediate and prioritized care during emergencies.
- **Preconditions:**
  1. The hospital staff is logged into the system and has access to real-time data.
  2. The patient has reported an emergency or emergency cases have been automatically logged into the system.
- **Success Guarantee (Postcondition):**
  - Resources are allocated based on the priority of emergency cases reported by the patient.

- **Main Success Scenario:**

1. Patient logs into the System and Selects "Emergency"
2. Patient Selects a Hospital from the List and Fills in type of Emergency and it's Details
3. System notifies all the Staffs of the Selected Hospital about the Emergency
4. The staff logs into the system & selects the allocate resources tab.
5. The system shows the data for emergency (emergency id, patient name, hospital name, type , status) in a table format.
6. The staff selects data from the list and clicks "Confirm Button"
7. The system changes the status from Pending to Allocated

- **Extensions:**
    - **1a**: If login details are incorrect, the system asks patient to log in again
    - **6a.** If the staff doesn't select any data and clicks confirm, the system gives an alert to select specific data.
    - **4a**: If login details are incorrect, the system asks staff to log in again

- **Special Requirements:**
  - The system must support real-time updates on resource allocation and emergency case status.
  - The system should enable dynamic reallocation of resources during ongoing emergencies as new cases are reported.
- **Technology and Data Variations List:**
  - Accessible through different versions of windows.
  - Real-time change of resource updates.

2.4.11 Use Case Name: Manage Inventory (Ayna Sulaiman)

- **Scope:** CureTrack
- **Level:** User Goal
- **Primary Actor:** Staff
- **Stakeholders and Interests:**
  - **Staff:** Wants to keep supplies stocked and organized.
  - **Admin:** Focused on budget management and efficiency.
- **Preconditions:**
  - The user is logged in as Staff.
  - Inventory records exist in the system.

- **Success Guarantee (Postcondition):**
  - The inventory is updated with accurate stock levels, and low stock alerts are sent.
- **Main Success Scenario:**

1. Staff logs into the system.
2. The manager selects the "Manage Inventory" option.
3. The system displays current inventory levels.
4. The manager updates inventory through a slider and clicks on a “Save Button”.
5. The system confirms the changes and updates the database.

- **Extensions:**
  - **1a**: If login details are incorrect, the system asks staff to log in again
  - **3a:** If any item details are incomplete, the system prompts for completion.
  - **5a:** If stock updates fail due to an error, the system provides an error message.
- **Special Requirements:**
  - The text should be visible from 3 meters.
  - User-friendly GUI.
- **Technology and Data Variations List:**
  - Different Desktop access for inventory updates.

2.4.12 Use Case Name: Pay Bills (Ayna Sulaiman)

- **Scope:** CureTrack
- **Level:** User Goal
- **Primary Actor:** Patient
- **Stakeholders and Interests:**
  - **Patients:** Expect a simple, transparent process to view and pay their bills
- **Preconditions:**
  - The patient has a treatment/service history and unpaid bills available in the system.
  - The patient is logged in.
- **Success Guarantee (Postcondition):**
  - The invoice is paid, and payment records are updated.
- **Main Success Scenario:**

1. Patient logs into the system and navigates to the "Pay Bills" section on their dashboard.
2. The system displays details of all unpaid bills such as Bill Id, description, amount and status.
3. The patient selects an unpaid bill from the list.
4. The patient clicks on pay.
5. The system changes the status from “pending” to "paid."

Extensions:

- **1a**: If login details are incorrect, the system asks patient to log in again
- **4a:** If the patient doesn’t select any bill and clicks “pay”, the system gives an alert to select a specific bill.

- **Special Requirements:**
  - The text should be visible from 3 meters.
  - User-friendly GUI.

2.4.13 Use Case Name: Submit and View Feedback (Ayna Sulaiman)

- **Scope:** CureTrack
- **Level:** User Goal
- **Primary Actor:** Patient, Doctor
- **Stakeholders and Interests:**
  - **Patients:** Want to express their experience about specific doctors .
  - **Doctors:** Want to see feedback regarding their services.
- **Preconditions:**
  - The patient or doctor is logged into the system.
- **Success Guarantee (Postcondition):**
  - Feedback is recorded, and doctors can view the feedback.
- **Main Success Scenario:**

1. Patient logs in and navigates to the feedback section.

2. The system prompts the patient to enter all the required fields.

3. Patients select the specific doctor to provide feedback.

4. Patient fills out the feedback form with the experience and Recommendations and submits it.

5. The doctor logs in and selects “feedback”.

6 The system shows a table having all the feedback of that specific doctor.

7. The Doctor can view the feedback.

- **Extensions:**
  - **1a**: If login details are incorrect, the system asks patient to log in again.
  - **4a:** If any of the fields except comment are missing the system gives an alert that all fields MUST be filled.
  - **5a:** If login details are incorrect, the system asks the doctor to log in again.
- **Special Requirements:**
  - The text should be visible from 3 meters.
  - User-friendly GUI.

2.4.14 Use Case Name: Availability (Ayna Sulaiman)

- **Scope:** CureTrack
- **Level:** User Goal
- **Primary Actor:** Doctor
- **Stakeholders and Interests:**
  - **Doctors:** Need to manage their appointments efficiently,
  - **Patient:** Needs to be notified about the cancellation of the appointment.

- **Preconditions:**
  - The doctor is logged into the system.
  - The doctor’s availability has been established.
- **Success Guarantee (Postcondition):**
  - The doctor's schedule is updated across all platforms, patients are notified of changes, and any rescheduled appointments are confirmed.
- **Main Success Scenario:**

1. The doctor logs in and views their current schedule.
2. The doctor selects the availability button.
3. The system shows all appointments of that specific doctor in a table view.
4. The doctor selects a specific appointment and clicks on the cancel appointment button.
5. The systems deletes that appointment from the table and the database and notifies the patient to reschedule appointment

- **Extensions:**
  - **1a**: If the login details are incorrect then the patient is asked to log in again.
  - **4a:** if no appointment is selected and cancel appointment button is clicked the system shows an alert that an appointment should be selected
- **Special Requirements:**
  - The text should be visible from 3 meters.
  - User-friendly GUI
- **Technology and Data Variations List:**
  - Page can be accessed on different windows versions.

- Page cant be accessed on Linux.

2.4.15 Use Case 15: Remove Doctors/Patient/Staff

- **Scope:** CureTrack
- **Level:** User Goal
- **Primary Actor:** Admin
- **Stakeholders and Interests:**
  - **Admin:** Need to remove Doctor/Patient/Staff to manage Hospital operations effectively.
- **Preconditions:**
  - The Admin is logged into the system.
- **Success Guarantee (Postcondition):**
  - The user's account to be deleted has been successfully terminated.
- **Main Success Scenario:**

1. Admin logs into the system

2. Admin Selects either Remove Doctor/Remove Patient/Remove Staff from Home page according to his requirement

3. System displays a drop-down menu of all existing user

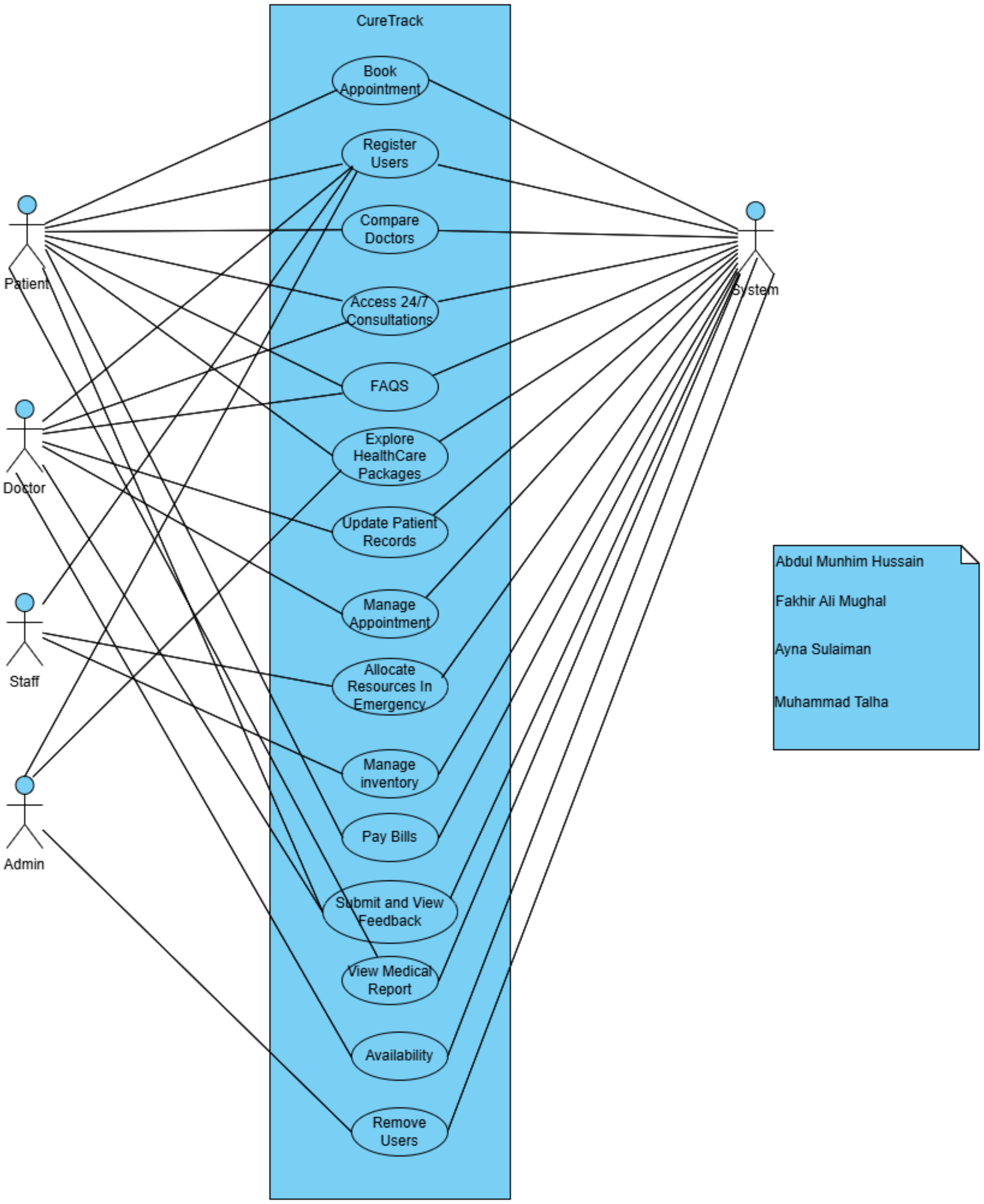
4. Admin Selects the User Name and clicks on “Delete”

5. The System Deletes the User from Database and updates it.

- **Extensions:**
    - **1a**: If the login details are incorrect then the Admin is asked to log in again.
    - **4a**. If Admin doesn’t select any user and clicks on delete then system shows a alert to select a user

- **Special Requirements:**
    - The text should be visible from 3 meters.
    - User-friendly GUI

2.5 Use Case Diagram



3. Other Nonfunctional Requirements

3.1 Performance Requirements

- The system must handle at least **500 simultaneous requests** without performance degradation.
- **Appointment booking**, **billing**, and **video consultation functionalities** should complete within **3 seconds** under standard operating conditions.
- Support **real-time room allocation and availability updates** without delays.
- Account creation within fraction of seconds
- Database updates within fraction of seconds
- Feedback table is updated quickly
- Hospitals are added quickly
- If admin removes staff, doctor or patient so it is done very quickly.

3.2 Safety Requirements

- Implement **role-based access control** to prevent unauthorized access to sensitive data such as patient records and hospital resource details.
- Only **Admin** can add hospitals
- Only Admin can remove doctors, patients and staffs.
- Only authorized **Staff** can manage resources
- **Patient** cannot do changes in hospitals
- Patient cannot add or remove doctors
- Patient cannot add or remove staffs
- **Doctors** cannot add or remove patients
- Doctors cannot add or remove staffs

- Staff cannot add or remove patients
- Staff cannot add or remove doctors
- Doctor cannot view other doctors’ queries

3.3 Security Requirements

- Username is matched with account’s password in database.
- Accounts’ passwords have input validations 8 characters long string/
- Each entity such as doctor, patient, staff and admin have separate database for login credentials.

3.4 Software Quality Attributes

- **Usability:** The interfaces are user-friendly. A person with basic skills can use it.
- **Reliability:** The system has an uptime of 99.99%. It is very fast and optimized.
- **Scalability:** Support expansion to additional hospitals without significant redevelopment. New hospitals can be added easily by an admin account.

3.5 Business Rules

1. **Admins** can add healthcare packages, remove doctors, remove patient, remove staff and add hospitals.
2. **Staff** can manage inventory, allocate resources and rooms.
3. **Doctors** can manage appointments, video consult, review feedback, update availability and answer FAQs.
4. **Patients** can book appointments, compare hospitals/doctors, FAQs, use emergency POS, give feedback and make payments.

3.6 Operating Environment

- The software support devices commonly found in hospitals, such as desktop computers, Laptops, Chromebooks and KIOSK screens.
- The software supports the following Operating Systems
  1. Window XP, 7, 8, 9, 10, 11, 12 and VISTA.
  2. Windows 10/11 IoT Enterprise.
  3. macOS Big Sur (11).
  4. macOS Monterey (12).
  5. macOS Ventura (13).
  6. macOS Sonoma (14).
- It does not support Linux or Ubuntu

3.7 User Interfaces

The user interface of Cure Track is designed to ensure simplicity and ease of use for all types of users, including hospital administrators, medical staff, doctors and patients. Below are the logical characteristics and guidelines for the interface:

1. **Interface for Patients**

   - **Book Appointment Page**: A form-based interface to select Hospital, Specialty, doctor, appointment date, and time slot.
   - **Compare Page:** A form-based interface to compare and doctors.
   - **FAQs Page:** A form-based interface to ask question to specific doctors and to view previously asked questions.
   - **Emergency Page:** This interface is used for emergency cases to find closest hospitals nearby based on selected specialty.

- **Feedback Page:** A form-based interface to give feedback to specific doctors of specific hospitals and experience specifically “Excellent,” “Good,” “Average” or “Poor”.
- **Payment Page**: A form-based interface to pay unpaid bills to specific doctors of specific hospitals.
- **Menu Bar:** Shows list of Notification

2. **Interface for Doctors**
    - **Manage Appointment**: A form-based interface to confirm or delete appointment.
    - **Video Consultation**: A form-based interface to consult the appointment by online video call.
    - **Feedback**: A form-based interface to view feedbacks given by patients.
    - **Availability:** A form-based interface to update availability and unavailability for the appointments.
    - **FAQs:** A form-based interface to answer questions asked by patients particularly to logged in doctor.
    - **Menu Bar:** Shows list of Notification

3. **Interface for Staff**

    - **Manage Inventory**: A form-based interface for real-time management of medicines, stocks and miscellaneous in the inventory. It uses 12 scroll bars 1 for each stuff.
    - **Allocate Resources**: A form-based interface to allocate rooms, resources and staffs to hospitals.
    - **Room Allocation**: A form-based interface for viewing rooms’ statuses , alerts and high alerts for maintenance.
    - **Menu Bar:** Shows list of Notification

4. **Interface for Admin**
    - **Add Healthcare Package:** A form-based interface for adding new packages by giving package name, start date, end date, hospital name, price and description. It also contains a table for viewing current packages.

- **Remove Doctor:** A form-based interface used to remove doctor by selection from a drop-down menu.
- **Remove Patient:** A form-based interface used to remove patient by selection from a drop-down menu.
- **Remove Staff:** A form-based interface used to remove staff by selection from a drop-down menu.
- **Add Hospital:** A form-based interface used to remove hospital by selection from a drop-down menu.

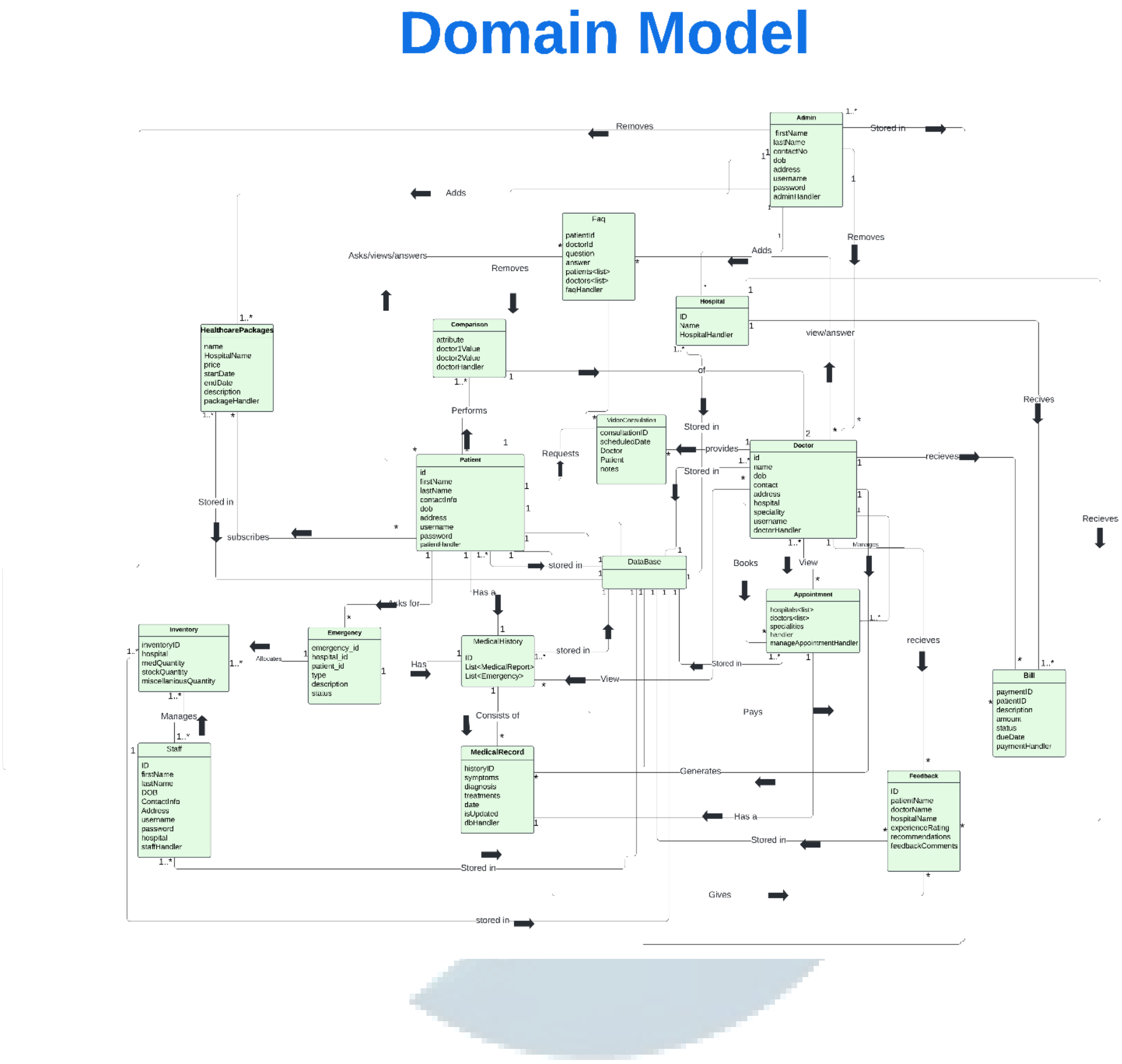
5. **Design Guidelines**

- **GUI Standards**: Adherence to Material Design principles for a consistent and modern look. CureTrack's GUI follows 12 principals of Material Design principals:
    1. contrast
    2. balance
    3. emphasis
    4. proportion
    5. hierarchy
    6. repetition
    7. rhythm
    8. pattern
    9. white space
    10. movement
    11. variety
    12. unity.
- **Screen Layout Constraints**: Good design ensuring usability on desktops, laptops, and KIOSK Screens.
- **Standard Buttons and Functions**:
    1. A logout button is available on home screens after login to logout.
    2. Registration for all entities such as doctors, patients, staffs and admins.
    3. Login for all entities such as doctors, patients, staffs and admins.

4. "**Ask**", "**Add**" and "**Save**" Buttons for FAQs, hospital addition and package addition in required places for respective forms.
5. Consistent placement of primary actions "**Confirm**," "**Cancel**" and "**Modify**" across forms accordingly.
6. A "**Back**" button for easy navigation.
7. A "**Pay**" button to pay bills
8. Proper Feedback system connected with database
9. Proper Appointment Management system's functionality linked with database for interlinkage of patient and doctor.

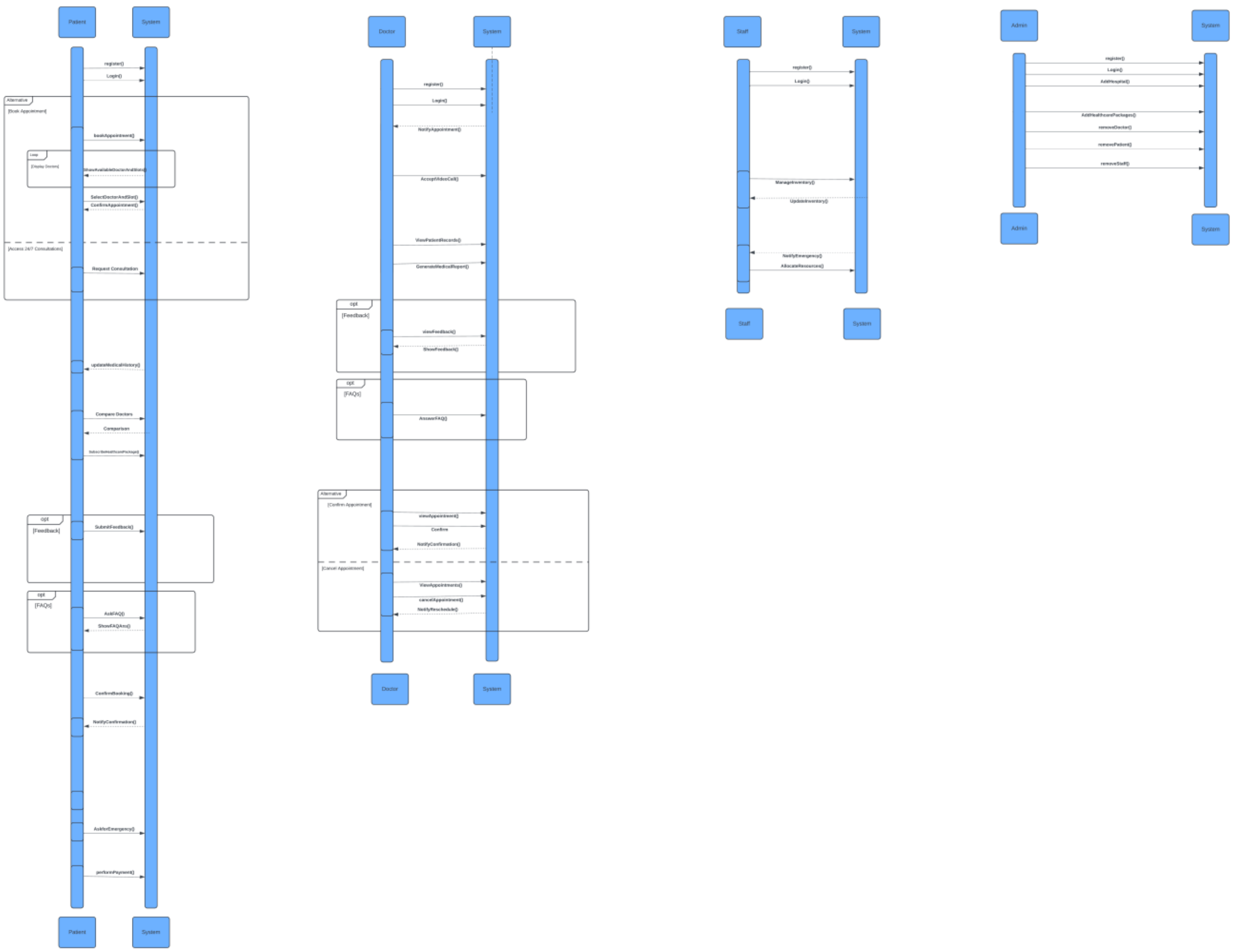
- **Error Message Display**:
  1. Real-time validation errors:
     - "Please enter a valid date of Birth"
     - "You cannot select a date that has passed").
     - "Invalid username or password"
     - "Contact number must be 11 digits"
     - "Password must be 8 characters long"
     - "Please fill all fields before confirming the appointment"
     - "Please enter an appointment ID to delete"
- **Success Message Displays:**
  1. Messages that display on success scenario are given below:
     - "The appointment has been deleted"
     - "Admin registered successfully"
     - "Patient registered successfully"
     - "Doctor registered successfully"
     - "Staff registered successfully"

4. Domain model

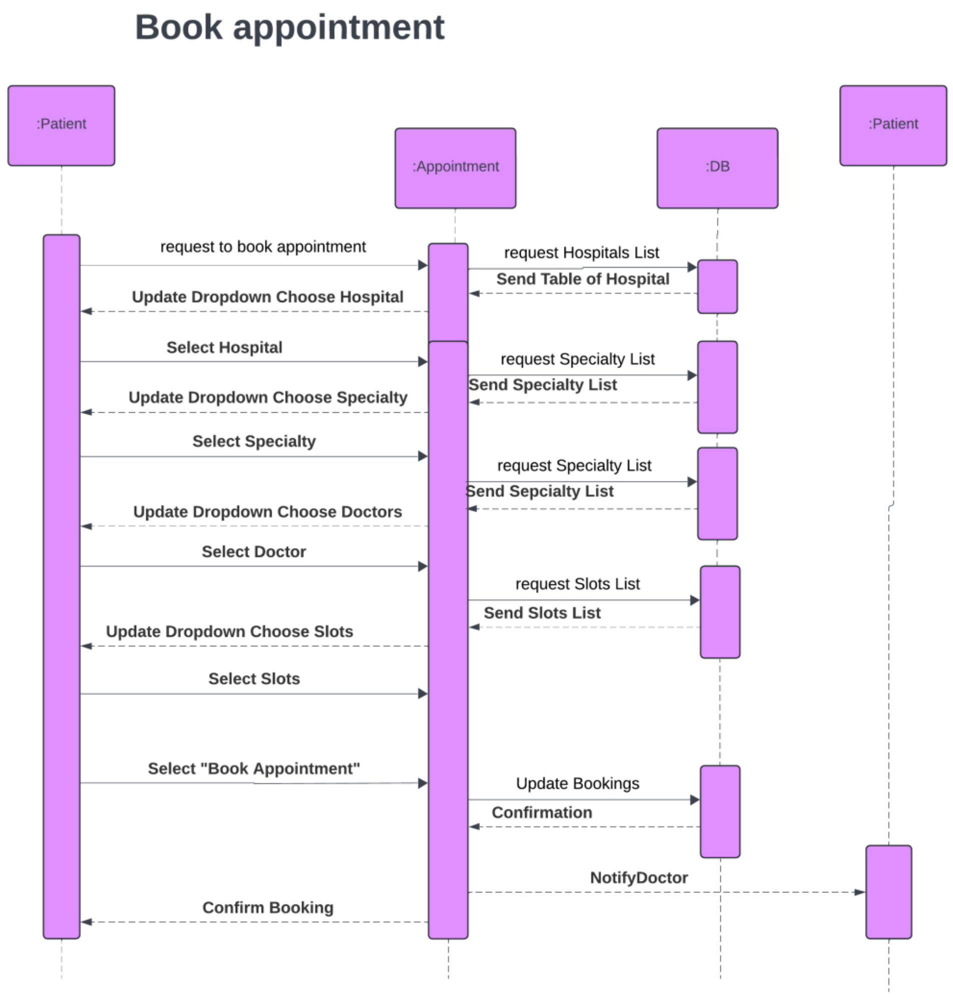


5. System Sequence

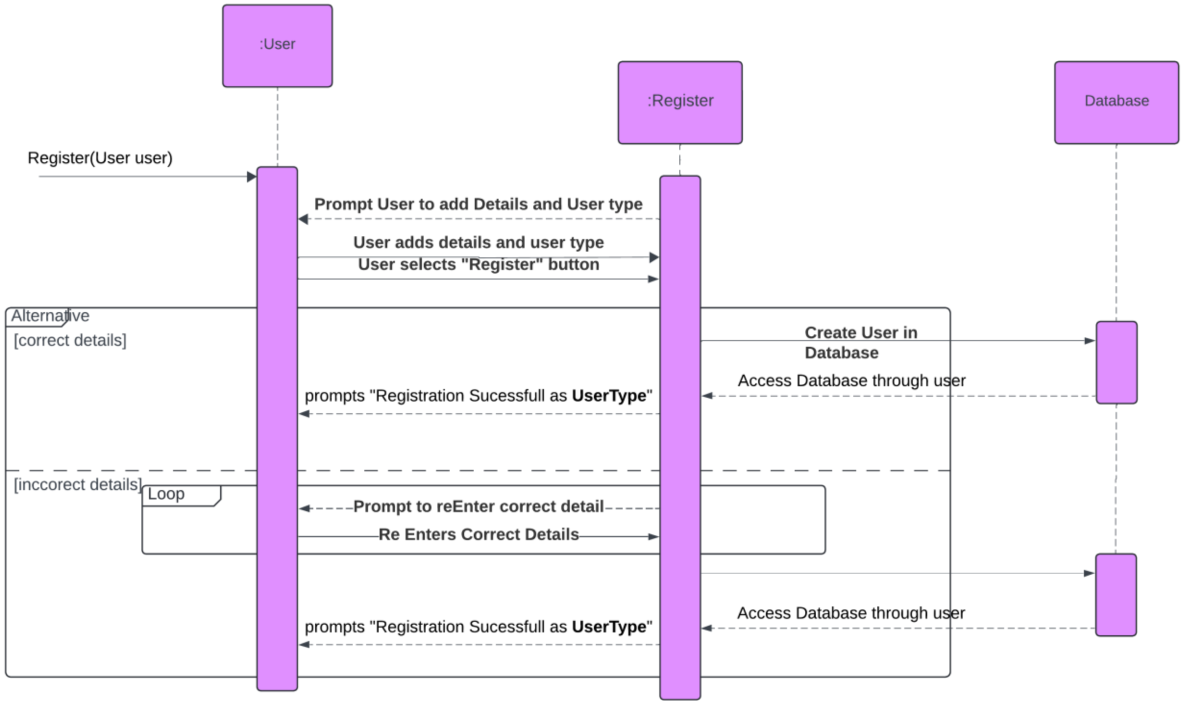
System Sequence



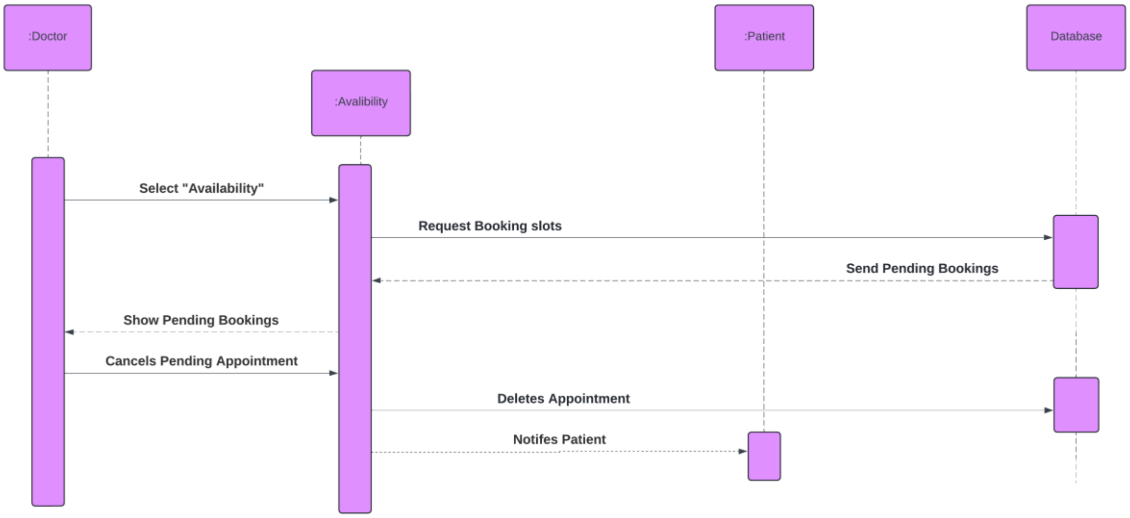
6. Action Sequence



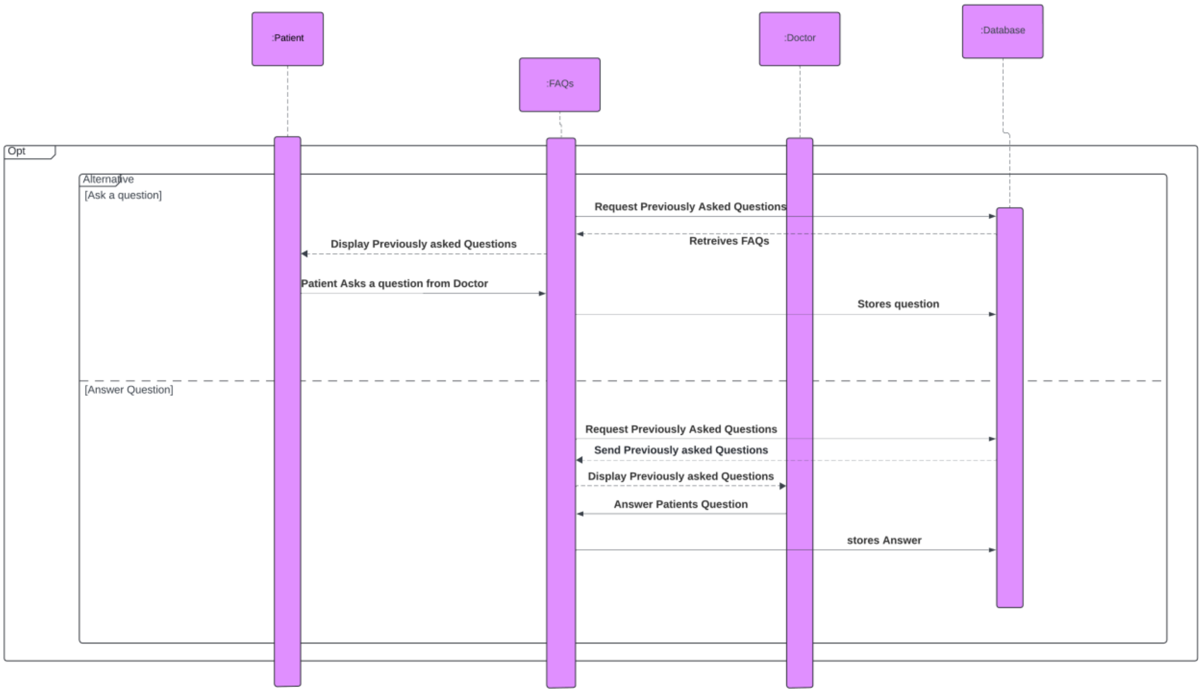
Register Doctor/Patients/Staff

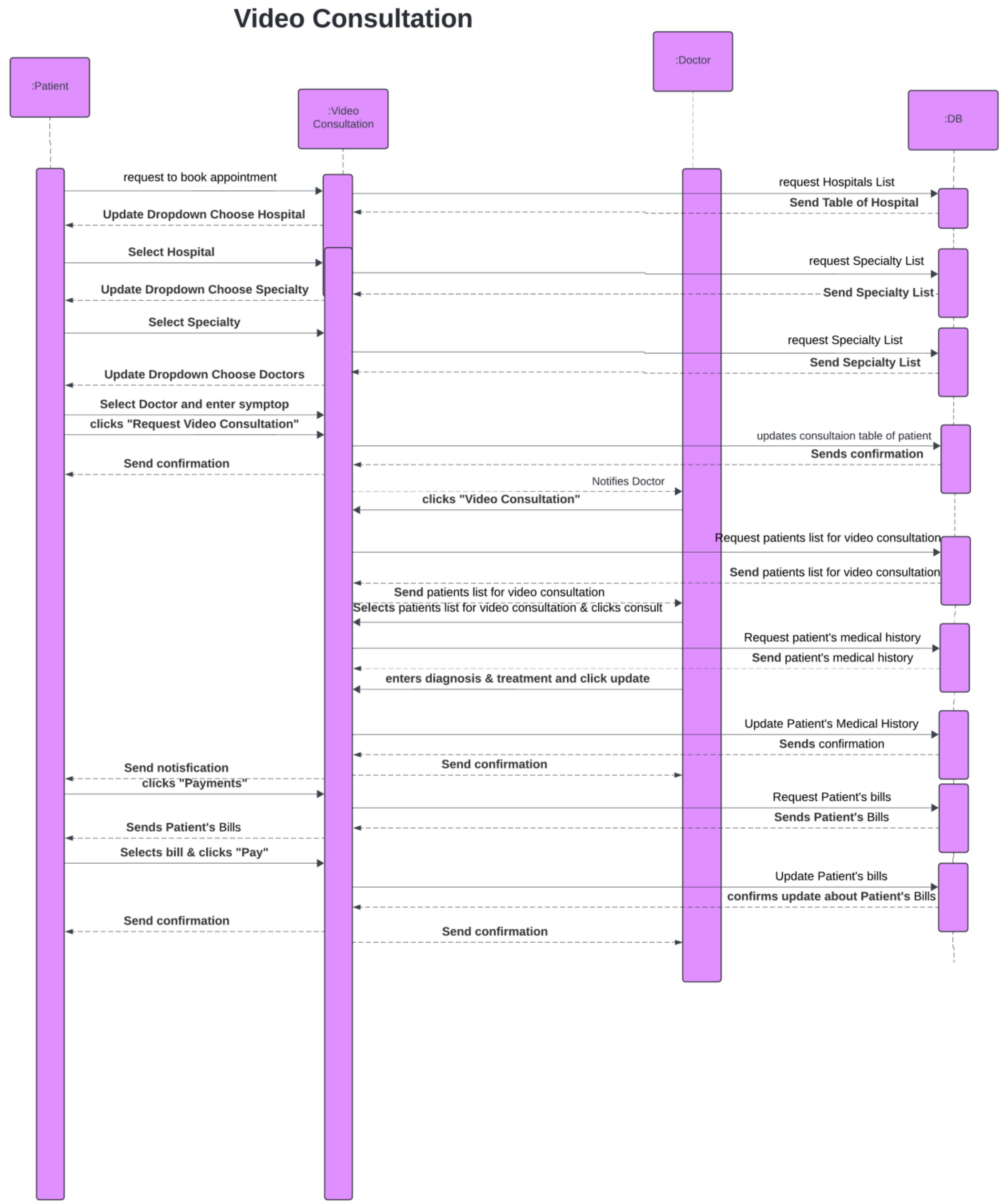

:User
:Register
Database
Register(User user)
Prompt User to add Details and User type
User adds details and user type
User selects "Register" button
Alternative
[correct details]
Create User in Database
Access Database through user
prompts "Registration Sucessfull as **UserType**"
[inccorect details]
Loop
Prompt to reEnter correct detail
Re Enters Correct Details
Access Database through user
prompts "Registration Sucessfull as **UserType**"

Avalibility

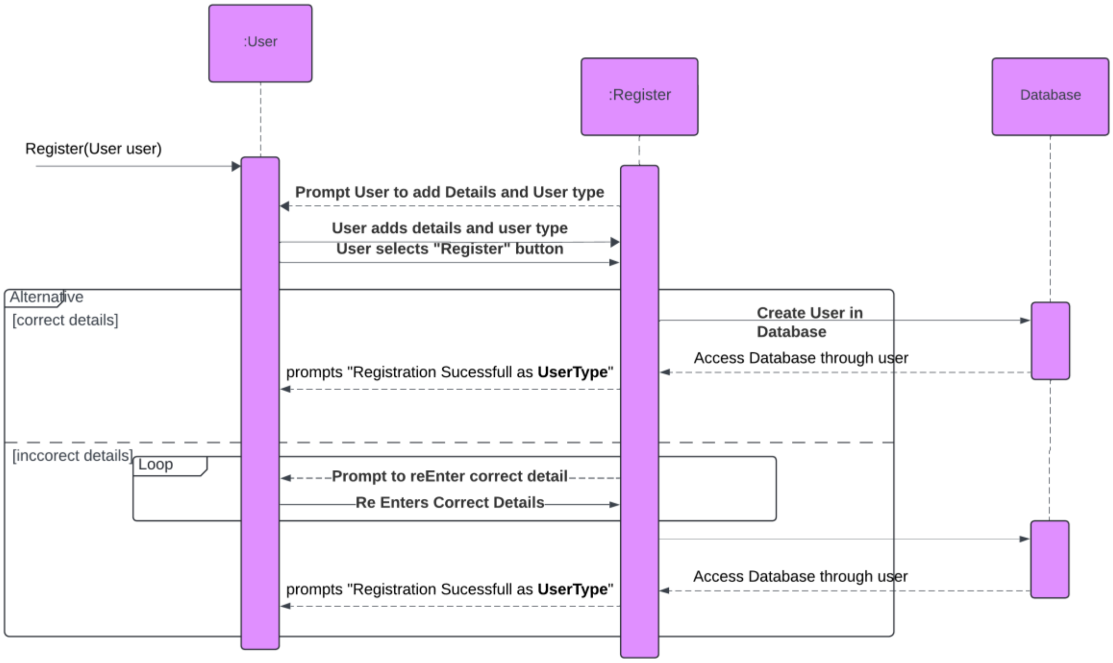


FAQs

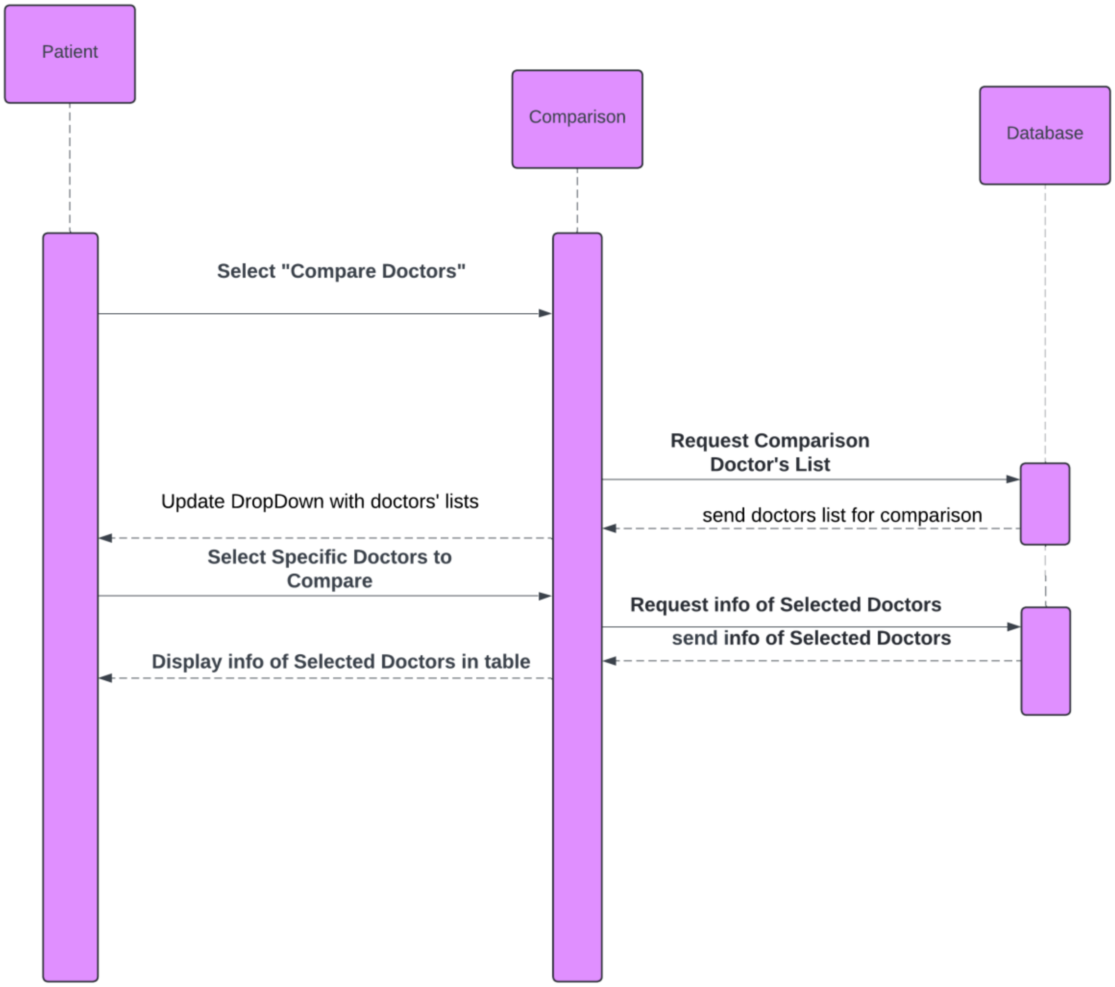

:Patient
:FAQs
:Doctor
:Database
Opt
Alternative
[Ask a question]
Request Previously Asked Questions
Retreives FAQs
Display Previously asked Questions
Patient Asks a question from Doctor
Stores question
[Answer Question]
Request Previously Asked Questions
Send Previously asked Questions
Display Previously asked Questions
Answer Patients Question
stores Answer


Video Consultation
:Patient
:Video Consultation
:Doctor
:DB
request to book appointment
request Hospitals List
Send Table of Hospital
Update Dropdown Choose Hospital
Select Hospital
request Specialty List
Update Dropdown Choose Specialty
Send Specialty List
Select Specialty
request Specialty List
Send Sepcialty List
Update Dropdown Choose Doctors
Select Doctor and enter symptop
clicks "Request Video Consultation"
updates consultaion table of patient
Sends confirmation
Send confirmation
Notifies Doctor
clicks "Video Consultation"
Request patients list for video consultation
Send patients list for video consultation
Send patients list for video consultation
Selects patients list for video consultation & clicks consult
Request patient's medical history
Send patient's medical history
enters diagnosis & treatment and click update
Update Patient's Medical History
Sends confirmation
Send confirmation
Send notisfication
clicks "Payments"
Request Patient's bills
Sends Patient's Bills
Sends Patient's Bills
Selects bill & clicks "Pay"
Update Patient's bills
confirms update about Patient's Bills
Send confirmation
Send confirmation

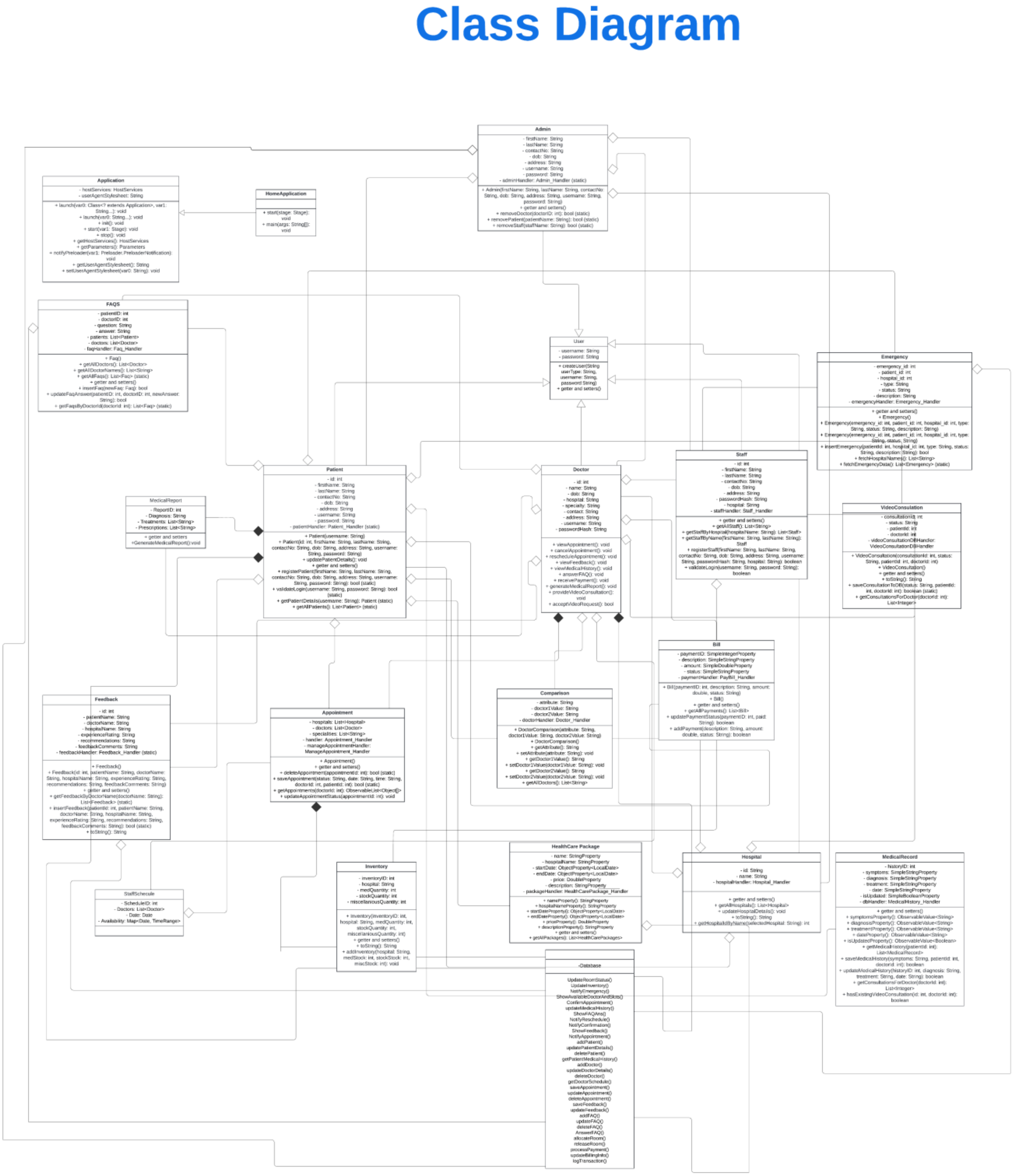
Register Doctor/Patients/Staff


:User
:Register
Database
Register(User user)
Prompt User to add Details and User type
User adds details and user type
User selects "Register" button
Alternative
[correct details]
Create User in Database
Access Database through user
prompts "Registration Sucessfull as **UserType**"
[inccorect details]
Loop
Prompt to reEnter correct detail
Re Enters Correct Details
Access Database through user
prompts "Registration Sucessfull as **UserType**"

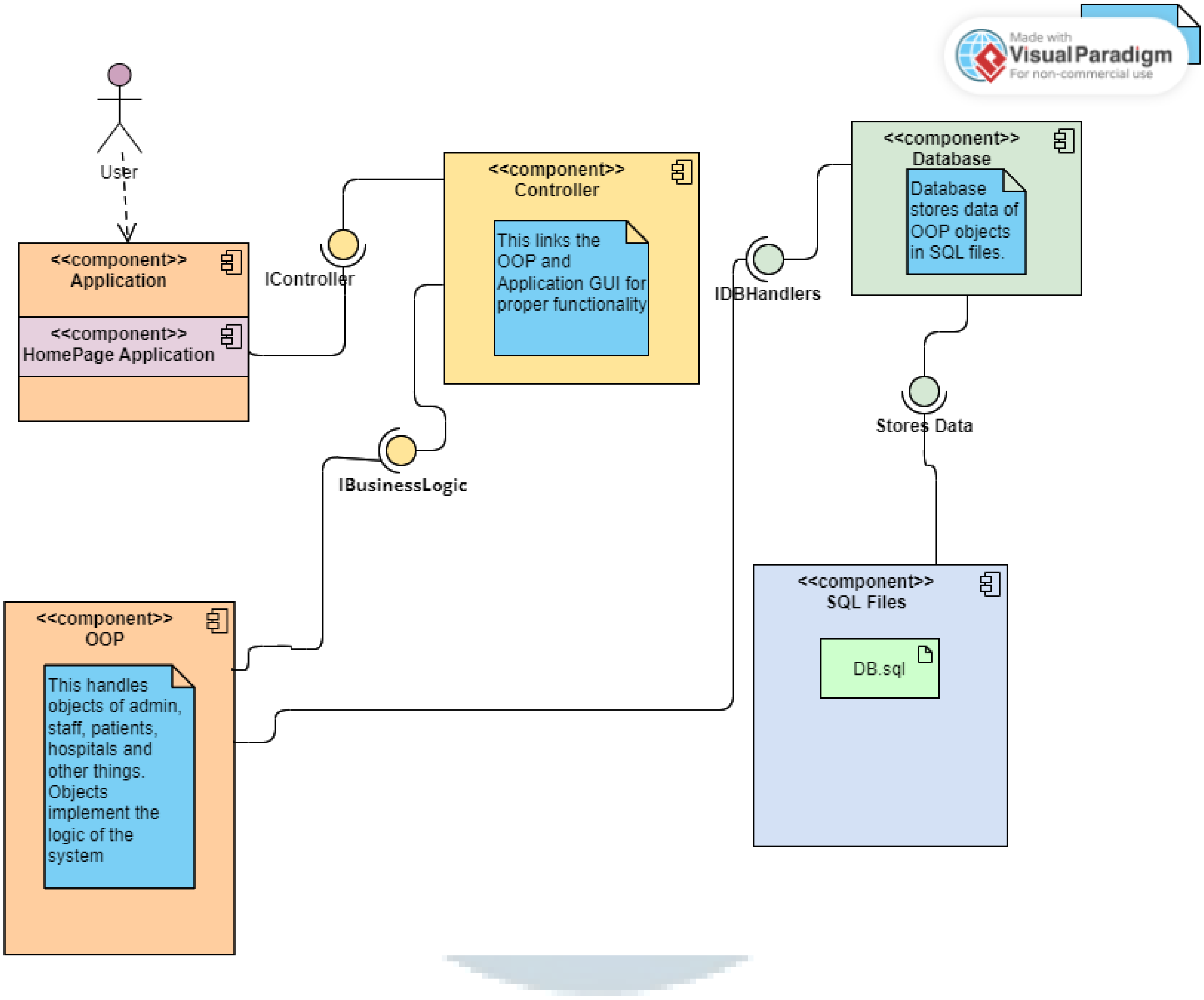
Compare Doctors


Patient
Comparison
Database
Select "Compare Doctors"
Request Comparison Doctor's List
Update DropDown with doctors' lists
send doctors list for comparison
Select Specific Doctors to Compare
Request info of Selected Doctors
send info of Selected Doctors
Display info of Selected Doctors in table

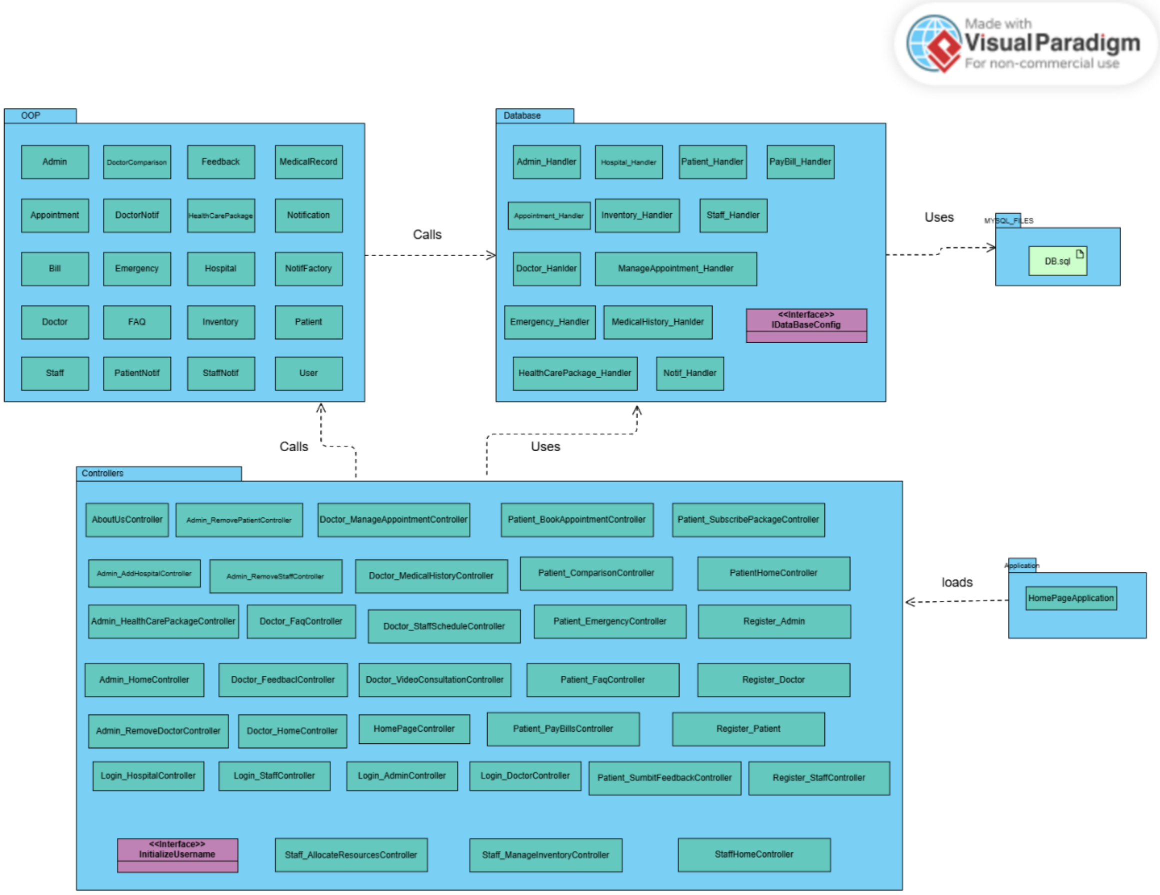
7. Class Diagram


Class Diagram
Admin
- firstName: String
- lastName: String
- contactNo: String
- dob: String
- address: String
- username: String
- password: String
- adminHandler: Admin_Handler (static)
+ Admin(firstName: String, lastName: String, contactNo: String, dob: String, address: String, username: String, password: String)
+ getter and setters()
+ removeDoctor(doctorID: int): bool (static)
+ removePatient(patientName: String): bool (static)
+ removeStaff(staffName: String): bool (static)
Application
- hostServices: HostServices
- userAgentStylesheet: String
+ launch(var0: Class<? extends Application>, var1: String...): void
+ launch(var0: String...): void
+ init(): void
+ start(var1: Stage): void
+ stop(): void
+ getHostServices(): HostServices
+ getParameters(): Parameters
+ notifyPreloader(var1: Preloader.PreloaderNotification): void
+ getUserAgentStylesheet(): String
+ setUserAgentStylesheet(var0: String): void
HomeApplication
+ start(stage: Stage): void
+ main(args: String[]): void
FAQS
- patientID: int
- doctorID: int
- question: String
- answer: String
- patients: List<Patient>
- doctors: List<Doctor>
- faqHandler: Faq_Handler
+ Faq()
+ getAllDoctors(): List<Doctor>
+ getAllDoctorNames(): List<String>
+ getAllFaqs(): List<Faq> (static)
+ getter and setters()
+ insertFaq(newFaq: Faq): bool
+ updateFaqAnswer(patientID: int, doctorID: int, newAnswer: String): bool
+ getFaqsByDoctorId(doctorId: int): List<Faq> (static)
User
- username: String
- password: String
+ createUser(String userType: String, username: String, password:String)
+ getter and setters()
Emergency
- emergency_id: int
- patient_id: int
- hospital_id: int
- type: String
- status: String
- description: String
- emergencyHandler: Emergency_Handler
+ getter and setters()
+ Emergency()
+ Emergency(emergency_id: int, patient_id: int, hospital_id: int, type: String, status: String, description: String)
+ Emergency(emergency_id: int, patient_id: int, hospital_id: int, type: String, status: String)
+ insertEmergency(patientId: int, hospital_id: int, type: String, status: String, description: String): bool
+ fetchHospitalNames(): List<String>
+ fetchEmergencyData(): List<Emergency> (static)
MedicalReport
- ReportID: int
- Diagnosis: String
- Treatments: List<String>
- Prescriptions: List<String>
+ getter and setters
+GenerateMedicalReport():void
Patient
- id: int
- firstName: String
- lastName: String
- contactNo: String
- dob: String
- address: String
- username: String
- password: String
- patientHandler: Patient_Handler (static)
+ Patient(username: String)
+ Patient(id: int, firstName: String, lastName: String, contactNo: String, dob: String, address: String, username: String, password: String)
+ updatePatientDetails(): void
+ getter and setters()
+ registerPatient(firstName: String, lastName: String, contactNo: String, dob: String, address: String, username: String, password: String): bool (static)
+ validateLogin(username: String, password: String): bool (static)
+ getPatientDetails(username: String): Patient (static)
+ getAllPatients(): List<Patient> (static)
Doctor
- id: int
- name: String
- dob: String
- hospital: String
- specialty: String
- contact: String
- address: String
- username: String
- passwordHash: String
+ viewAppointment(): void
+ cancelAppointment(): void
+ rescheduleAppointment(): void
+ viewFeedback(): void
+ viewMedicalHistory(): void
+ answerFAQ(): void
+ receivePayment(): void
+ generateMedicalReport(): void
+ provideVideoConsultation(): void
+ acceptVideoRequest(): bool
Staff
- id: int
- firstName: String
- lastName: String
- contactNo: String
- dob: String
- address: String
- passwordHash: String
- hospital: String
- staffHandler: Staff_Handler
+ getter and setters()
+ getAllStaff(): List<String>
+ getStaffByHospital(hospitalName: String): List<Staff>
+ getStaffByName(firstName: String, lastName: String): Staff
+ registerStaff(firstName: String, lastName: String, contactNo: String, dob: String, address: String, username: String, passwordHash: String, hospital: String): boolean
+ validateLogin(username: String, password: String): boolean
VideoConsulation
- consultationId: int
- status: String
- patientId: int
- doctorId: int
- videoConsultationDBHandler: VideoConsultationDBHandler
+ VideoConsultation(consultationId: int, status: String, patientId: int, doctorId: int)
+ VideoConsultation()
+ getter and setters()
+ toString(): String
+ saveConsultationToDB(status: String, patientId: int, doctorId: int): boolean (static)
+ getConsultationsForDoctor(doctorId: int): List<Integer>
Bill
- paymentID: SimpleIntegerProperty
- description: SimpleStringProperty
- amount: SimpleDoubleProperty
- status: SimpleStringProperty
- paymentHandler: PayBill_Handler
+ Bill(paymentID: int, description: String, amount: double, status: String)
+ Bill()
+ getter and setters()
+ getAllPayments(): List<Bill>
+ updatePaymentStatus(paymentID: int, paid: String): boolean
+ addPayment(description: String, amount: double, status: String): boolean
Comparison
- attribute: String
- doctor1Value: String
- doctor2Value: String
- doctorHandler: Doctor_Handler
+ DoctorComparison(attribute: String, doctor1Value: String, doctor2Value: String)
+ DoctorComparison()
+ getAttribute(): String
+ setAttribute(attribute: String): void
+ getDoctor1Value(): String
+ setDoctor1Value(doctor1Value: String): void
+ getDoctor2Value(): String
+ setDoctor2Value(doctor2Value: String): void
+ getAllDoctors(): List<String>
Feedback
- id: int
- patientName: String
- doctorName: String
- hospitalName: String
- experienceRating: String
- recommendations: String
- feedbackComments: String
- feedbackHandler: Feedback_Handler (static)
+ Feedback()
+ Feedback(id: int, patientName: String, doctorName: String, hospitalName: String, experienceRating: String, recommendations: String, feedbackComments: String)
+ getter and setters()
+ getFeedbackByDoctorName(doctorName: String): List<Feedback> (static)
+ insertFeedback(patientId: int, patientName: String, doctorName: String, hospitalName: String, experienceRating: String, recommendations: String, feedbackComments: String): bool (static)
+ toString(): String
Appointment
- hospitals: List<Hospital>
- doctors: List<Doctor>
- specialities: List<String>
- handler: Appointment_Handler
- manageAppointmentHandler: ManageAppointment_Handler
+ Appointment()
+ getter and setters()
+ deleteAppointment(appointmentId: int): bool (static)
+ saveAppointment(status: String, date: String, time: String, doctorId: int, patientId: int): bool (static)
+ getAppointments(doctorId: int): ObservableList<Object[]>
+ updateAppointmentStatus(appointmentId: int): void
StaffSchecule
- ScheduleID: int
- Doctors: List<Doctor>
- Date: Date
- Availability: Map<Date, TimeRange>
Inventory
- inventoryID: int
- hospital: String
- medQuantity: int
- stockQuantity: int
- miscellaniousQuantity: int
+ Inventory(inventoryID: int, hospital: String, medQuantity: int, stockQuantity: int, miscellaniousQuantity: int)
+ getter and setters()
+ toString(): String
+ addInventory(hospital: String, medStock: int, stockStock: int, miscStock: int): void
HealthCare Package
- name: StringProperty
- hospitalName: StringProperty
- startDate: ObjectProperty<LocalDate>
- endDate: ObjectProperty<LocalDate>
- price: DoubleProperty
- description: StringProperty
- packageHandler: HealthCarePackage_Handler
+ nameProperty(): StringProperty
+ hospitalNameProperty(): StringProperty
+ startDateProperty(): ObjectProperty<LocalDate>
+ endDateProperty(): ObjectProperty<LocalDate>
+ priceProperty(): DoubleProperty
+ descriptionProperty(): StringProperty
+ getter and setters()
+ getAllPackages(): List<HealthCarePackages>
Hospital
- id: String
- name: String
- hospitalHandler: Hospital_Handler
+ getter and setters()
+ getAllHospitals(): List<Hospital>
+ updateHospitalDetails(): void
+ toString(): String
+ getHospitalIdByName(selectedHospital: String): int
MedicalRecord
- historyID: int
- symptoms: SimpleStringProperty
- diagnosis: SimpleStringProperty
- treatment: SimpleStringProperty
- date: SimpleStringProperty
- isUpdated: SimpleBooleanProperty
- dbHandler: MedicalHistory_Handler
+ getter and setters()
+ symptomsProperty(): ObservableValue<String>
+ diagnosisProperty(): ObservableValue<String>
+ treatmentProperty(): ObservableValue<String>
+ dateProperty(): ObservableValue<String>
+ isUpdatedProperty(): ObservableValue<Boolean>
+ getMedicalHistory(patientId: int): List<MedicalRecord>
+ saveMedicalHistory(symptoms: String, patientId: int, doctorId: int): boolean
+ updateMedicalHistory(historyID: int, diagnosis: String, treatment: String, date: String): boolean
+ getConsultationsForDoctor(doctorId: int): List<Integer>
+ hasExistingVideoConsultation(id: int, doctorId: int): boolean
-Database
UpdateRoomStatus()
UpdateInventory()
NotifyEmergency()
ShowAvailableDoctorAndSlots()
ConfirmAppointment()
updateMedicalHistory()
ShowFAQAns()
NotifyReschedule()
NotifyConfirmation()
ShowFeedback()
NotifyAppointment()
addPatient()
updatePatientDetails()
deletePatient()
getPatientMedicalHistory()
addDoctor()
updateDoctorDetails()
deleteDoctor()
getDoctorSchedule()
saveAppointment()
updateAppointment()
deleteAppointment()
saveFeedback()
updateFeedback()
addFAQ()
updateFAQ()
deleteFAQ()
AnswerFAQ()
allocateRoom()
releaseRoom()
processPayment()
updateBillingInfo()
logTransaction()

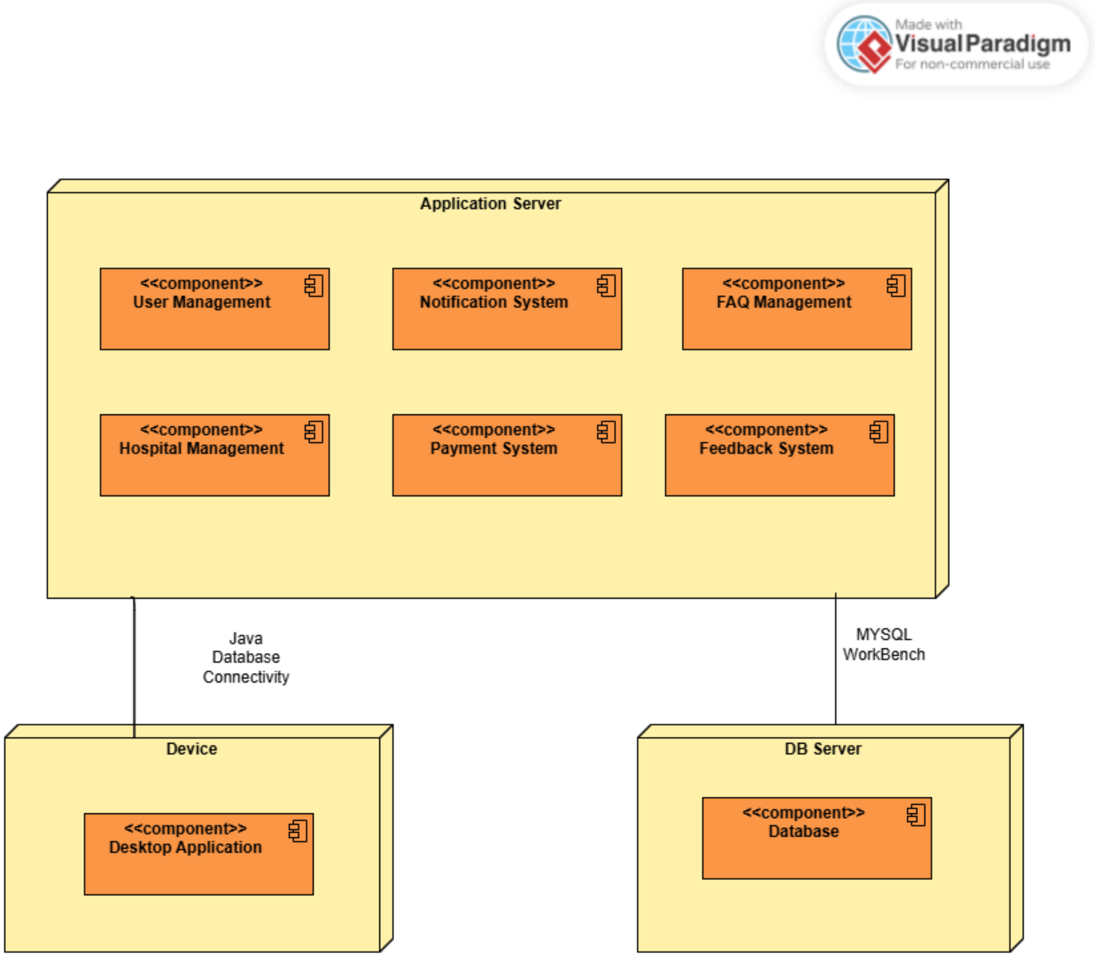
8. Component Diagram


Made with Visual Paradigm For non-commercial use
User
<<component>> Application
<<component>> HomePage Application
IController
<<component>> Controller
This links the OOP and Application GUI for proper functionality
IBusinessLogic
IDBHandlers
<<component>> Database
Database stores data of OOP objects in SQL files.
Stores Data
<<component>> OOP
This handles objects of admin, staff, patients, hospitals and other things. Objects implement the logic of the system
<<component>> SQL Files
DB.sql

9. Package Diagram



10. Deployment Diagram


Made with Visual Paradigm For non-commercial use
Application Server
<<component>> User Management
<<component>> Notification System
<<component>> FAQ Management
<<component>> Hospital Management
<<component>> Payment System
<<component>> Feedback System
Java Database Connectivity
MYSQL WorkBench
Device
<<component>> Desktop Application
DB Server
<<component>> Database