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 implementation.

1.2 Product Scope

Cure Track is a hospital management system aimed at streamlining hospital operations by centralizing all the operations within the hospital. The system provides core functionalities like user registration, user login, appointment booking, feedback, FAQs, video consultation, billing, staff scheduling and adding and subscribing healthcare packages. These are just some of the functionalities of CureTrack.

1.3 Title

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

1.4 Objectives

- **Centralize Hospital Operations**

  Streamline hospital management by integrating core functions into a single, unified system.
- **Enhance Patient Experience**
  Simplify patient interactions through features such as online appointment booking, video consultations, and access to FAQs.
- **Facilitate Feedback and Improvement**
  Collect and analyze user feedback to continuously improve hospital services.

- **Facilitate Doctor-Patient Communication**
  Enable proper question-and-answer interactions between patients and doctors to address health concerns effectively.
- **Handle Emergency Cases**
  Provide features to prioritize and manage emergency cases promptly and efficiently.
- **Empower Admins with Advanced Management Tools**
  Allow administrators to add healthcare packages, add hospitals, and remove doctors, staff, and patients as needed for efficient hospital operations.
- **Enable Real-Time Updates**
  Provide live updates for appointments, schedules, and notifications.

1.5 Problem Statement

The issue of inefficient and decentralized hospital management across a city impacts hospital administrators, medical staff, and patients, leading to delays, poor resource utilization, and reduced patient satisfaction. A centralized hospital management system offers an effective solution by integrating all operations into a single platform. This system would empower administrators to efficiently manage resources, add new hospitals, and handle staff or patient records. Patients could easily book appointments, make payments, and access healthcare services, while doctors could manage appointments and notify patients of any unavailability. Enhanced features like feedback collection and FAQs would further improve patient satisfaction. By digitizing hospital operations, this system would drive a significant transformation in healthcare management, making it more efficient and accessible 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 and to bridge the gap between patient and healthcare facilities. It is not a replacement, but a new platform designed for centralized control of hospital operations. It provides interfaces for admins,doctors, staff and patients, ensuring a seamless experience. Unlike traditional systems CureTrack integrates modern tools and features to streamline all the operations and ensure smooth coordination between all the users involved in the platform.

2.2 Product Functions

- User registration for admin, staff, doctors and patients.
- Appointment booking
- Appointment management.
- Video consultation
- Staff Scheduling
- Patient Feedback for Doctors.
- Emergency Cases 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
- Online payment system for consultation fees, treatments and healthcare packages

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 thefields.

- **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 the 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 Consultation.
- **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

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

4. 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 in a TableView .
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 or PC.
  - 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 "FAQ" 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-downmenu 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 alongwith 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 thefeedback 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 system 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 can’t 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

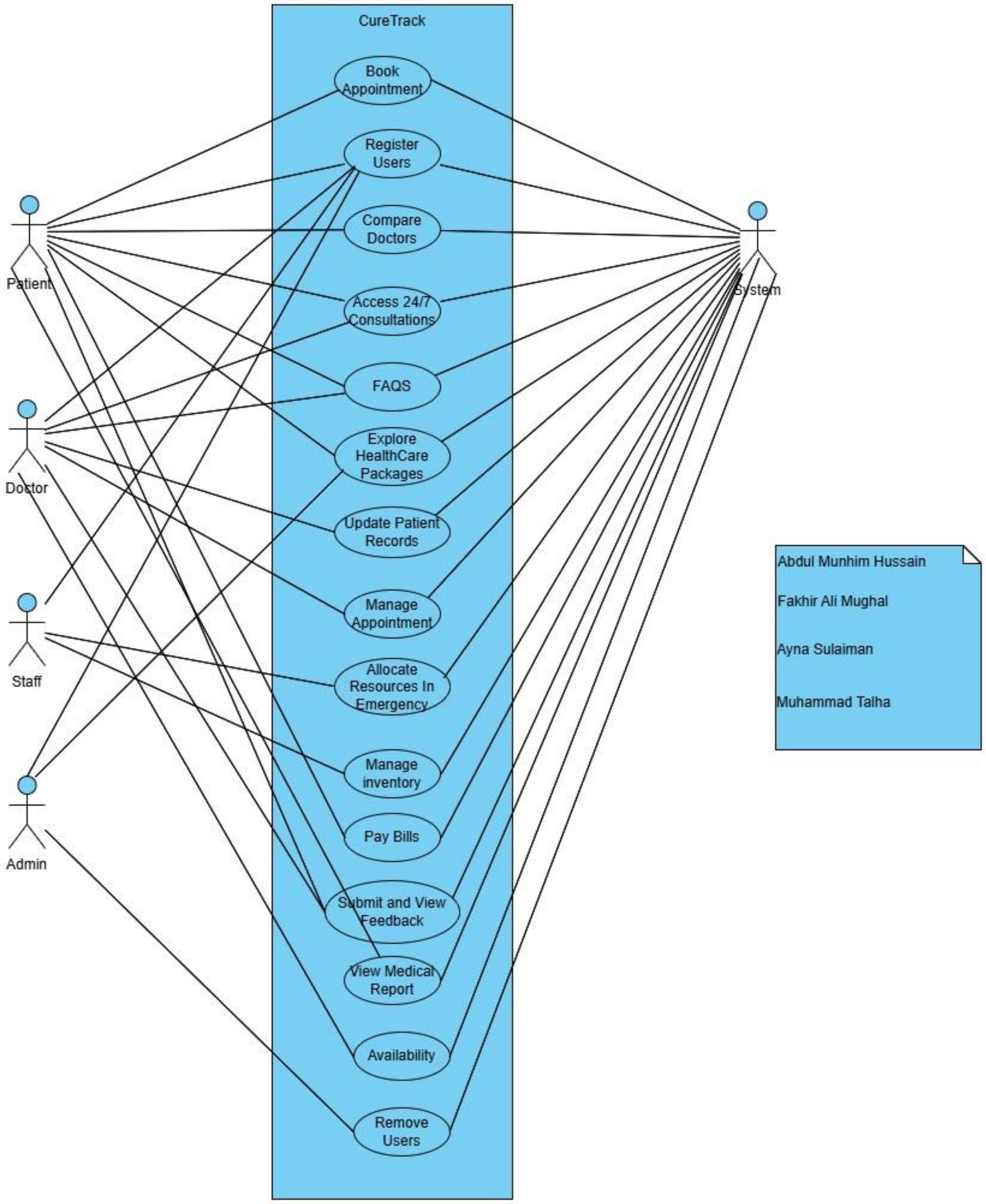
4. Admin Selects the User Name and clicks on “Delete”

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

System Performance

- Startup Time: The application should launch within 5 seconds on standard hardware (e.g., 8GB RAM, i5 processor, SSD storage).
- Response Time:
  - User interactions (e.g., button clicks, form submissions) should execute within 200 milliseconds.
  - Navigation between views (e.g., from the login page to the dashboard) should occur within 1 second.

User Interface

- Smooth Animations: JavaFX animations and transitions (e.g., during scene changes) must run smoothly at 60 FPS on supported hardware.
- Rendering Speed: Large tables, charts, or lists (e.g., patient or appointment data) should render within 1 second for up to 1,000 entries.

Data Handling

- Data Processing Speed: Transactions (appointment confirmation, video consultation, payment processing) should be completed within 2 second.
- Database Query Time: Queries to retrieve patient records or appointment schedules should execute in under 3 seconds.

3.2 Safety Requirements

Error Handling and Data Recovery

- The system must implement error handling to prevent accidental data loss or unauthorized changes, especially regarding medical records and billing information.

- In case of a system failure, the system must allow for the recovery of lost or unsaved work, with minimal downtime or disruption to hospital operations.

Compliance with Healthcare Regulations

- The system must comply with applicable healthcare data protection regulations.

- The system must ensure that patient data is handled and stored in compliance with privacy laws, providing necessary safeguards for sensitive health information.

Preventing Unauthorized Access to Critical Data

- The system must ensure that only authorized personnel, such as hospital administrators, can modify or delete critical patient data.

- The system must implement access logs to track all actions performed on patient records, providing an audit trail for review.

Audit and Monitoring:

- All system notifications and changes must be logged for auditing purposes

3.3 Security Requirements

Access Control and Authentication:

- ○ The system must implement **role-based access control (RBAC)** for users with access to sensitive information, ensuring only authorized personnel can access critical data

Data Encryption:

- ○ The application must encrypt sensitive user data (e.g., login credentials, medical records) both in transit and at rest using secure encryption protocols (e.g., TLS for transmission)

Data Security:

○ All sensitive data, including personal information, payment details, and event records, must be secured in the database

3.4 Software Quality Attributes

- **Usability**: The system must have an intuitive and user-friendly interface for hospital staff and patients.
- **Performance**: The system should respond quickly, even with multiple users accessing it simultaneously.
- **Reliability**: The system must be stable, with minimal downtime and effective error recovery.
- **Security**: Sensitive data must be encrypted, and access should be controlled via user authentication and role-based permissions.
- **Scalability**: The system should handle growth in users and data without performance issues.
- **Maintainability**: The system must be easy to update and fix, with well-structured code and documentation.

3.5 Business Rules

Role-Based Access Control: Every user has access and control of the system depending on their roles

1. **Administrator**
   - Add Healthcare Packages
   - Remove Doctors
   - Remove Patients
   - Remove Staff

2. **Patient**
   - Book Appointment
   - Compare Doctors
   - Ask for video consultation
   - Ask FAQS
   - Report Emergency
   - Provide Feedback
   - Pay Bills
   - Explore Healthcare Packages

3. **Doctors**
   - Manage Appointments
   - Provide video consultation
   - See feedback
   - Manage availability
   - Answer FAQS

4. **Staff**
   - Allocate Resources
   - Manage Inventory

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

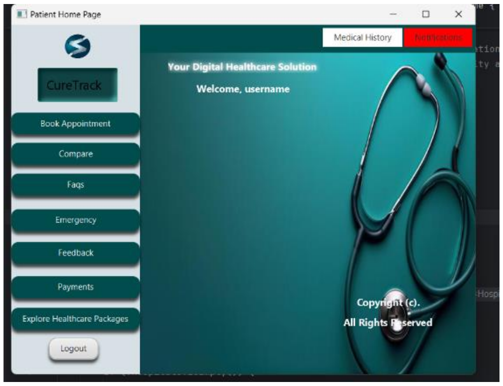
3.7 User Interfaces

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.

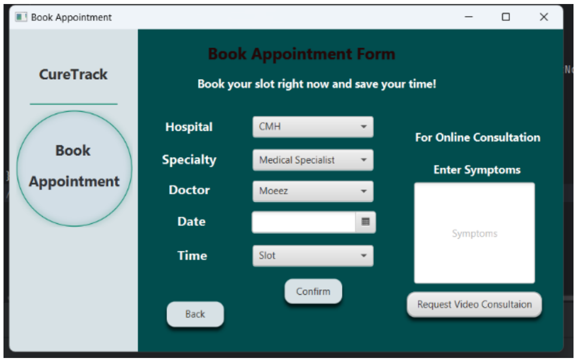
- **Feedback Page:** A form-based interface to give feedback to specific doctors of specific hospitals and experience specifically “Excellent,” “Good,” “Average” or “Poor”.
- **Payment Page**: A form-based interface to pay unpaid bills to specific doctors of specific hospitals.
- **Menu Bar:** Shows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



Your Digital Healthcare Solution

Welcome, username

Copyright (c).
All Rights Reserved



Book Appointment Form

Book your slot right now and save your time!

Hospital: CMH

Specialty: Medical Specialist

Doctor: Moeez

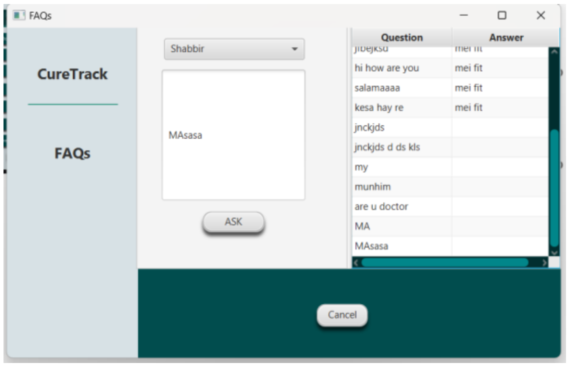
Date:

Time: Slot

For Online Consultation

Enter Symptoms

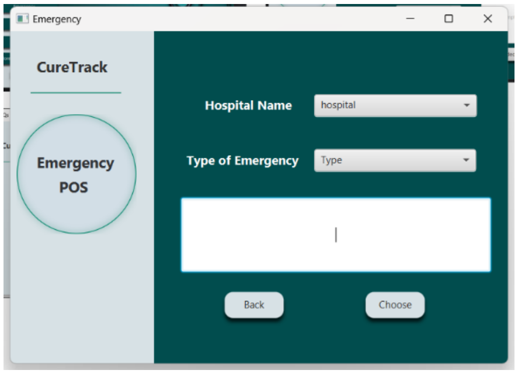
Symptoms



Shabbir

MAsasa

Question	Answer
jibejksd	mei fit
hi how are you	mei fit
salamaaaa	mei fit
kesa hay re	mei fit
jnckjds	
jnckjds d ds kls	
my	
munhim	
are u doctor	
MA	
MAsasa	

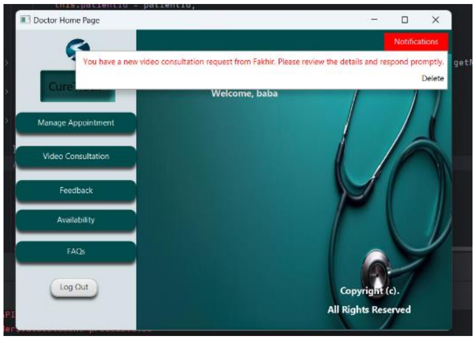


Hospital Name: hospital

Type of Emergency: Type

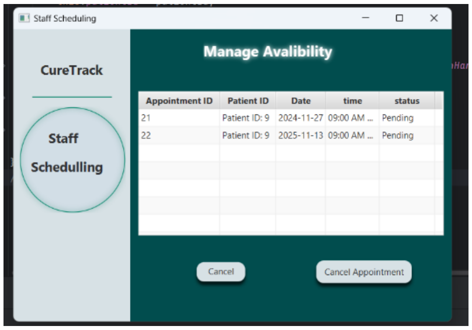
2. Interface for Doctors

- **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

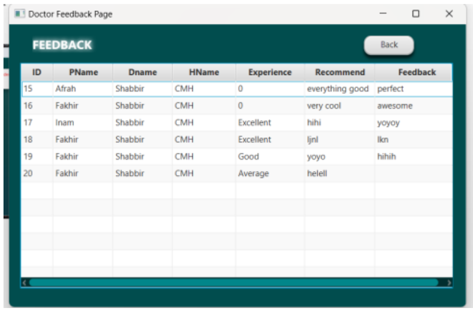


You have a new video consultation request from Fakhir. Please review the details and respond promptly.

Welcome, baba



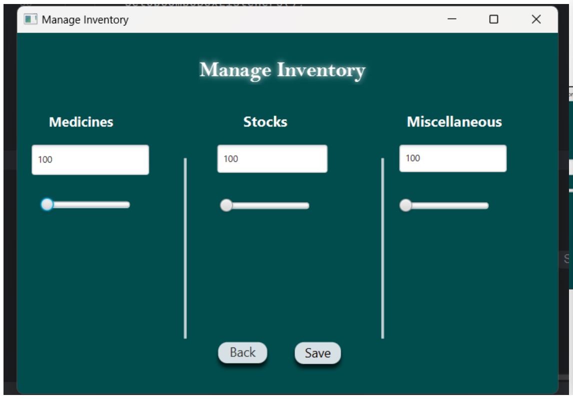
Appointment ID	Patient ID	Date	time	status
21	Patient ID: 9	2024-11-27	09:00 AM ...	Pending
22	Patient ID: 9	2025-11-13	09:00 AM ...	Pending

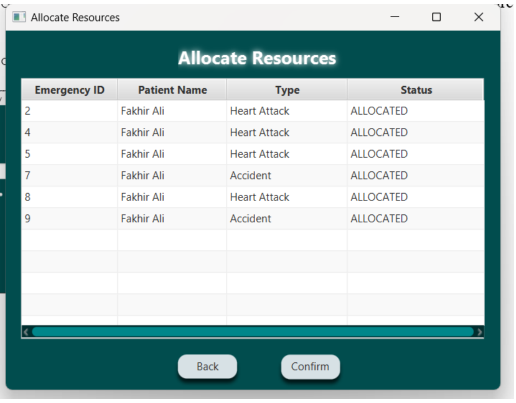


ID	PName	Dname	HName	Experience	Recommend	Feedback
15	Afrah	Shabbir	CMH	0	everything good	perfect
16	Fakhir	Shabbir	CMH	0	very cool	awesome
17	Inam	Shabbir	CMH	Excellent	hihi	yoyoy
18	Fakhir	Shabbir	CMH	Excellent	ljnl	lkn
19	Fakhir	Shabbir	CMH	Good	yoyo	hihih
20	Fakhir	Shabbir	CMH	Average	helell	

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.
- **Menu Bar:** Shows list of Notification

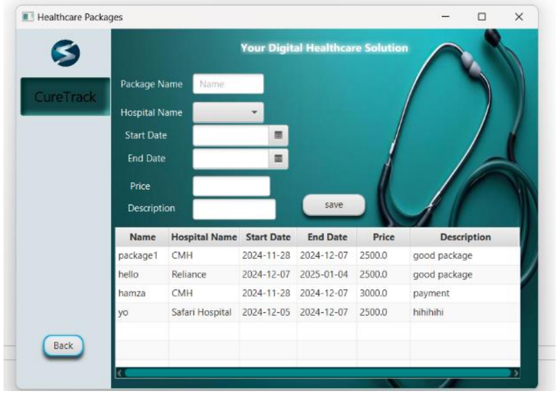




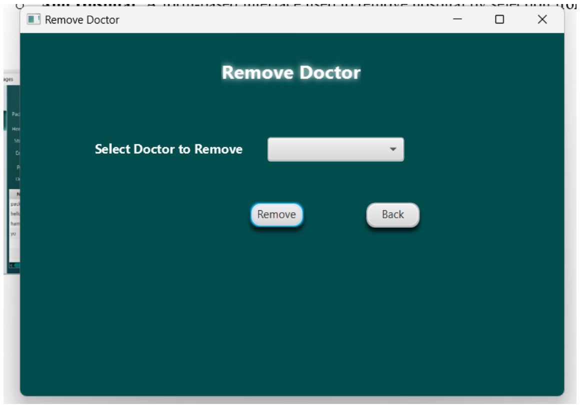
Emergency ID	Patient Name	Type	Status
2	Fakhir Ali	Heart Attack	ALLOCATED
4	Fakhir Ali	Heart Attack	ALLOCATED
5	Fakhir Ali	Heart Attack	ALLOCATED
7	Fakhir Ali	Accident	ALLOCATED
8	Fakhir Ali	Heart Attack	ALLOCATED
9	Fakhir Ali	Accident	ALLOCATED

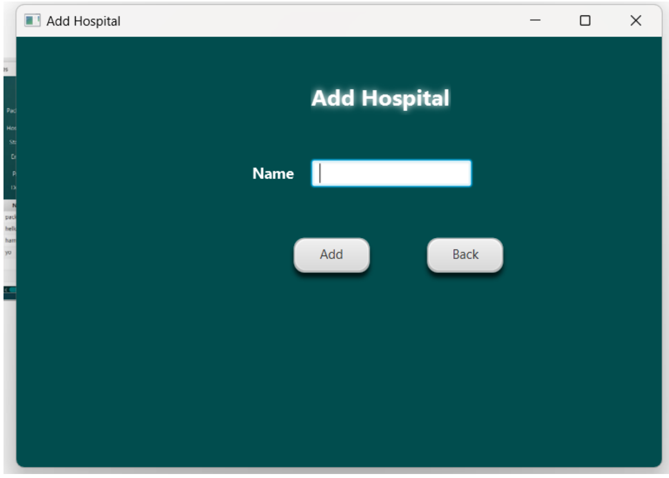
4. Interface for Admin

- **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.



Name	Hospital Name	Start Date	End Date	Price	Description
package1	CMH	2024-11-28	2024-12-07	2500.0	good package
hello	Reliance	2024-12-07	2025-01-04	2500.0	good package
hamza	CMH	2024-11-28	2024-12-07	3000.0	payment
yo	Safari Hospital	2024-12-05	2024-12-07	2500.0	hihihihi



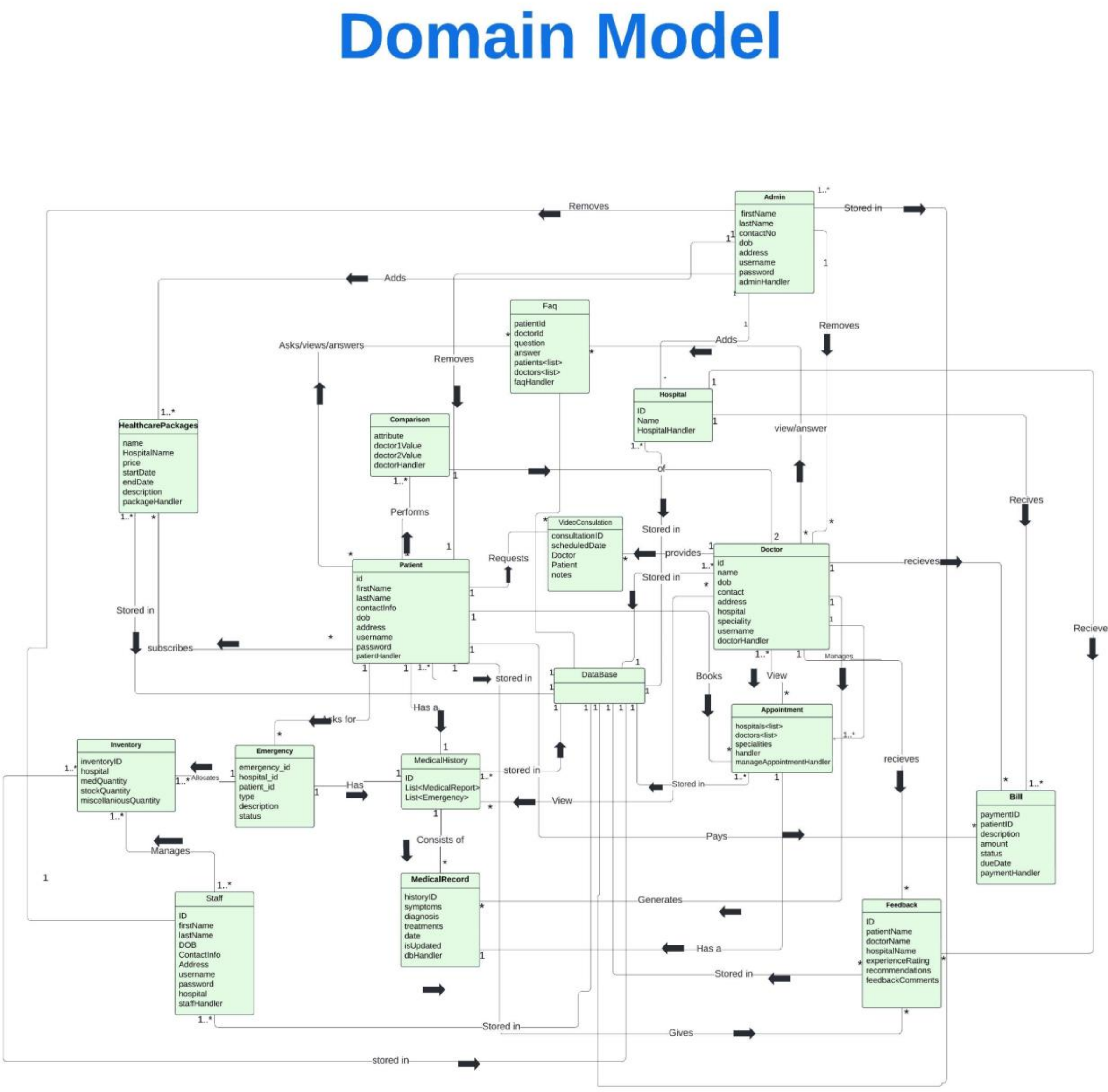


5. Design Guidelines

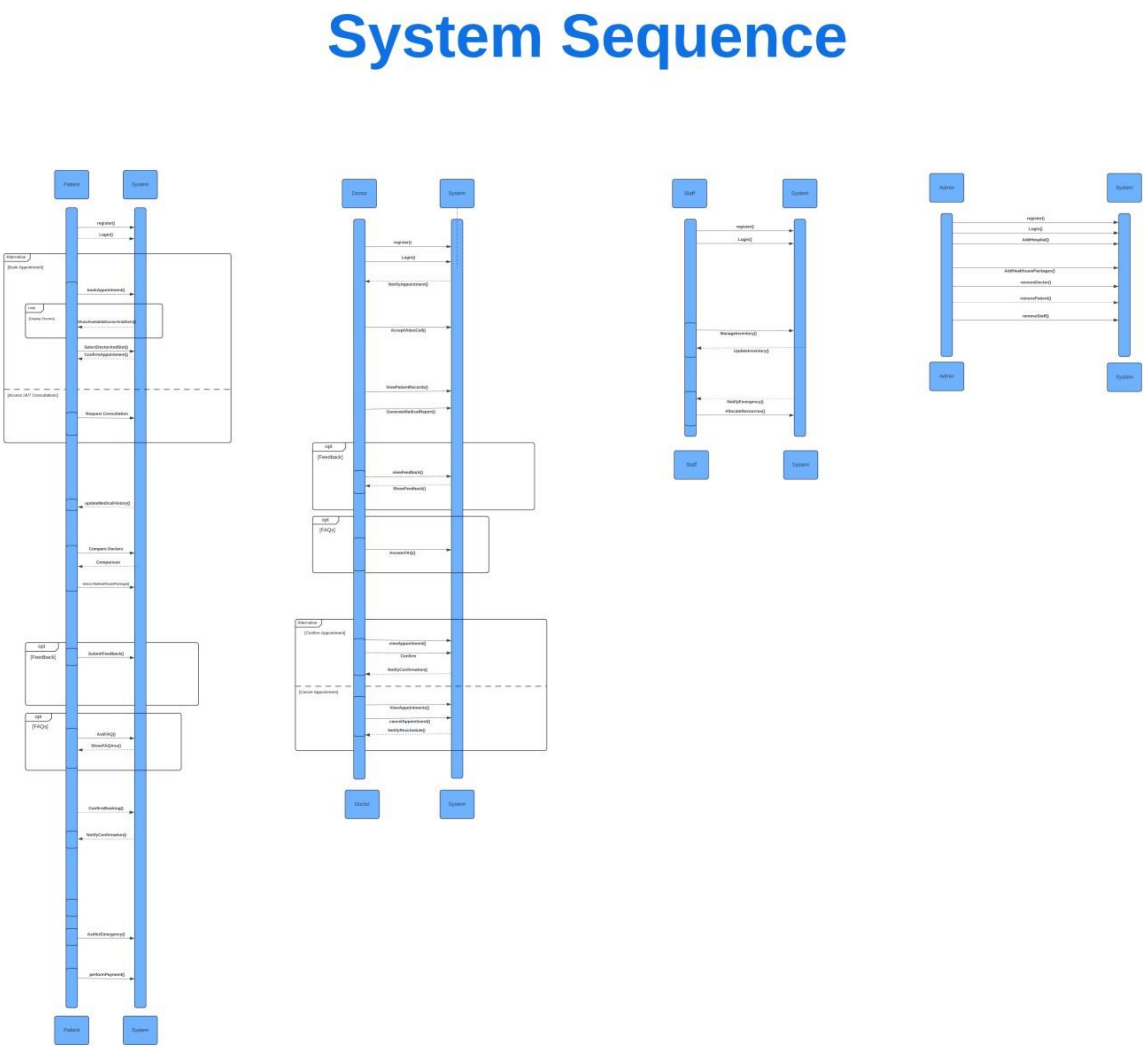
- **GUI Standards**: Adherence to Material Design principles for a consistent and modern look. CureTrack's GUI follows 12 principals of Material Design principals:
    1. Contrast
    2. Balance
    3. Emphasis
    4. Proportion
    5. Hierarchy
    6. Repetition
    7. Rhythm
    8. Pattern
    9. White 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”

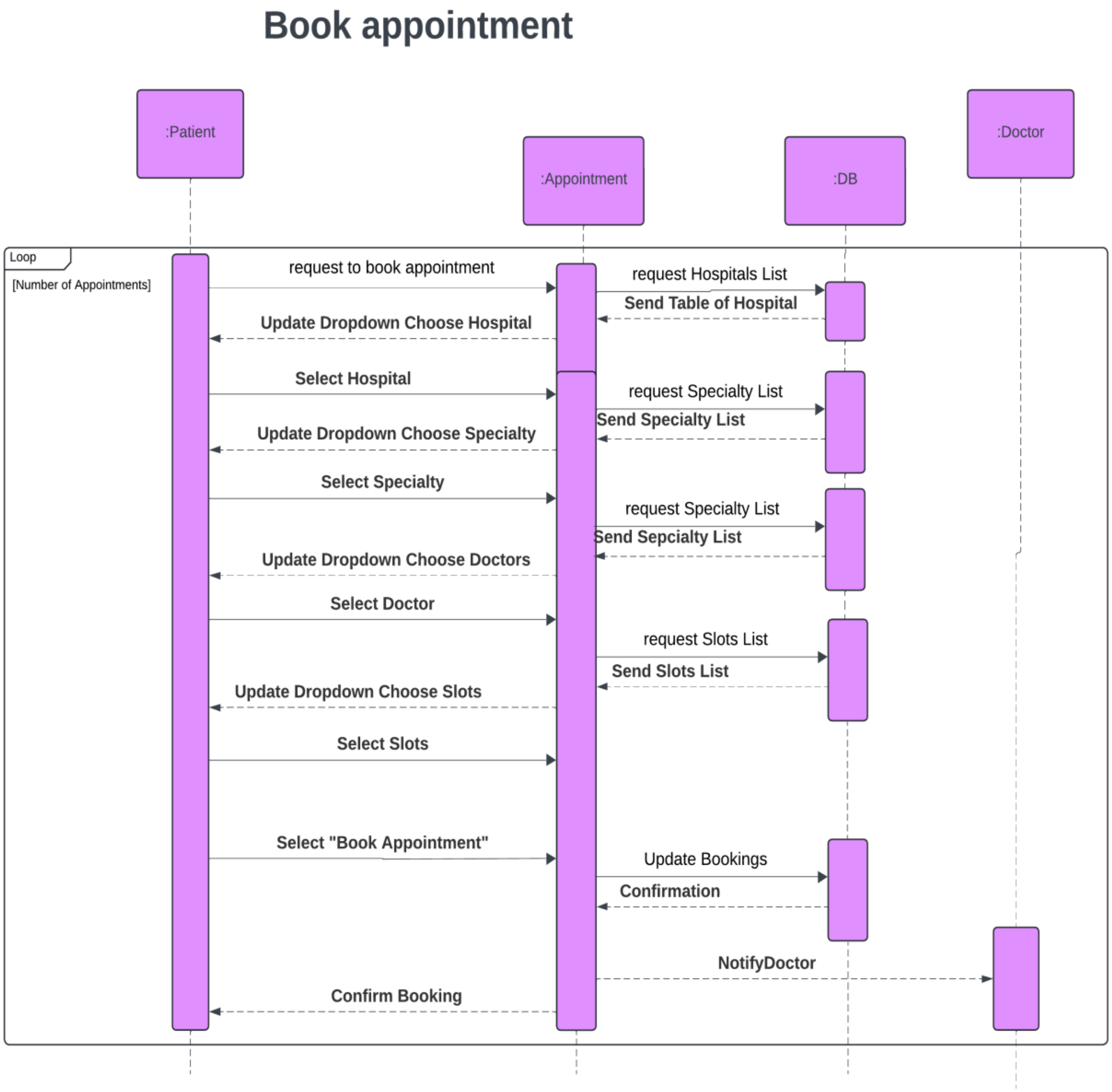
4. Domain model



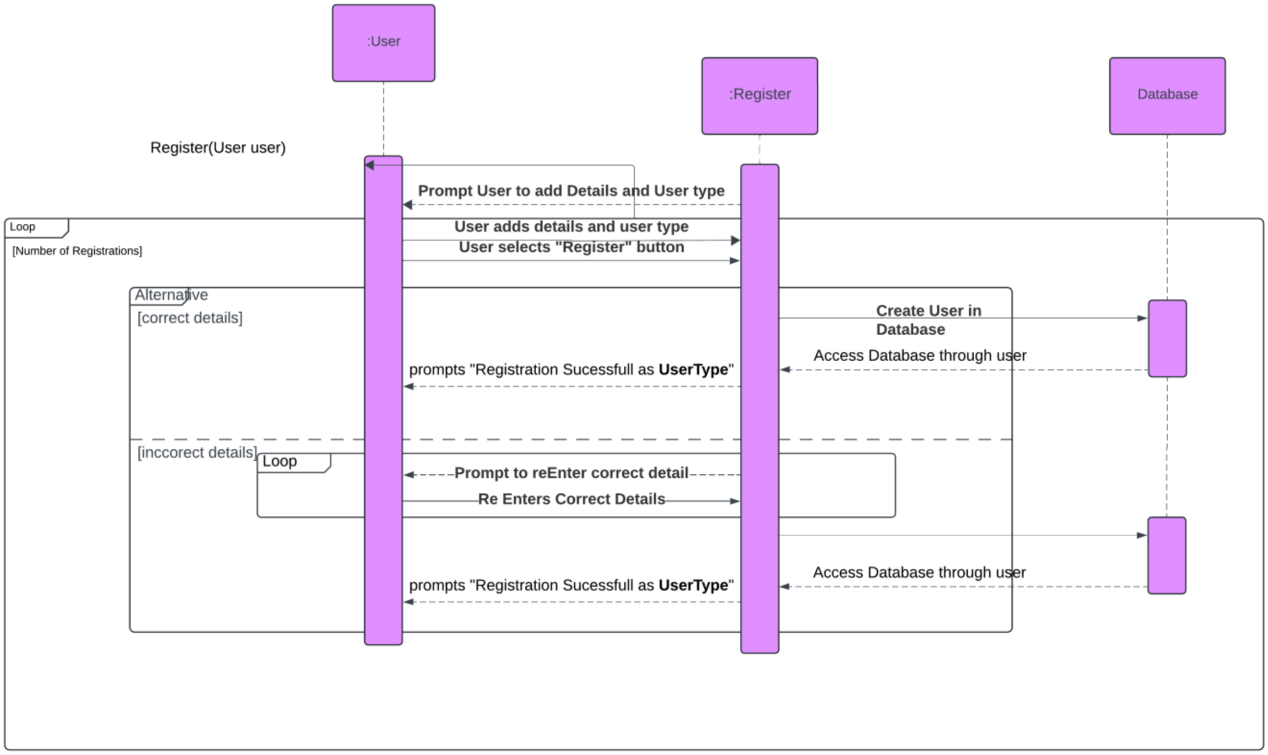
5. System Sequence



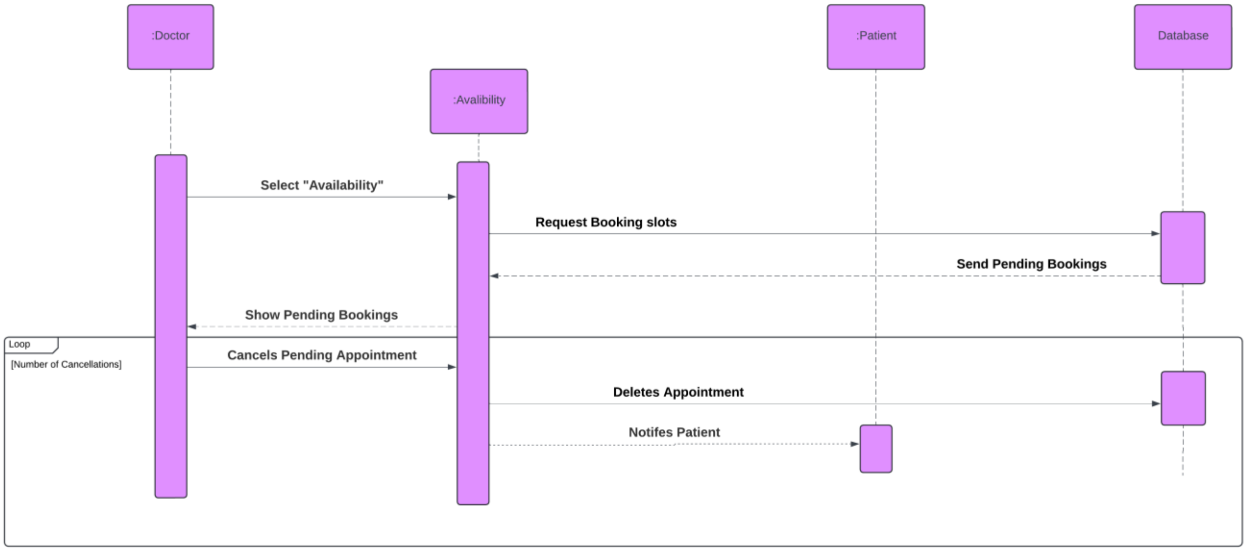
6. Action Sequence



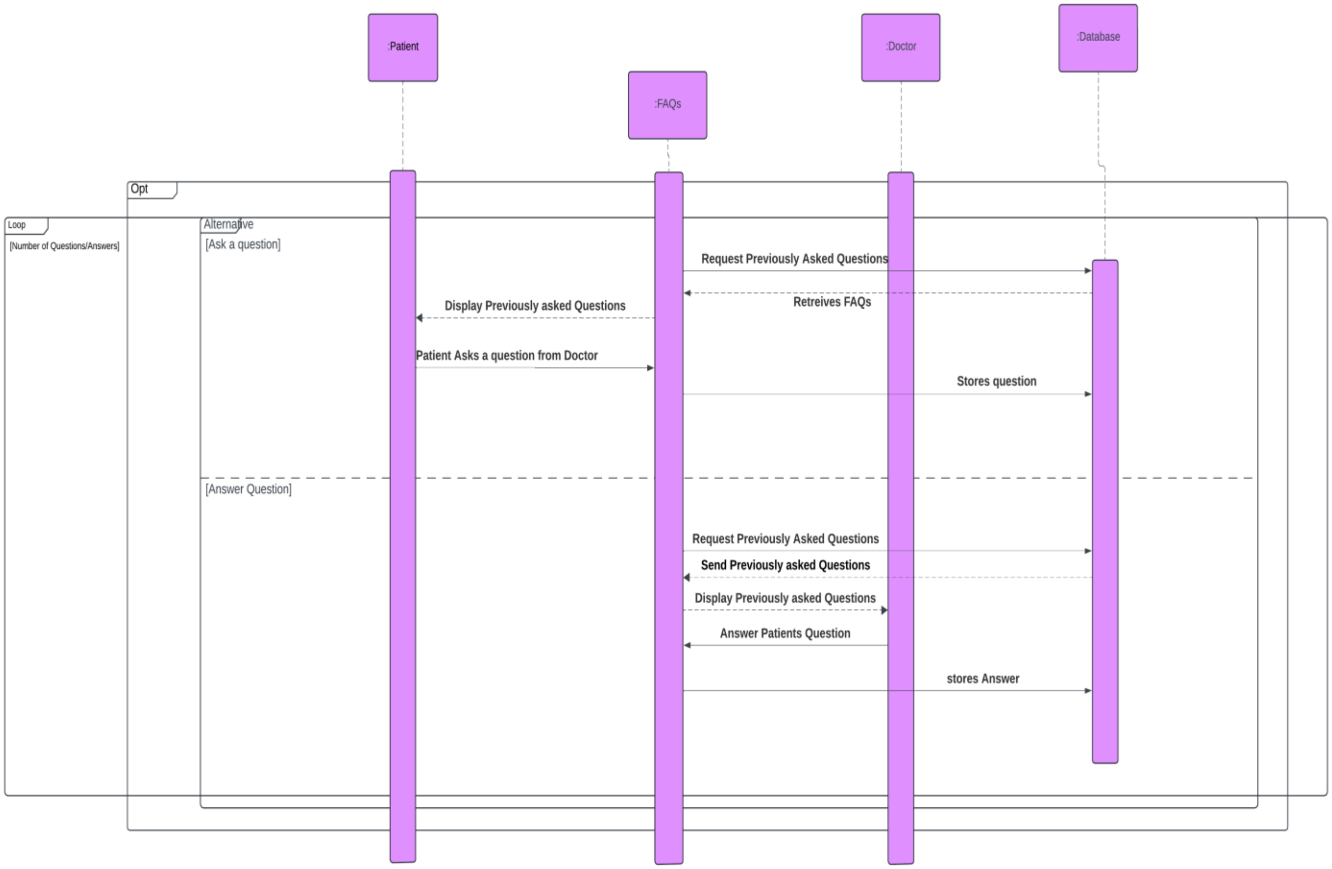
Register Doctor/Patients/Staff


:User
:Register
Database
Register(User user)
Prompt User to add Details and User type
Loop
[Number of Registrations]
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**"

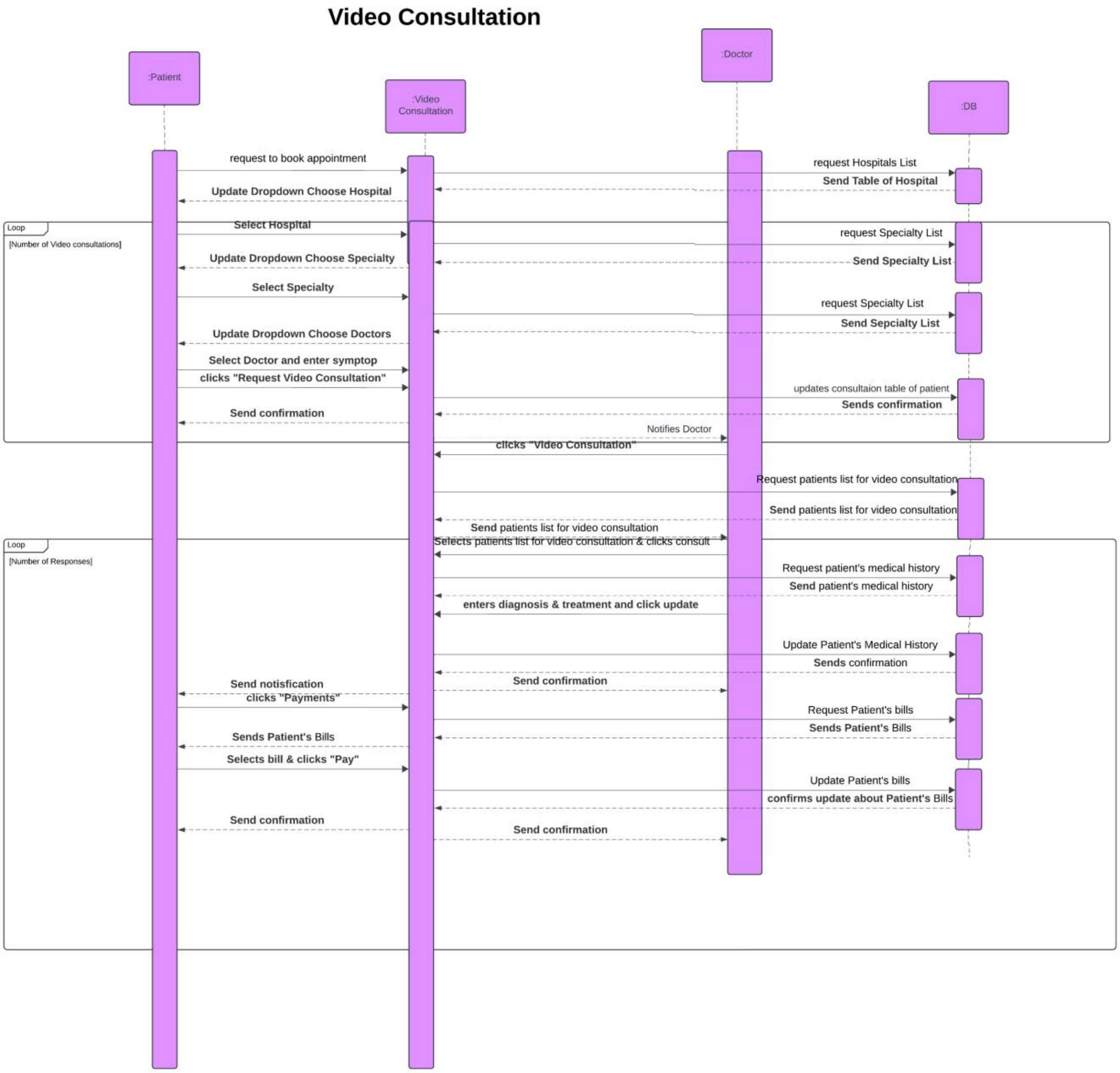
Availability

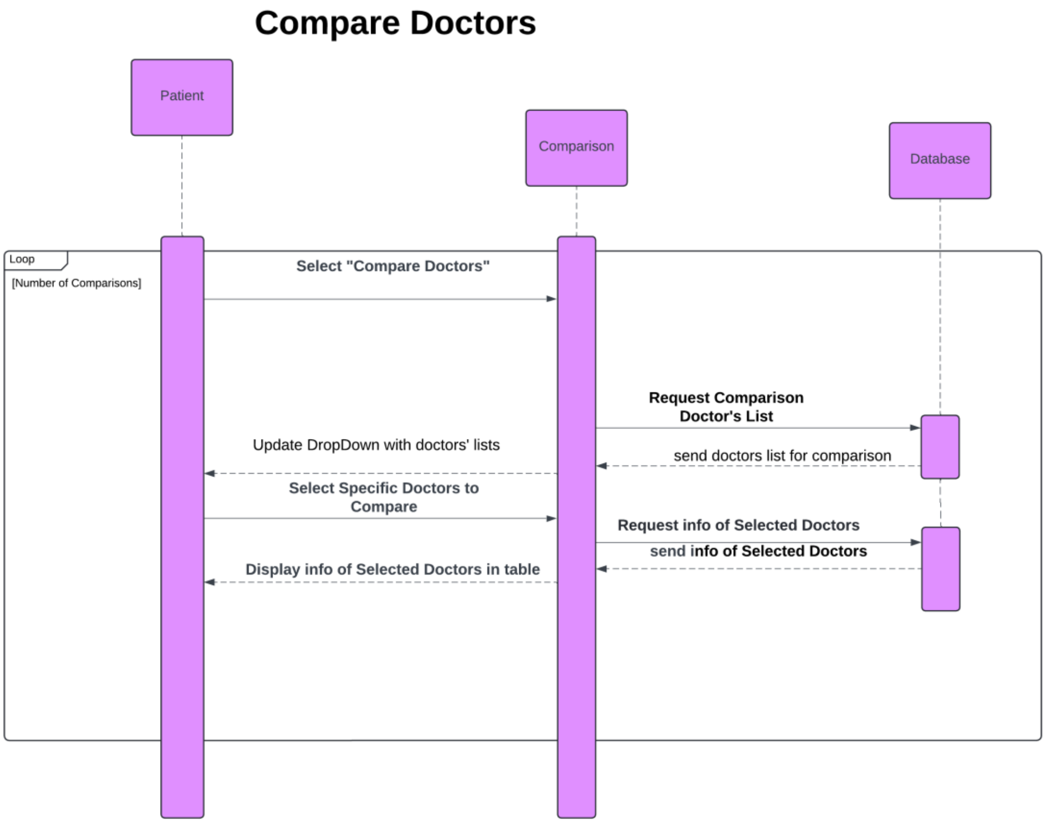


FAQs

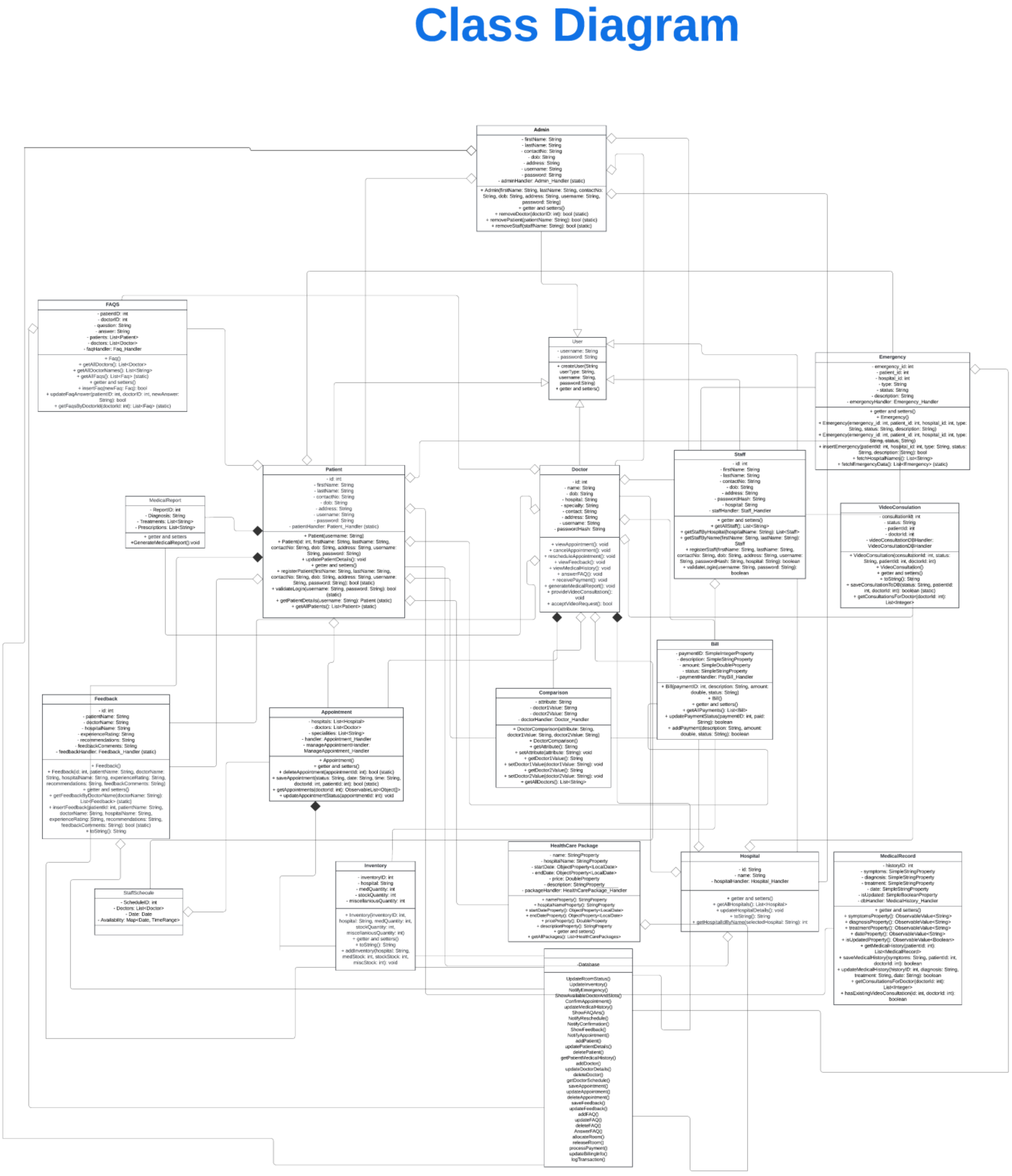

:Patient
:FAQs
:Doctor
:Database
Opt
Loop
[Number of Questions/Answers]
Alternative
[Ask a question]
Request Previously Asked Questions
Retreives FAQs
Display Previously asked Questions
Patient Asks a question from Doctor
Stores question
[Answer Question]
Request Previously Asked Questions
Send Previously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

Video Consultation

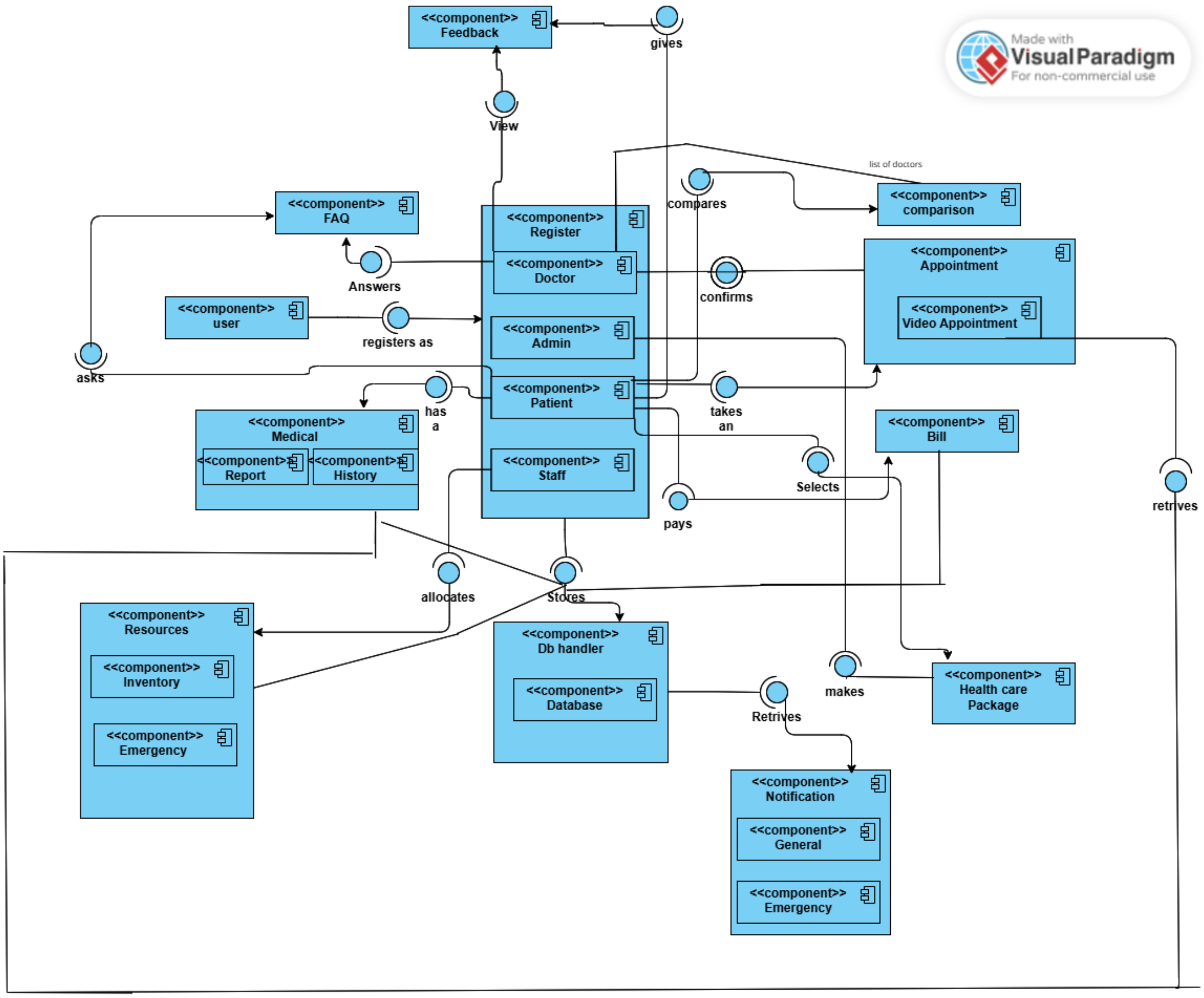

Video Consultation
:Patient
:Video Consultation
:Doctor
:DB
request to book appointment
request Hospitals List
Send Table of Hospital
Update Dropdown Choose Hospital
Loop
[Number of Video consultations]
Select Hospital
request Specialty List
Update Dropdown Choose Specialty
Send Specialty List
Select 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
Loop
[Number of Responses]
Request patient's medical history
Send patient's medical history
enters diagnosis & treatment and click update
Update Patient's Medical History
Sends confirmation
Send notisfication
Send confirmation
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


Compare Doctors
Patient
Comparison
Database
Loop
[Number of Comparisons]
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

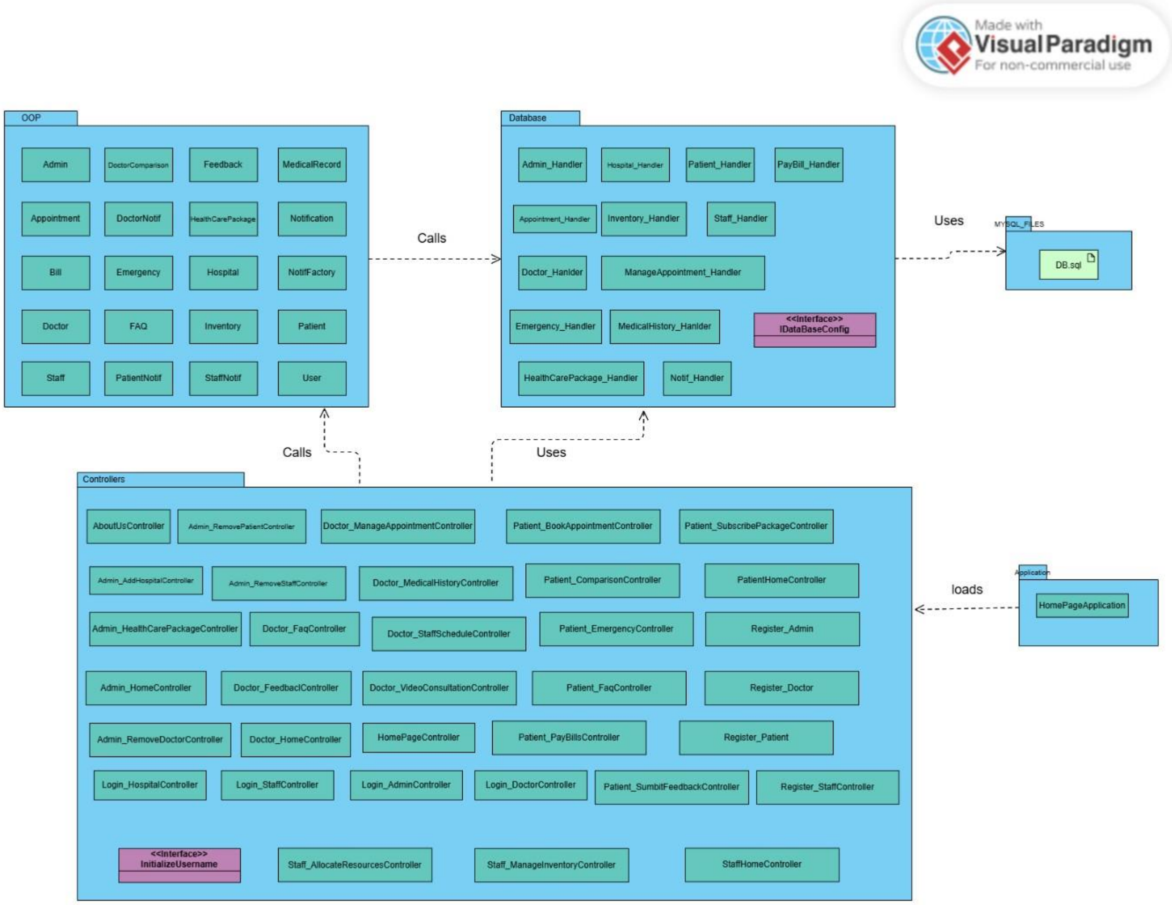
7. Class Diagram


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)
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()

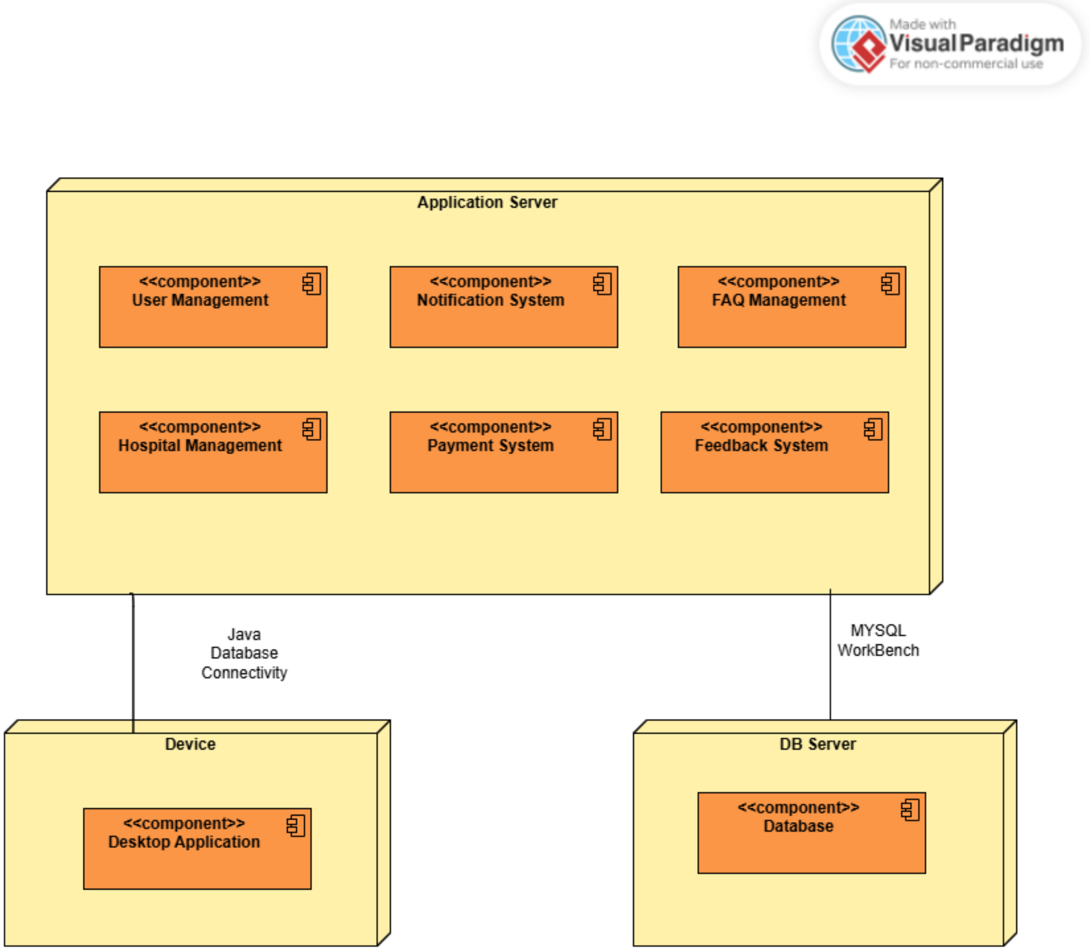
8. Component Diagram



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