PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Object Oriented Analysis and Design using Java (UE20CS352)

Mini Project

PROJECT TITLE:

HOSPITAL MANAGAMENT SYSTEM

Team No. 1

TEAM MEMBERS:

SAMPREET A PATIL	PES2UG20CS306
SAMYAK P SHETTY	PES2UG20CS307
SHAIK MOHAMMAD JEELAN	PES2UG20CS321

Submitted to:

Prof. Saranya Devi

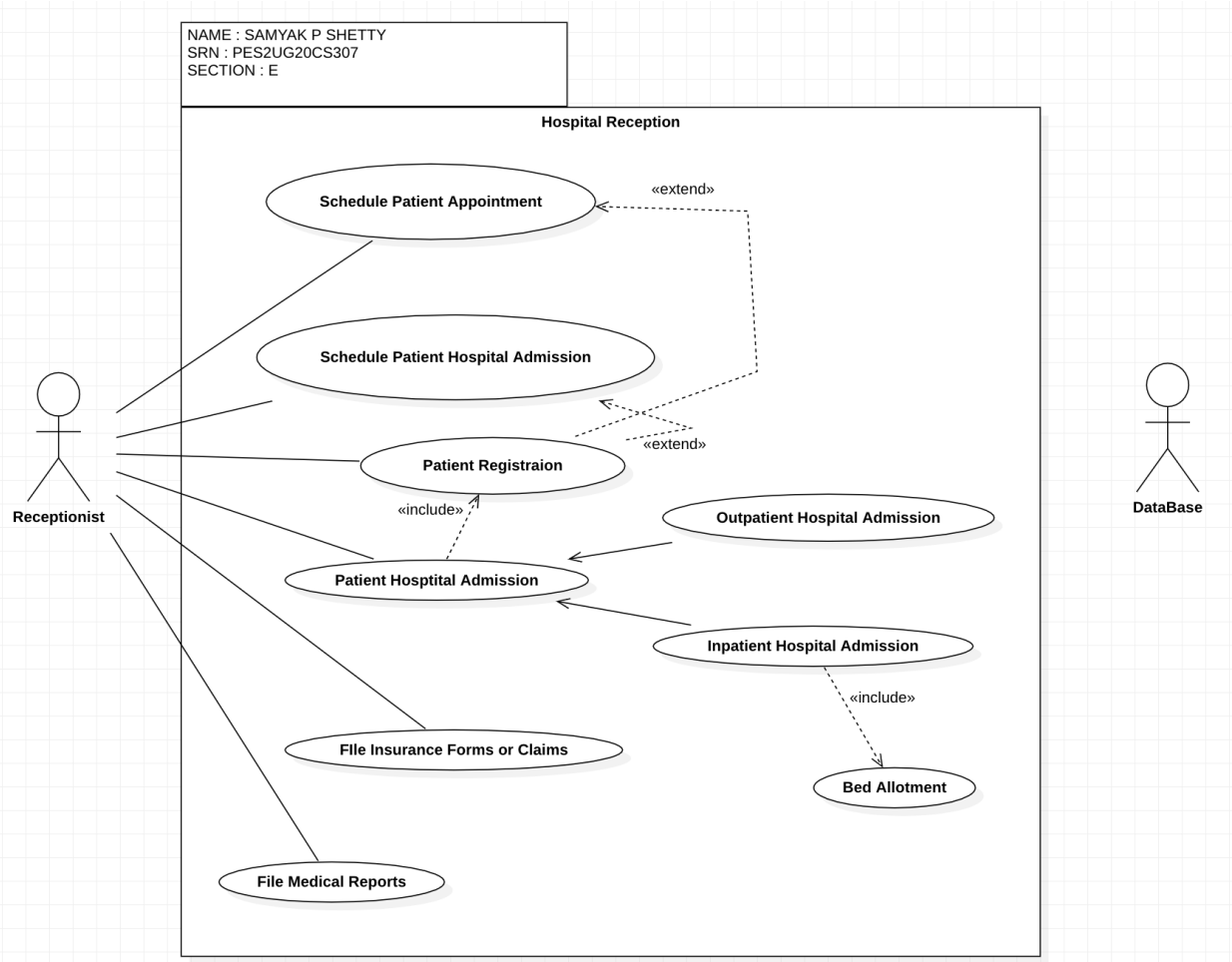
Problem statement:

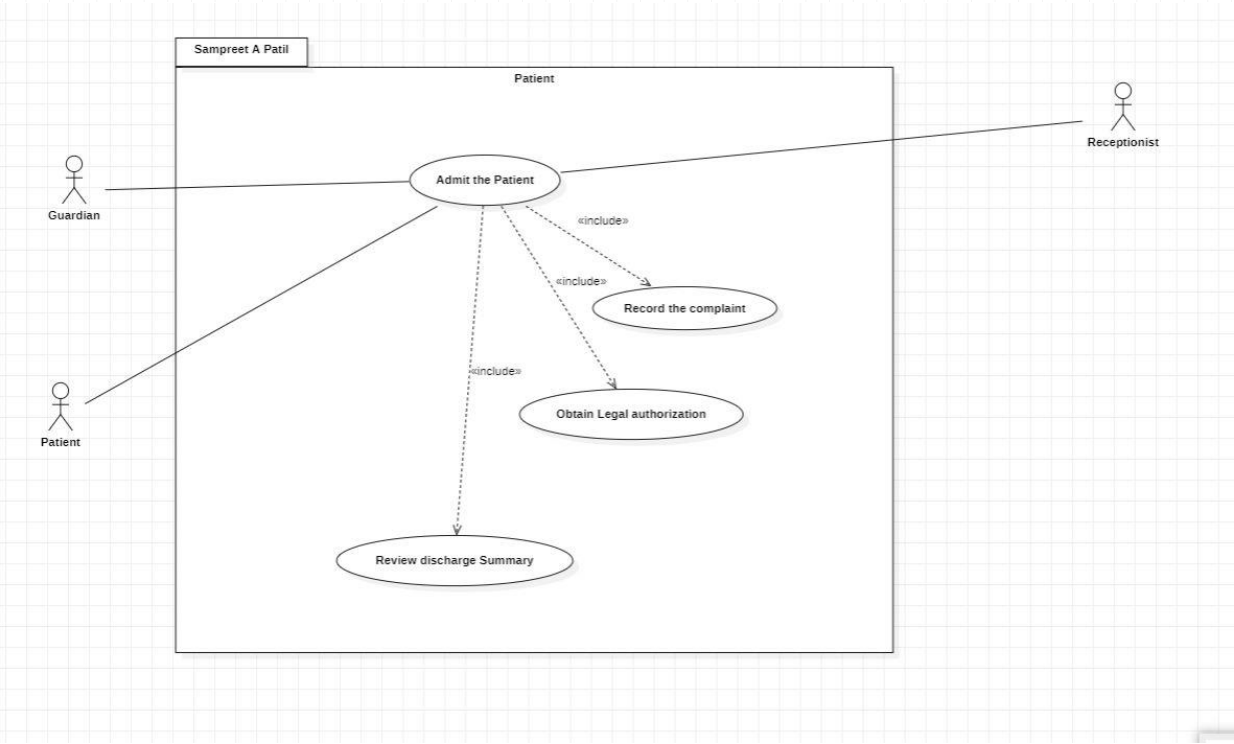
We have taken up the topic of Hospital Management system database project. This project would have patient records consisting of the patient's basic details. Each patient would be assigned to one doctor. However, each doctor could treat multiple patients. The system would also include some basic details pertaining to the individual doctor. Also, each patient would be assigned to a wardroom/ operation theater/ ICU. Additionally, the patient care staff including the nurses and ward boys would be assigned some rooms.
We are going to consider two user types – Receptionist and Doctor in this system. The system will be accessible only using valid usernames and passwords for each user type.

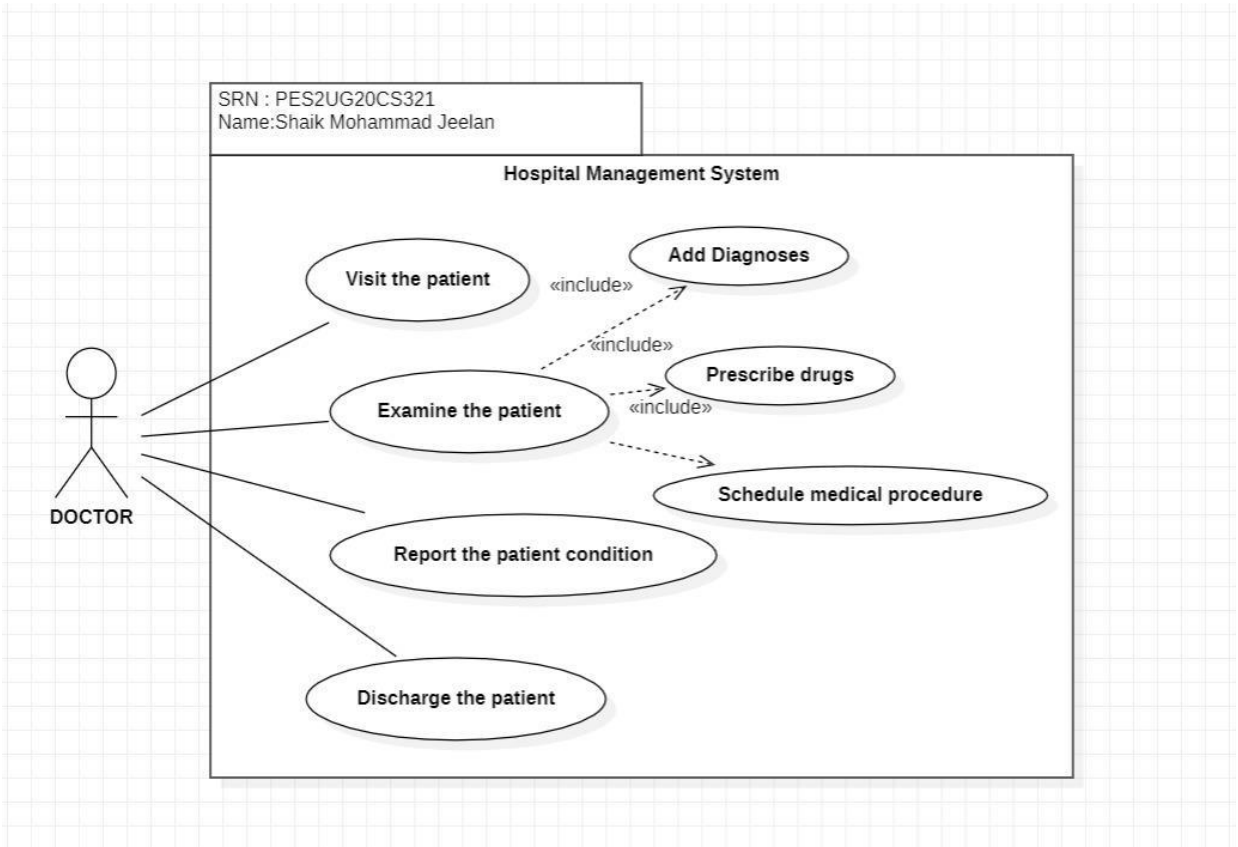
1. Receptionist:
   - The receptionist here has full access to the system.
   - They can register a new doctor or another receptionist to the system
   - They can add a new record for patient, doctor and other staff in the system
   - They can retrieve existing record for patient, doctor and other staff
   - They can edit patient and staff details and update it in the database
   - They can create an appointment for a patient and assign him a doctor
   - They can create/update and delete a record for a room/operation theatre/ ICU
   - They can obtain the room/operation theatre/ ICU that are unassigned
   - They can allocate/deallocate a room/operation theatre/ ICU to a patient and update its status in the database
   - They can assign or remove assignment of the staff (nurse and ward boys) to the rooms/operation theater/ICU

2. Doctor:
   - The doctor will login to the system with his username and password.
   - The doctor will be able to add and update his own details in the system. They will not be able to add or update details of other doctors in the system
   - They will be able to view details about other doctors in the hospital database
   - They will be able to retrieve the patient records as well as the patients assigned to them
   - They will also be able to view the room/operation theatre/ ICU the patient is assigned to
   - They can also add/update a diagnosis and prescription for the patient

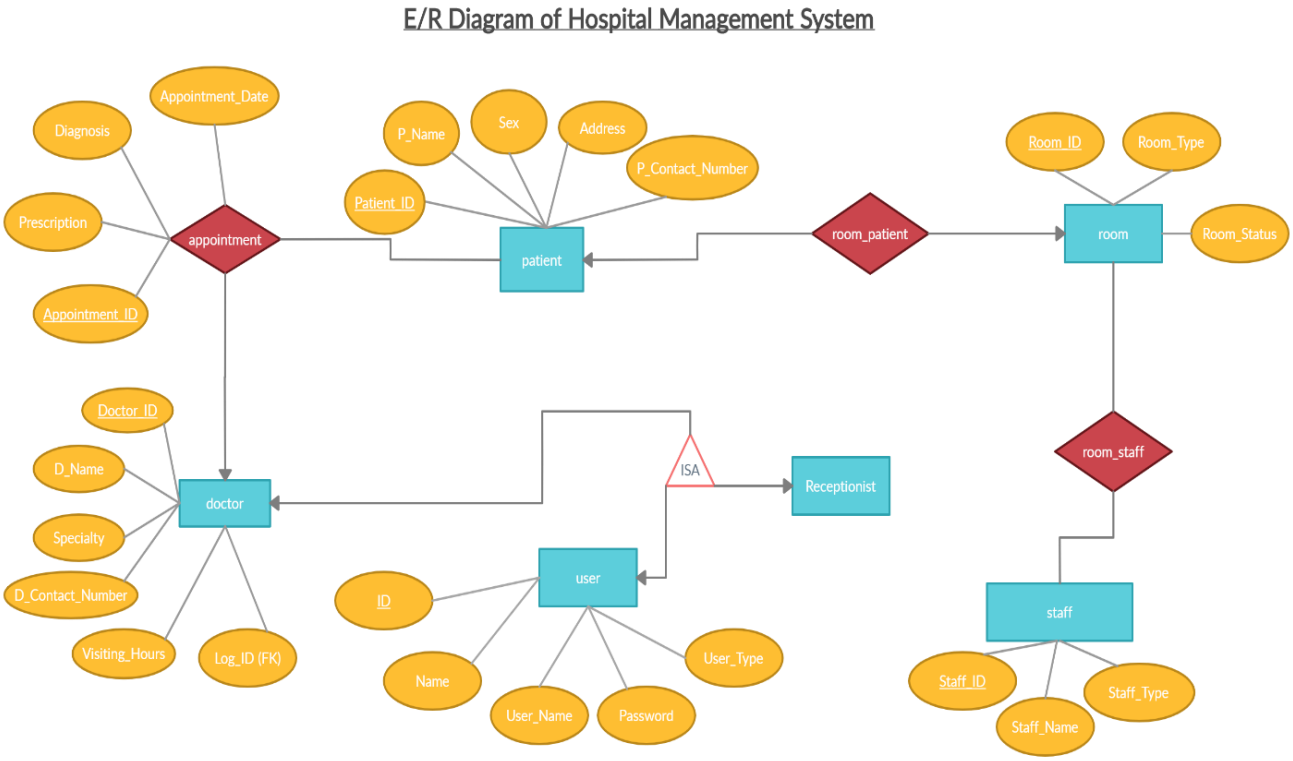
Models:


NAME : SAMYAK P SHETTY
SRN : PES2UG20CS307
SECTION : E
Hospital Reception
Schedule Patient Appointment
«extend»
Schedule Patient Hospital Admission
Patient Registraion
«extend»
«include»
Outpatient Hospital Admission
Patient Hosptital Admission
Inpatient Hospital Admission
«include»
FIle Insurance Forms or Claims
Bed Allotment
File Medical Reports
Receptionist
DataBase


Sampreet A Patil
Patient
Admit the Patient
«include»
Record the complaint
«include»
Obtain Legal authorization
«include»
Review discharge Summary
Guardian
Patient
Receptionist


SRN : PES2UG20CS321
Name:Shaik Mohammad Jeelan
Hospital Management System
Visit the patient
Add Diagnoses
«include»
«include»
Examine the patient
Prescribe drugs
«include»
Schedule medical procedure
Report the patient condition
Discharge the patient
DOCTOR

ER Diagram:


E/R Diagram of Hospital Management System
Appointment_Date
Diagnosis
Prescription
appointment
Appointment_ID
P_Name
Sex
Address
P_Contact_Number
Patient_ID
patient
room_patient
Room_ID
Room_Type
room
Room_Status
Doctor_ID
D_Name
Specialty
D_Contact_Number
Visiting_Hours
Log_ID (FK)
doctor
ISA
Receptionist
room_staff
ID
user
Name
User_Name
Password
User_Type
staff
Staff_ID
Staff_Name
Staff_Type

Architecture Patterns:

Model-View-Controller (MVC): The e-commerce application project uses the MVC pattern, which separates the application into three main components: the Model (the data and business logic), the View (the user interface), and the Controller (the logic that handles user input and manages the communication between the Model and the View). This pattern promotes modularity, flexibility, and maintainability by reducing the coupling between different components of the system.

Design Principles:

1. Single Responsibility Principle (SRP): Each class should have only one responsibility or reason to change. Here, Recieptionist class is responsible for managing the users.
2. Open-Closed Principle (OCP): Classes should be open for extension but closed for modification. This means that you should be able to add new functionality to a class without having to modify its existing code. For example, you should be able to add a new appointment method without having to change the Recieptionist class.
3. Liskov Substitution Principle (LSP): Subtypes should be substitutable for their base types. This means that any subclass of a class should be able to replace its superclass without affecting the correctness of the program. For example, a Appointment class should be able to replace a other class without causing any issues.
4. Interface Segregation Principle (ISP): Clients should not be forced to depend on interfaces they do not use. This means that interfaces should be kept small and focused on a specific task. For example, a Appointment interface should only include methods related to Appointment and not methods related to adding users.
5. Dependency Inversion Principle (DIP): High-level modules should not depend on low-level modules. Both should depend on abstractions. This means that classes should depend on interfaces or abstract classes rather than concrete implementations. For example, a Appointment class should depend on an interface like CreateAppointment rather than aconcrete implementation like CreateNewPatient.

Design Patterns:

1. Factory Pattern: This pattern can be used to create different types of medical equipment or medications without exposing the details of their creation to the client. For example, the hospital management system can use the factory pattern to create different types of medical devices such as ventilators or X-ray machines.

2. Singleton Pattern: This pattern can be used to manage a single instance of classes such as Patient or Doctor. It ensures that only one instance of these classes is created and provides a global point of access to it. In a hospital management system, the singleton pattern can be used to manage the patient's medical history or the doctor's schedule.

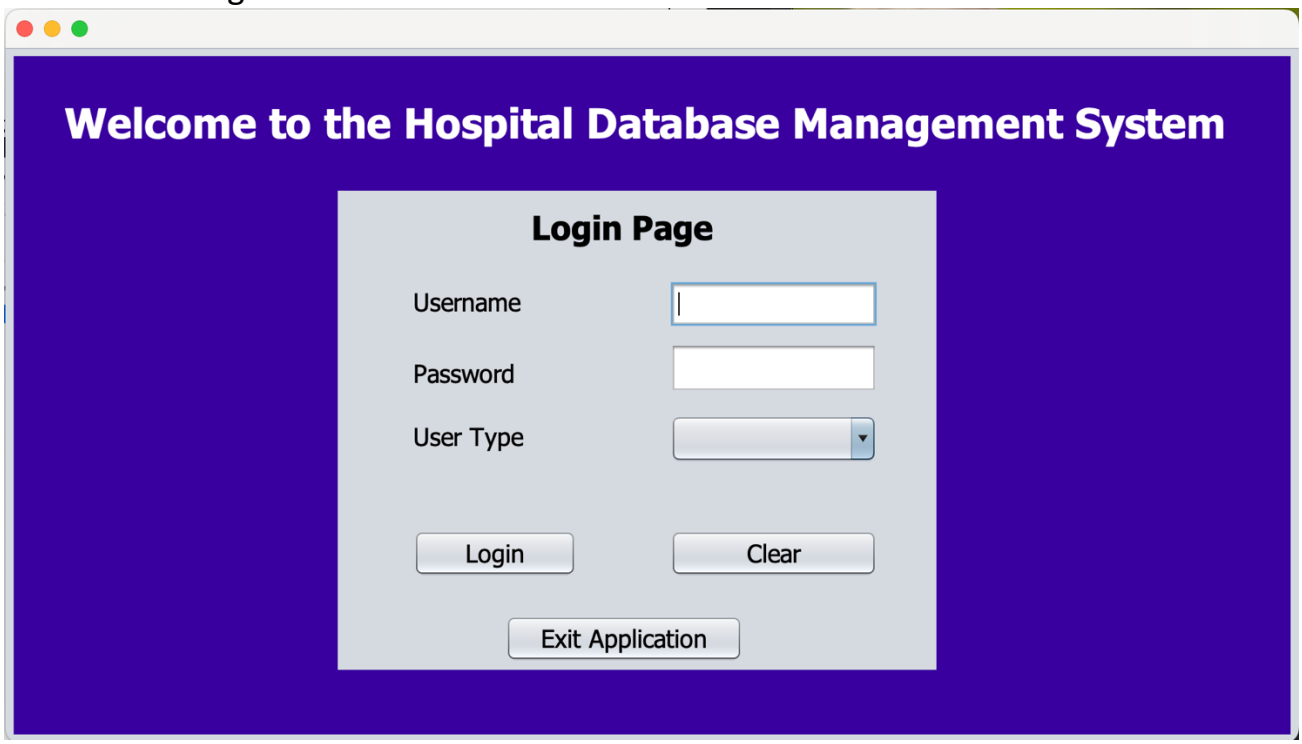
3. Strategy Pattern: This pattern can be used to manage different medical procedures, such as surgery or chemotherapy, by encapsulating them in separate classes. In a hospital management system, the strategy pattern can be used to manage different types of medical treatments, such as pharmacological or non-pharmacological treatments.

4. Observer Pattern: This pattern can be used to notify doctors or patients of medical updates or changes automatically. In a hospital management system, the observer pattern can be used to notify doctors of changes in a patient's medical status or to inform patients of appointment updates.

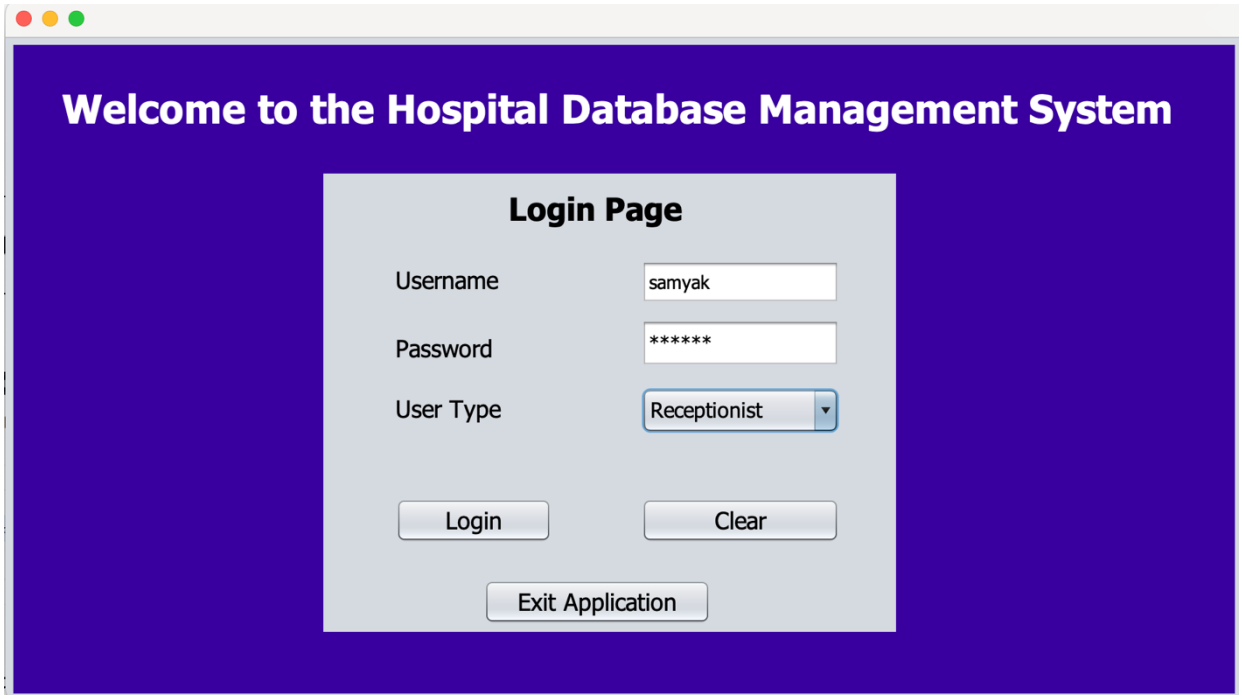
5. Decorator Pattern: This pattern can be used to add new functionality to a patient or a doctor by wrapping them in a decorator class that provides additional behavior. For example, in a hospital management system, the decorator pattern can be used to add features such as adding a new medical procedure to a patient's treatment or adding a new specialization to a doctor's profile.

Screenshots:

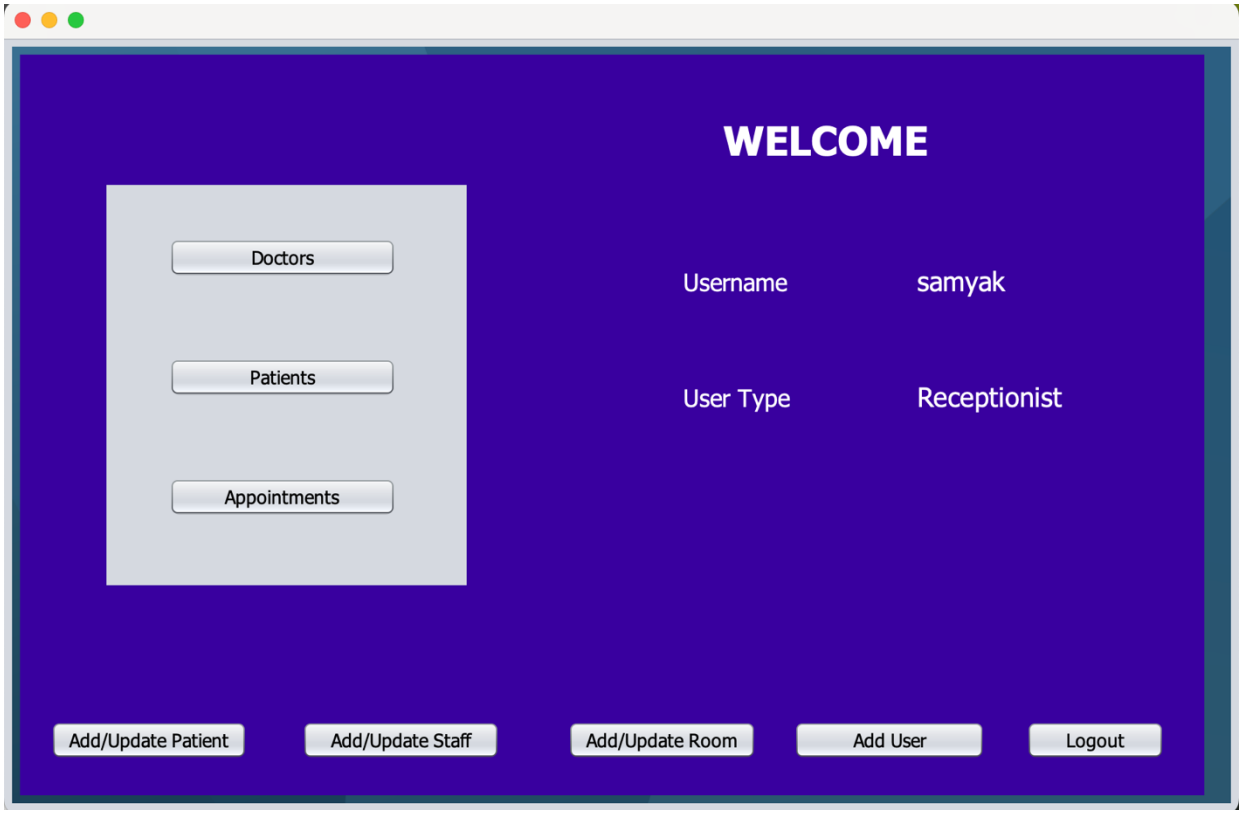
1. User Login



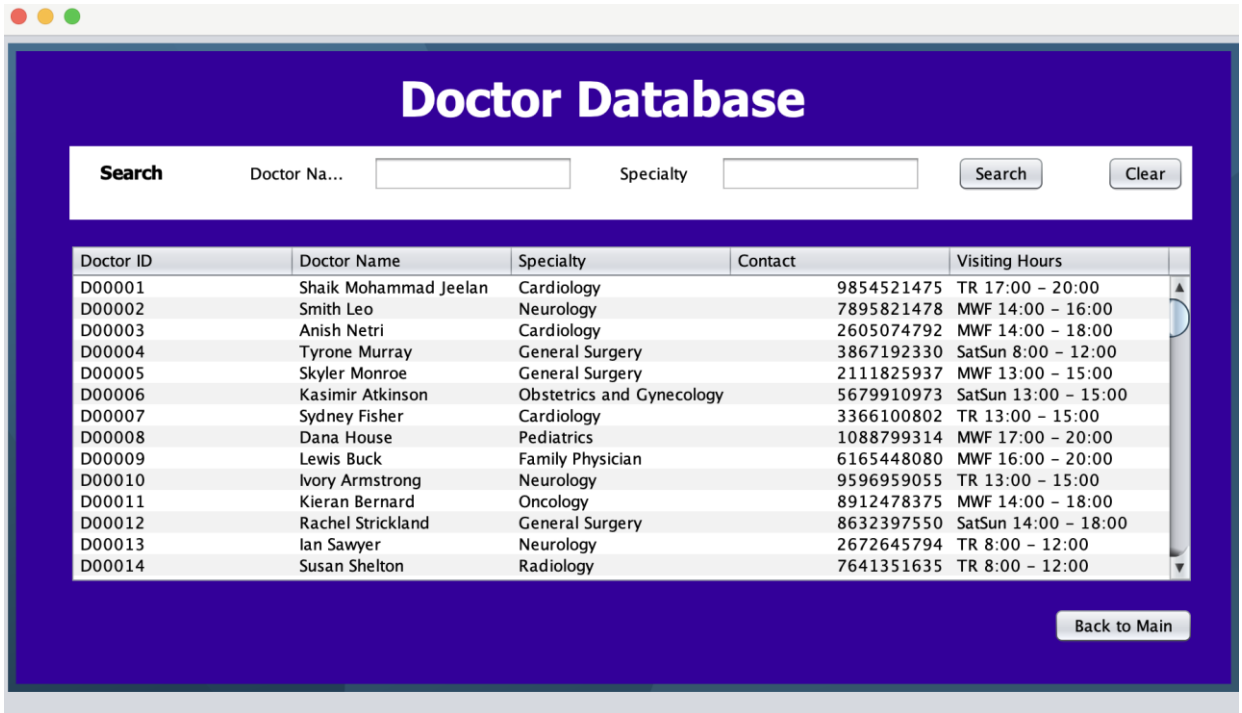
2. Receptionist Profile



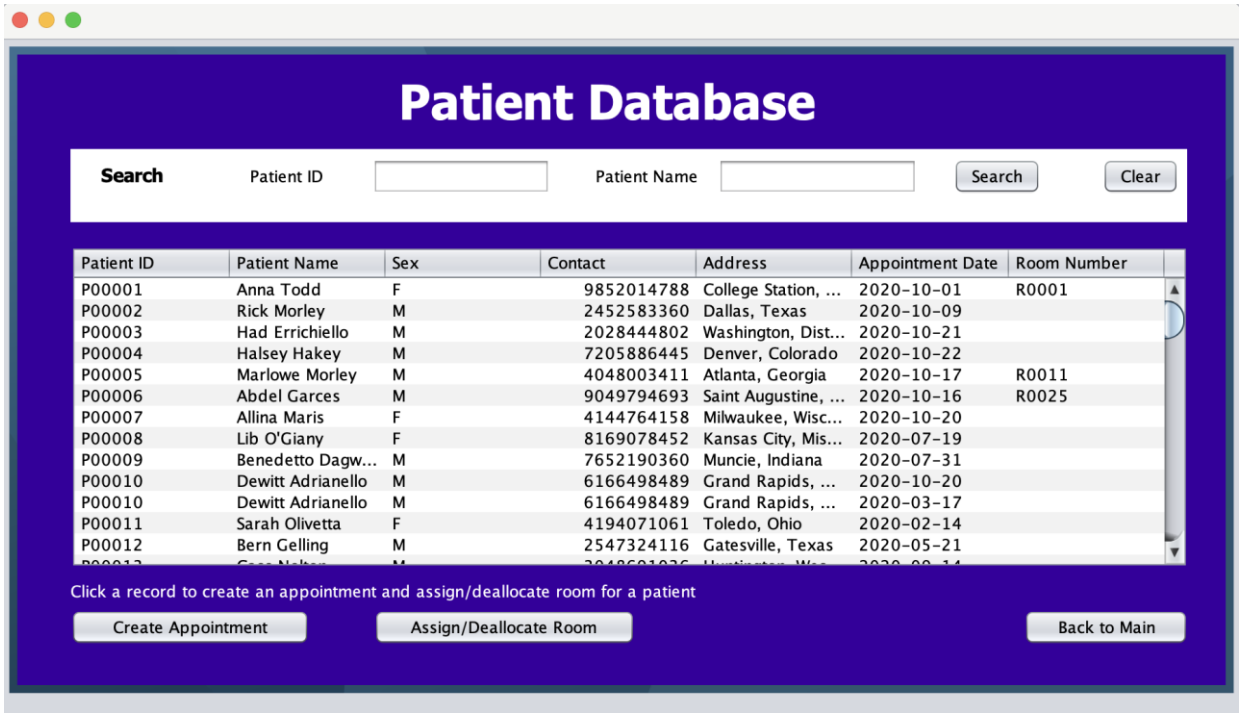
3. Receptionist Front Page



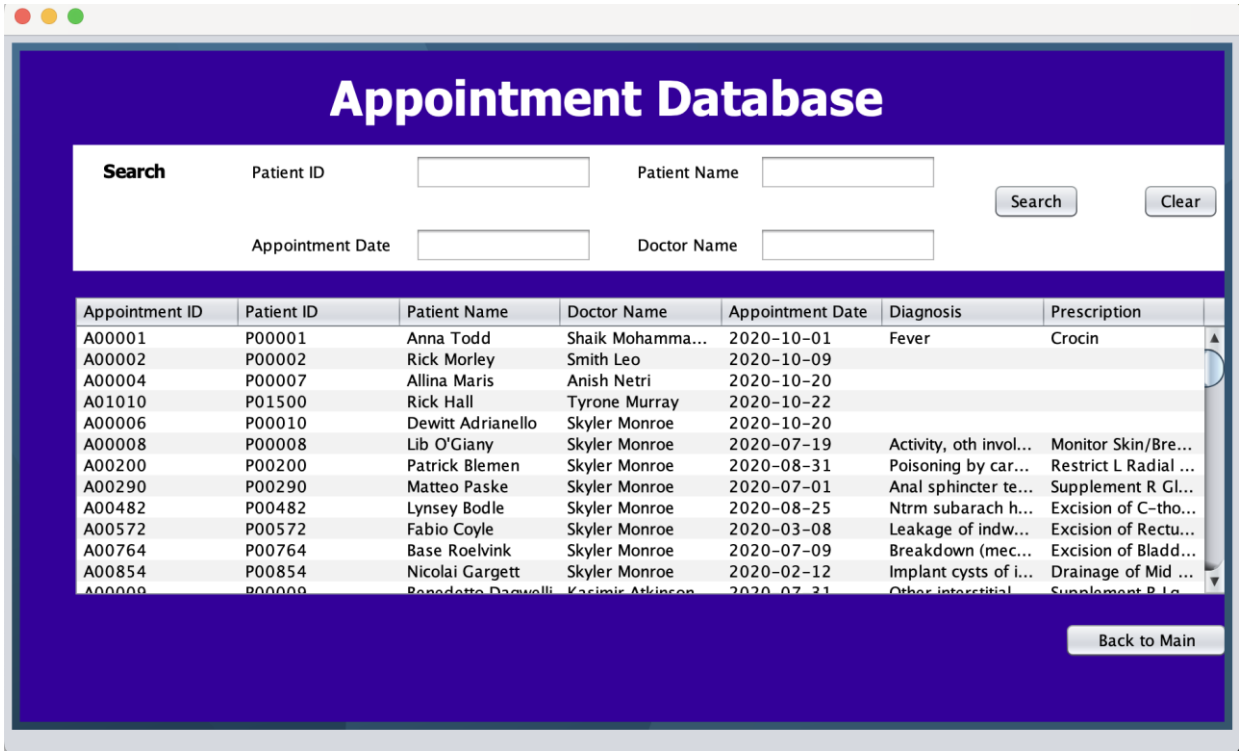
4. Doctors



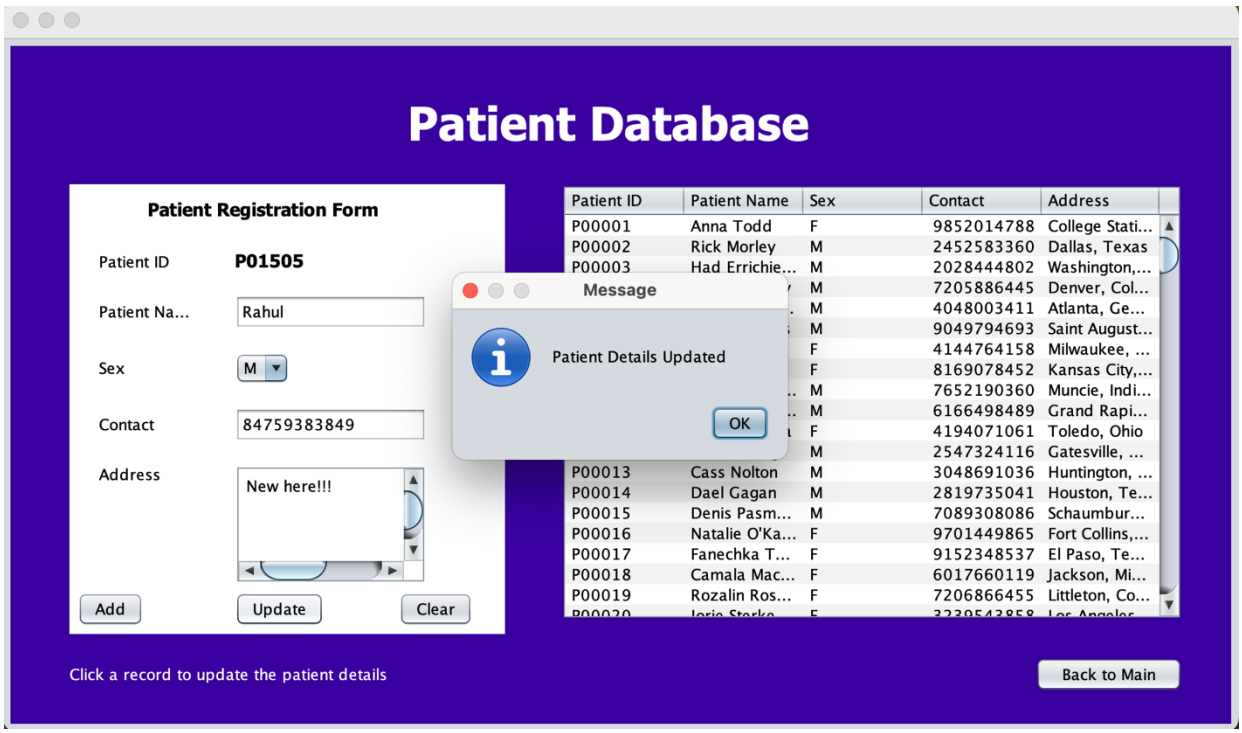
5. Patients



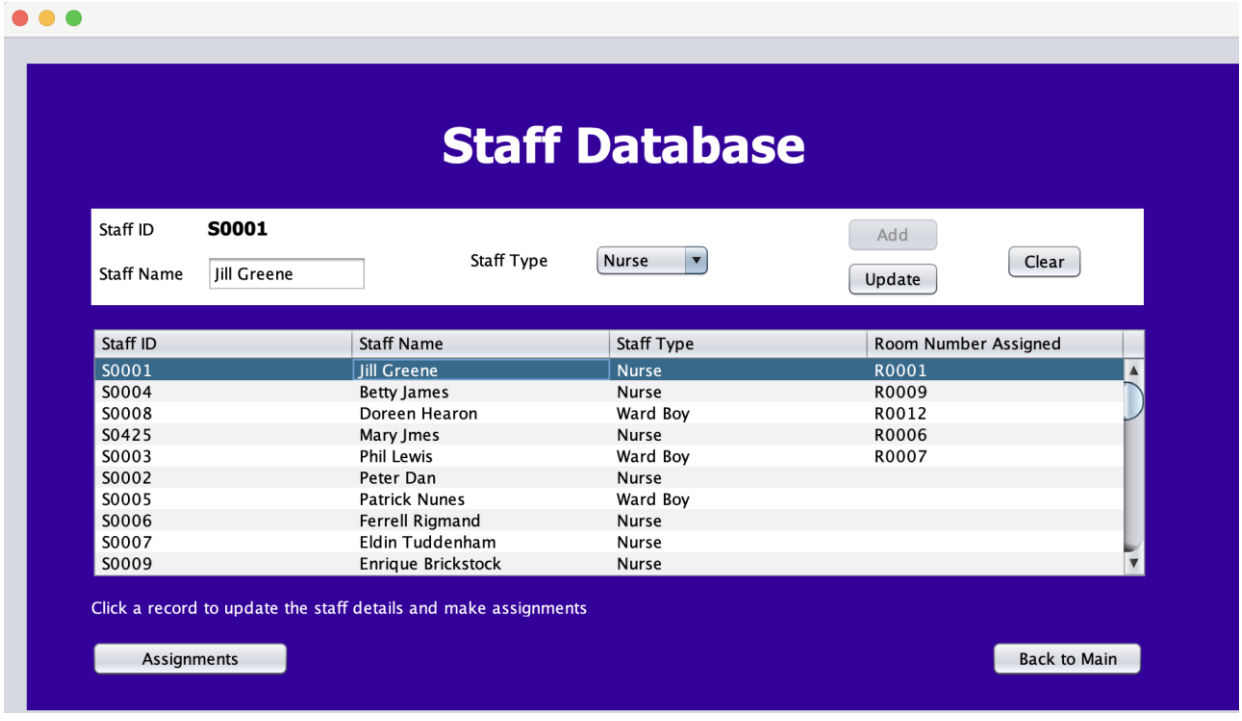
6. Appointments



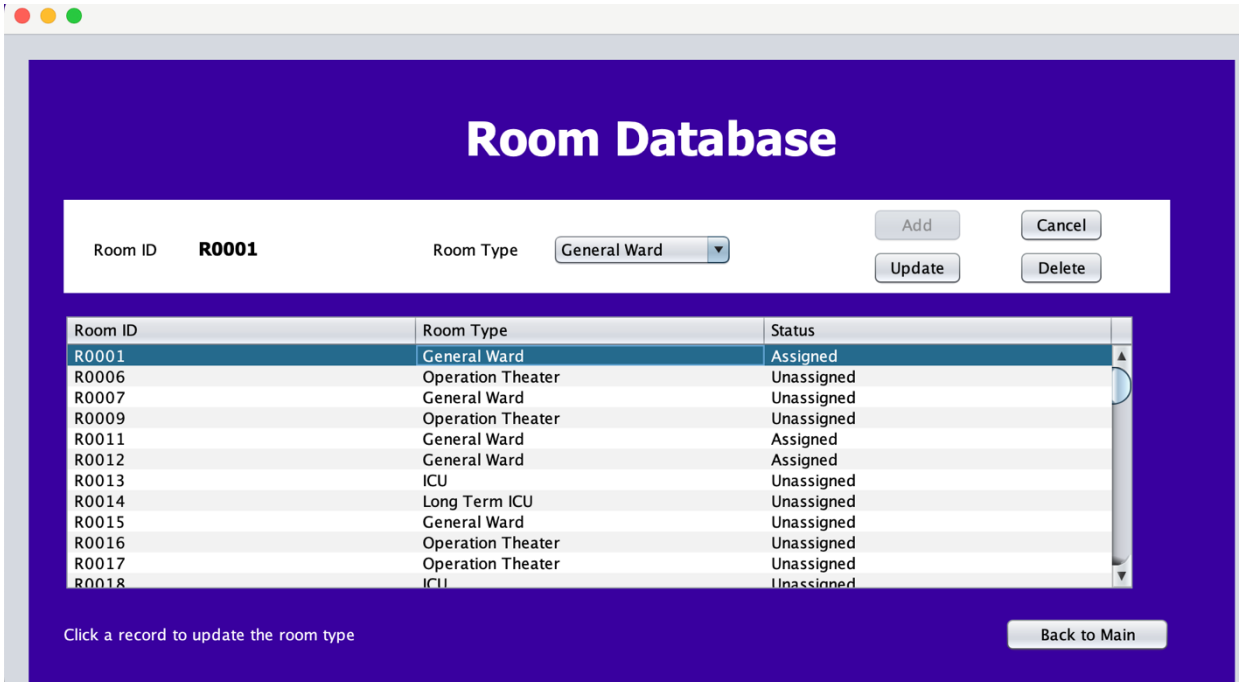
7. ADD/UPDATE Patient



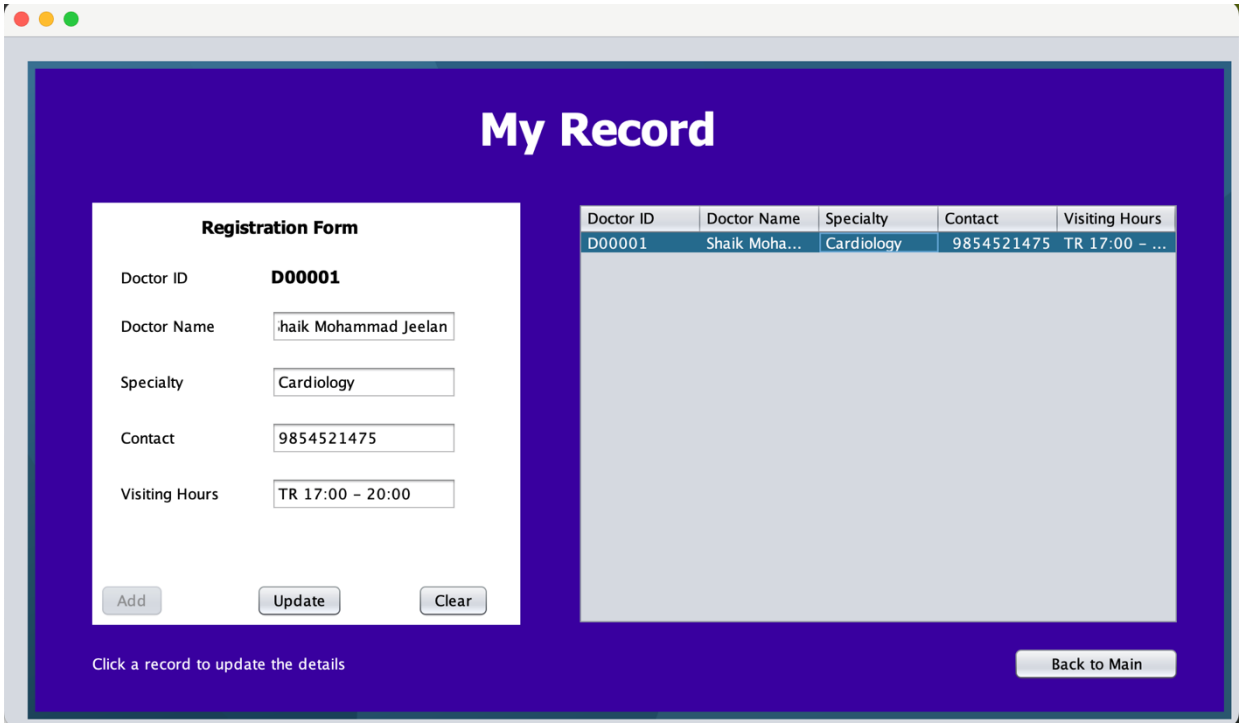
8. ADD/UPDATE Staff



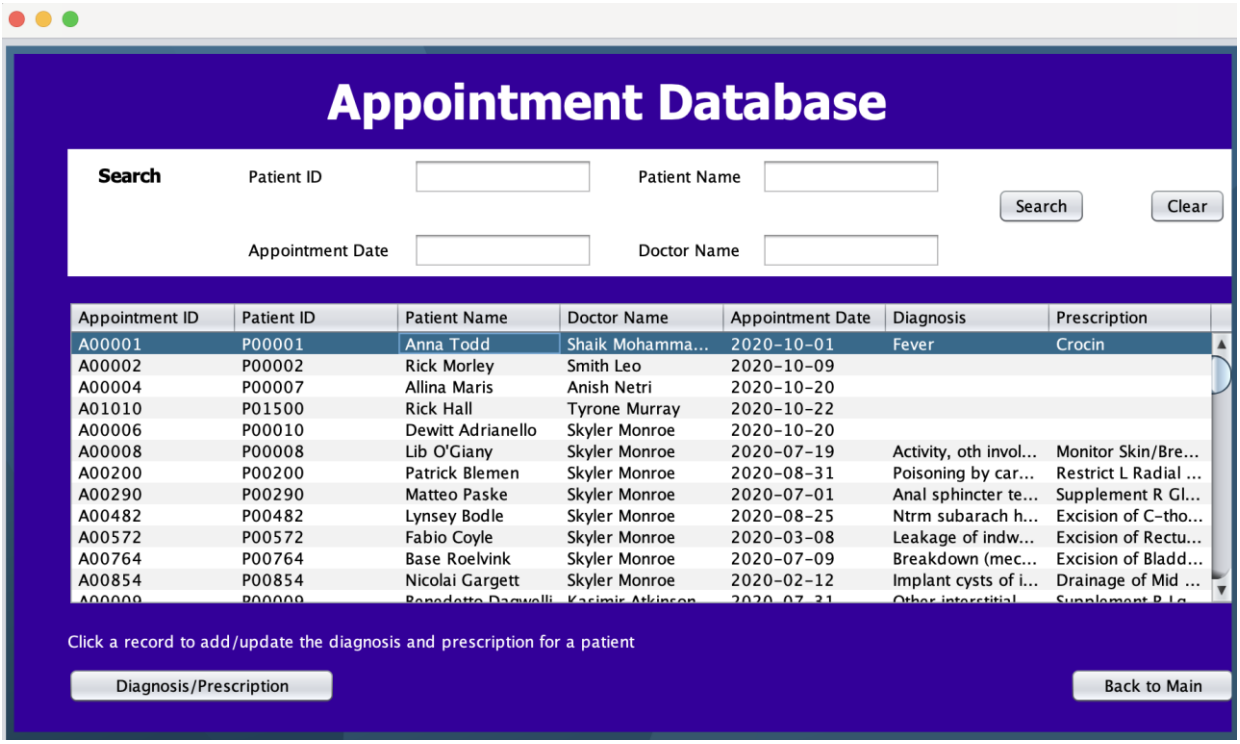
9. ADD/UPDATE Room



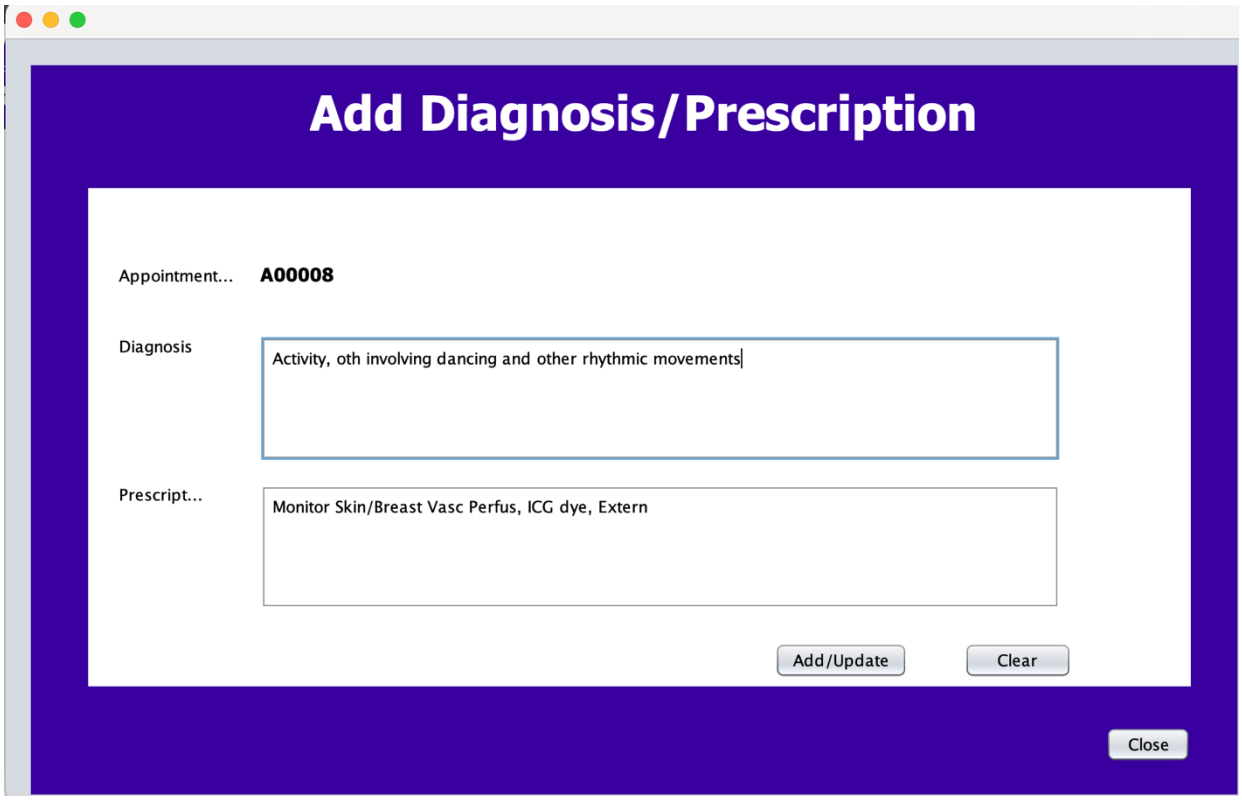
10. Doctor Page



11. Appointments



12. Diagnosis / Prescription



GITHUB LINK :

https://github.com/samyakpshetty/UE20CS352_PROJECT_HMS.git