**Project Problem Statement: Healthcare Management Portal**

**1. Background**

Healthcare systems today need to manage **patients, doctors, appointments, prescriptions, and billing** efficiently. In many hospitals and clinics, traditional record-keeping leads to:

* Long waiting times for appointments
* Manual records that are error-prone
* No centralized system to link **patients, doctors, and treatments**
* Lack of transparency in billing and payments

The **Healthcare Management Portal** aims to provide a **centralized system** that digitizes patient records, manages appointments, and streamlines doctor-patient interactions. It will be built with **Core Java (for backend & console menus), PostgreSQL (data storage), and Selenium (testing automation).**

**2. Problem Statement**

The goal is to build a **Healthcare Management Portal** that:

* Allows **patients** to register, search doctors, book/cancel appointments, and view prescriptions.
* Allows **doctors** to manage appointments, diagnose patients, and prescribe medicines.
* Allows **admins** to manage doctors, departments, and billing.
* Uses **PostgreSQL** for secure, reliable data storage.
* Uses **Selenium** to automate key workflows like login, booking, and billing validation.

**3. Objectives**

1. Implement **Core Java console-based application** (OOP, JDBC, exception handling, multithreading for concurrent bookings).
2. Store and manage healthcare data in **PostgreSQL** (patients, doctors, appointments, billing).
3. Provide **role-based access** (Patient, Doctor, Admin).
4. Implement **appointment scheduling system** with real-time availability checks.
5. Automate test flows (login, appointment booking, billing) using **Selenium**.
6. Ensure scalability and maintainability for future extensions (telemedicine, pharmacy, lab results).

**4. System Requirements**

**Functional Requirements**

* **Patient Module**
  + Register & Login
  + Search Doctors (by specialization, availability, hospital)
  + Book/Cancel Appointments
  + View Medical History & Prescriptions
  + Pay Bills Online (dummy payment simulation)
* **Doctor Module**
  + Doctor Login
  + View Today’s Appointments
  + Update Diagnosis & Prescriptions
  + Mark Availability (time slots)
* **Admin Module**
  + Admin Login
  + Add/Update/Delete Doctors & Departments
  + Manage Appointments & Billing Records
  + Generate Reports (daily patients, revenue, cancellations)
* **Billing Module**
  + Generate bills after consultation
  + Track payment status (Success/Failed)
  + Store transaction records
* **Testing Module (Selenium)**
  + Automated test cases for:
    - Patient/Doctor/Admin login
    - Appointment booking & cancellation
    - Prescription updates
    - Billing and payment flows

**Non-Functional Requirements**

* **Performance:** Handle multiple patients booking appointments simultaneously
* **Security:** Protect sensitive patient data (HIPAA-like principles)
* **Reliability:** Ensure accurate appointment & billing records
* **Usability:** Simple **menu-driven console navigation**

**5. Database Schema (PostgreSQL)**

**Tables**

**1. Users (Patients/Doctors/Admins)**

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15),

role VARCHAR(10) CHECK (role IN ('PATIENT','DOCTOR','ADMIN')) NOT NULL

);

**2. Departments**

CREATE TABLE departments (

dept\_id SERIAL PRIMARY KEY,

name VARCHAR(50) NOT NULL UNIQUE

);

**3. Doctors**

CREATE TABLE doctors (

doctor\_id SERIAL PRIMARY KEY,

user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,

dept\_id INT REFERENCES departments(dept\_id),

specialization VARCHAR(100),

availability VARCHAR(100) -- e.g., "Mon-Fri 10am-2pm"

);

**4. Patients**

CREATE TABLE patients (

patient\_id SERIAL PRIMARY KEY,

user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,

age INT,

gender VARCHAR(10),

blood\_group VARCHAR(5)

);

**5. Appointments**

CREATE TABLE appointments (

appointment\_id SERIAL PRIMARY KEY,

patient\_id INT REFERENCES patients(patient\_id) ON DELETE CASCADE,

doctor\_id INT REFERENCES doctors(doctor\_id) ON DELETE CASCADE,

appointment\_date TIMESTAMP NOT NULL,

status VARCHAR(20) CHECK (status IN ('BOOKED','COMPLETED','CANCELLED')) DEFAULT 'BOOKED'

);

**6. Prescriptions**

CREATE TABLE prescriptions (

prescription\_id SERIAL PRIMARY KEY,

appointment\_id INT REFERENCES appointments(appointment\_id) ON DELETE CASCADE,

doctor\_notes TEXT,

medicines TEXT

);