UE22CS352B - Object Oriented Analysis & Design

Hospital Management System

Submitted by:

Nikhil Jha	PES1UG22CS385
Nirnay Jain	PES1UG22CS394
Parv Mehta	PES1UG22CS408

VI Semester G-section

Dr. Bhargavi Mokashi

Associate Professor

PES University

January - May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

TABLE OF CONTENTS

PROBLEM STATEMENT :

The project Hospital Management System includes registration of patients, storing their details into the system, and also computerised billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient automatically. TheHospital Management System can be accessed by different users using a username and password. It is accessible by an administrator, receptionist, doctor and a pharmacist. Only the admin can register users into the database. The data can be retrieved easily. The interface is very user-friendly. The data is well protected for personal use and makes the data processing very fast.

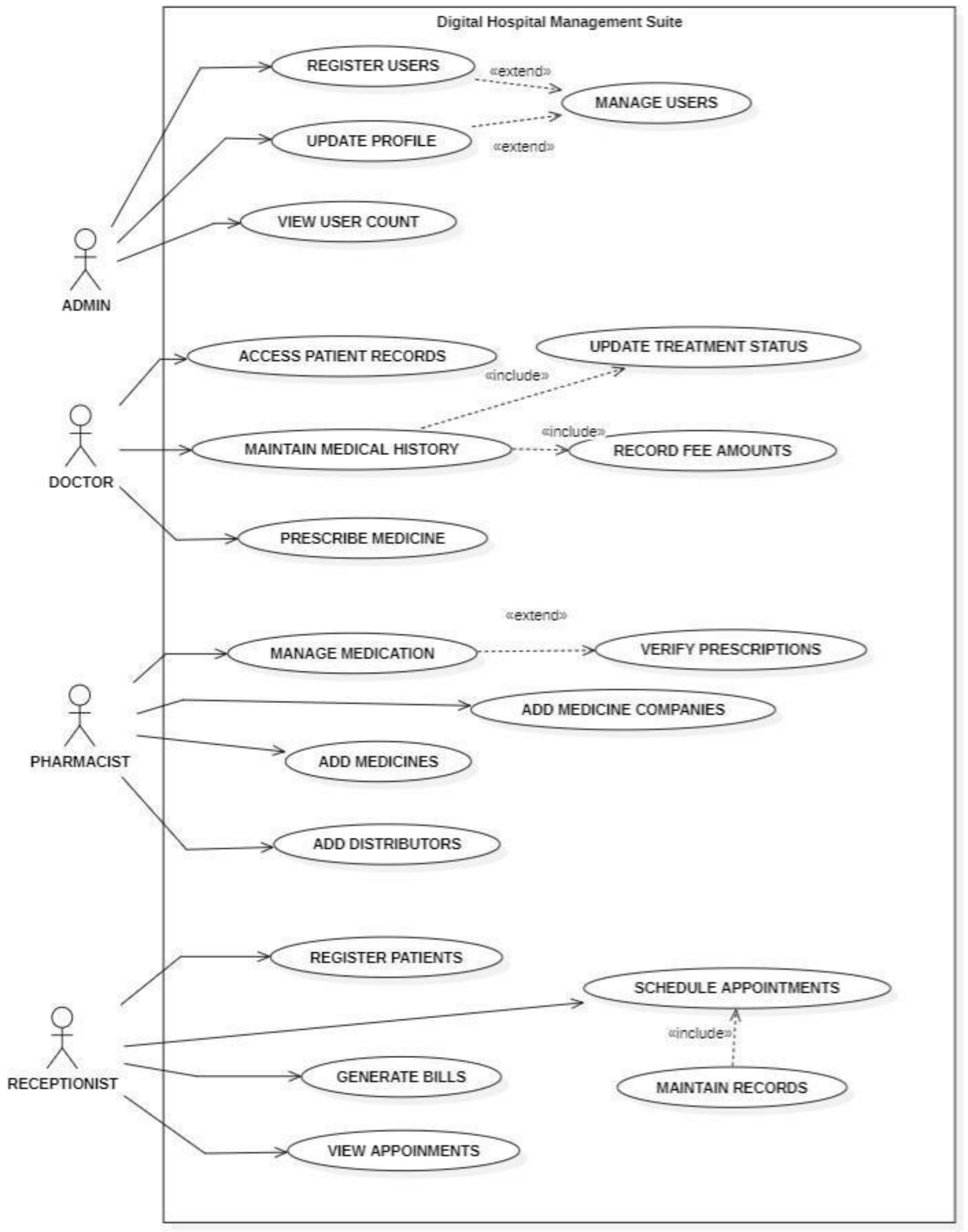
The Hospital Management System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals.

Hospital Management System is designed for multispeciality hospitals, to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration in a seamless flow.

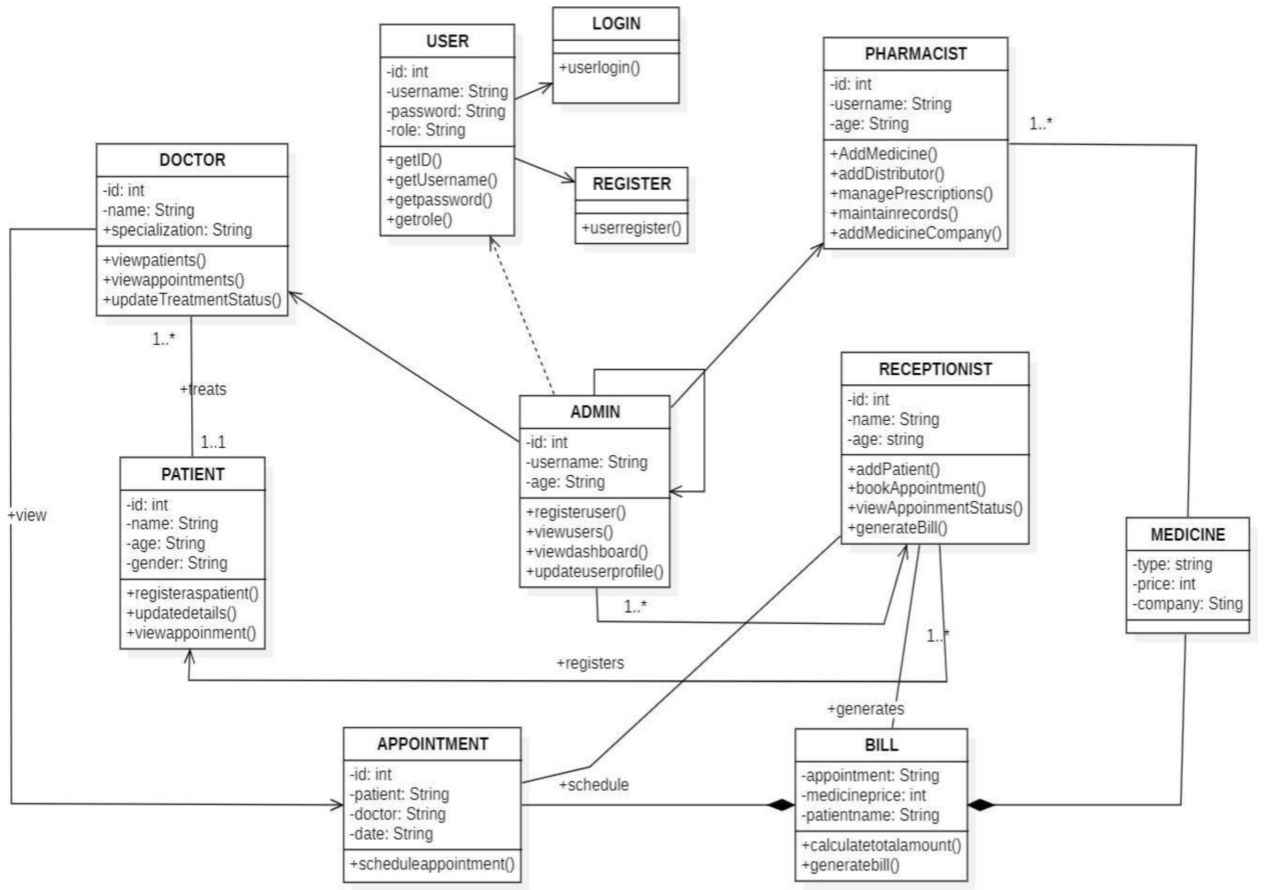
Hospital Management System enables you to develop your organisation and improve its effectiveness and quality of work.

MODELS:

USE CASE DIAGRAM:



CLASS DIAGRAM:



ARCHITECTURE PATTERN:

The architecture pattern chosen for our project is Model-View-Controller(MVC). This pattern separates the system into three interconnected components:

Model:

The model component represents the data and business logic of the Hospital Management System . It consists of entities such as Doctor, Patient, Appointment, and Medicine, which define the structure of the data and contain methods for data manipulation. Each entity encapsulates properties and behaviors relevant to its domain, ensuring data integrity and consistency throughout the application.

View:

The view component encompasses the user interface (UI) elements of the Hospital Management System , responsible for presenting information to users in a human-readable format. Views are implemented using technologies like JSP (JavaServer Pages), Thymeleaf, or HTML templates, which render data dynamically to create interactive and visually appealing interfaces. Views display information retrieved from the model and allow users to interact with the system's functionalities.

Controller:

The controller component acts as an intermediary between the model and the view in the Hospital Management System . Controllers receive user input from the UI, process it, and determine the appropriate response or view to be rendered based on the input and application logic. Controllers are implemented as classes annotated with @Controller or @RestController, handling incoming HTTP requests and invoking methods on the service layer (model) to perform business logic operations. By separating concerns, controllers enable modular development and facilitate the maintenance and testing of the application's functionality.

DESIGN PRINCIPLES:

1. Single Responsibility Principle (SRP):

Each class and method is designed to have a single responsibility. For example, service classes are responsible for handling business logic related to their respective entities.

2. Open/Closed Principle (OCP):

The codebase is designed to be open for extension but closed for modification. For example, new functionalities can be added by creating new classes or extending existing ones without modifying the core implementation.

3. Liskov Substitution Principle (LSP):

Subtypes (e.g., service implementations, DAO implementations) can be substituted for their base types (e.g., service interfaces, DAO interfaces) without affecting the system's behavior. This enables polymorphism and facilitates easier code maintenance and scalability.

4. Interface Segregation Principle (ISP):

Service interfaces are designed with specific methods relevant to their corresponding entities. This ensures that clients only depend on methods they use, reducing the risk of interface pollution and making the system more cohesive.

DESIGN PATTERNS :

Factory Method Pattern : This pattern is implemented with the use of Spring's @Service annotation acts as a form of factory method pattern. This annotation instructs the Spring framework to manage the creation and lifecycle of service beans. By leveraging dependency injection and inversion of control, Spring dynamically instantiates and injects service instances into other components, promoting loose coupling and flexibility in the application's architecture.

DAO (Data Access Object) Pattern: The DAO pattern is utilised in the codebase to separate the data access logic from the business logic. Each service class interacts with the underlying database through its corresponding DAO interface (e.g., DoctorDao, PatientDao) and implementation (e.g., DoctorDaoImpl, PatientDaoImpl). This abstraction allows for interchangeable data access implementations and promotes code reusability, scalability, and maintainability.

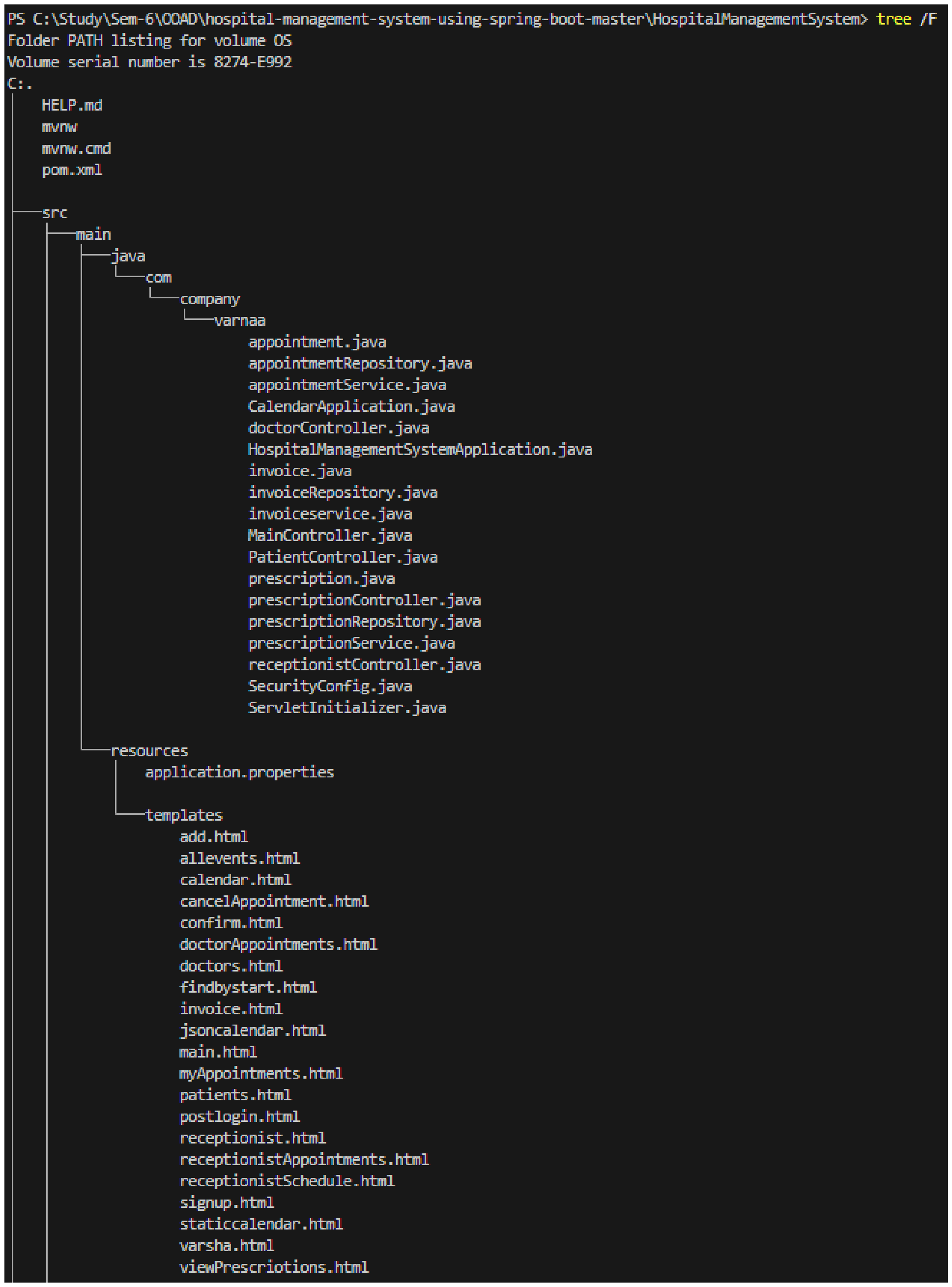
Builder Pattern : This pattern implemented using builder classes, the creation of complex objects, such as entities and DTOs, often involves setting multiple attributes. This aligns with the builder pattern's concept of constructing objects step by step, where each method call configures a specific aspect of the object. While the builder pattern is not directly instantiated in the codebase, the process of constructing objects in a systematic manner follows its principles, enhancing readability and maintainability.

Service Layer Pattern: The service layer pattern is employed in the codebase by defining service interfaces (e.g., DoctorService, PatientService) and their corresponding implementations (e.g., DoctorServiceImpl, PatientServiceImpl). Each service class encapsulates the business logic related to its respective domain entity, such as adding, deleting, updating, or retrieving data. This modular approach promotes separation of concerns and facilitates easier maintenance and testing of the application.

GITHUB REPOSITORY :

https://github.com/NJMINION/HospitalManagement

CODE BASE STRUCTURE :



```
PS C:\Study\Sem-6\OOAD\hospital-management-system-using-spring-boot-master\HospitalManagementSystem> tree /F
Folder PATH listing for volume OS
Volume serial number is 8274-E992
C:.
│   HELP.md
│   mvnw
│   mvnw.cmd
│   pom.xml
│
├───src
│   ├───main
│   │   ├───java
│   │   │   └───com
│   │   │       └───company
│   │   │           └───varnaa
│   │   │                   appointment.java
│   │   │                   appointmentRepository.java
│   │   │                   appointmentService.java
│   │   │                   CalendarApplication.java
│   │   │                   doctorController.java
│   │   │                   HospitalManagementSystemApplication.java
│   │   │                   invoice.java
│   │   │                   invoiceRepository.java
│   │   │                   invoiceservice.java
│   │   │                   MainController.java
│   │   │                   PatientController.java
│   │   │                   prescription.java
│   │   │                   prescriptionController.java
│   │   │                   prescriptionRepository.java
│   │   │                   prescriptionService.java
│   │   │                   receptionistController.java
│   │   │                   SecurityConfig.java
│   │   │                   ServletInitializer.java
│   │   │
│   │   └───resources
│   │       │   application.properties
│   │       │
│   │       └───templates
│   │               add.html
│   │               allevents.html
│   │               calendar.html
│   │               cancelAppointment.html
│   │               confirm.html
│   │               doctorAppointments.html
│   │               doctors.html
│   │               findbystart.html
│   │               invoice.html
│   │               jsoncalendar.html
│   │               main.html
│   │               myAppointments.html
│   │               patients.html
│   │               postlogin.html
│   │               receptionist.html
│   │               receptionistAppointments.html
│   │               receptionistSchedule.html
│   │               signup.html
│   │               staticcalendar.html
│   │               varsha.html
│   │               viewPrescriotions.html
│   │
```

INDIVIDUAL CONTRIBUTION :

Rishav Banerjee - Admin

- Register doctors, pharmacists, and receptionists into the system.
- Access a comprehensive view of the total user count.
- Update personal profile details with ease.
- Manage user profiles efficiently by updating or deleting them.
- Utilise a user-friendly dashboard for accessing and analysing systemgenerated data.

Rudra Khatri - Doctor

- Access patient records and their scheduled appointments effortlessly.
- Update treatment statuses for patients under their care.
- Prescribe medicines to patients according to their diagnosis and treatment plan.
- Record and manage fee amounts for individual patients seamlessly.
- Maintain detailed medical histories for each patient, including diagnoses, treatments, and progress notes.

Pravar Raj Singh : Patient signup

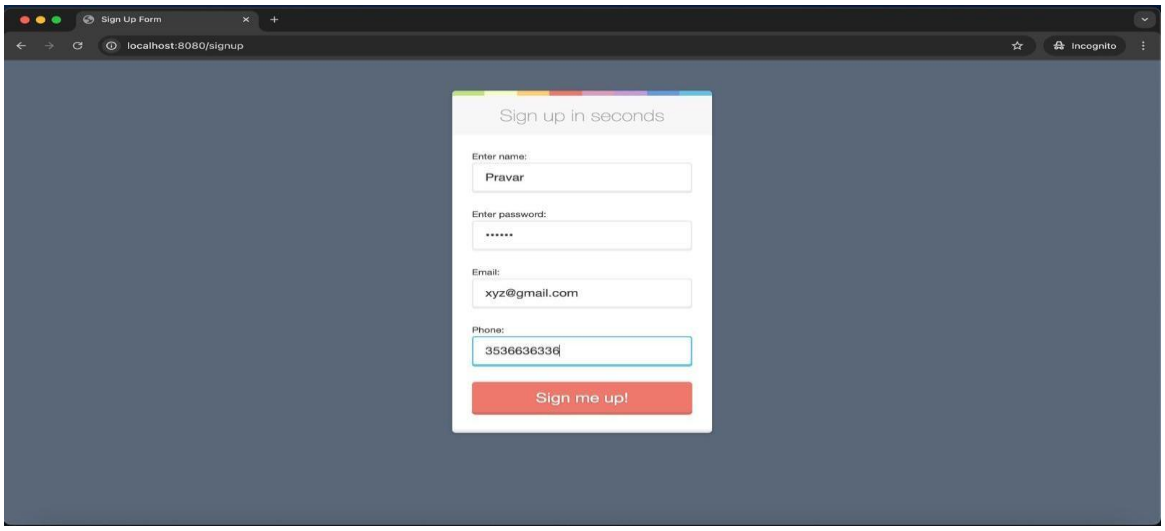
- Sign up and log in to the system to access personalized features.
- Book new appointments using the calendar and scheduling interface.
- View and cancel existing appointments.
- Browse doctor profiles and availability.
- View prescribed Medicine.

Prateek S Nyamagoudar - Receptionist

- Register Patients and schedule appointments.
- Monitor and update treatment progress.
- Generate bills for patient services.
- Manage inventory and medication distribution.
- Maintain patient records and medical history.

OUTPUT SCREENSHOTS :

- Register Page



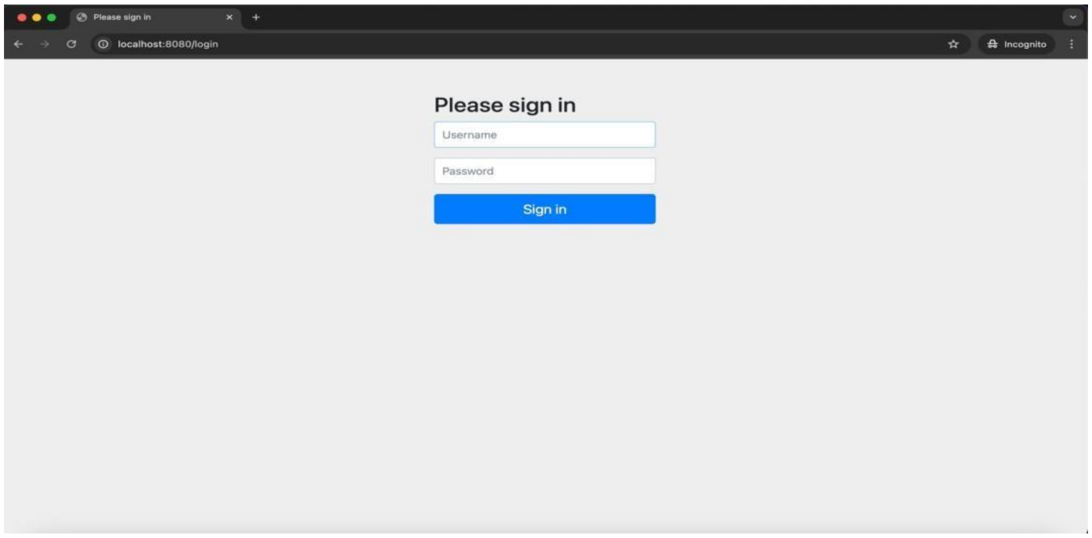
Enter name:
Pravar

Enter password:
••••••

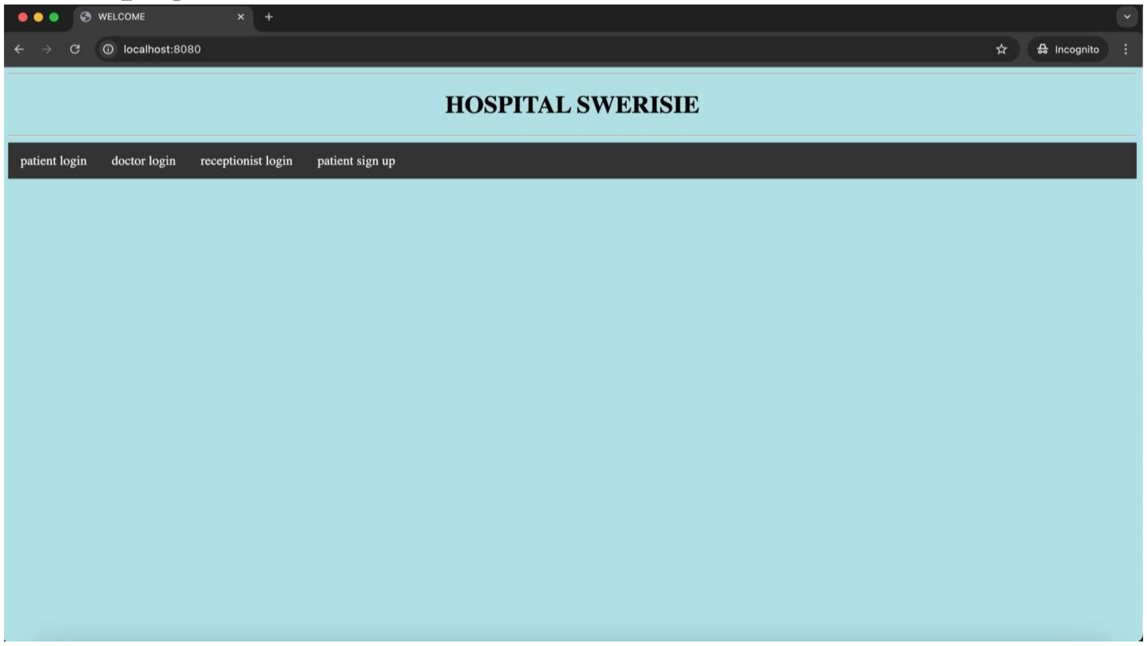
Email:
xyz@gmail.com

Phone:
3536636336

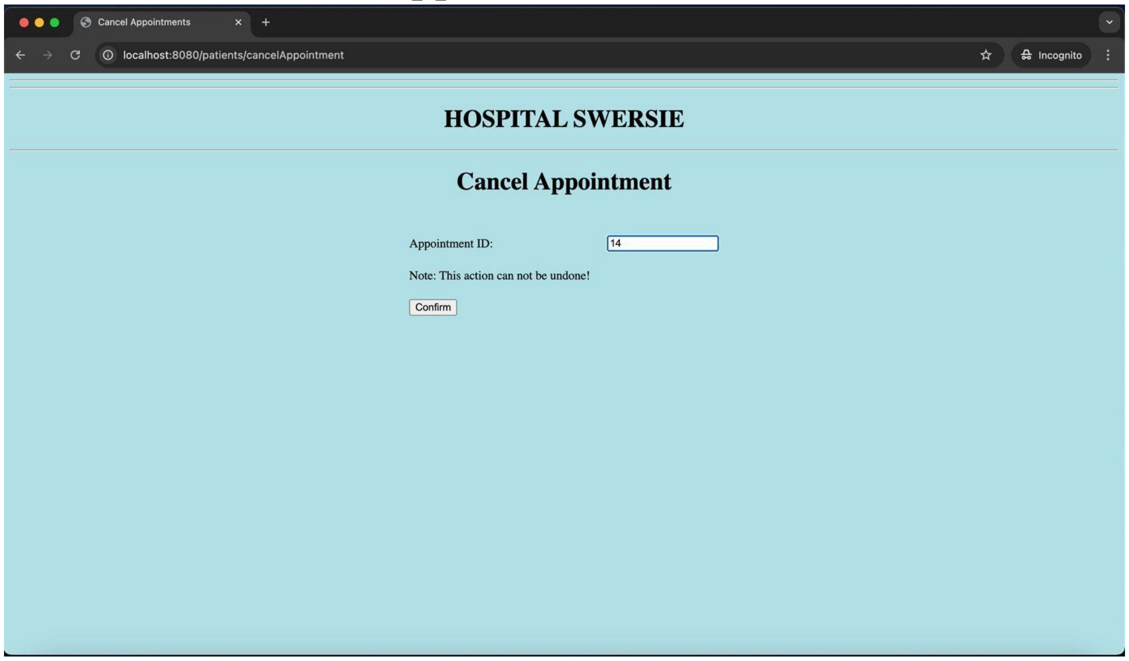
- Admin Register



- Homepage

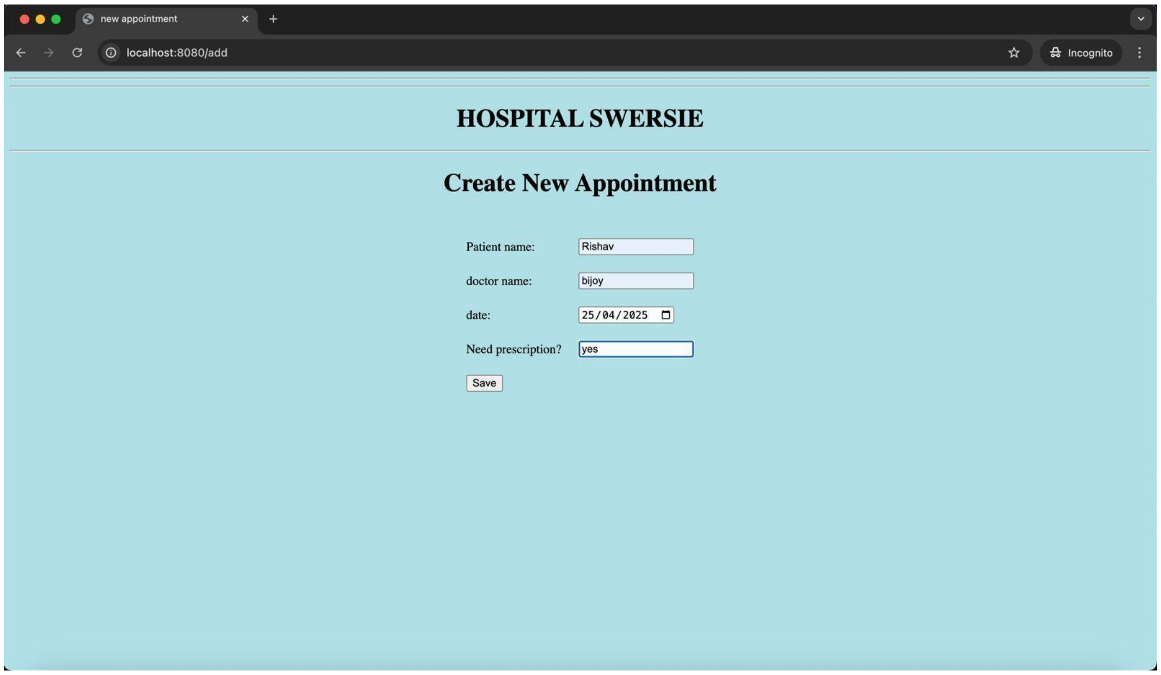


- Create And Cancel Appointment



Appointment ID: 14

Note: This action can not be undone!

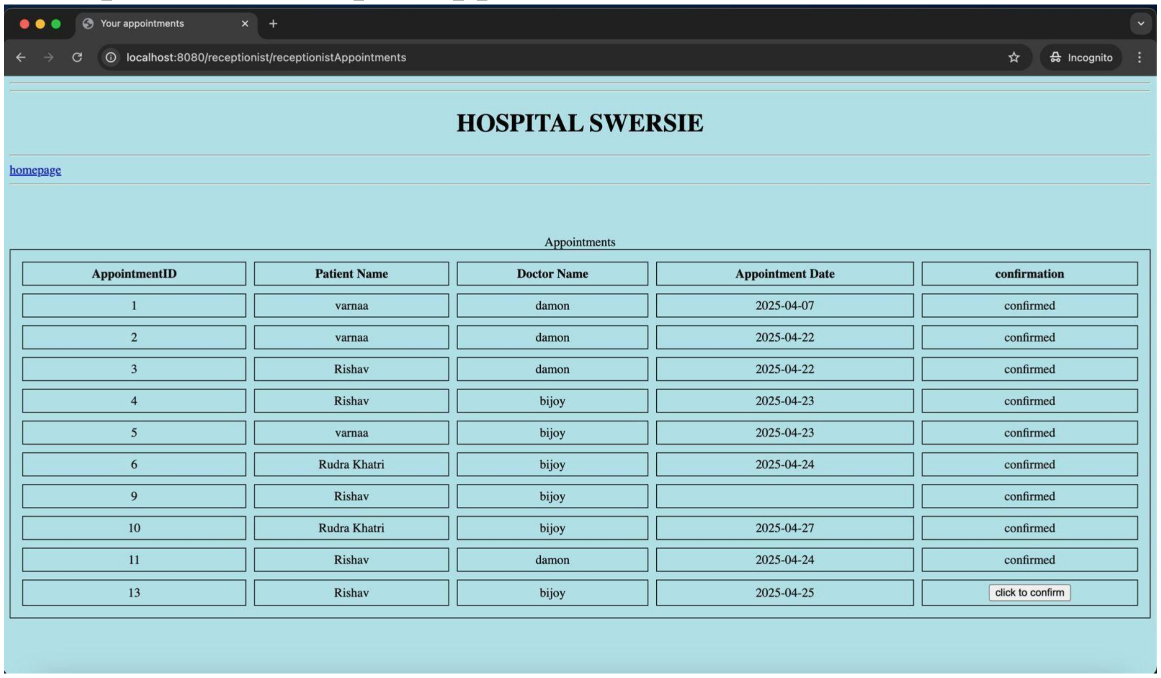


HOSPITAL SWERSIE

Create New Appointment

Patient name:	Rishav
doctor name:	bijoy
date:	25/04/2025
Need prescription?	yes

- Receptionist Accepts Appointments



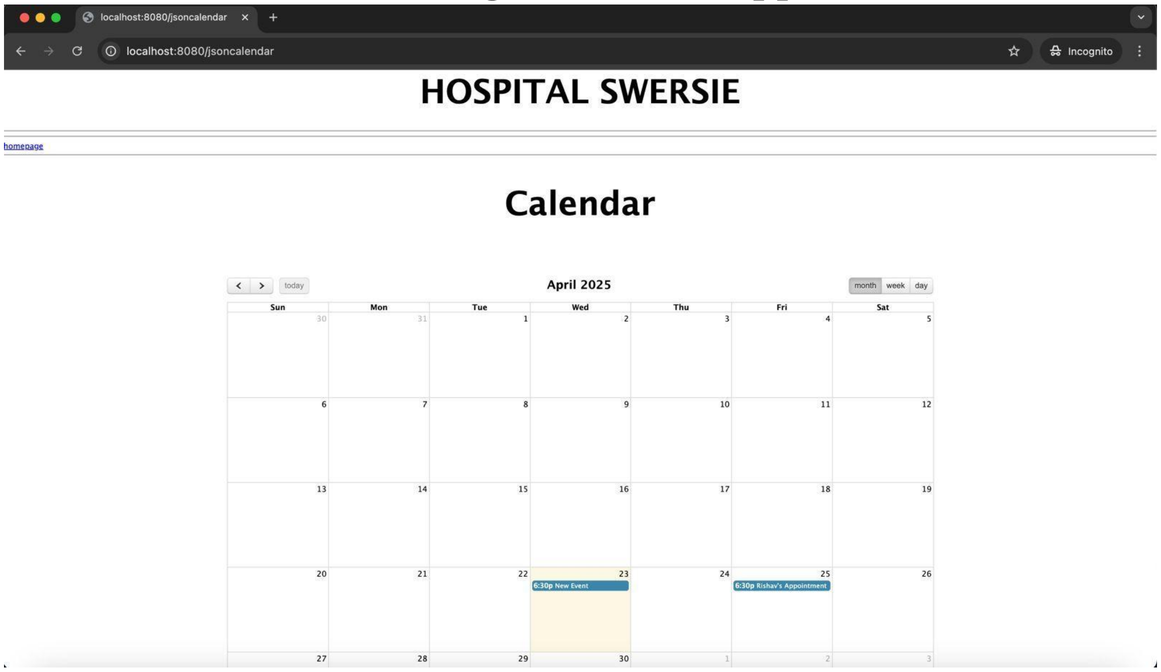
HOSPITAL SWERSIE

homepage

Appointments

AppointmentID	Patient Name	Doctor Name	Appointment Date	confirmation
1	varnaa	damon	2025-04-07	confirmed
2	varnaa	damon	2025-04-22	confirmed
3	Rishav	damon	2025-04-22	confirmed
4	Rishav	bijoy	2025-04-23	confirmed
5	varnaa	bijoy	2025-04-23	confirmed
6	Rudra Khatri	bijoy	2025-04-24	confirmed
9	Rishav	bijoy		confirmed
10	Rudra Khatri	bijoy	2025-04-27	confirmed
11	Rishav	damon	2025-04-24	confirmed
13	Rishav	bijoy	2025-04-25	click to confirm

- Doctors calendar showing scheduled Appointments



HOSPITAL SWERSIE

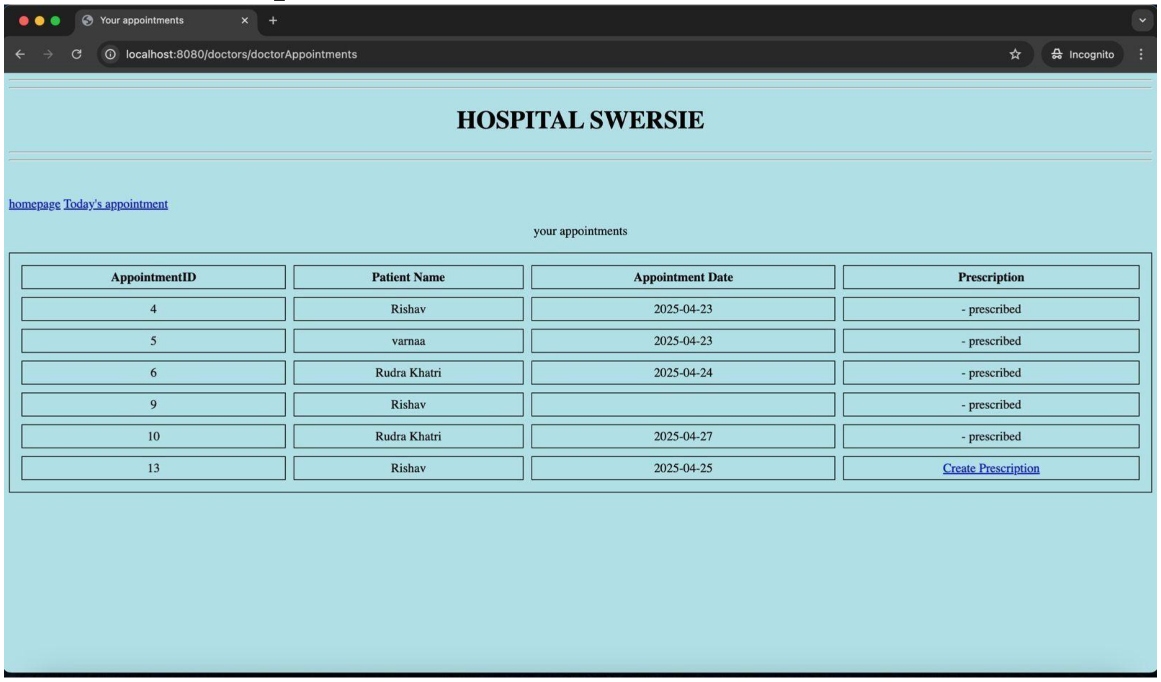
homepage

Calendar

April 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23 6:30p New Event	24	25 6:30p Rishav's Appointment	26
27	28	29	30	1	2	3

- Create Prescriptions



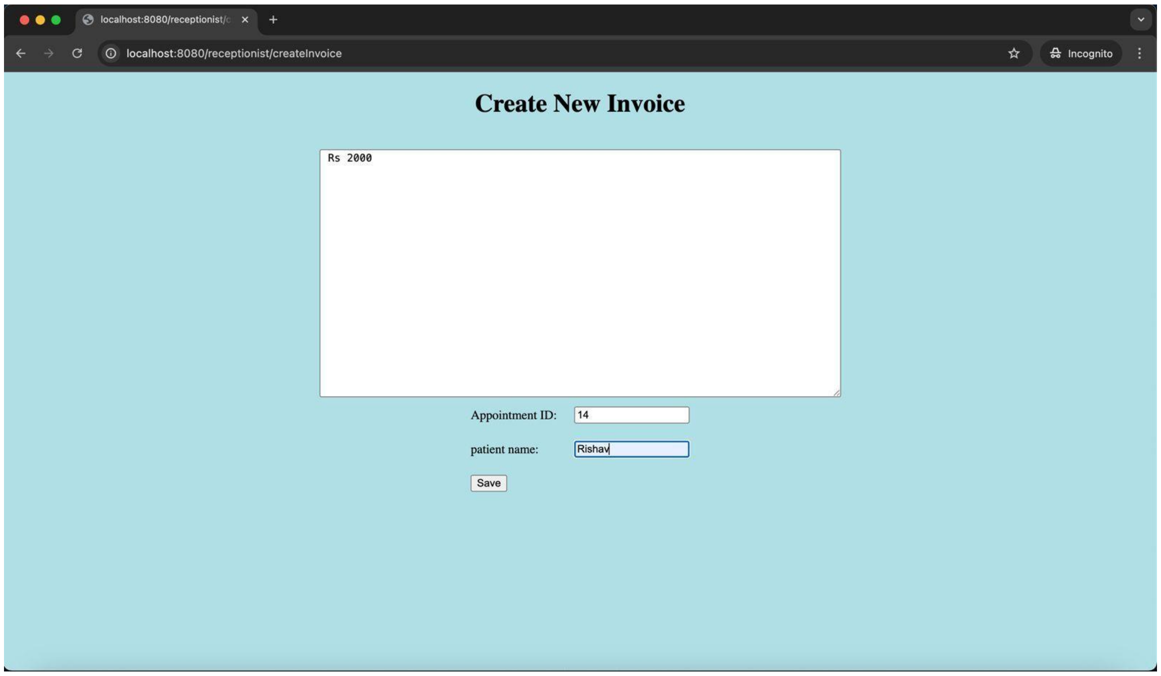
HOSPITAL SWERSIE

homepage Today's appointment

your appointments

AppointmentID	Patient Name	Appointment Date	Prescription
4	Rishav	2025-04-23	- prescribed
5	varnaa	2025-04-23	- prescribed
6	Rudra Khatri	2025-04-24	- prescribed
9	Rishav		- prescribed
10	Rudra Khatri	2025-04-27	- prescribed
13	Rishav	2025-04-25	Create Prescription

- Invoice



localhost:8080/receptionist/createInvoice

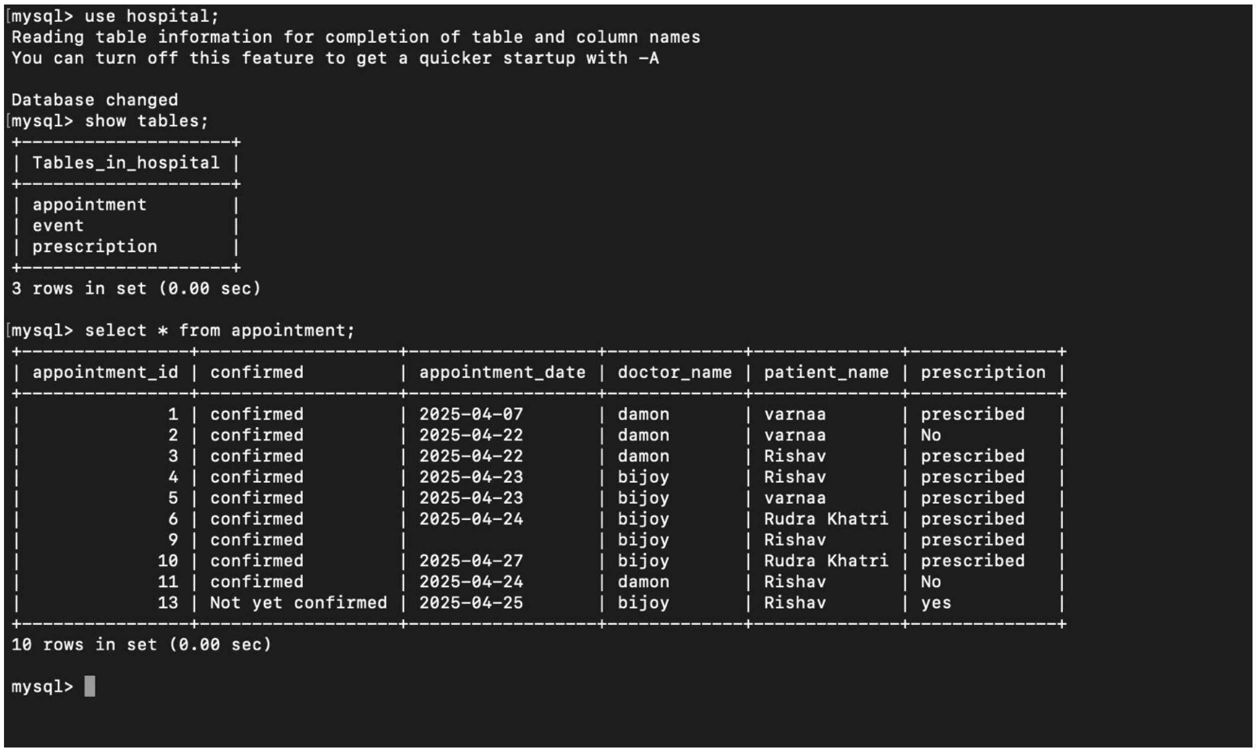
Create New Invoice

```
Rs 2000
```

Appointment ID: 14

patient name: Rishav

DATABASE SCREENSHOTS:

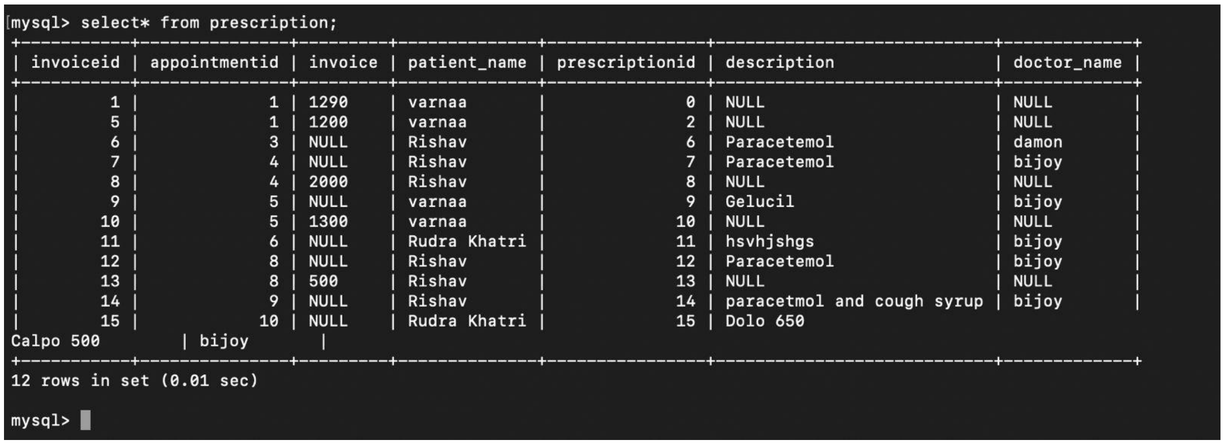


```
mysql> use hospital;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------+
| Tables_in_hospital |
+---------------------+
| appointment        |
| event              |
| prescription       |
+---------------------+
3 rows in set (0.00 sec)

mysql> select * from appointment;
+----------------+-------------------+------------------+-------------+--------------+--------------+
| appointment_id | confirmed         | appointment_date | doctor_name | patient_name | prescription |
+----------------+-------------------+------------------+-------------+--------------+--------------+
|              1 | confirmed         | 2025-04-07       | damon       | varnaa       | prescribed   |
|              2 | confirmed         | 2025-04-22       | damon       | varnaa       | No           |
|              3 | confirmed         | 2025-04-22       | damon       | Rishav       | prescribed   |
|              4 | confirmed         | 2025-04-23       | bijoy       | Rishav       | prescribed   |
|              5 | confirmed         | 2025-04-23       | bijoy       | varnaa       | prescribed   |
|              6 | confirmed         | 2025-04-24       | bijoy       | Rudra Khatri | prescribed   |
|              9 | confirmed         |                  | bijoy       | Rishav       | prescribed   |
|             10 | confirmed         | 2025-04-27       | bijoy       | Rudra Khatri | prescribed   |
|             11 | confirmed         | 2025-04-24       | damon       | Rishav       | No           |
|             13 | Not yet confirmed | 2025-04-25       | bijoy       | Rishav       | yes          |
+----------------+-------------------+------------------+-------------+--------------+--------------+
10 rows in set (0.00 sec)

mysql> 
```



```
mysql> select* from prescription;
+-----------+---------------+---------+--------------+----------------+-----------------------------+-------------+
| invoiceid | appointmentid | invoice | patient_name | prescriptionid | description                 | doctor_name |
+-----------+---------------+---------+--------------+----------------+-----------------------------+-------------+
|         1 |             1 | 1290    | varnaa       |              0 | NULL                        | NULL        |
|         5 |             1 | 1200    | varnaa       |              2 | NULL                        | NULL        |
|         6 |             3 | NULL    | Rishav       |              6 | Paracetemol                 | damon       |
|         7 |             4 | NULL    | Rishav       |              7 | Paracetemol                 | bijoy       |
|         8 |             4 | 2000    | Rishav       |              8 | NULL                        | NULL        |
|         9 |             5 | NULL    | varnaa       |              9 | Gelucil                     | bijoy       |
|        10 |             5 | 1300    | varnaa       |             10 | NULL                        | NULL        |
|        11 |             6 | NULL    | Rudra Khatri |             11 | hsvhjshgs                   | bijoy       |
|        12 |             8 | NULL    | Rishav       |             12 | Paracetemol                 | bijoy       |
|        13 |             8 | 500     | Rishav       |             13 | NULL                        | NULL        |
|        14 |             9 | NULL    | Rishav       |             14 | paracetmol and cough syrup  | bijoy       |
|        15 |            10 | NULL    | Rudra Khatri |             15 | Dolo 650
Calpo 500        | bijoy       |
+-----------+---------------+---------+--------------+----------------+-----------------------------+-------------+
12 rows in set (0.01 sec)

mysql> 
```