**Patient Medicine and Appointment System**

**Project Abstract: Patient Medicine and Appointment System**

The **Patient Medicine and Appointment System** is a comprehensive web application designed to streamline patient registration, appointment scheduling, and medication management. Built using **Spring Boot** on the backend and **HTML, CSS, Thymeleaf** on the frontend, this system provides an efficient and user-friendly platform for managing healthcare appointments and prescriptions.

The core features of the system include:

1. **User Registration and Authentication**: Patients can securely register and log in to the system. The registration process captures essential details such as contact information, medical history, and other relevant data. The authentication mechanism ensures data security and personalized access.
2. **Appointment Scheduling**: Patients can view available appointment slots, select preferred doctors, and book appointments based on their schedules. The system dynamically updates appointment availability, preventing double bookings and ensuring seamless scheduling.
3. **Medication Management**: Patients can view and manage their prescribed medications. The system allows for the addition, updating, and removal of medications, giving patients control over their medication information.
4. **Database Management**: Using **MongoDB**, the system stores patient, doctor, appointment, and medication data efficiently. The relational data model ensures scalability and proper management of entity relationships such as patient-doctor-appointment links.
5. **RESTful APIs**: The backend exposes RESTful APIs for handling core operations, such as user registration, appointment booking, and medication management. These APIs ensure flexibility and potential integration with other systems, such as hospital management platforms.
6. **Responsive Frontend**: The application’s user interface is built with **Bootstrap** and **Thymeleaf**, ensuring a responsive design that works across various devices, including desktops, tablets, and smartphones. The clean, intuitive UI simplifies patient interactions with the system.

**Key Technologies:**

* **Backend**: Spring Boot, REST APIs
* **Frontend**: HTML, CSS, Bootstrap, Thymeleaf
* **Database**: MongoDB
* **Security**: User authentication, form validation

The **Patient Medicine and Appointment System** enhances the efficiency of healthcare service delivery by automating critical patient interactions, reducing the administrative burden on healthcare providers, and providing patients with easy access to their medical appointments and prescriptions. This project can be scaled to support integration with hospital systems and expanded to include features such as doctor reviews, notifications, and real-time health monitoring.

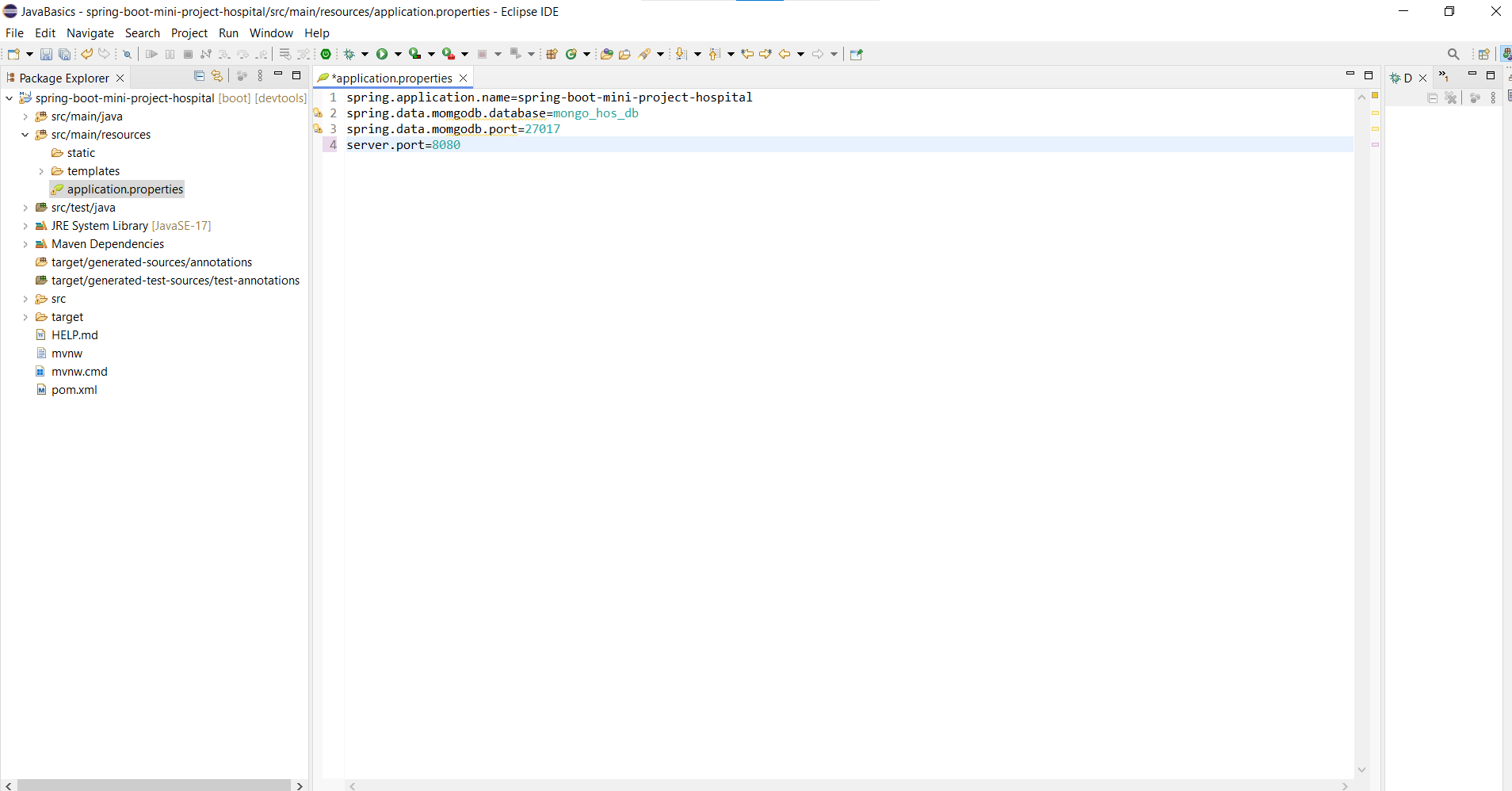
**1.Installation and Setup**

* **Prerequisites:**
  + Java Development Kit (JDK) 11 or higher
  + Maven
  + MongoDB
  + IDE (e.g., Eclipse)
* **Clone the Repository:**

git clone<https://github.com/Makitha03/Mini-project-2.git>.

* **Backend Setup:**

1. Navigate to the project directory.
2. Configure the database connection in application.properties



* **Build and run the application:**

1. Mvn clean install
2. Mvn spring-boot:run

**2.pom.xml on added:**

1. Spring Data Reactive MongoDB.
2. Spring Data MongoDB.
3. Thymeleaf
4. Spring web
5. Validation
6. Spring Boot DevTools.

The application will be accessible at htttp://localhost:8080.

**3.Database and Data Model**

* **Entities:**

**1.Patient:** Stores patient information.

**2.Doctor:** Stores doctor information.

**3.Apointment:** Tracks appointment details.

**4.Medication:** Stores prescribed medication.

* **Entity Relationships:**

1.One-to-many between Patient and Appointment.

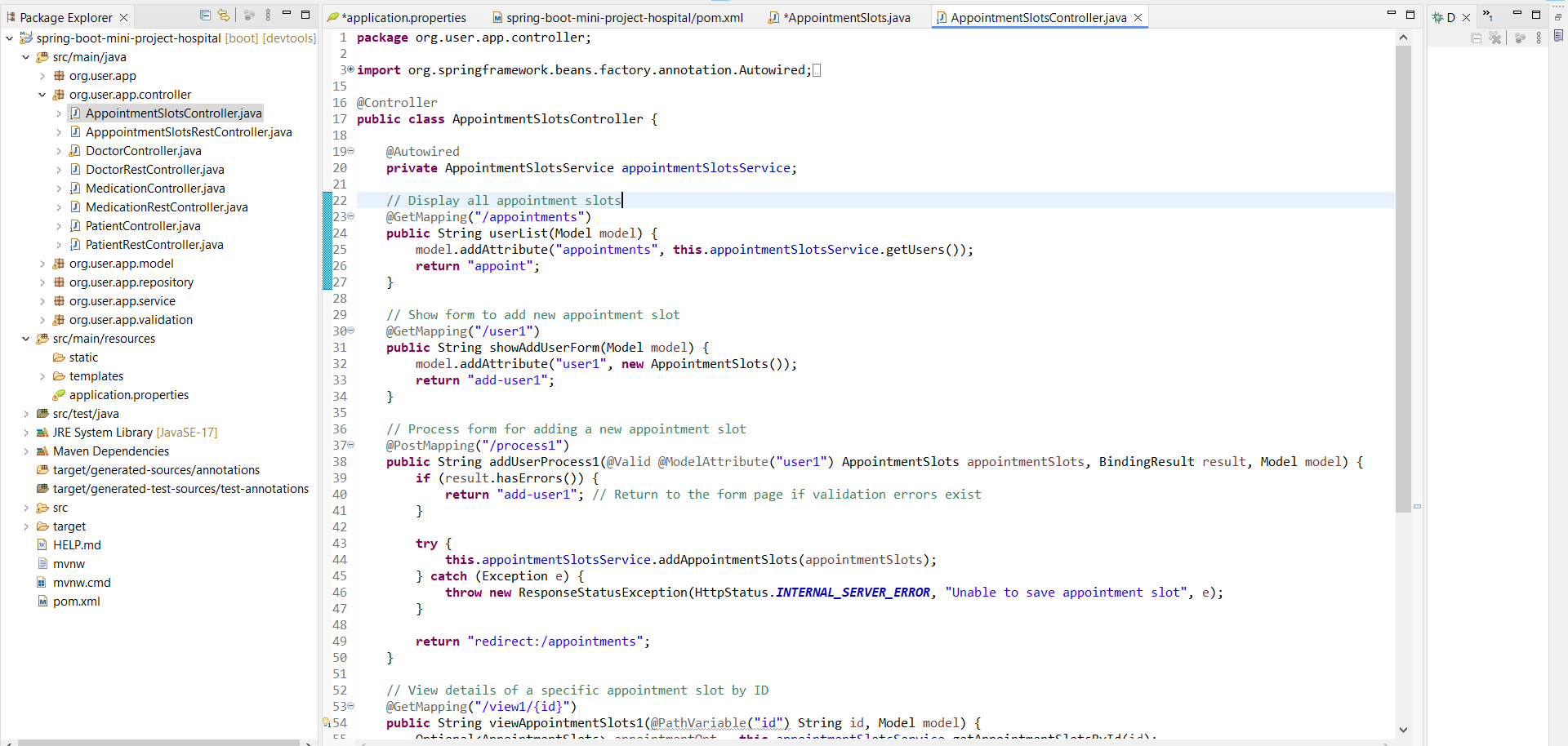
2.Many-to-one between Appointment and Doctor.

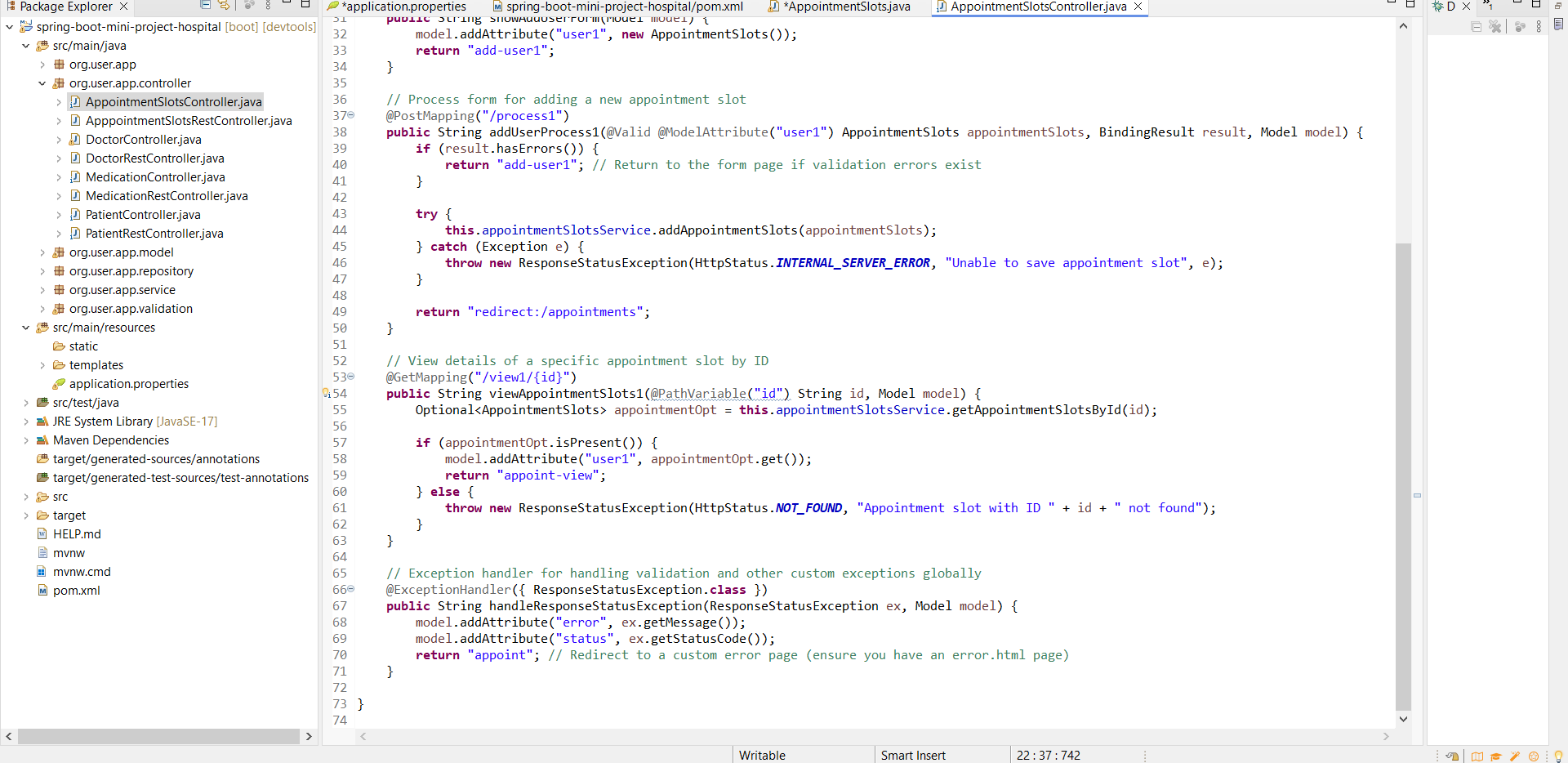
3.One-to-many between Patient and Medication.

**4.BackEnd Development (Spring boot )**

* **Coding:**The application uses MongoDB for database Interaction. Below the example of Appointment Controller.

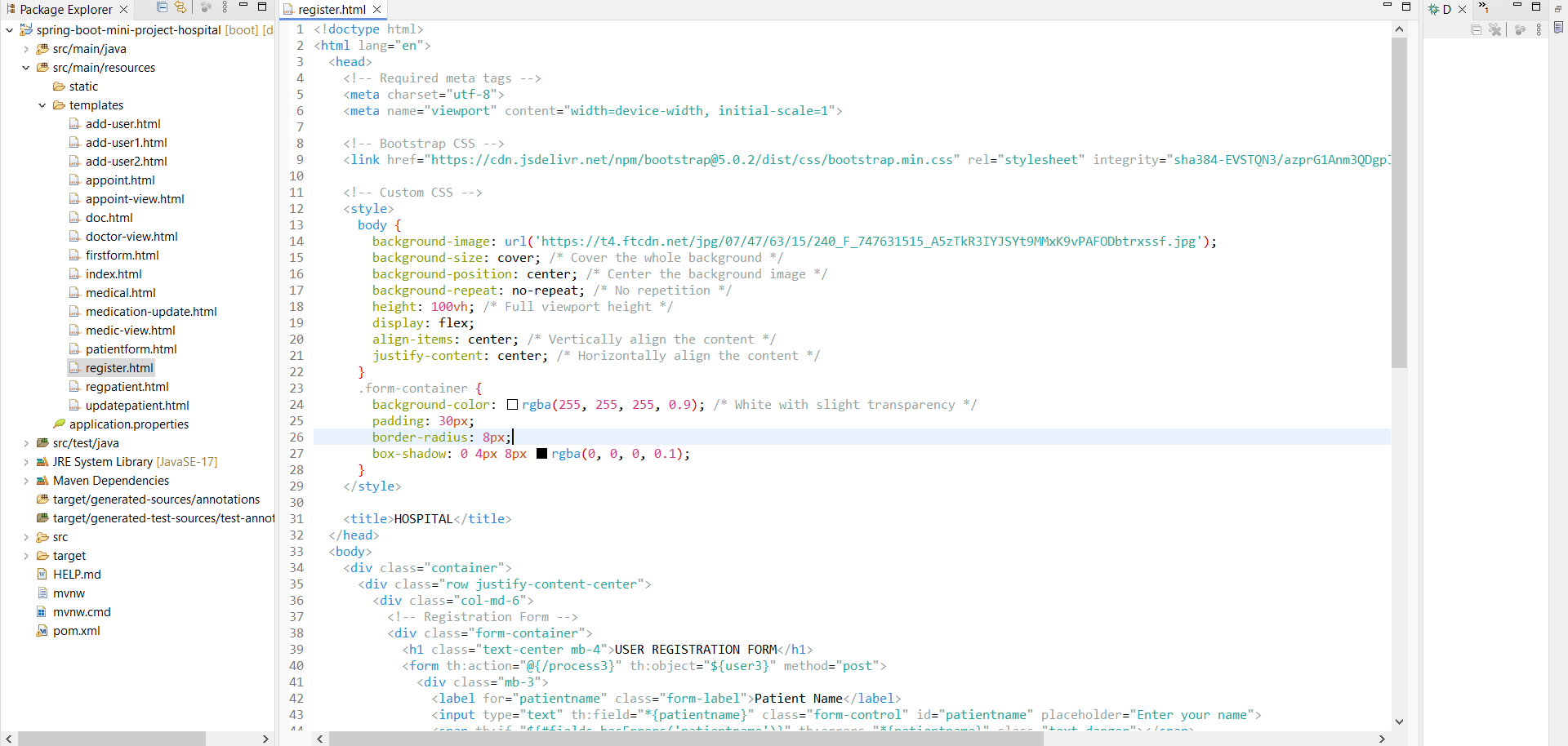
A Spring Boot controller handles incoming HTTP requests, mapping them to specific methods using annotations like @GetMapping or @PostMapping. It serves as the intermediary between the client and the backend, invoking service layer logic to process requests. Controllers use dependency injection to call services, handle validation, and return appropriate HTTP responses (e.g., JSON data or error codes). Each method is mapped to a specific URL pattern and HTTP method, managing actions like retrieval, creation, updating, and deletion of resources.

****

****

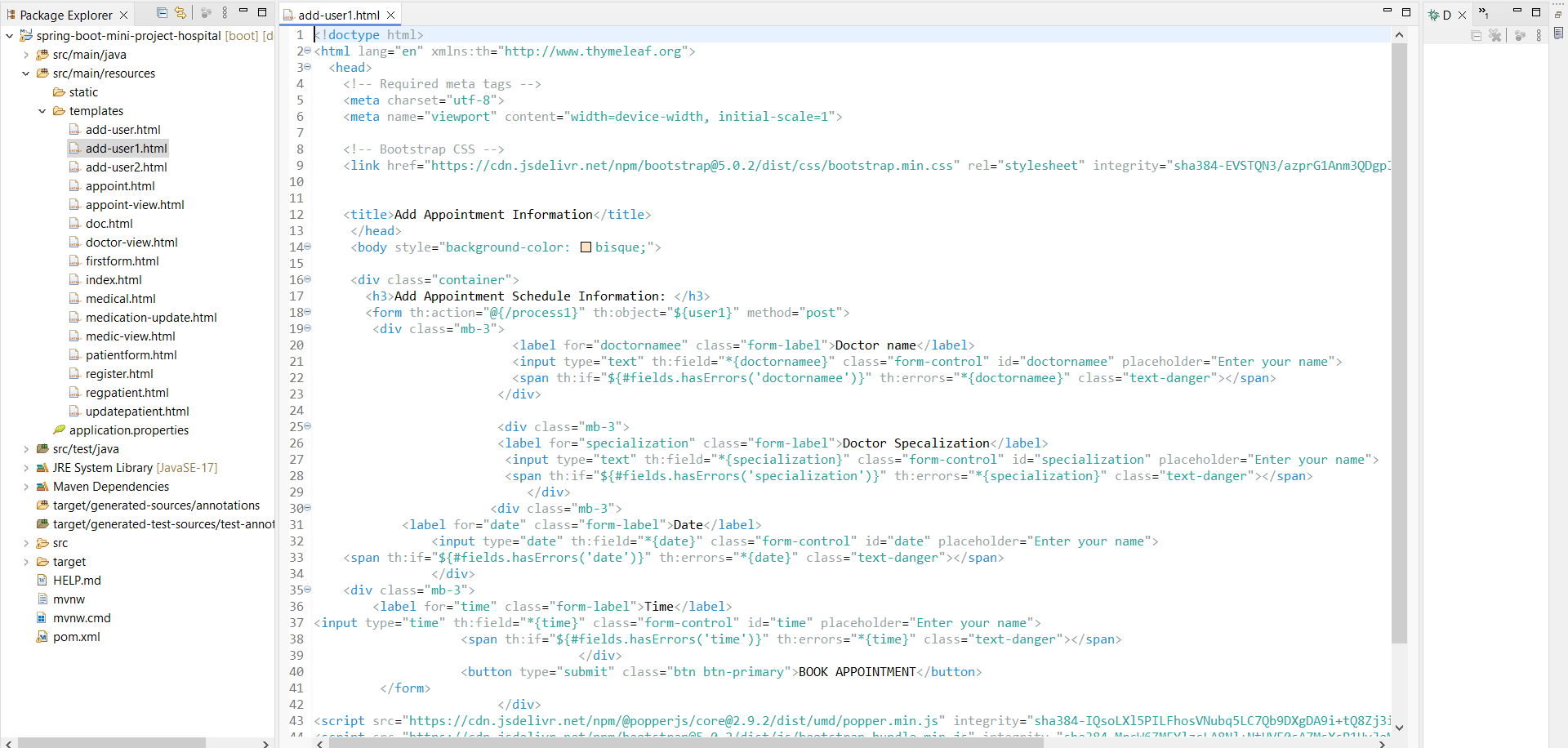
**6.FRONDEND CODING PAGES:**

* **REGISTRATION PAGE:**
* A form that collects patient details and sends them to the registration endpoints.

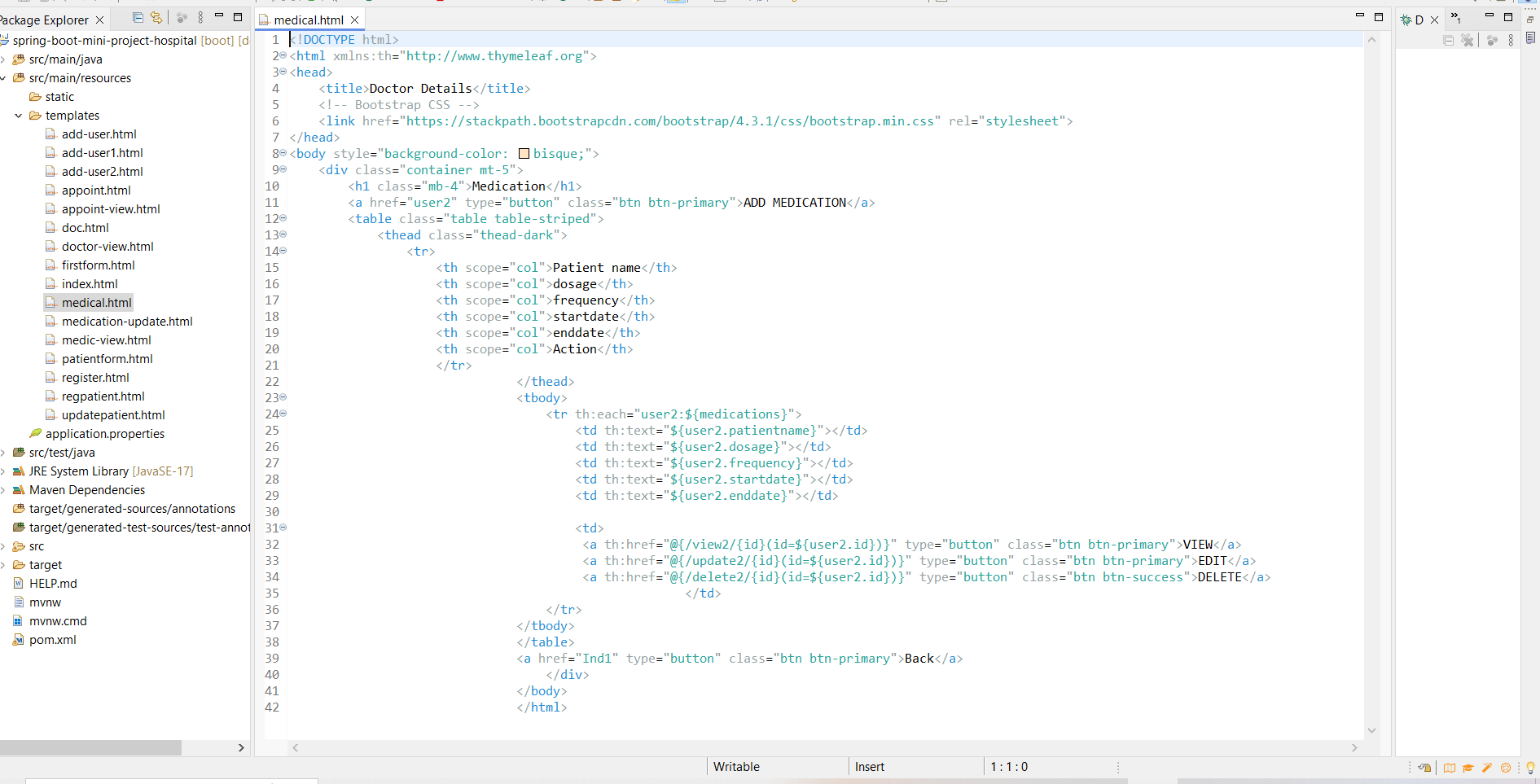




* **APPOINTMENT BOOKING CODING PAGE:**
* Display available time slots for doctor and allows patients to book an appointment**.**

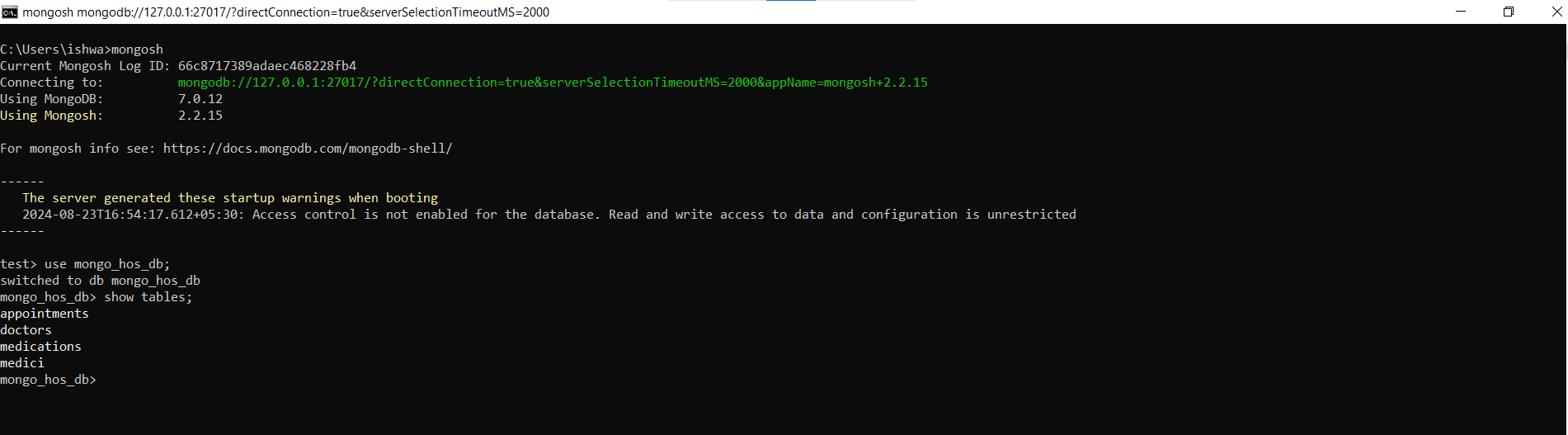
****

* **MEDICATION MANAGEMENT CODIING PAGE:**
* Display the patient’s current medication with options to add, update or delete them.
* **CSS STYLING-**Used Bootstrap to ensure the frondend is responsive and user-friendly.

****

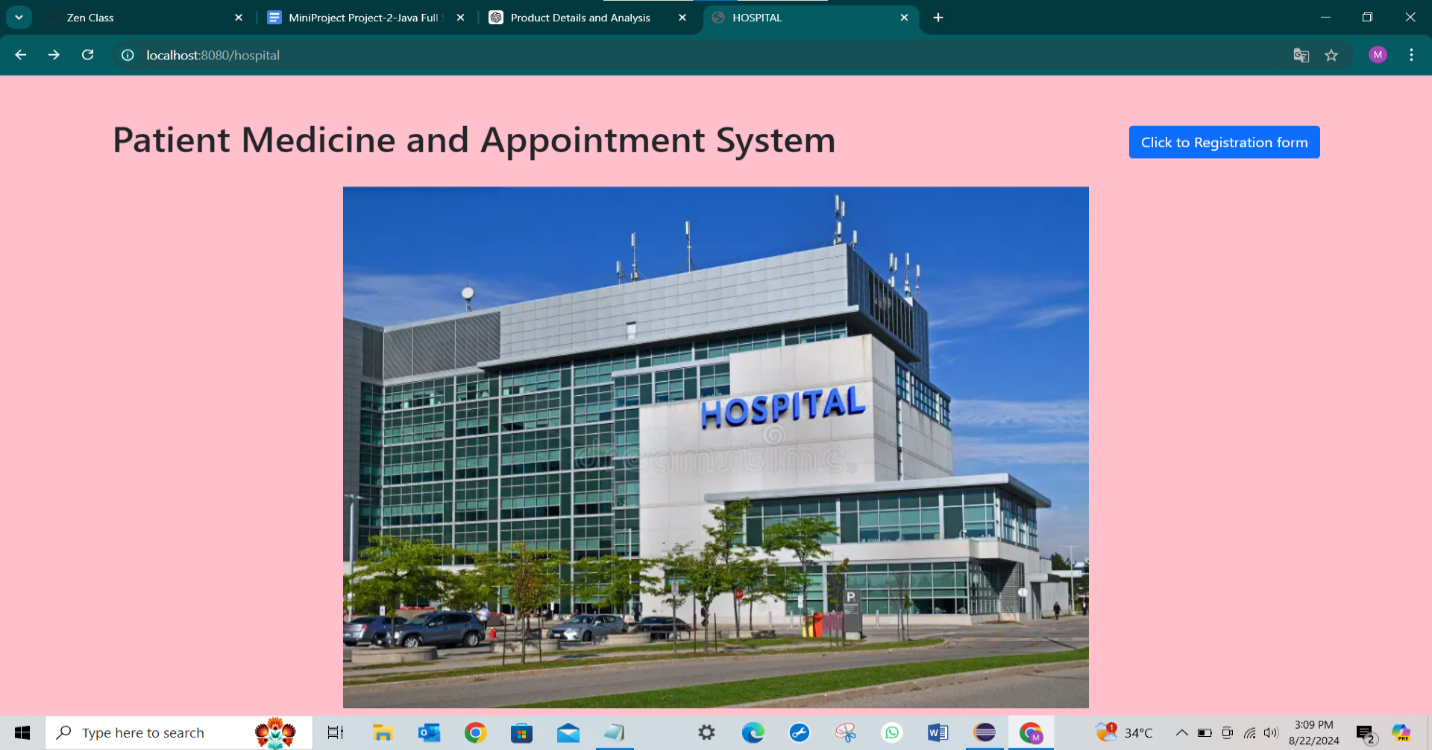
**6.Database and Data Model**.**:**

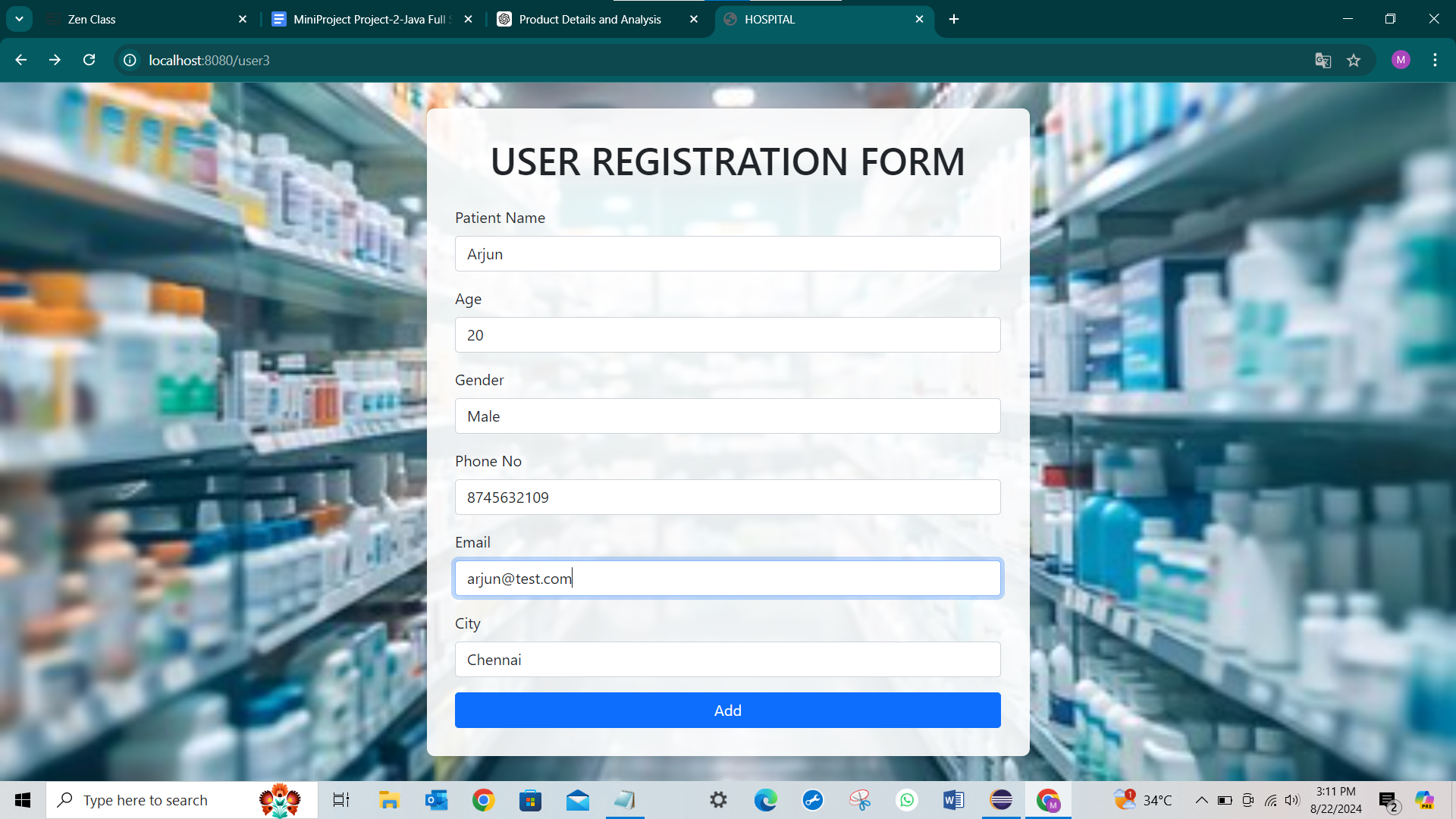
* MongoDB Setup Set up the database depending on your preference. Use the following commands to create the database:

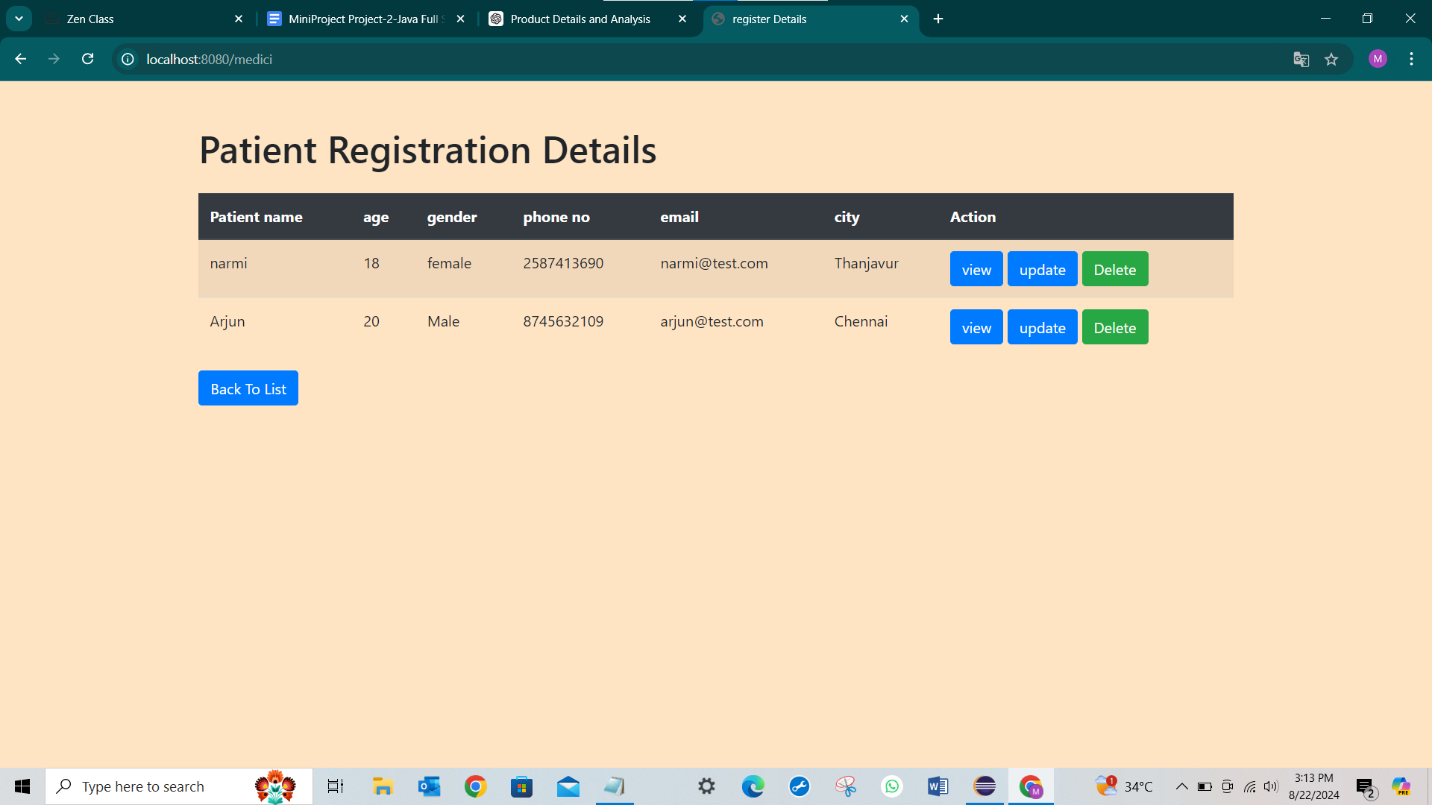
****

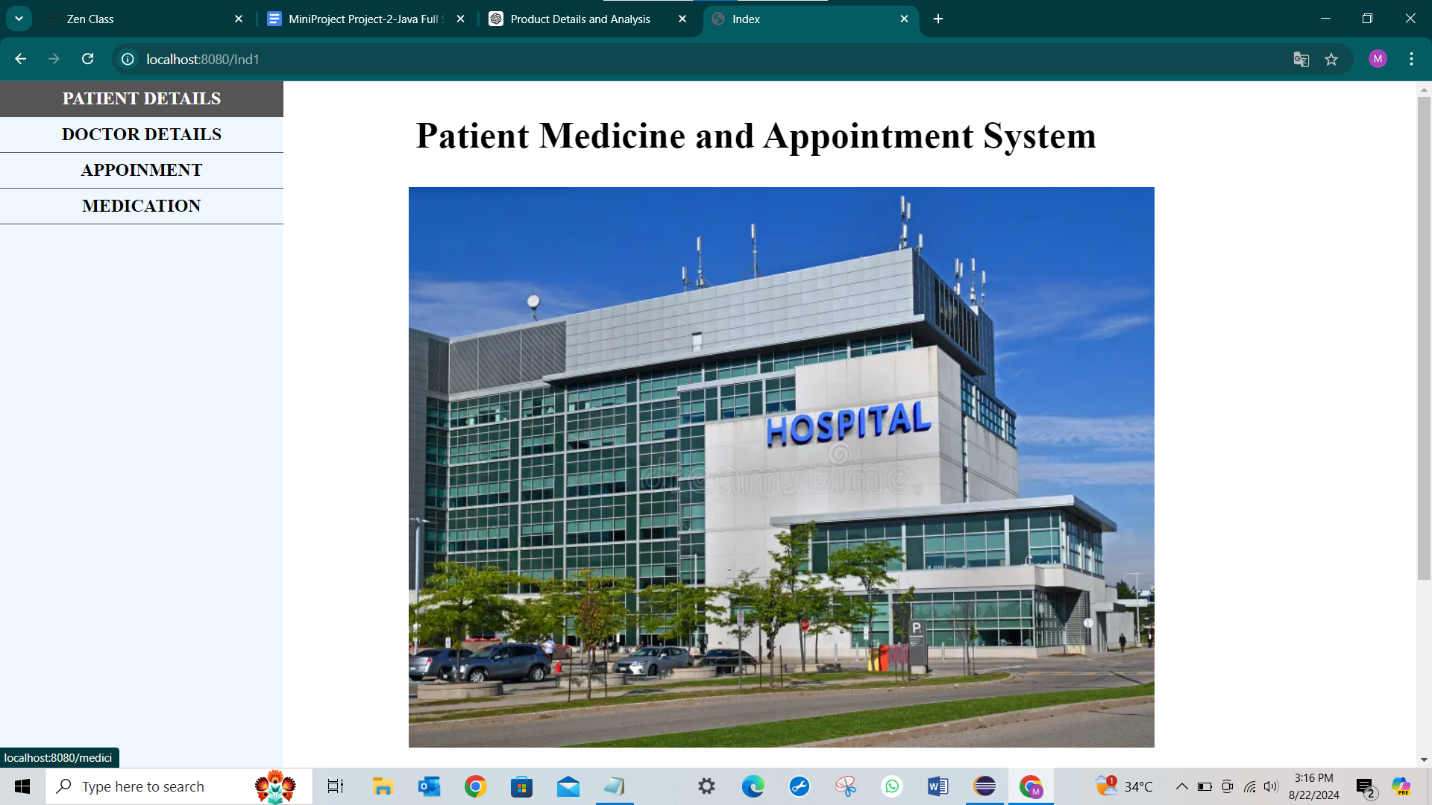
**7.OUPUT:**

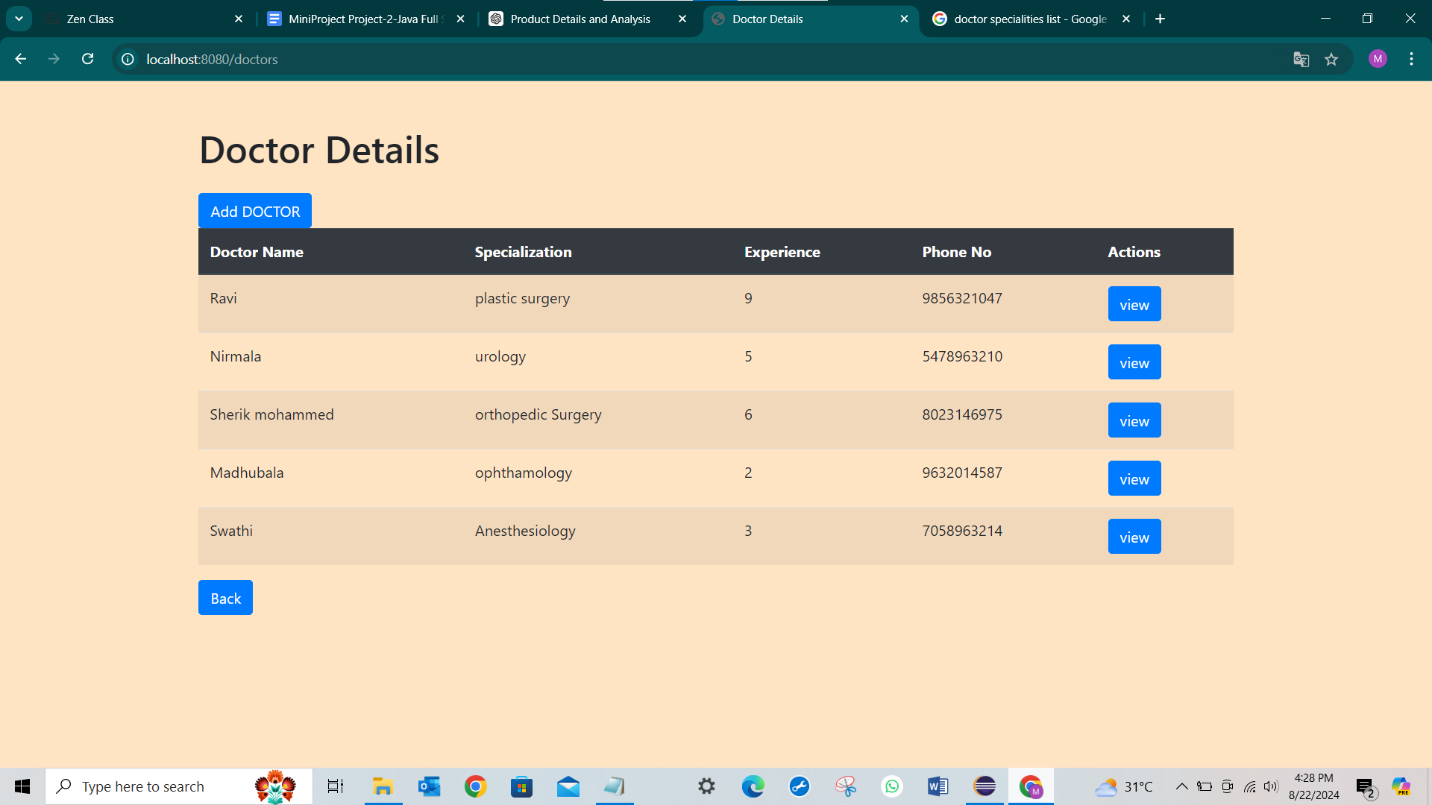
TO run Localhost on the Project.

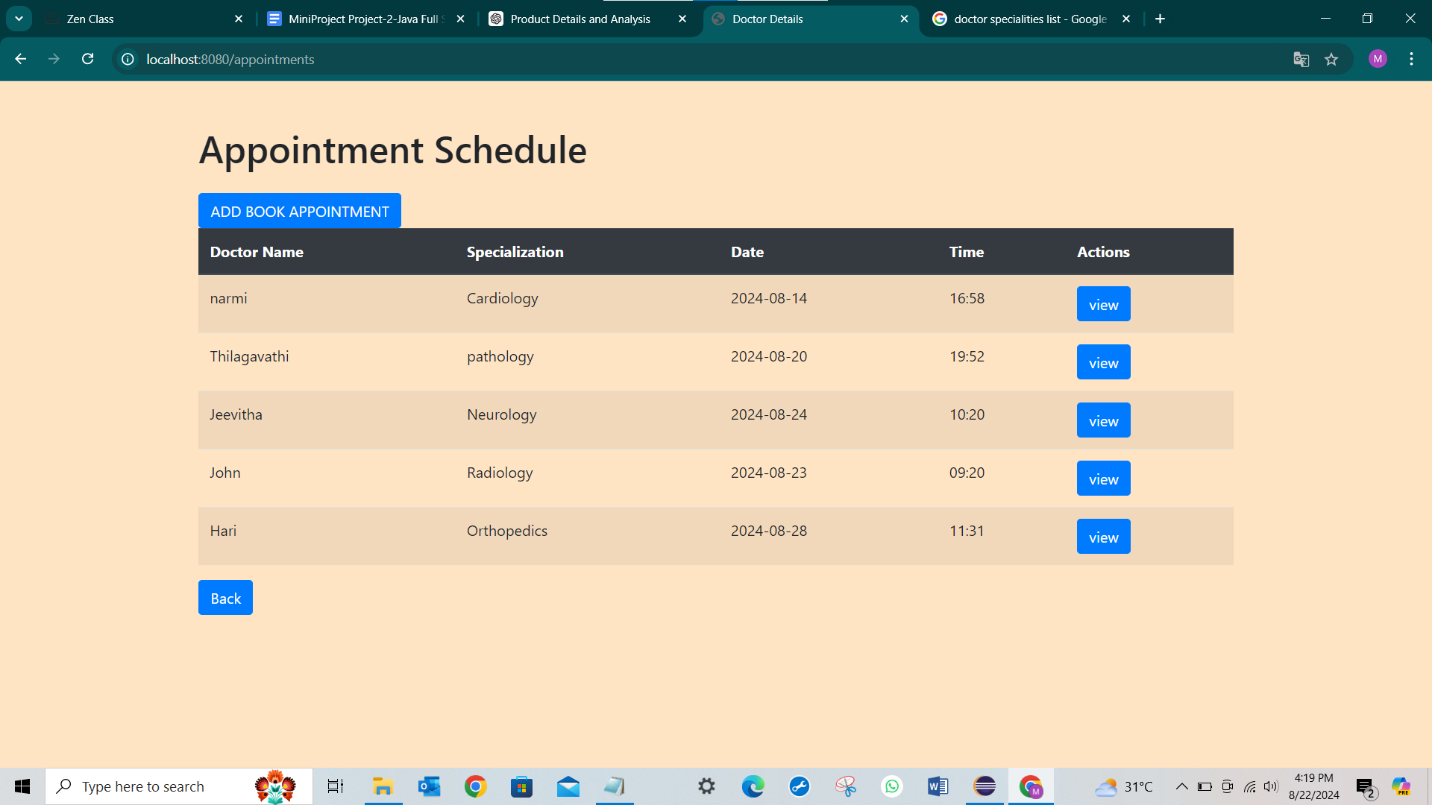


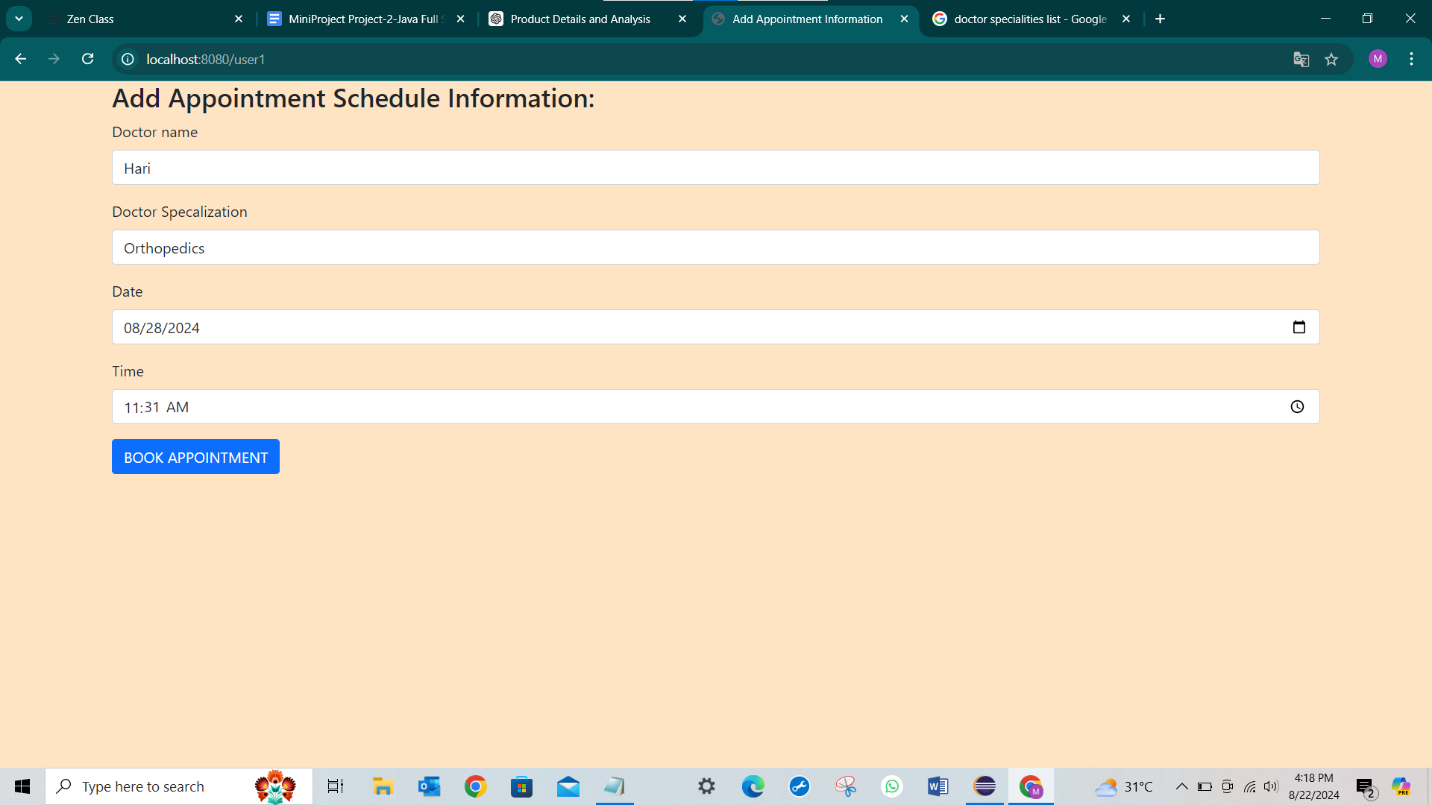


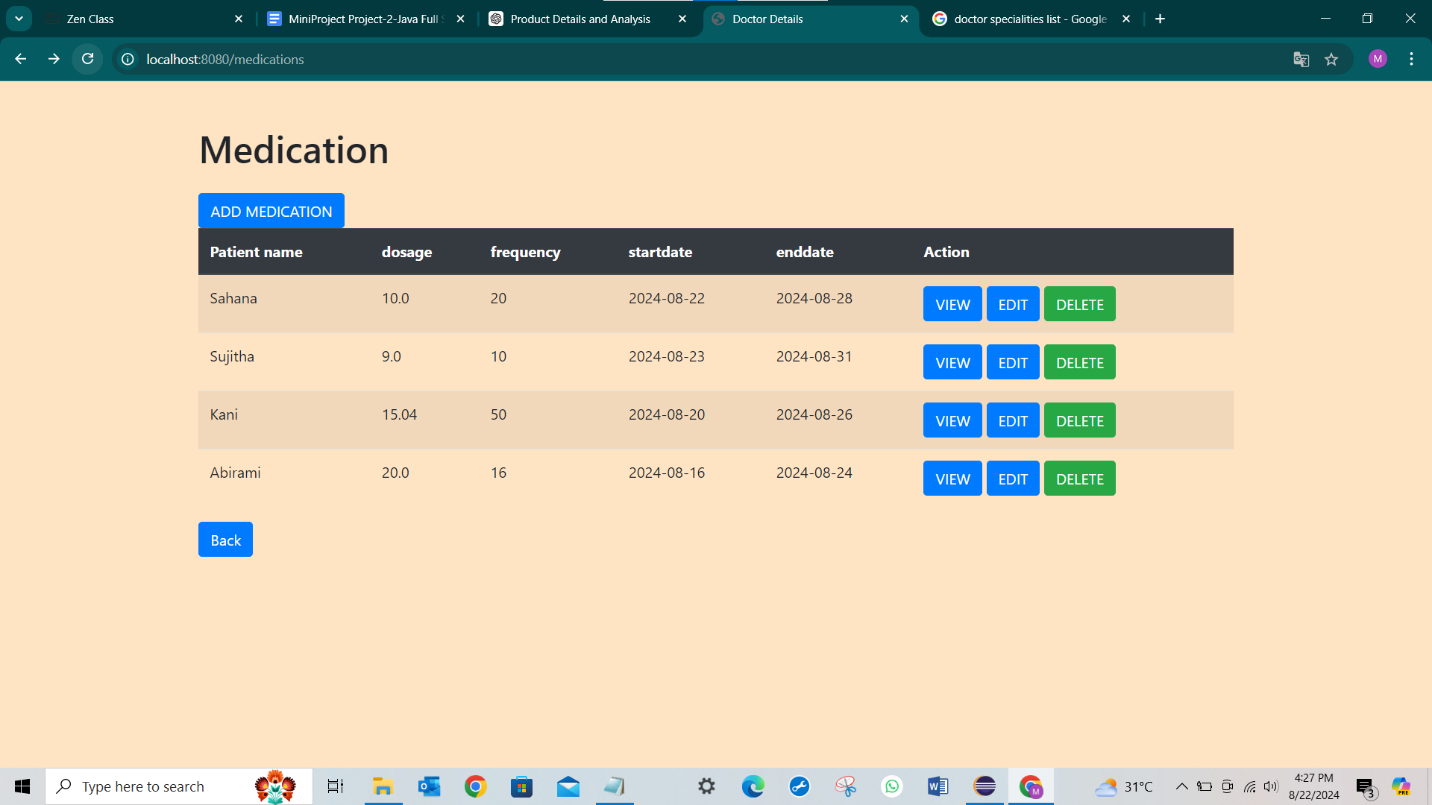


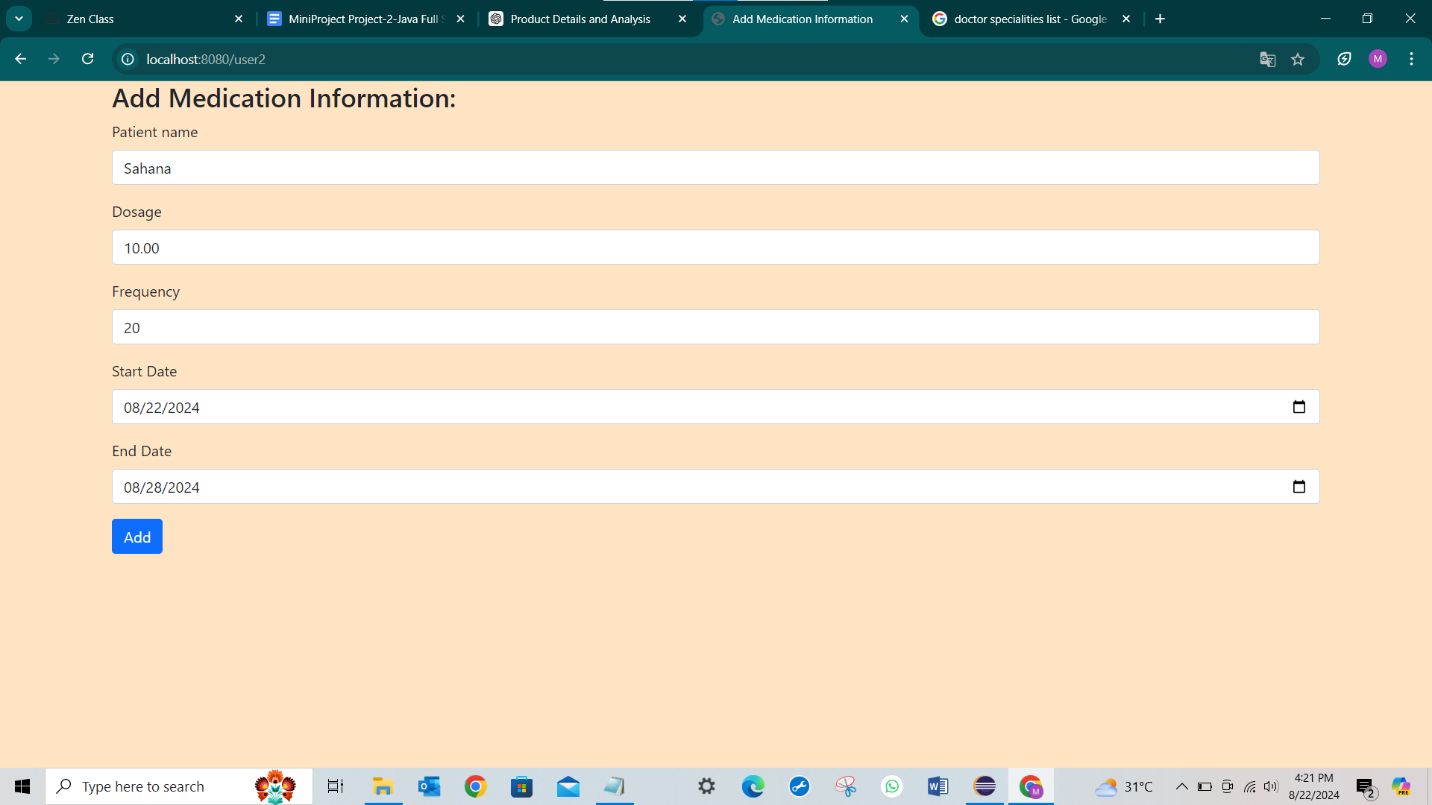












**8.Conclusion:**

The Patient Medicine and Appointment System project successfully integrates both backend and frontend technologies to streamline the process of managing patient information, scheduling appointments, and handling medication management. By using Spring Boot for backend development and HTML/CSS/Thymeleaf for the frontend, the application provides a robust and scalable solution for patient management.

The project facilitates secure patient registration, effective appointment scheduling, and detailed medication management while ensuring data integrity through well-structured APIs and database models. With the addition of unit testing, the application ensures reliability and performance across different functionalities. Thorough documentation provides clear instructions for setup, usage, and customization, making this system easy to implement and extend for future healthcare solutions.