WIPRO NGA Program – EP JAVA FULL STACK Batch (24NAG0970)

Capstone Project Presentation – 11 JULY 2024

Project Title Here – HealthCare - Hospital Management System

Presented by –Md Yaseen
Nikhil Kumar Saw
Mehul Tukaram Anjikhane
Munnangi Naga Naveena

Introduction

Wipro HealthCare is a comprehensive healthcare management system designed to streamline and enhance the management of healthcare services. The project leverages modern technologies to create a robust, scalable, and user-friendly application. The backend is developed using Java Spring Boot with a Maven structure, ensuring modularity and ease of maintenance. The database is managed using MySQL, providing a reliable and efficient storage solution. The frontend is built with Angular 15, incorporating Bootstrap for responsive and aesthetically pleasing UI design.

Technologies Used

- Backend: Java Spring Boot
- Frontend: Angular 15, Bootstrap
- Database: MySQL
- Build Tool: Maven
- Authentication: JWT (JSON Web Tokens).

Project Structure

The project is organized into several key modules, each focusing on a specific aspect of healthcare management. The modular approach ensures that each component can be developed, tested, and maintained independently, while still integrating seamlessly with the overall system.

Backend: Java Spring Boot

The backend of the Wipro HealthCare application is developed using Spring Boot, a powerful framework that simplifies the development of production-ready applications. The Maven structure is used for project management, ensuring that dependencies are managed efficiently.

Key Features

Spring Data JPA: Facilitates interaction with the MySQL database using JPA (Java Persistence API).
Spring Security: Provides authentication and authorization capabilities, ensuring that only authorized users can access the system.
JWT (JSON Web Token): Used for secure authentication. Tokens are generated upon successful login and are used to validate user sessions.
RESTful APIs: Exposes various endpoints for CRUD operations on different entities such as doctors, patients, and appointments.

Dependencies (pom.xml)
The pom.xml file defines the dependencies required for the project. Key dependencies include:
spring-boot-starter-data-jpa
spring-boot-starter-web
spring-boot-starter-security
mysql-connector-j
jjwt-api, jjwt-impl, jjwt-jackson for JWT authentication
lombok for reducing boilerplate code

Controllers Overview
The controllers in the Wipro HealthCare system play a crucial role in defining and managing the endpoints for interacting with different parts of the application. Each module has its own controller, responsible for handling HTTP requests and responses, ensuring a clear separation of concerns and modular architecture. Here's a detailed look at each controller and its responsibilities:

AdminController
The AdminController manages admin-related functionalities and serves as a central point for administering the system. Key responsibilities include:
Authentication: Provides endpoints for admin login and token generation using JWT, ensuring secure access to the system.
Admin CRUD Operations: Allows for the creation, updating, fetching, and deletion of admin records.
Doctor Management: Facilitates CRUD operations for doctor records, enabling admins to add, update, view, and delete doctor details.
Patient Management: Similar to doctor management, it allows CRUD operations on patient records, ensuring admins can manage patient data effectively.

DoctorController
The DoctorController is responsible for managing operations related to doctors. Its key functionalities include:
View Doctor Details: Provides endpoints for viewing detailed information about specific doctors.
Manage Appointments: Handles endpoints for doctors to view all appointments, upcoming appointments, and specific appointment details.
Consultations: Allows doctors to conduct and update consultations, ensuring that patient interactions and medical advice are properly recorded.
Medical Records: Provides endpoints for doctors to update patient medical records, maintaining an accurate health history.

PatientController
The PatientController focuses on managing patient-related operations. Key responsibilities include:
Patient Registration: Provides endpoints for registering new patients, ensuring that all necessary patient details are captured.
Update Patient Details: Allows for updating patient information, ensuring records are kept current.
Doctor Interaction: Enables patients to view doctor profiles and select doctors for their healthcare needs.
Appointment Management: Allows patients to view the status of their appointments and interact with the appointment system.

AppointmentController
The AppointmentController handles all aspects of appointment management. Key functionalities include:
Book Appointments: Provides endpoints for patients to book appointments with doctors, ensuring that the booking process is streamlined.
Update Appointments: Allows for the modification of existing appointments, accommodating changes in schedules.
View Appointment Details: Enables users to fetch details of specific appointments, ensuring clarity and proper planning.
Cancel Appointments: Provides functionality for canceling appointments, ensuring that both doctors and patients can manage their schedules effectively.

Key Functionalities

Each controller leverages Spring Boot's capabilities to provide a RESTful API, ensuring that the system is easy to interact with via HTTP requests. This modular approach not only simplifies development and maintenance but also enhances the scalability and flexibility of the application.

Integration with Other Components

The controllers interact with service layers that encapsulate the business logic. They also utilize repositories for data access, ensuring a clean separation of concerns and adherence to best practices in software design. The use of annotations like @Autowired for dependency injection, @PostMapping, @PutMapping, @GetMapping, and @DeleteMapping for defining endpoints, ensures that the code is concise and easy to manage.

Security and Authentication

Security is a critical aspect managed within the controllers, especially in the AdminController and DoctorController, where JWT tokens are generated and validated. This ensures that only authenticated users can access sensitive endpoints, maintaining the integrity and security of the system.

Angular Frontend

The frontend of Wipro HealthCare is developed using Angular 15. Angular provides a robust framework for building dynamic, single-page web applications. Bootstrap is used to ensure a responsive and visually appealing design.

Components

Each module has its own set of Angular components:

Doctor Module: Components for viewing doctor profiles, managing consultations, and updating medical records.

Admin Module: Components for admin management, including user registration, updates, and listing doctors and patients.

Patient Module: Components for patient registration, viewing doctor profiles, and appointment management.

Appointment Module: Components for booking, updating, and viewing appointments.

Services

Angular services are used to handle business logic and communicate with the backend APIs. Each module has its own service class:

DoctorService: Interacts with the backend for doctor-related operations.

AdminService: Manages admin functionalities including authentication and CRUD operations.

PatientService: Handles patient-related operations.

AppointmentService: Manages appointment bookings and updates.

Models

Models are defined for each entity to ensure type safety and clarity in data handling:
Doctor: Represents doctor entities.
Admin: Represents admin entities.
Patient: Represents patient entities.
Appointment: Represents appointment entities.

Routing

Angular routing is configured to handle navigation between different components. Each module has its own routes, and a shared home page acts as the dashboard.

Homepage and Dashboard

The homepage serves as the dashboard for the application, providing an overview and quick access to various functionalities. It includes links to different modules, enabling users to navigate easily to their desired section.

JWT Authentication

JSON Web Tokens (JWT) are used for secure authentication in the Wipro HealthCare system. When a user logs in, a JWT is generated and sent to the client. This token is then included in the header of subsequent requests to authenticate the user. JWTs are stateless and can be validated without storing session information on the server, making them ideal for scalable web applications.

Modules and Responsibilities:

1.Doctor Module - Handled by Md Yaseen

1. **Functionality**:
    1. **Authentication**: Verify doctor's credentials and generate authentication tokens.
    2. **Doctor Management**: Register, update, and delete doctor profiles.
    3. **Consultations**: Conduct and manage consultations, update medical records, and handle related tasks.
    4. **Appointments**: View all, upcoming, and specific appointments.

2.Admin Module - Handled by Nikhil

1. **Functionality**:
    1. **Authentication**: Admin authentication and token generation.
    2. **Admin Management**: Register and update admin profiles.
    3. **Doctor Management**: Oversee doctor registration, updates, and deletions.
    4. **Patient Management**: Access and manage patient records and profiles.
    5. **Medical Records**: Retrieve and manage patient medical records.

1.Patient Module - Handled by Mehul

1. **Functionality**:
    1. **Authentication**: Verify patient credentials and generate authentication tokens.
    2. **Patient Management**: Register, update, and view patient profiles.
    3. **Doctor Information**: Fetch information about doctors.
    4. **Appointment Status**: View the status of appointments.

2.Appointment Module - Handled by Naveena

1. **Functionality**:
    1. **Appointment Booking**: Schedule and book appointments.
    2. **Appointment Management**: Update, view, and delete appointments.
    3. **Appointment Status**: Provide status updates for booked appointments.

Controller Class Functionalities:

Each module has a dedicated controller class that defines the various REST API endpoints used to perform CRUD (Create, Read, Update, Delete) operations and other essential functionalities.

AdminController:

- **Admin Authentication**: Authenticate admins and generate JWT tokens.
- **Admin Management**: Register and update admin profiles.
- **Doctor Management**: Register, update, get details, and delete doctor profiles.
- **Patient Management**: Get details of all patients and specific patient profiles.
- **Medical Records**: Retrieve medical records of patients.

DoctorController:

- **Doctor Authentication**: Authenticate doctors and generate JWT tokens.
- **Doctor Profile Management**: View doctor's own profile.
- **Appointment Management**: View all, upcoming appointments, and specific appointments by doctor ID.
- **Consultations**: Conduct, update, and delete consultations.
- **Medical Records**: Update medical records.

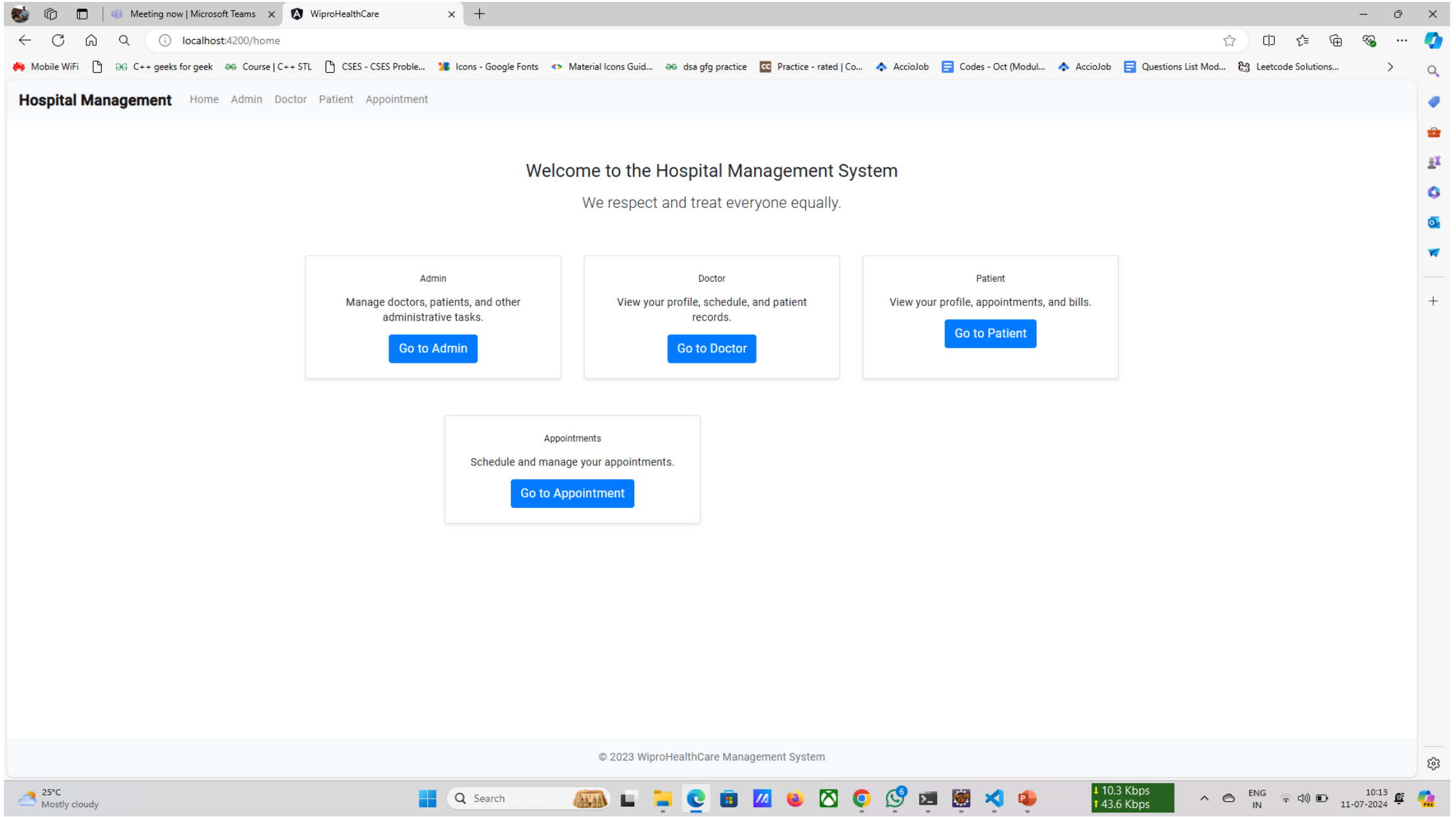
PatientController:

- **Patient Authentication**: Authenticate patients and generate JWT tokens.
- **Patient Profile Management**: Register, update, and view patient profiles.
- **Doctor Information**: Fetch details about specific doctors.
- **Appointment Status**: View status of specific appointments.

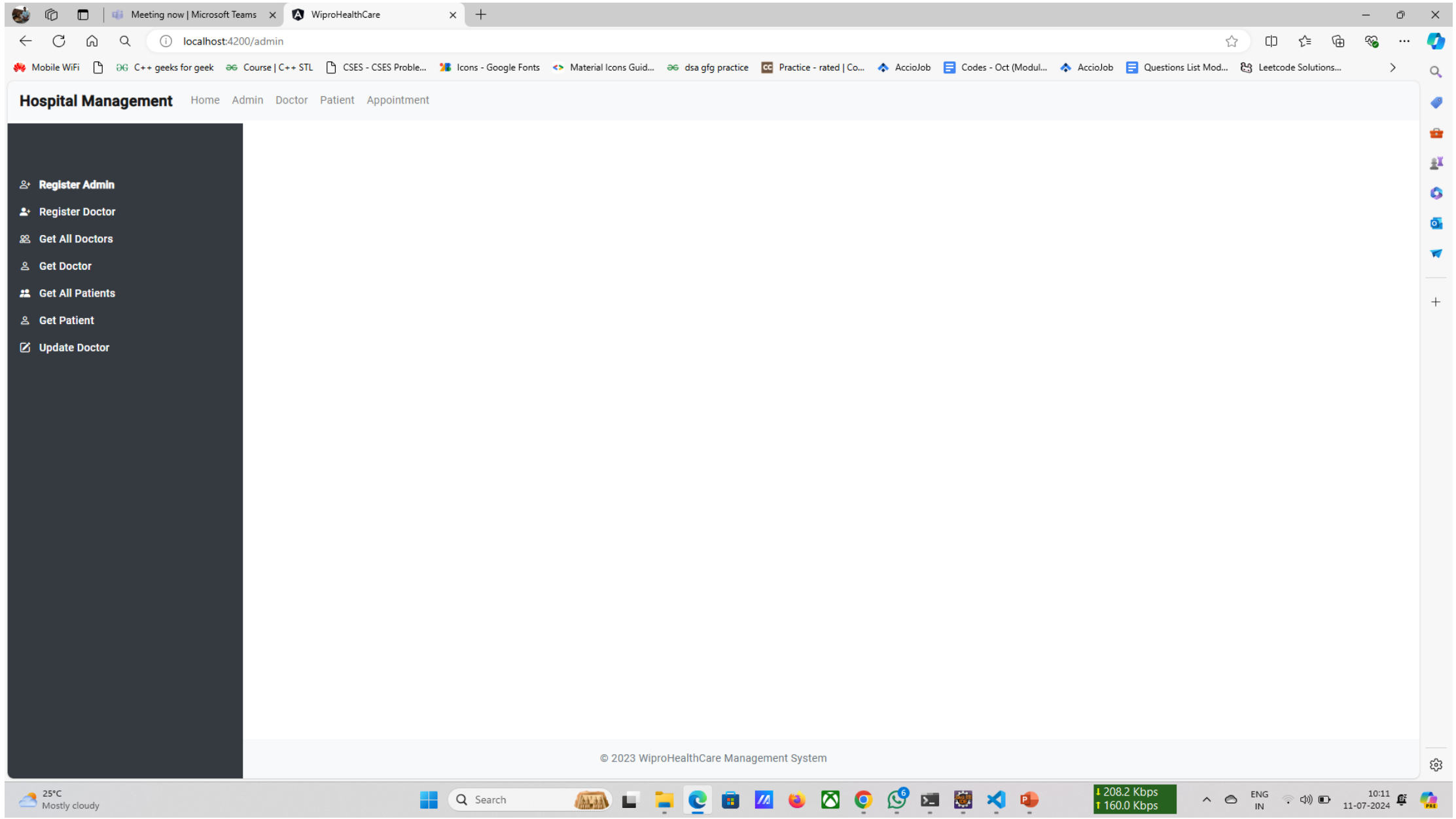
AppointmentController:

- **Appointment Booking**: Book new appointments.
- **Appointment Management**: Update, view, and delete appointments.
- **Appointment Status**: Provide status updates for appointments.

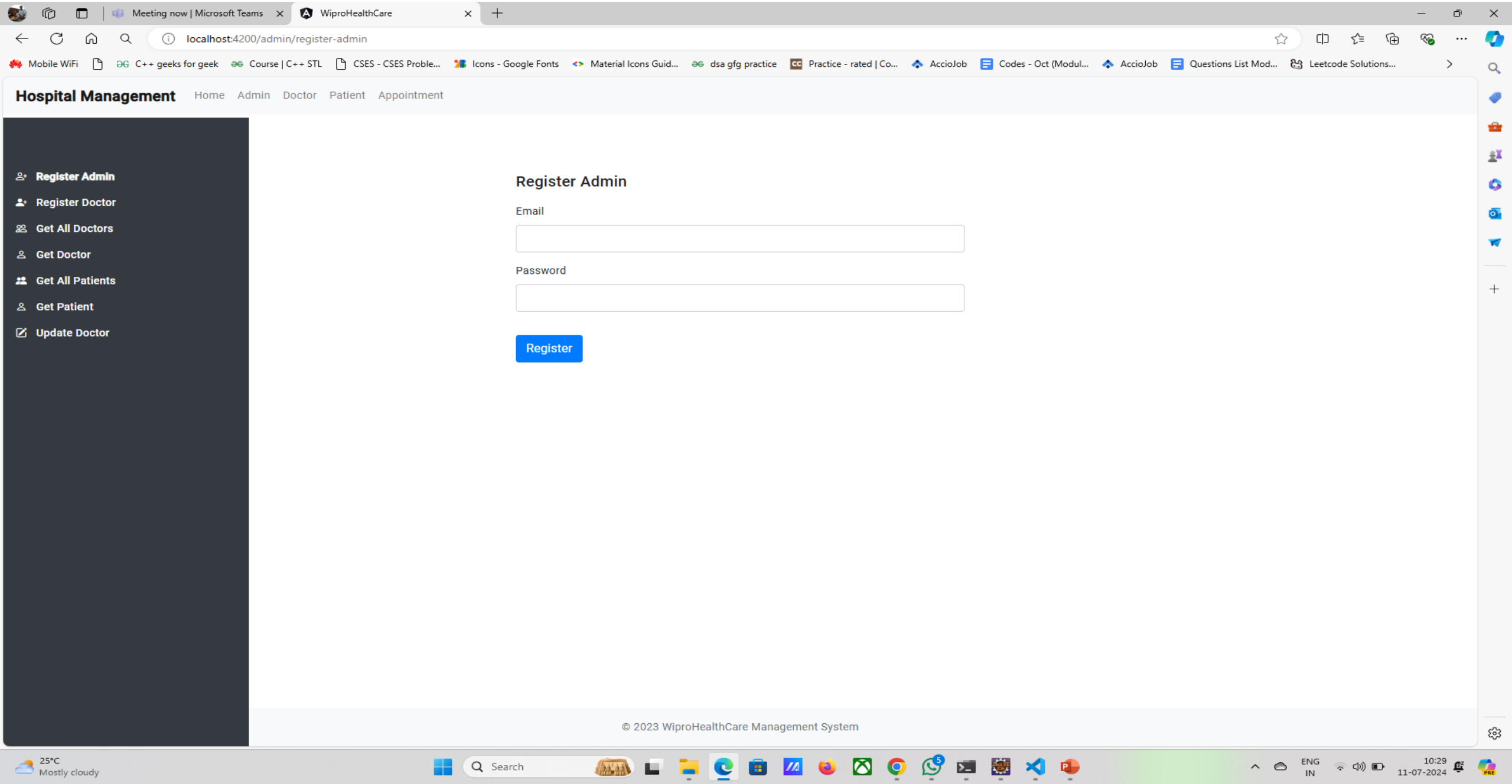
1. Home Page


Meeting now | Microsoft Teams
WiproHealthCare
localhost:4200/home
Hospital Management
Home
Admin
Doctor
Patient
Appointment
Welcome to the Hospital Management System
We respect and treat everyone equally.
Admin
Manage doctors, patients, and other administrative tasks.
Go to Admin
Doctor
View your profile, schedule, and patient records.
Go to Doctor
Patient
View your profile, appointments, and bills.
Go to Patient
Appointments
Schedule and manage your appointments.
Go to Appointment
© 2023 WiproHealthCare Management System

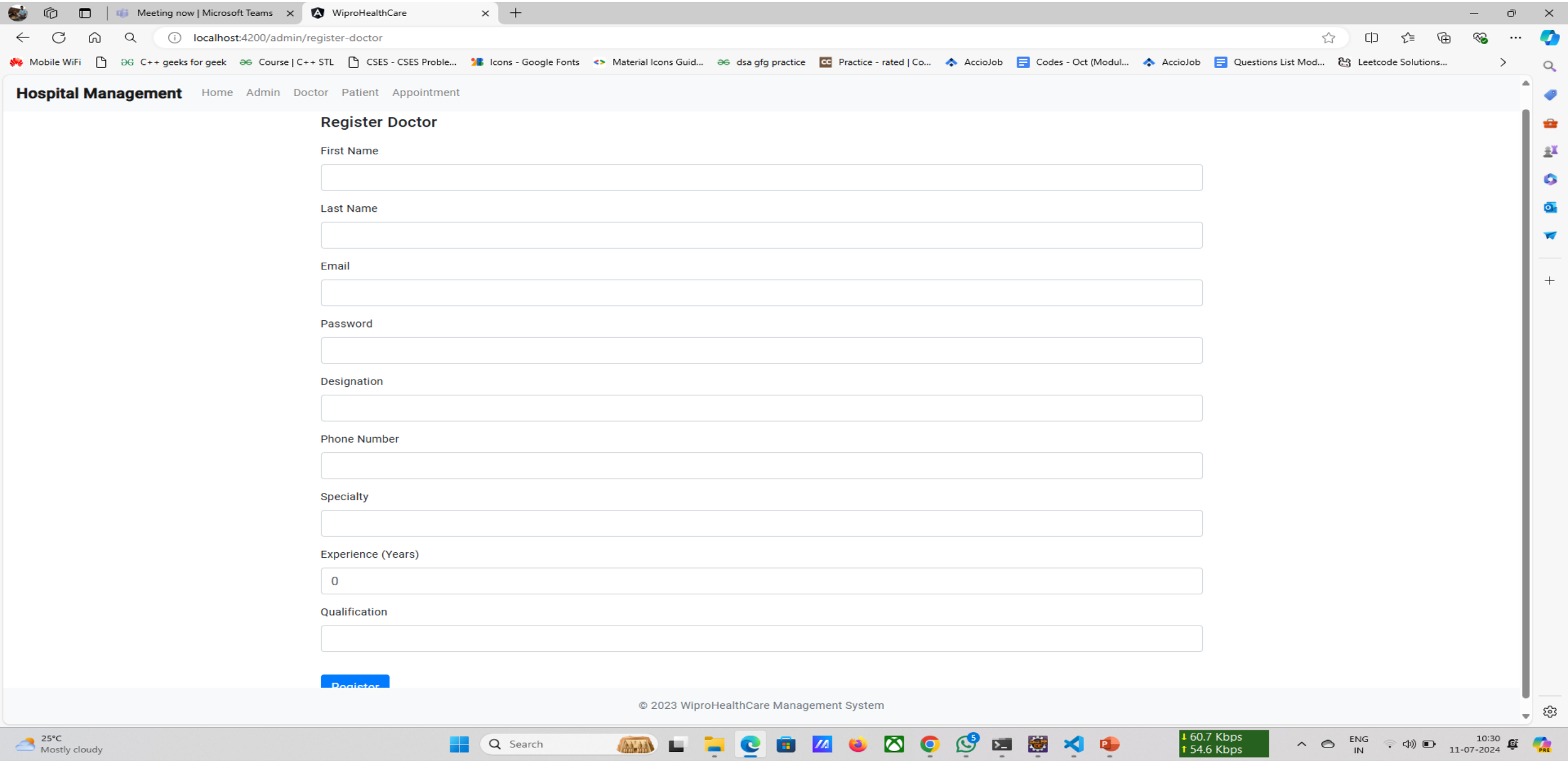
2. Admin Dashboard:



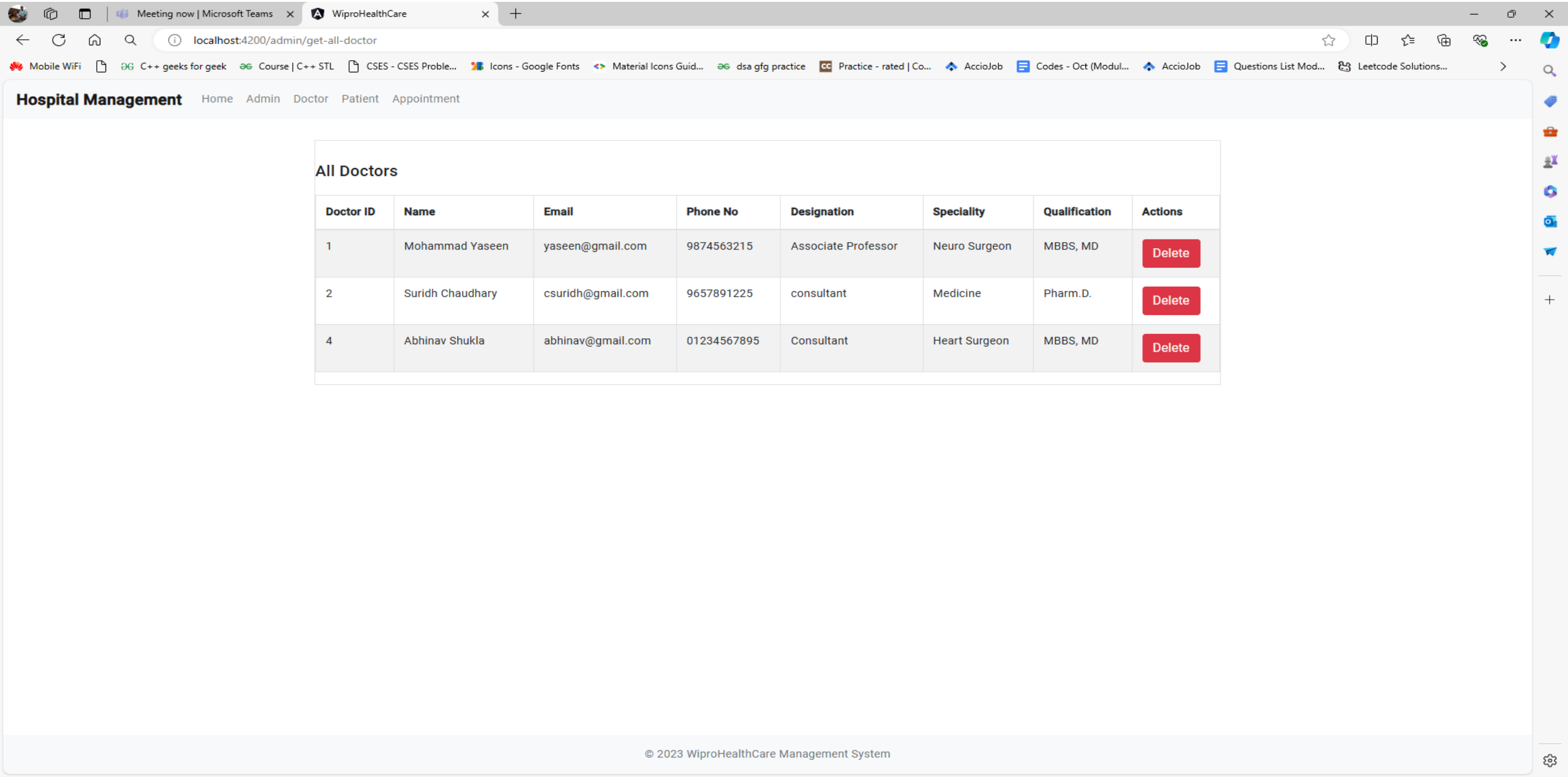
Register Admin



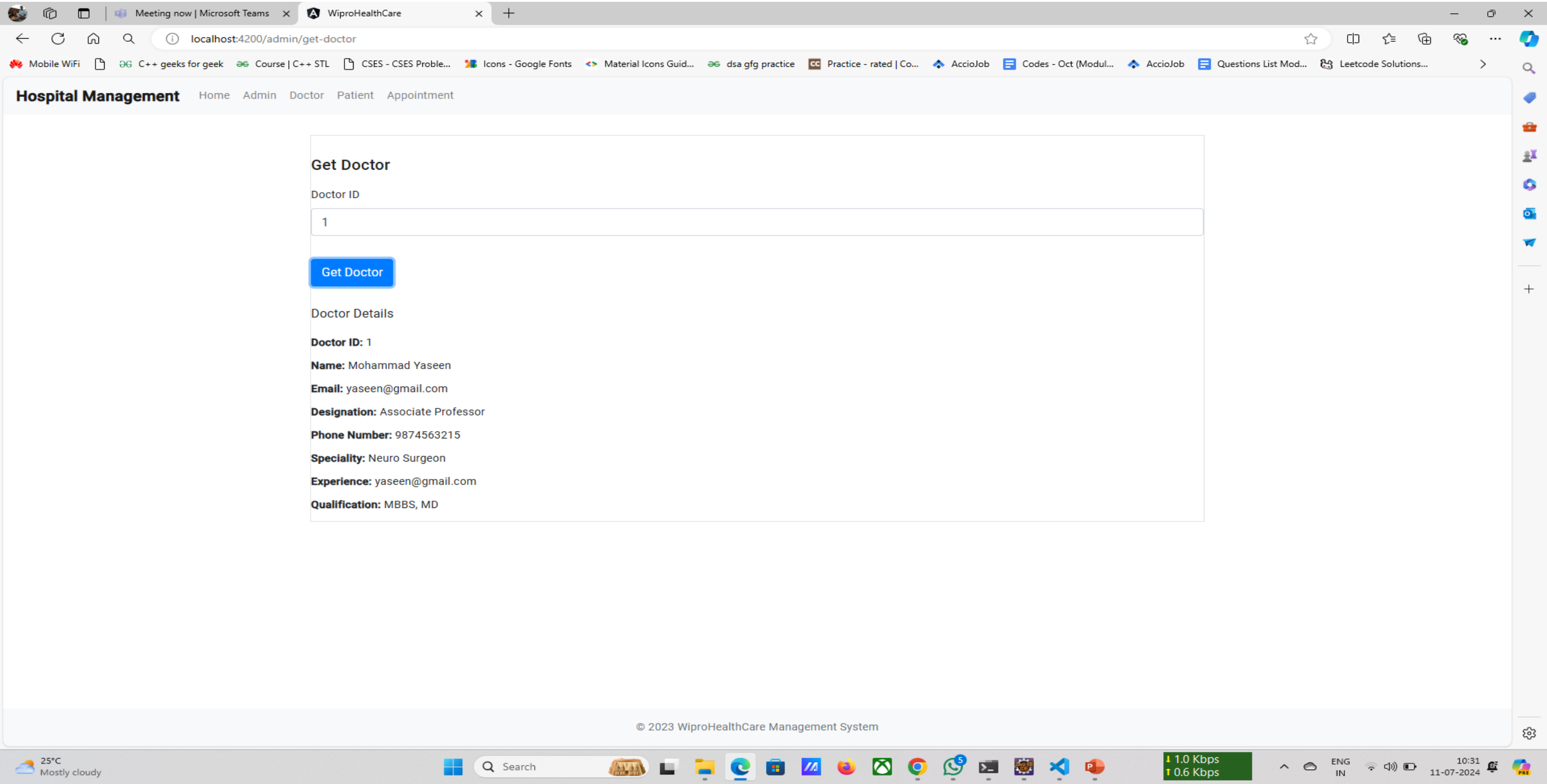
Register Doctor



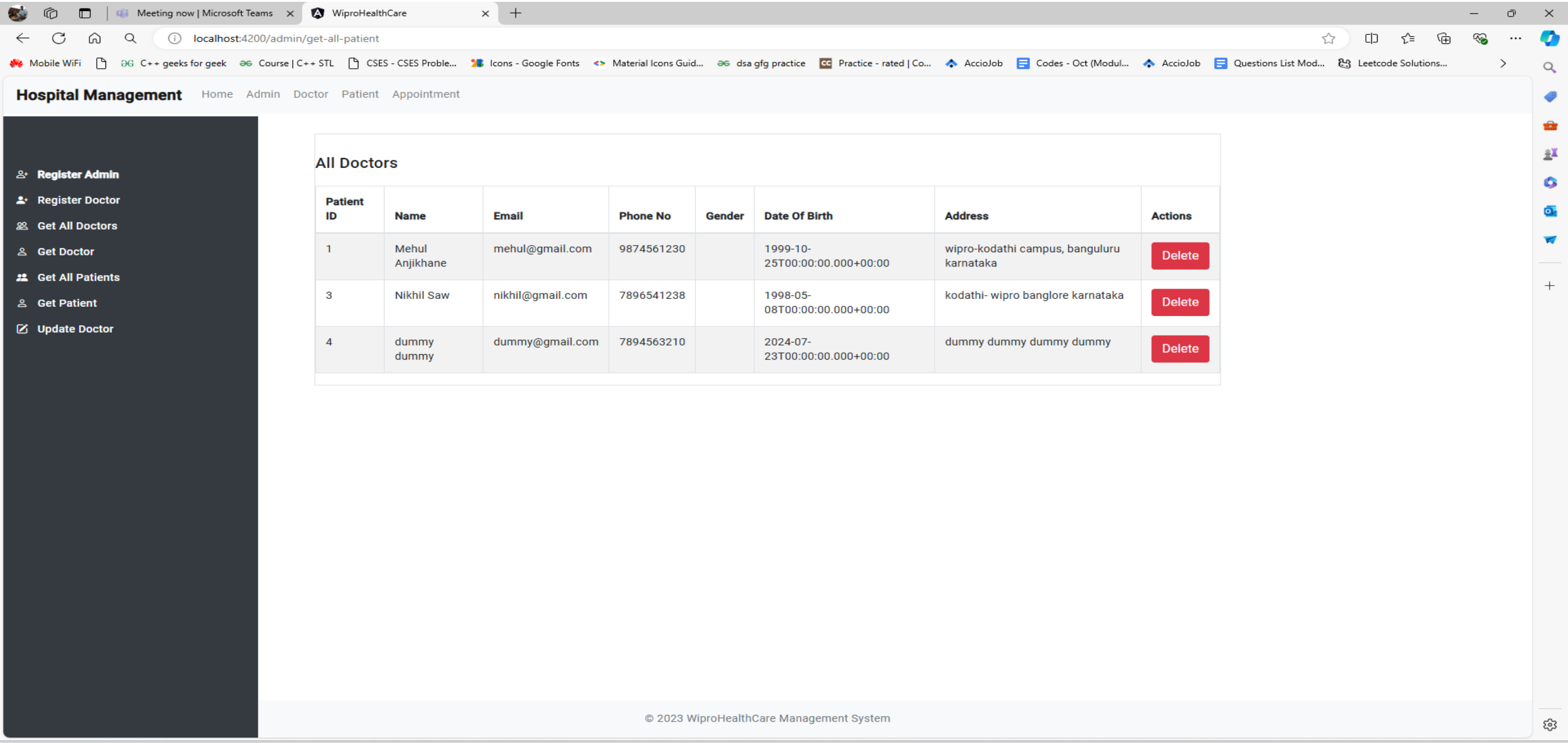
Get All Doctor List



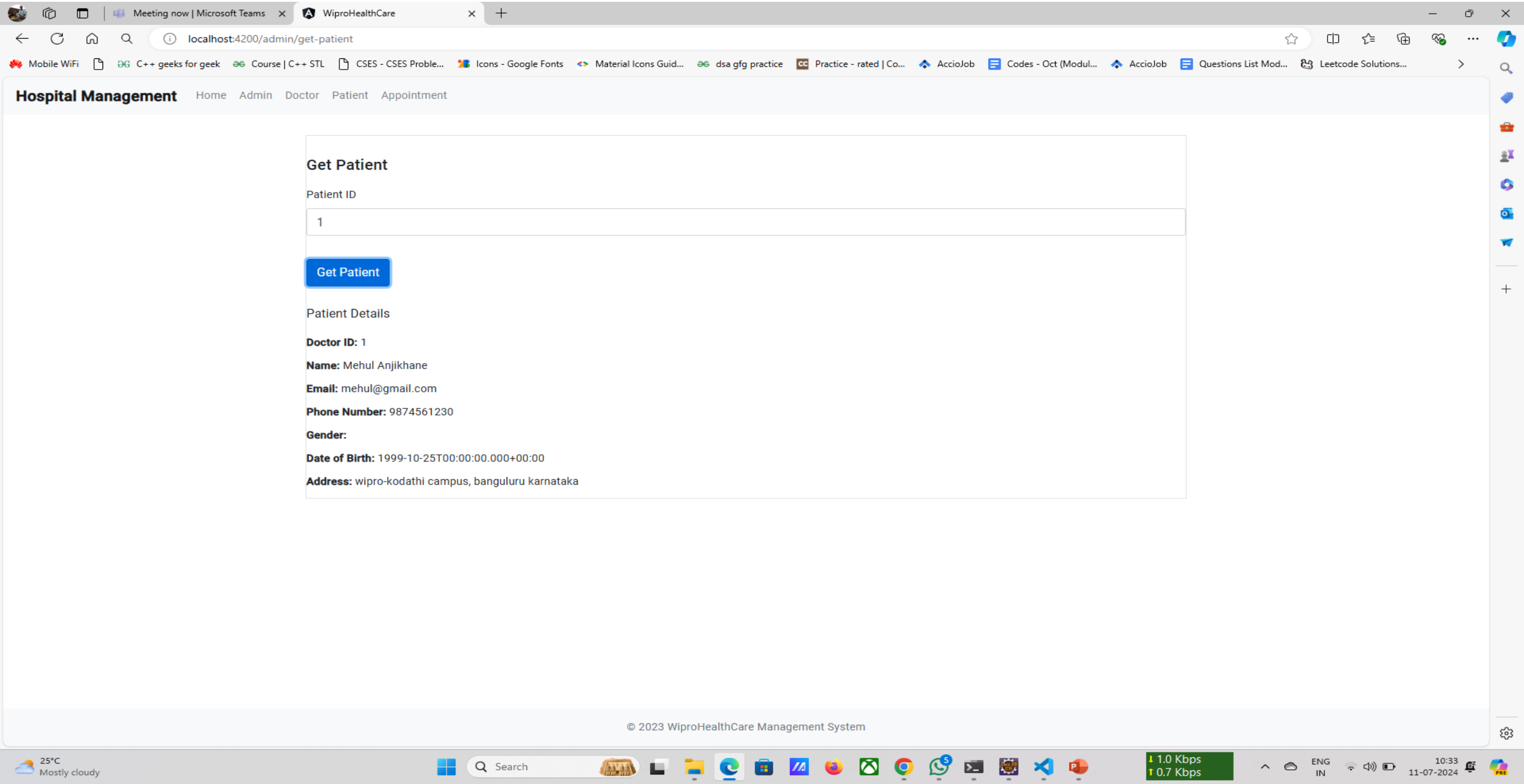
Get Doctor By Id



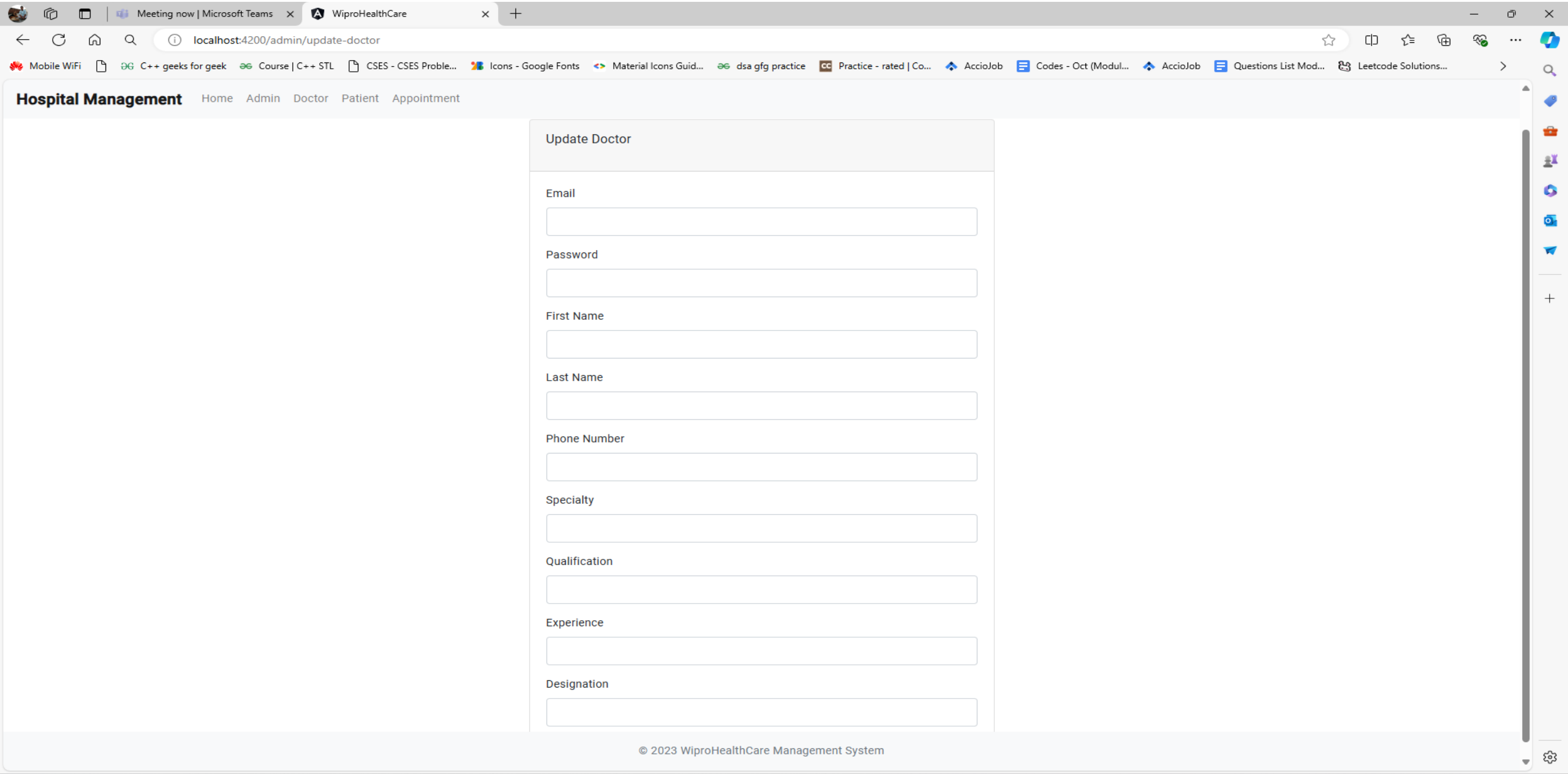
Get All Patient Details



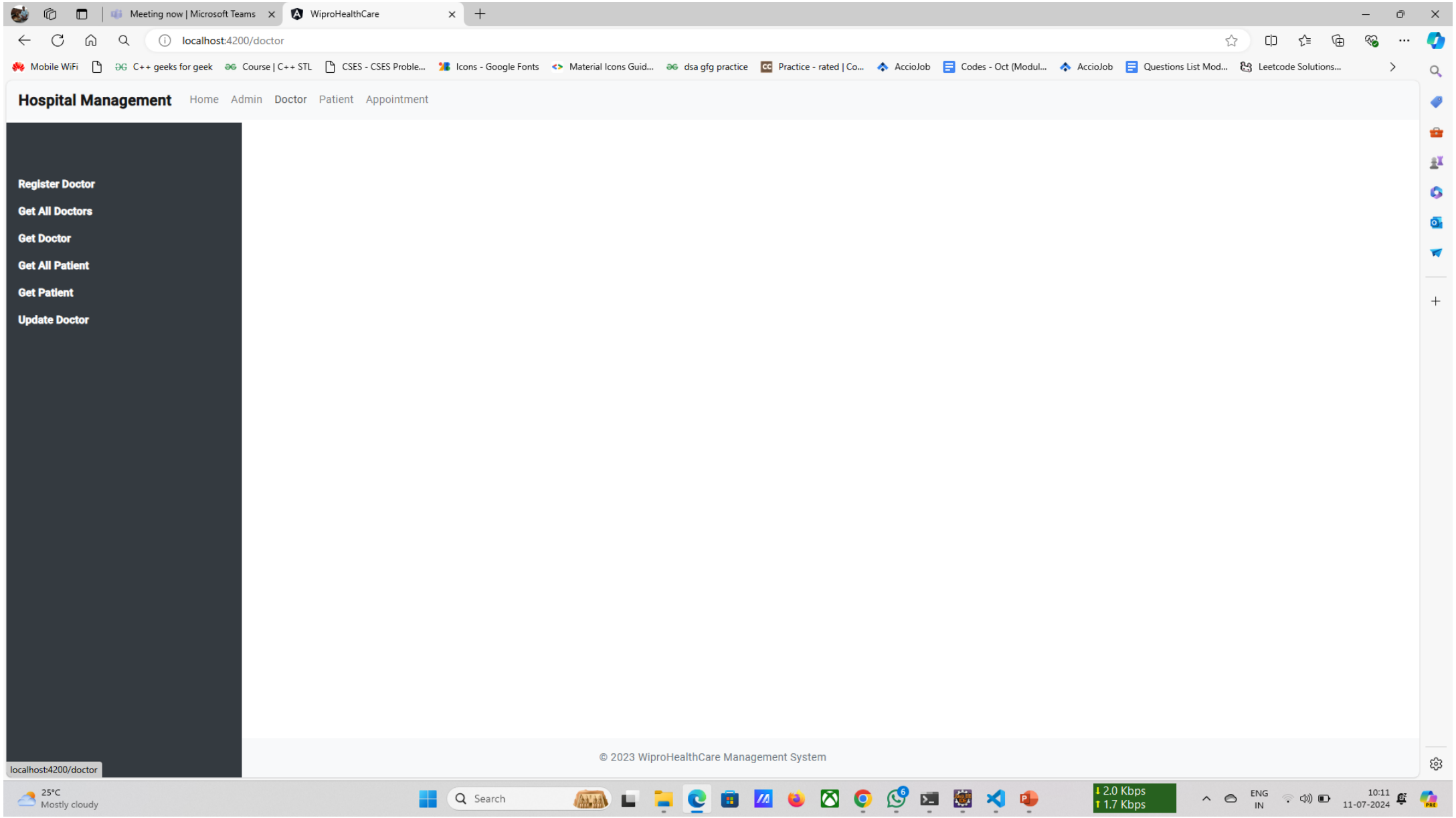
Get Patient By ID



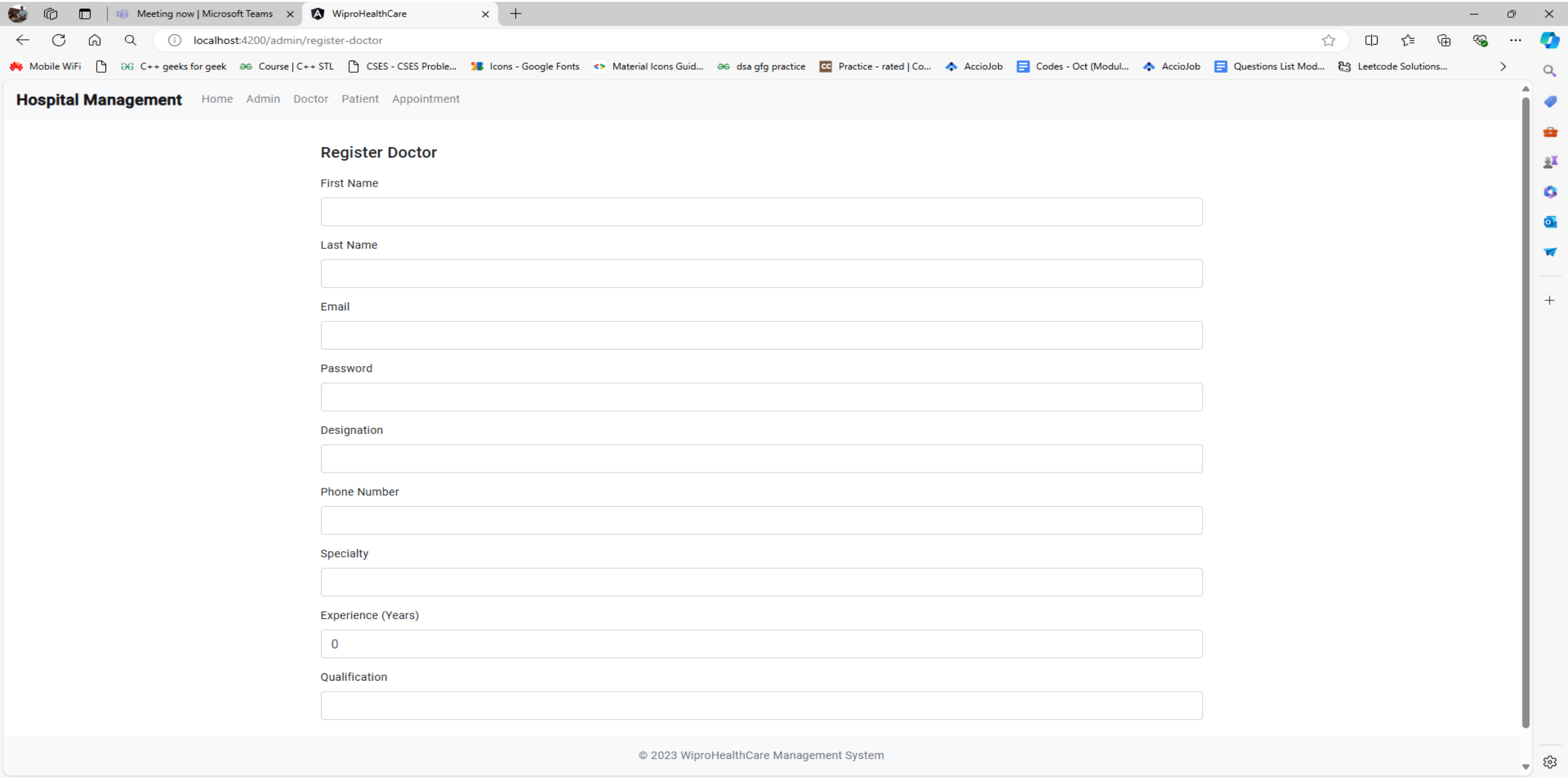
Update Doctor Details



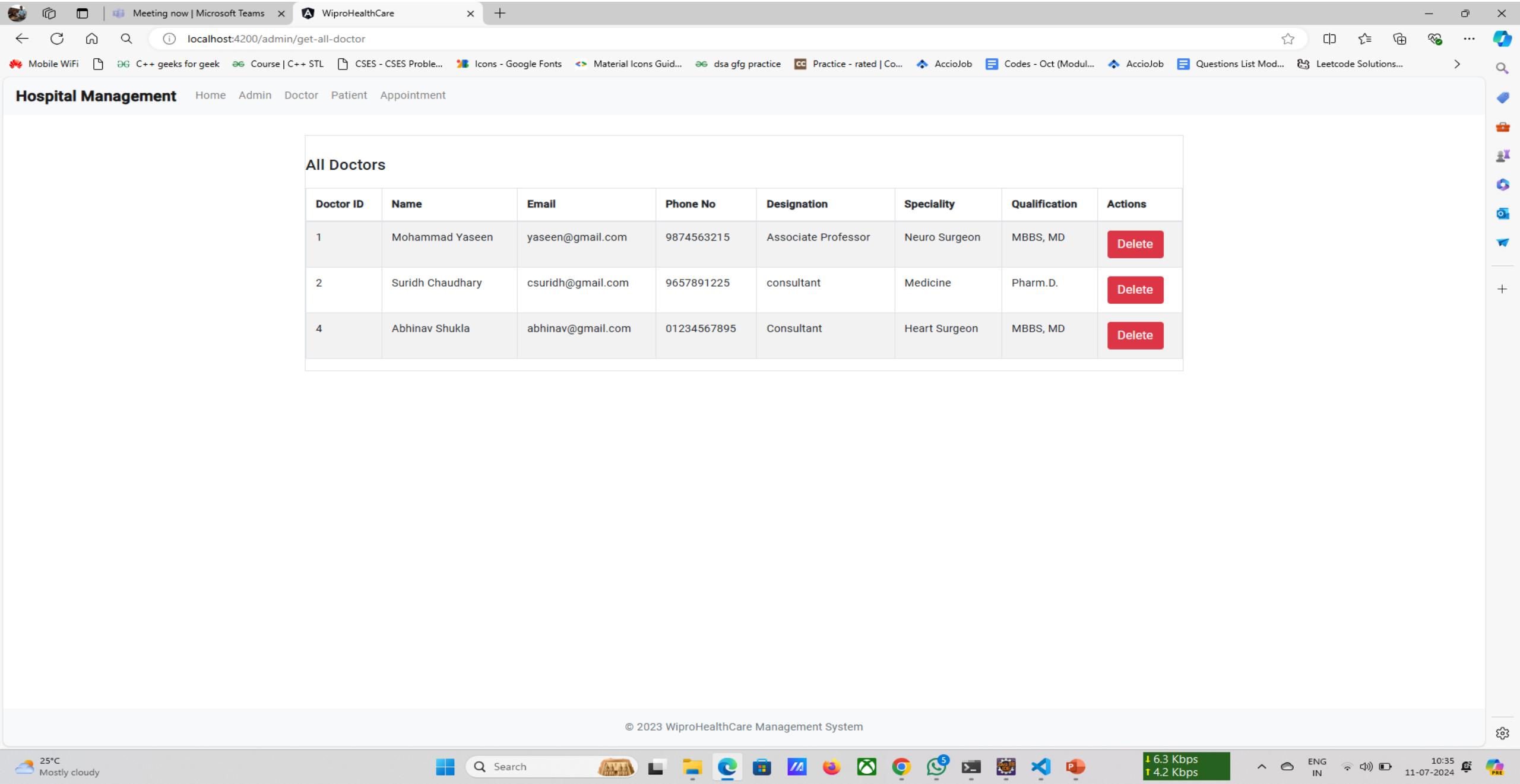
3. Doctor Dashboard:



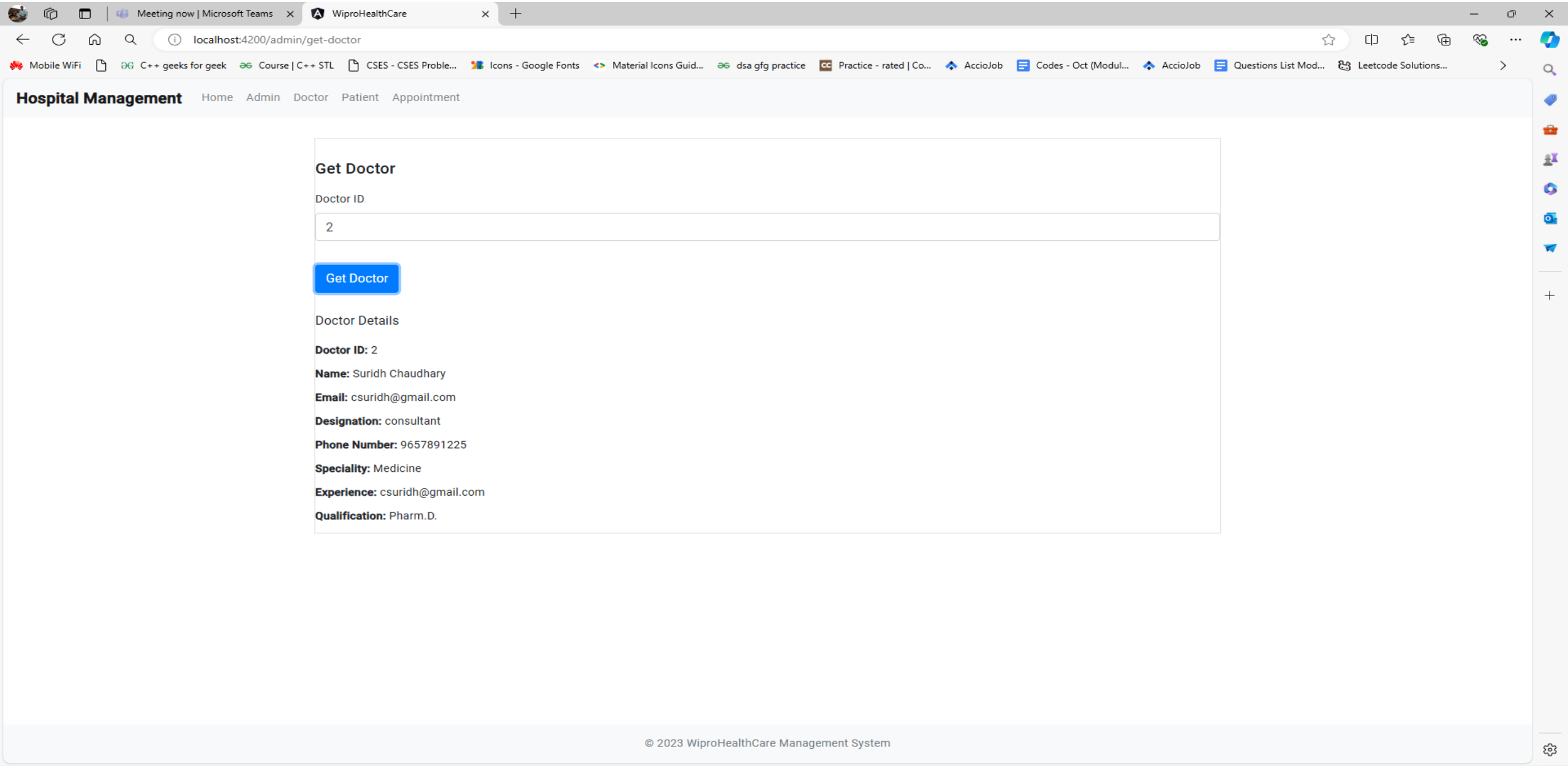
Register Doctor



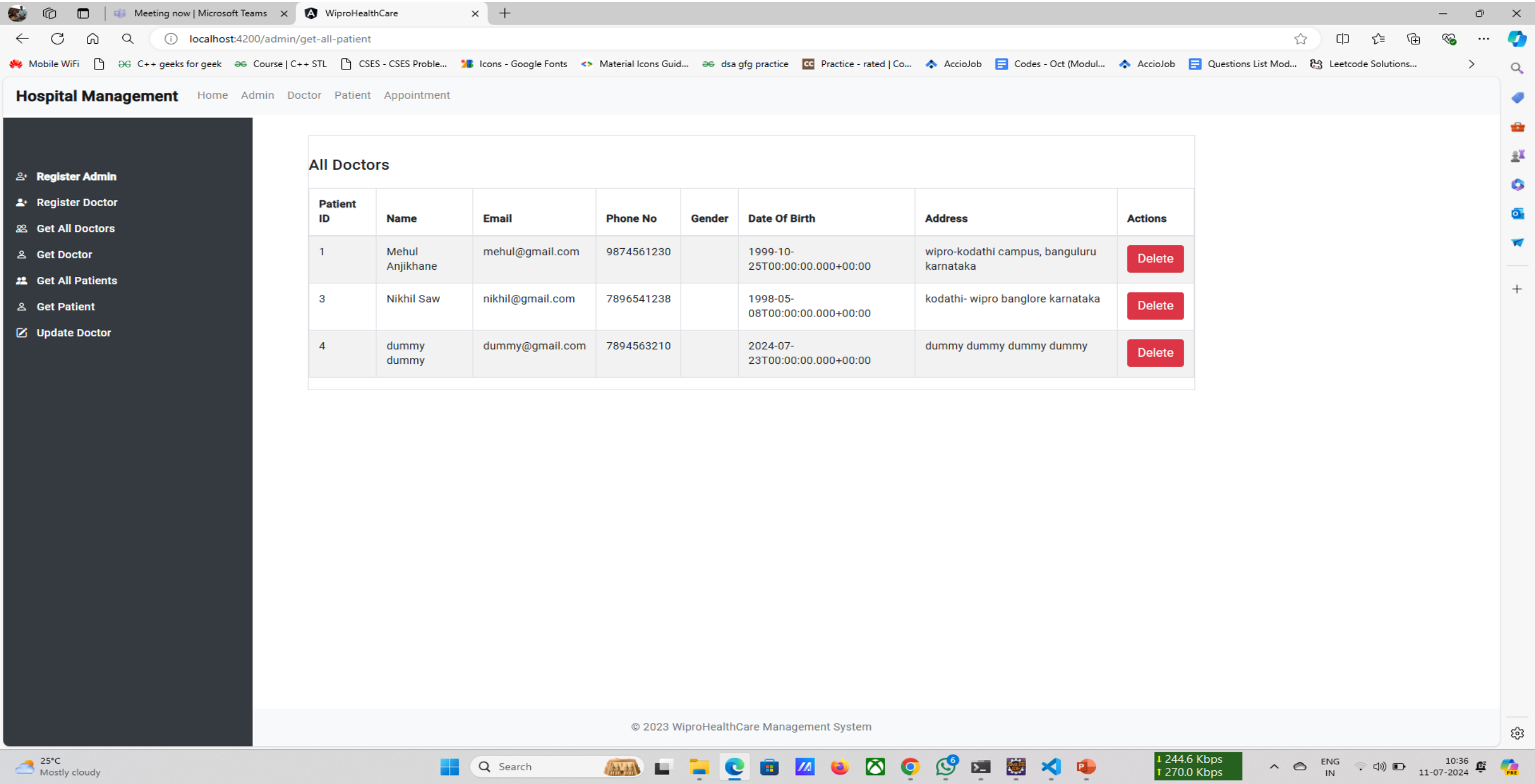
Get All Doctor Details



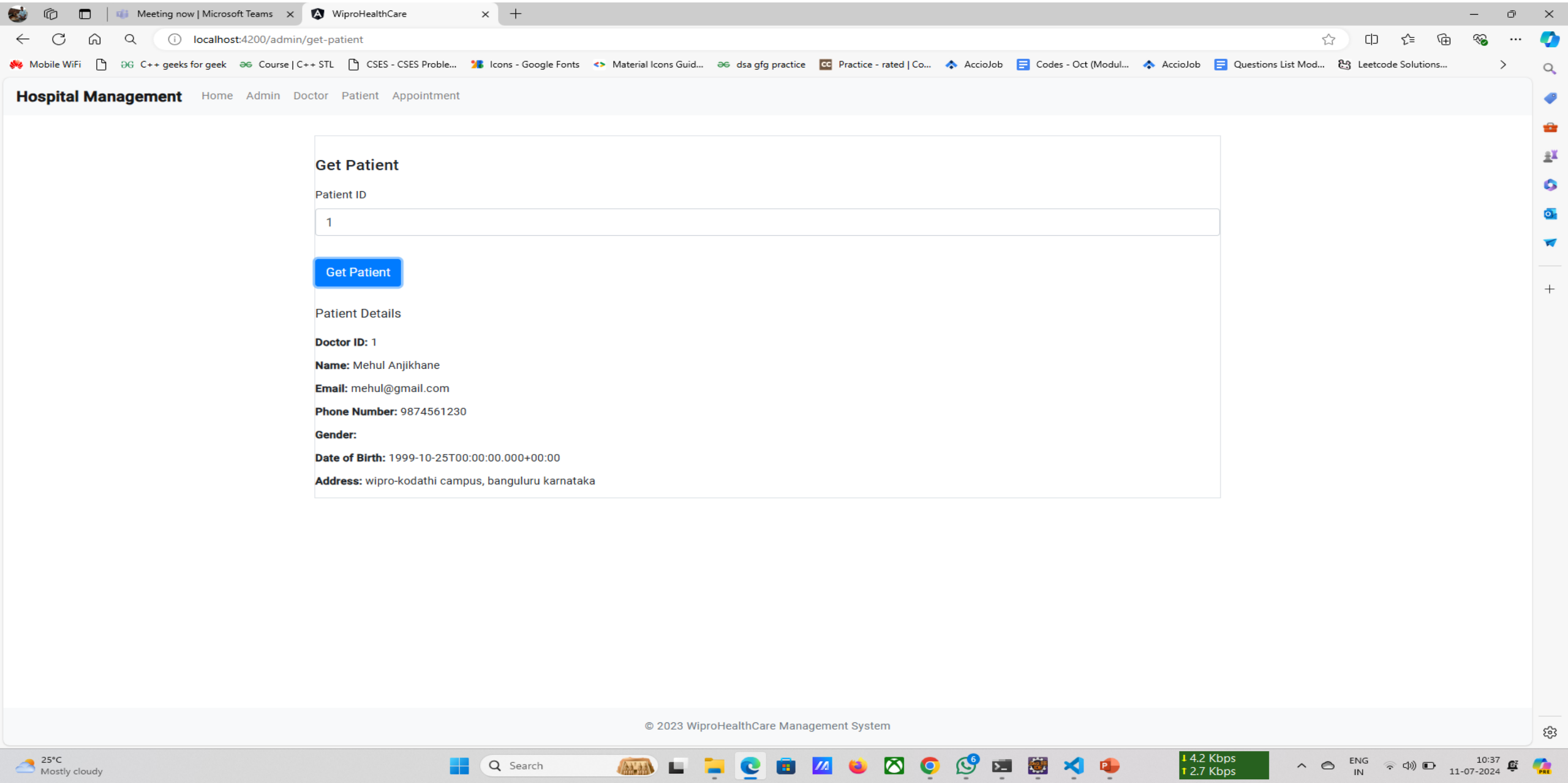
View Doctor By ID



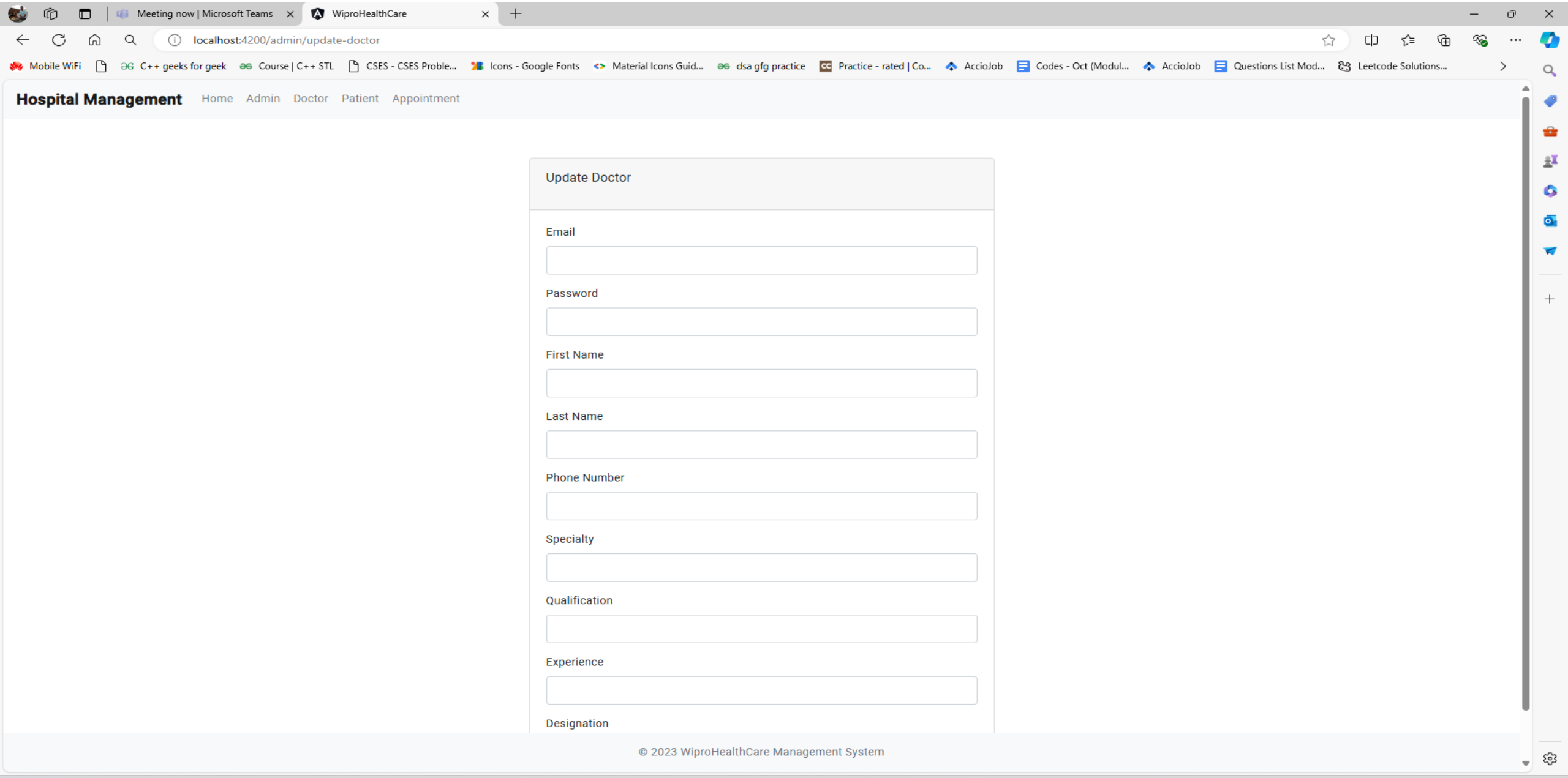
Get All Patient Details



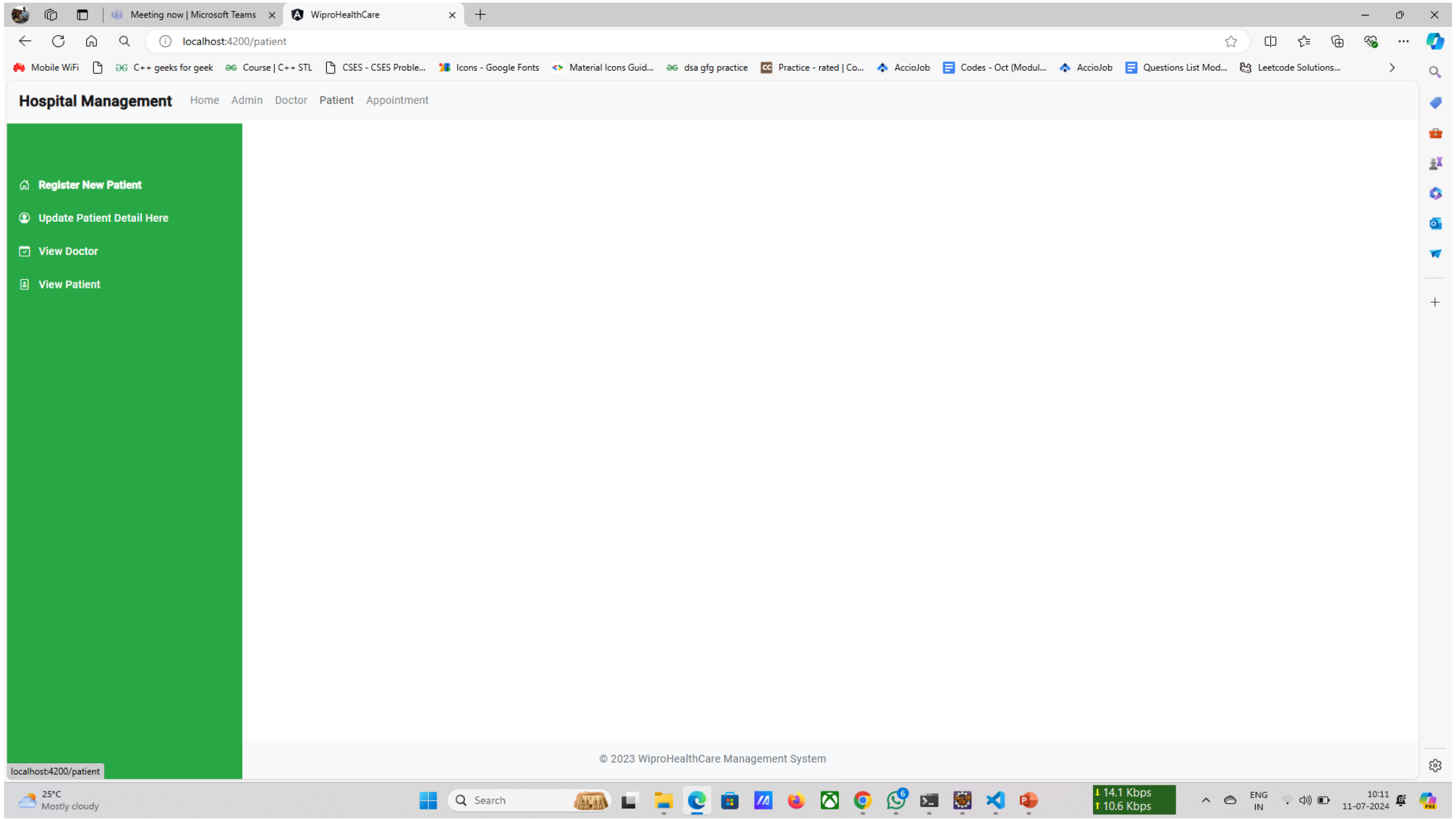
Get Patient By ID



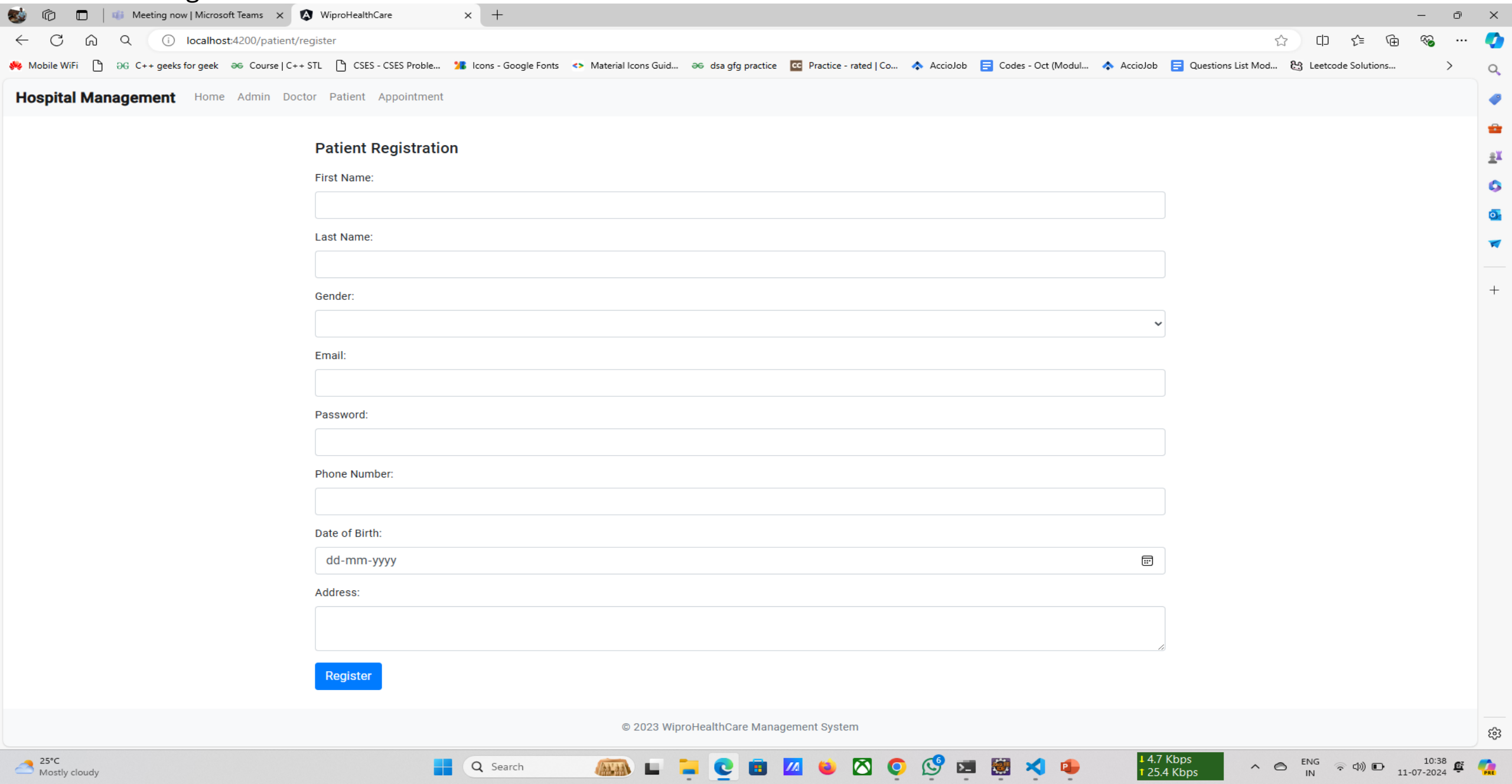
Update Doctor



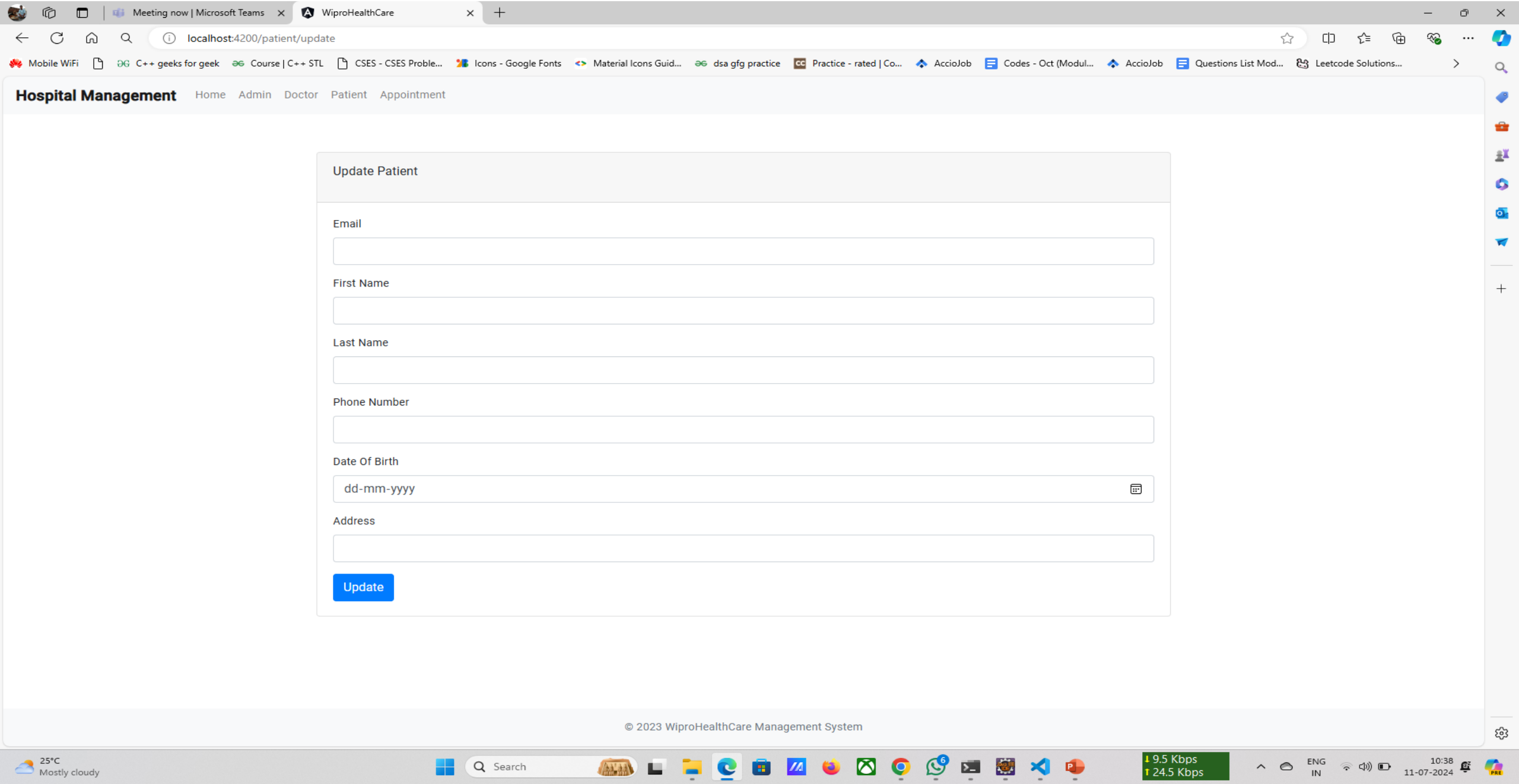
4.Patient Dashboard:



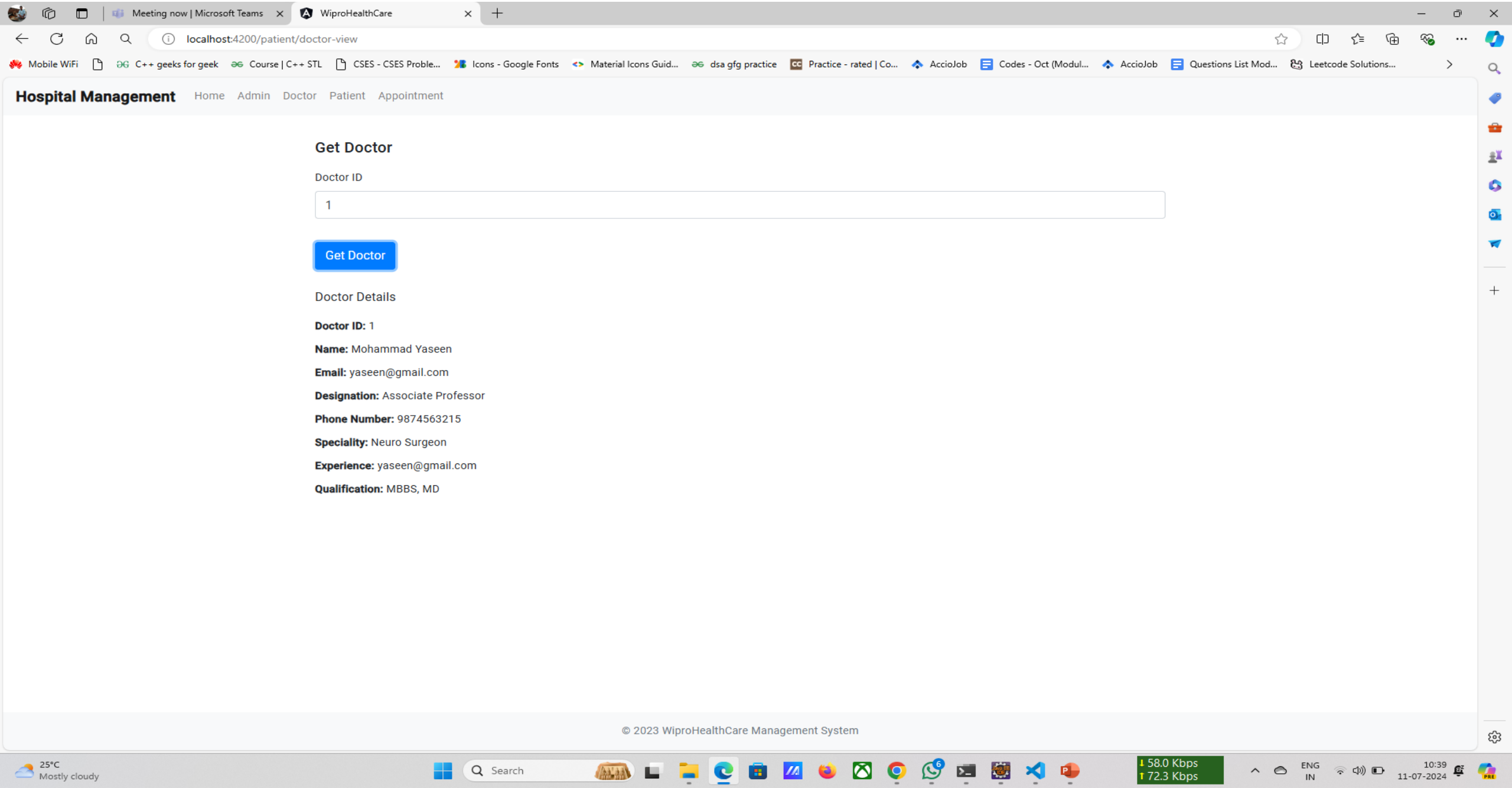
Patient Registration



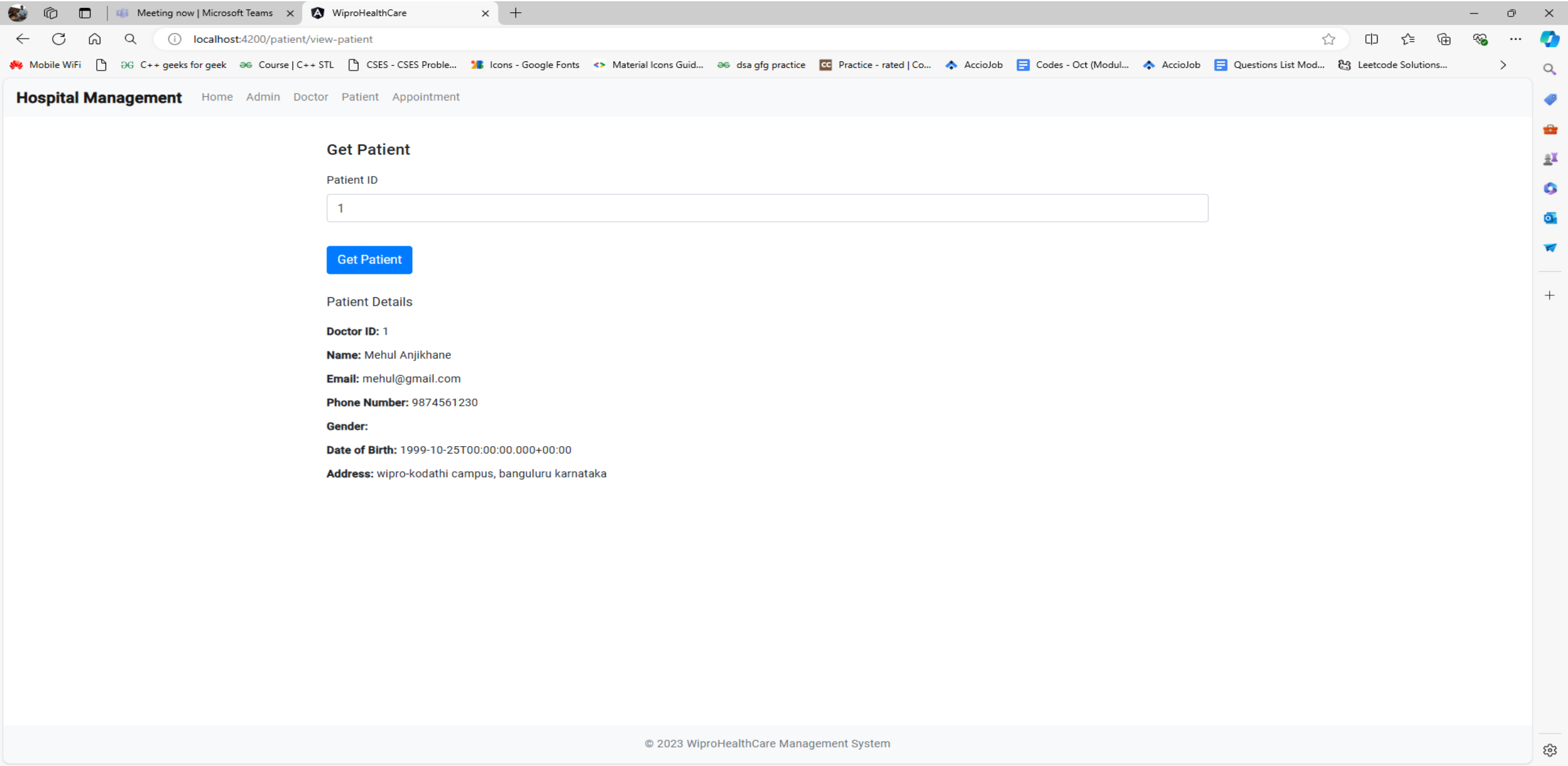
Update patient Details



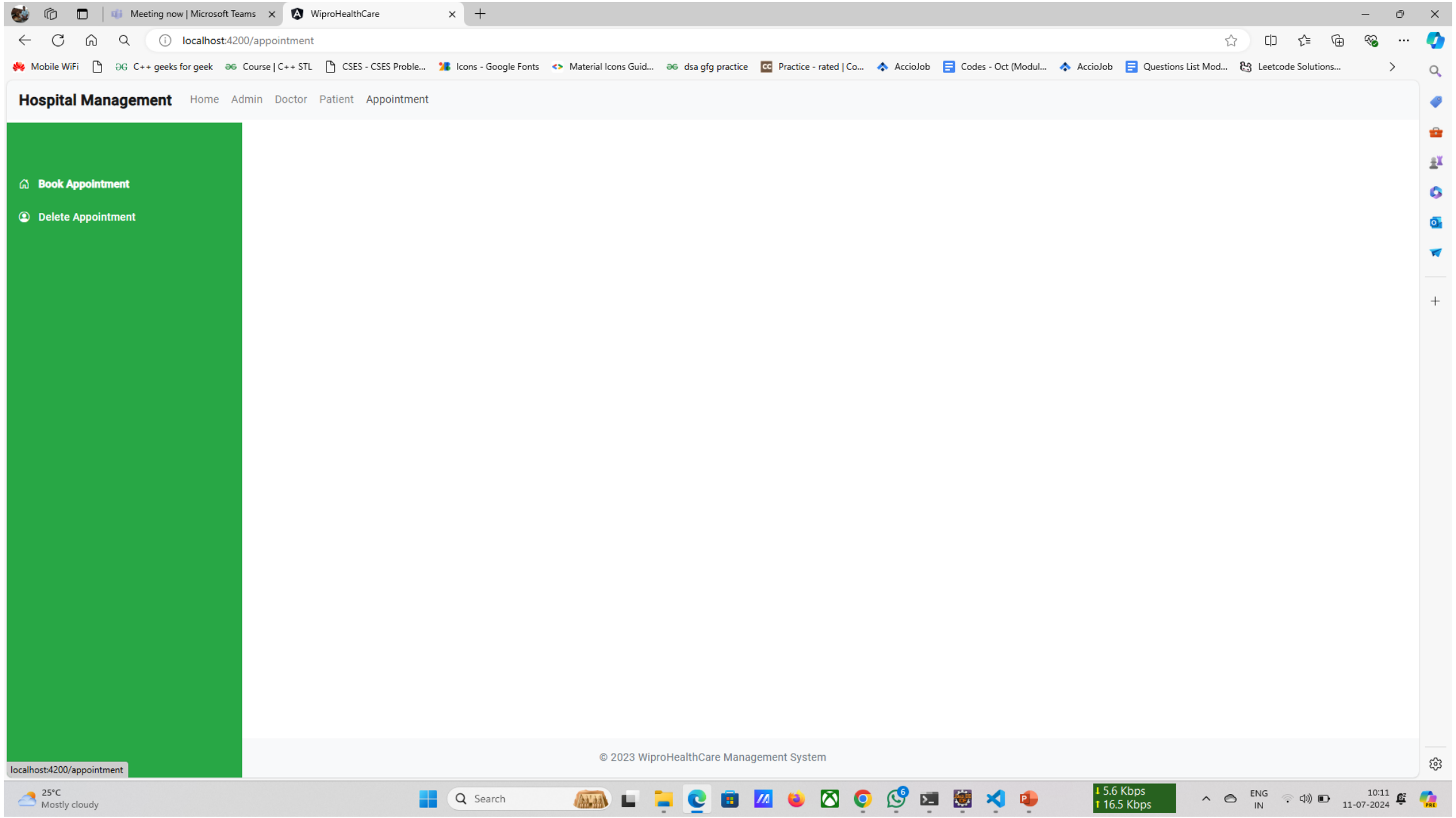
View Doctor by ID



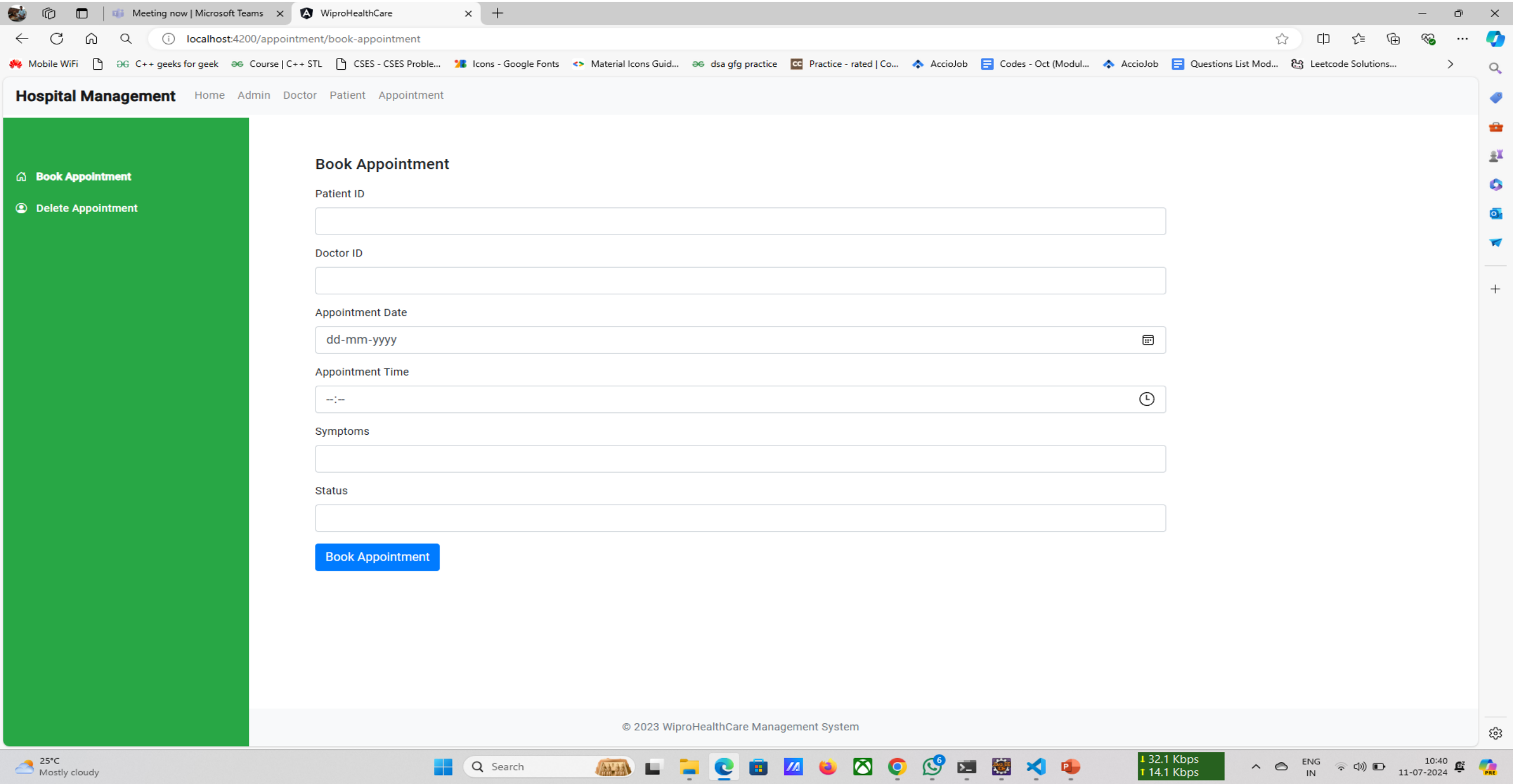
View Patient



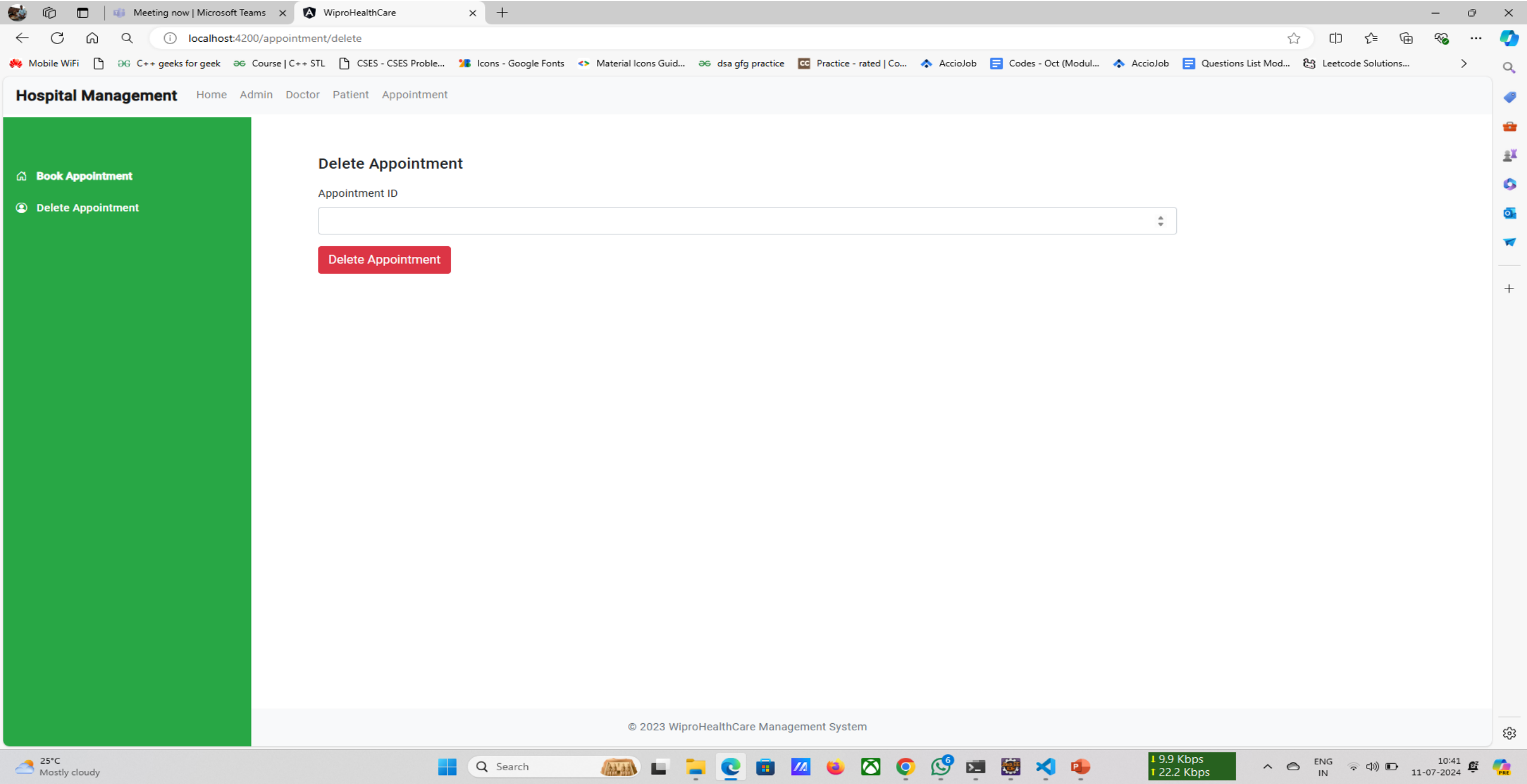
5. Appointment Dashboard :



Book Appointment



Delete Appointment



Conclusion

Wipro HealthCare is a well-structured, modular healthcare management system that utilizes modern technologies to provide a comprehensive solution for healthcare providers and patients. The use of Java Spring Boot, MySQL, and Angular ensures that the application is robust, scalable, and user-friendly. Each team member's contribution to their respective modules highlights the collaborative effort put into the development of this project.

Thank You