1. Title Page

- Project Title: *Fracto – Online Doctor Appointment Booking System*
- Your Name : Jaya Jetly
- Date: 07/09/2025
- Batch Name & Instructor Details : WIPRO NGA - .Net Full Stack Angular - FY26 - C3 & Mr. Ramesh Nediyadath

2. Table of Contents

3. Problem Definition and Objectives

- Problem: Difficulty for patients in finding doctors by city/specialization and booking appointments efficiently.
- Objectives: Provide users a platform to search, book, and manage doctor appointments; give admins full control to manage users, doctors, and appointments.

4. System Overview

- Fracto is an online doctor appointment booking system that allows patients to search for doctors by city, specialization, and availability. Users can book and cancel appointments, while admins can manage users, doctors, and appointments.

User Roles:

- User – Register, login, search doctors, book/cancel appointments.
- Admin – Manage doctors, users, appointments, and confirmations.

5. Frontend & Backend Architecture

- Tech Stack: Angular (frontend), ASP.NET Core MVC + Web API (backend), SQL Server (database), EF Core (ORM), Swagger (API docs).
- **Diagram**: Draw an architecture diagram (Frontend → API → Database + File Storage).

6. Component Breakdown & API Design

Frontend (Angular):

- **Angular Router for navigation**
- **Components:** Login, Register, Doctor Search, Appointment Booking, Admin Dashboard
- **State management (RxJS or services)**

Backend (ASP.NET Core MVC):

- **Controllers:** UserController, DoctorController, AppointmentController

- JWT Authentication for secure login/logout
- Swagger UI for API testing

API Endpoints:

Auth Endpoints

- POST /api/Auth/register → Register a new user
- POST /api/Auth/login → User login

User Endpoints

- GET /api/User → Get all users
- POST /api/User → Create new user
- GET /api/User/{id} → Get user by ID
- PUT /api/User/{id} → Update user by ID

Doctor Endpoints

- GET /api/Doctor → Get all doctors
- POST /api/Doctor → Add a new doctor
- GET /api/Doctor/search → Search doctors (city, specialization, rating, etc.)
- GET /api/Doctor/{id} → Get doctor by ID
- PUT /api/Doctor/{id} → Update doctor by ID
- DELETE /api/Doctor/{id} → Delete doctor by ID

Specialization Endpoints

- GET /api/Specialization → Get all specializations
- POST /api/Specialization → Add specialization
- GET /api/Specialization/{id} → Get specialization by ID
- PUT /api/Specialization/{id} → Update specialization by ID
- DELETE /api/Specialization/{id} → Delete specialization by ID

Appointment Endpoints

- GET /api/Appointment/doctor/{doctorId}/date/{date} → Get available slots for a doctor on a date
- POST /api/Appointment/book → Book an appointment
- GET /api/Appointment/user → Get appointments for logged-in user
- PUT /api/Appointment/cancel/{id} → Cancel appointment by ID
- GET /api/Appointment/admin/all → Get all appointments (admin)
- PUT /api/Appointment/admin/approve/{id} → Approve appointment (admin)
- PUT /api/Appointment/admin/cancel/{id} → Cancel appointment (admin)
- PUT /api/Appointment/admin/reject/{id} → Reject appointment (admin)

Rating Endpoints

- POST /api/Rating → Add rating for a doctor

7. Database Design & Optimization

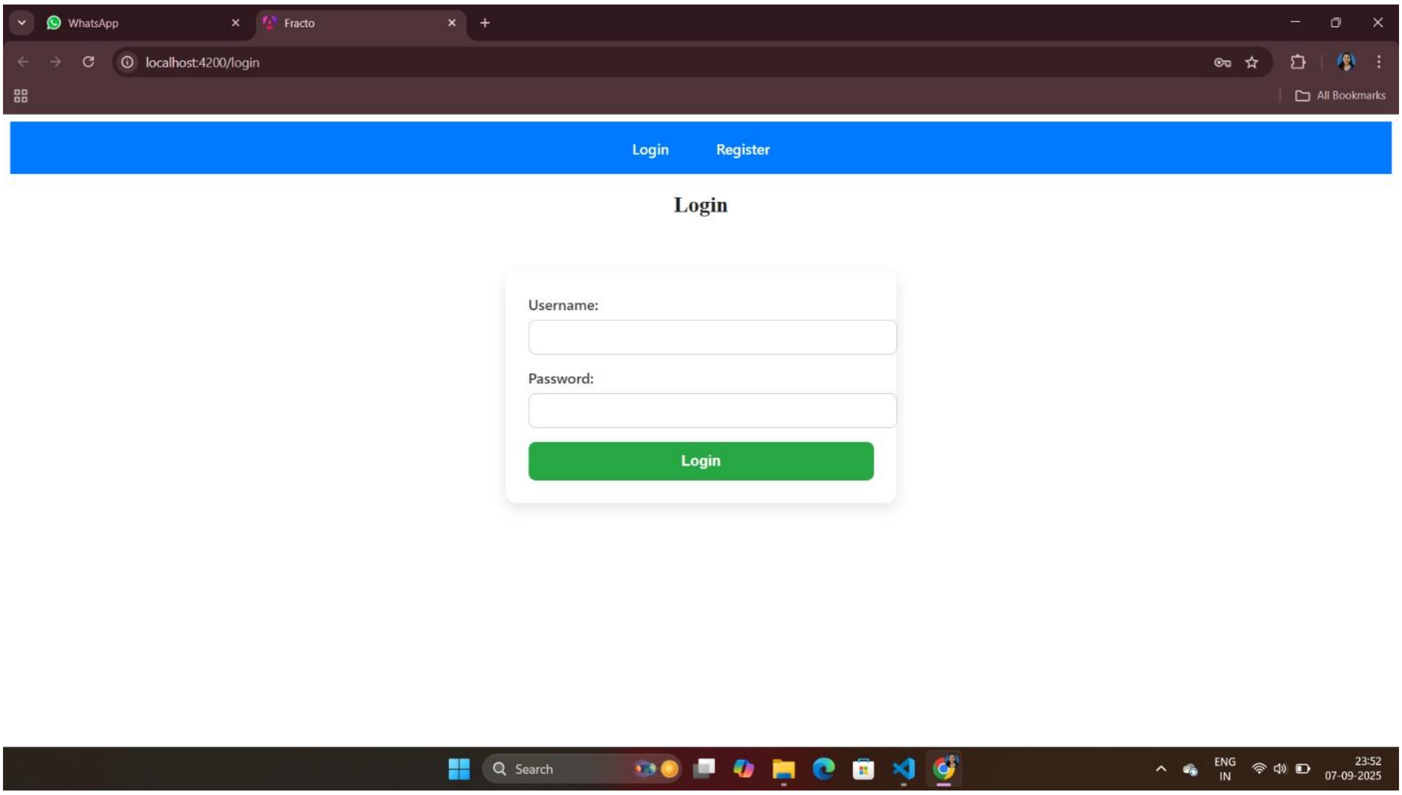
ERD Tables:

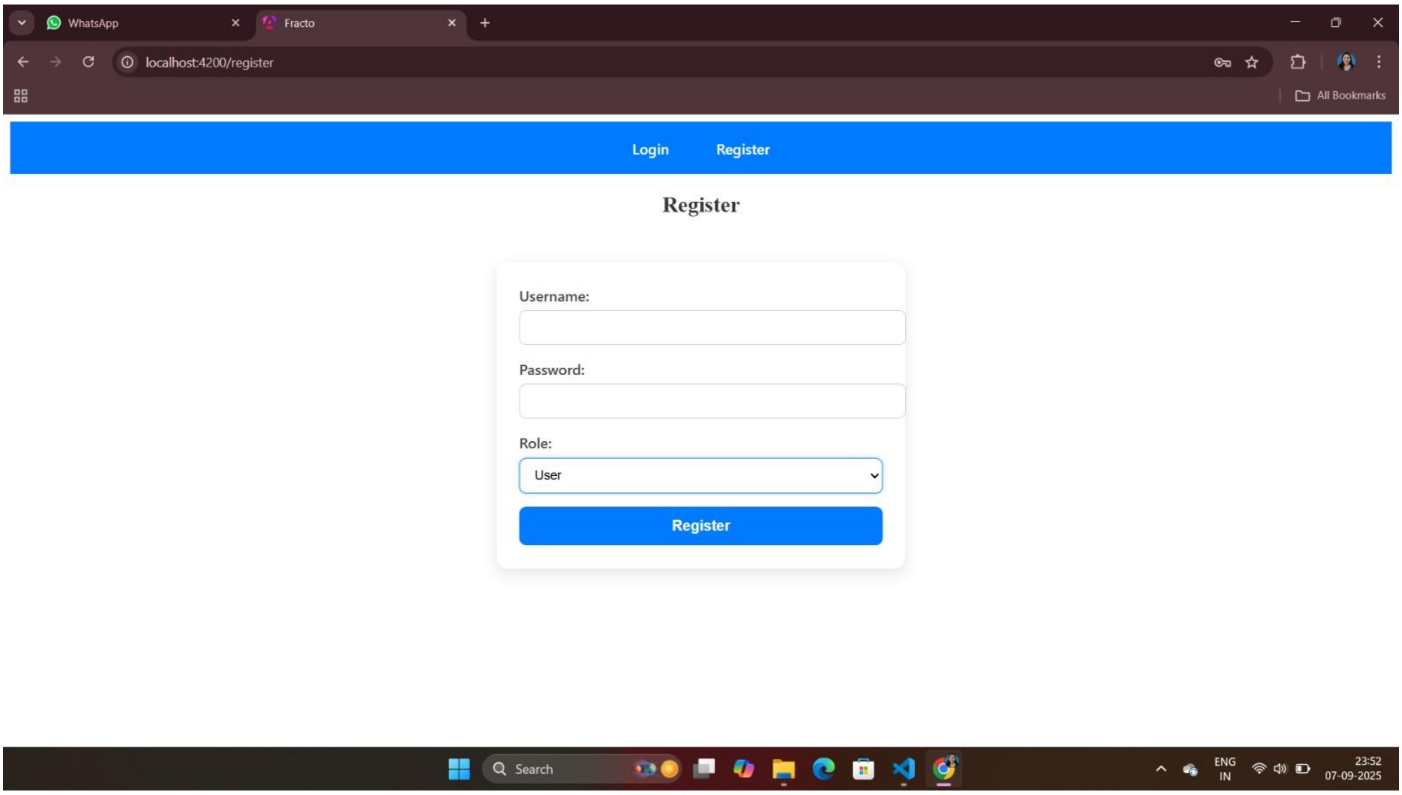
- Users (UserId, Username, Password, Role)
- Doctors (DoctorId, Name, SpecializationId, City, Rating)
- Specializations (SpecializationId, SpecializationName)
- Appointments (AppointmentId, UserId, DoctorId, AppointmentDate, TimeSlot, Status)
- Ratings (RatingId, DoctorId, UserId, Rating)

Optimization:

- Indexing on City and SpecializationId for faster queries.
- Relationships: One-to-Many (User → Appointments, Doctor → Appointments).

8. Screenshots





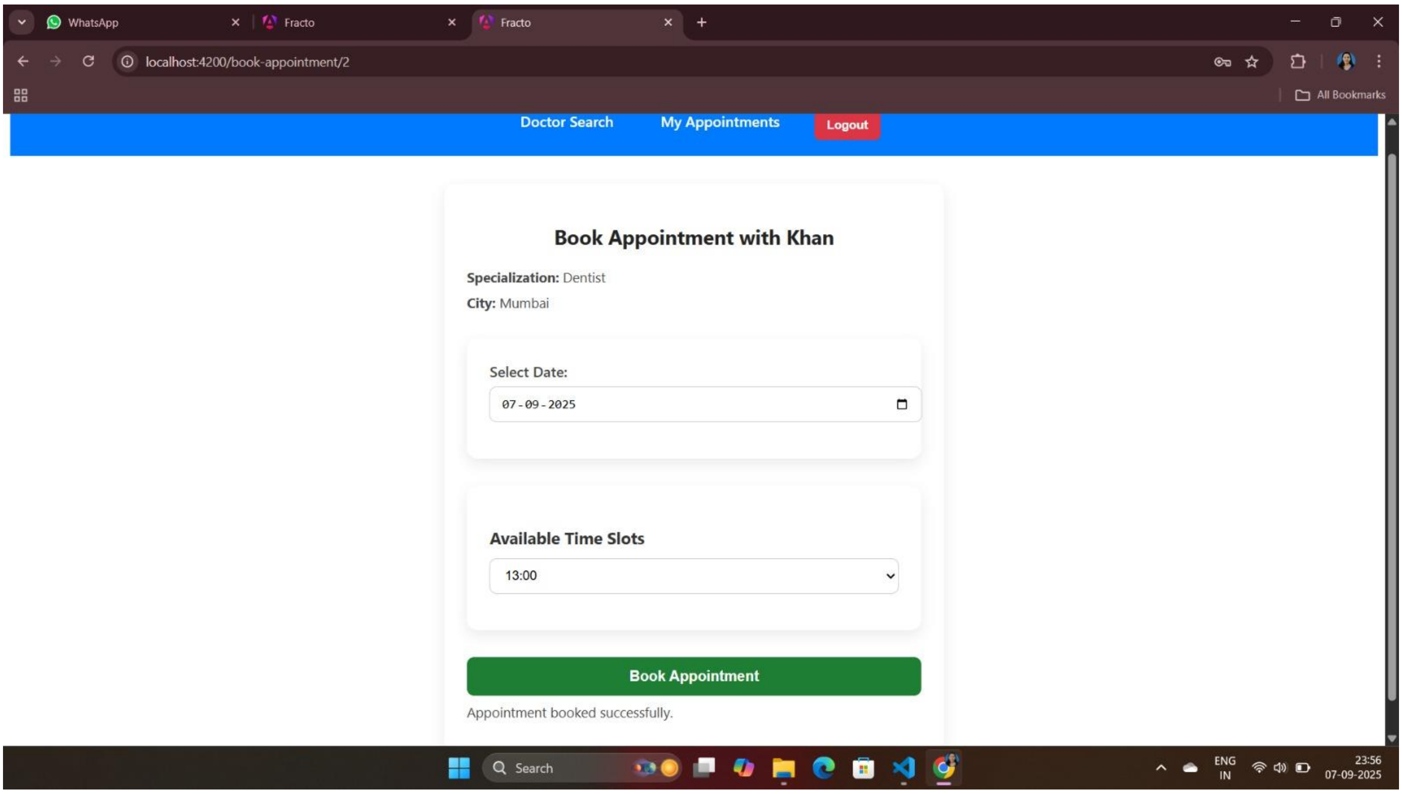
localhost:4200/register

Register

Username:

Password:

Role: User



localhost:4200/book-appointment/2

Book Appointment with Khan

Specialization: Dentist

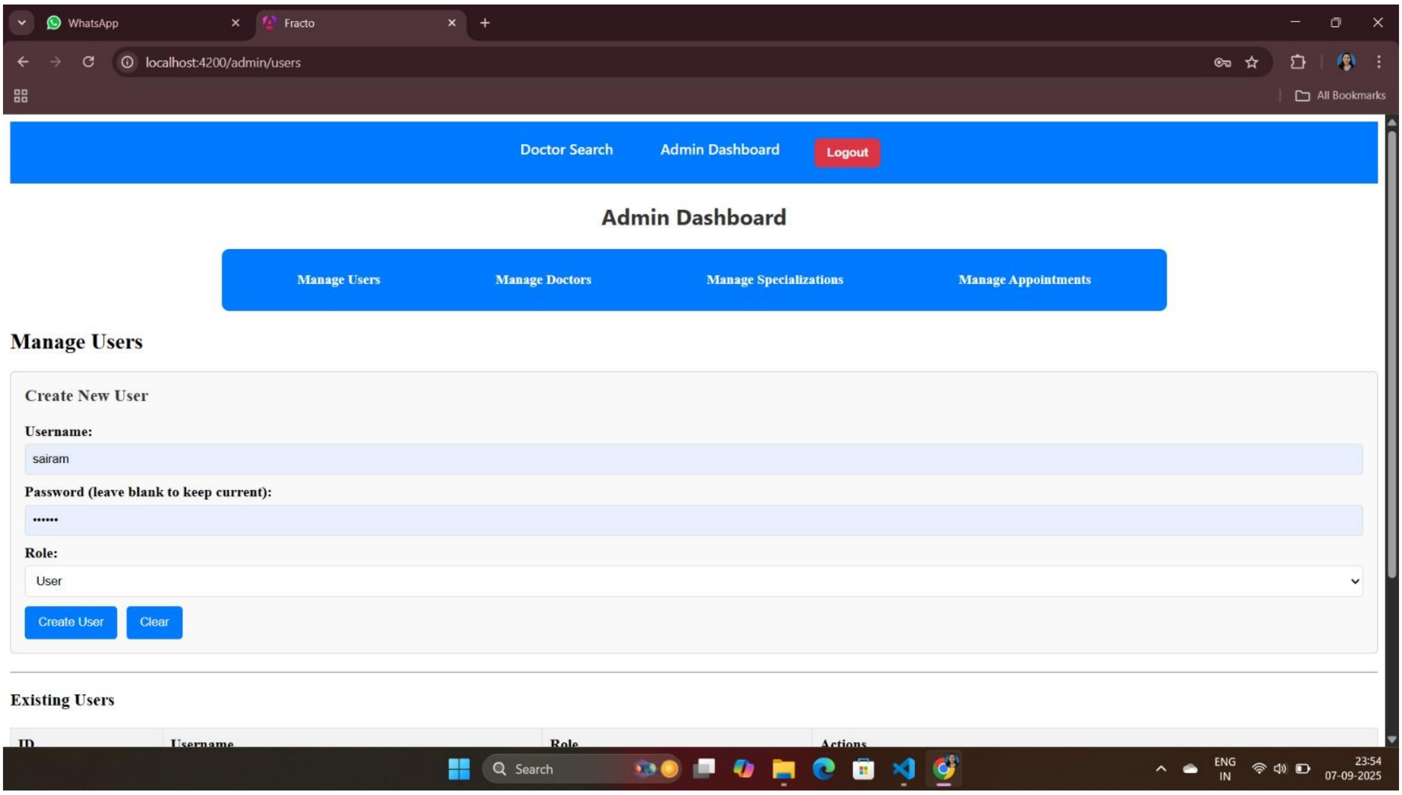
City: Mumbai

Select Date: 07-09-2025

Available Time Slots

13:00

Appointment booked successfully.



Admin Dashboard

Manage Users

Create New User

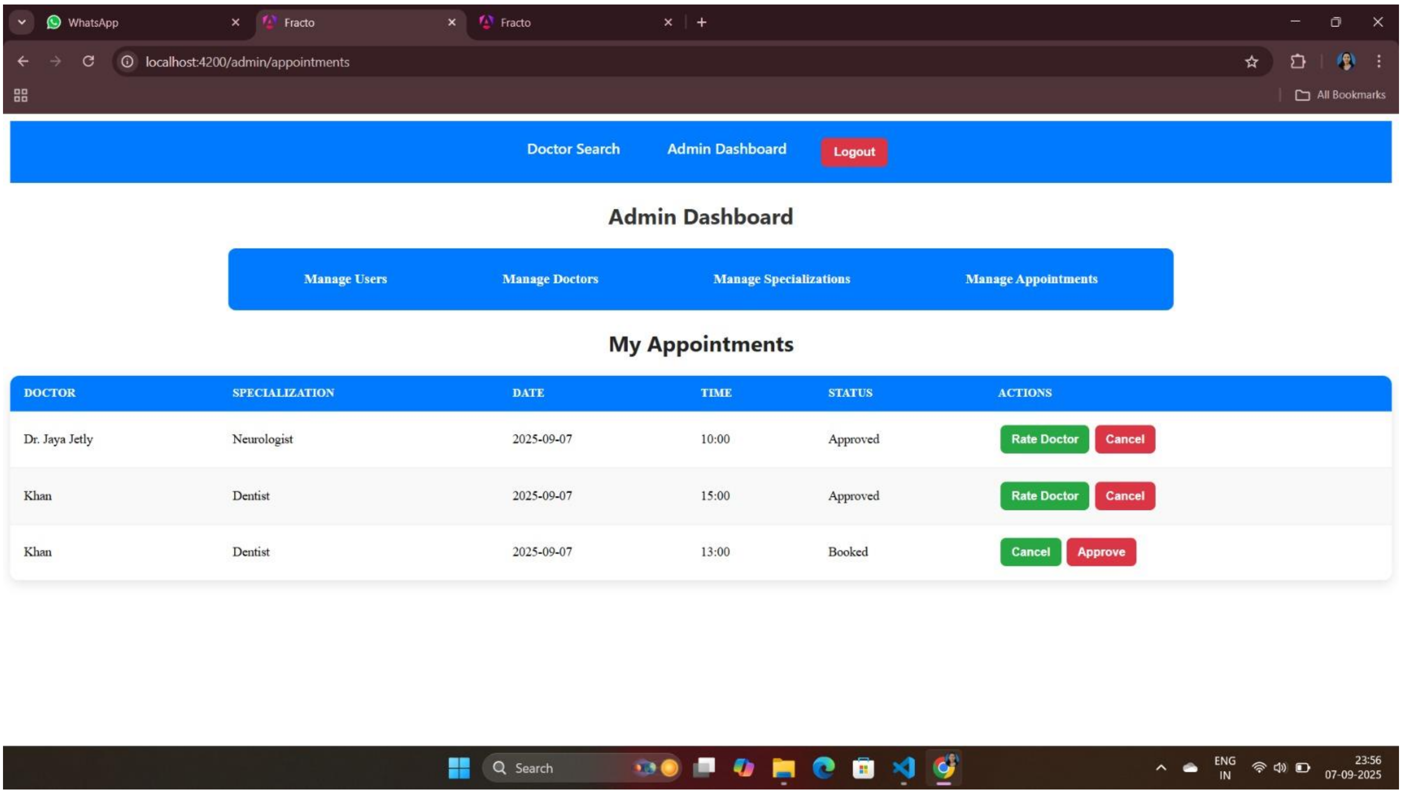
Username: sairam

Password (leave blank to keep current): ••••••

Role: User

Existing Users

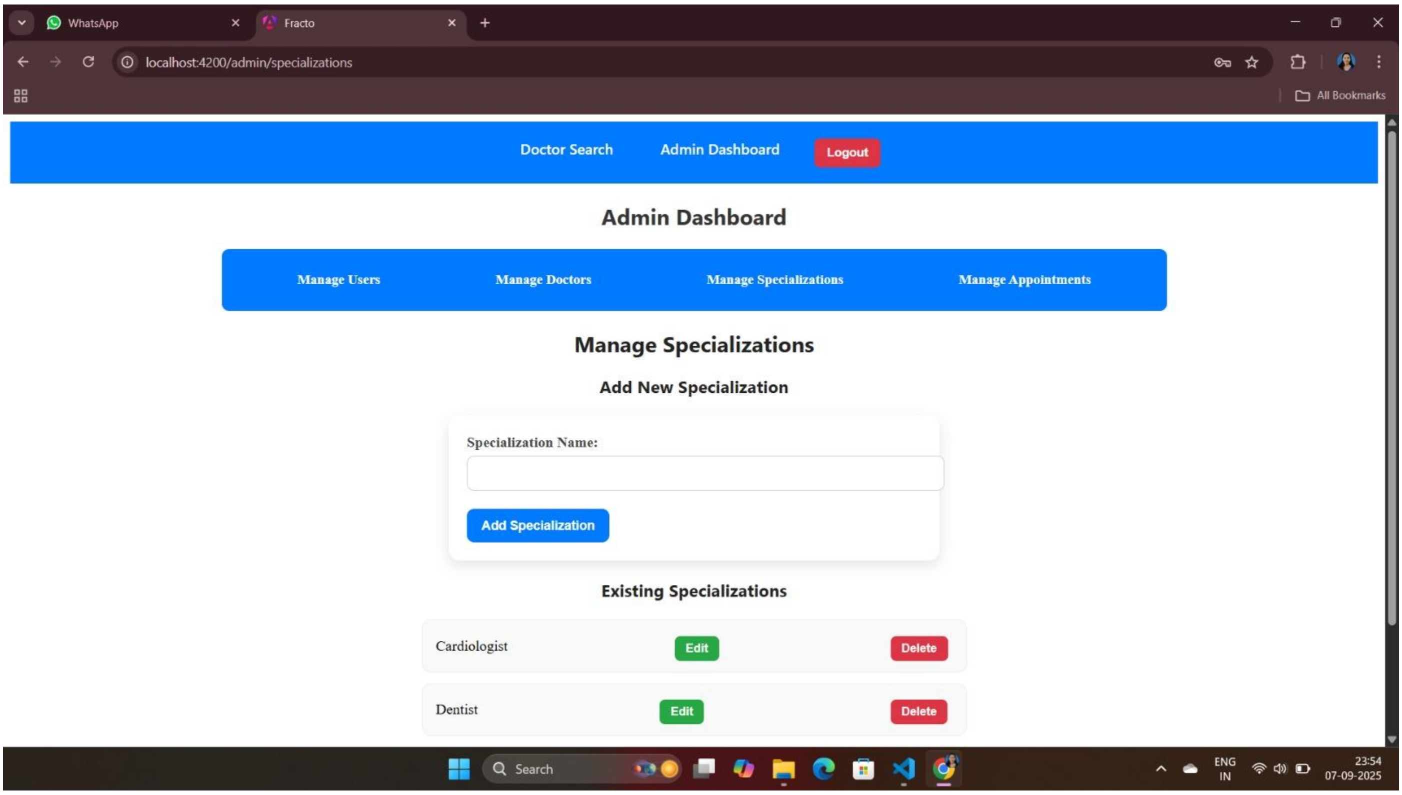
ID	Username	Role	Actions



Admin Dashboard

My Appointments

DOCTOR	SPECIALIZATION	DATE	TIME	STATUS	ACTIONS
Dr. Jaya Jetly	Neurologist	2025-09-07	10:00	Approved	Rate Doctor Cancel
Khan	Dentist	2025-09-07	15:00	Approved	Rate Doctor Cancel
Khan	Dentist	2025-09-07	13:00	Booked	Cancel Approve



Admin Dashboard

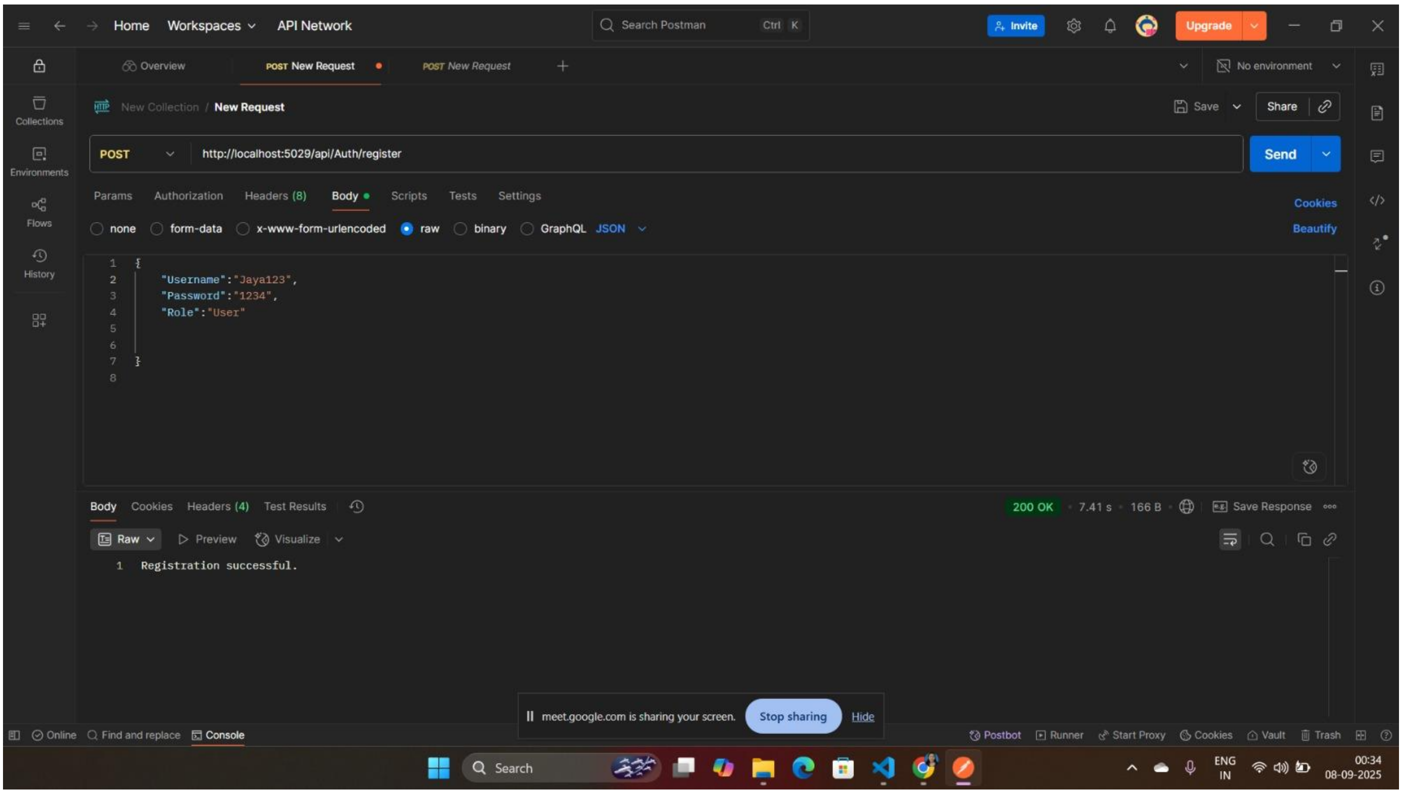
Manage Specializations

Add New Specialization

Specialization Name:

Existing Specializations

Cardiologist	Edit	Delete
Dentist	Edit	Delete

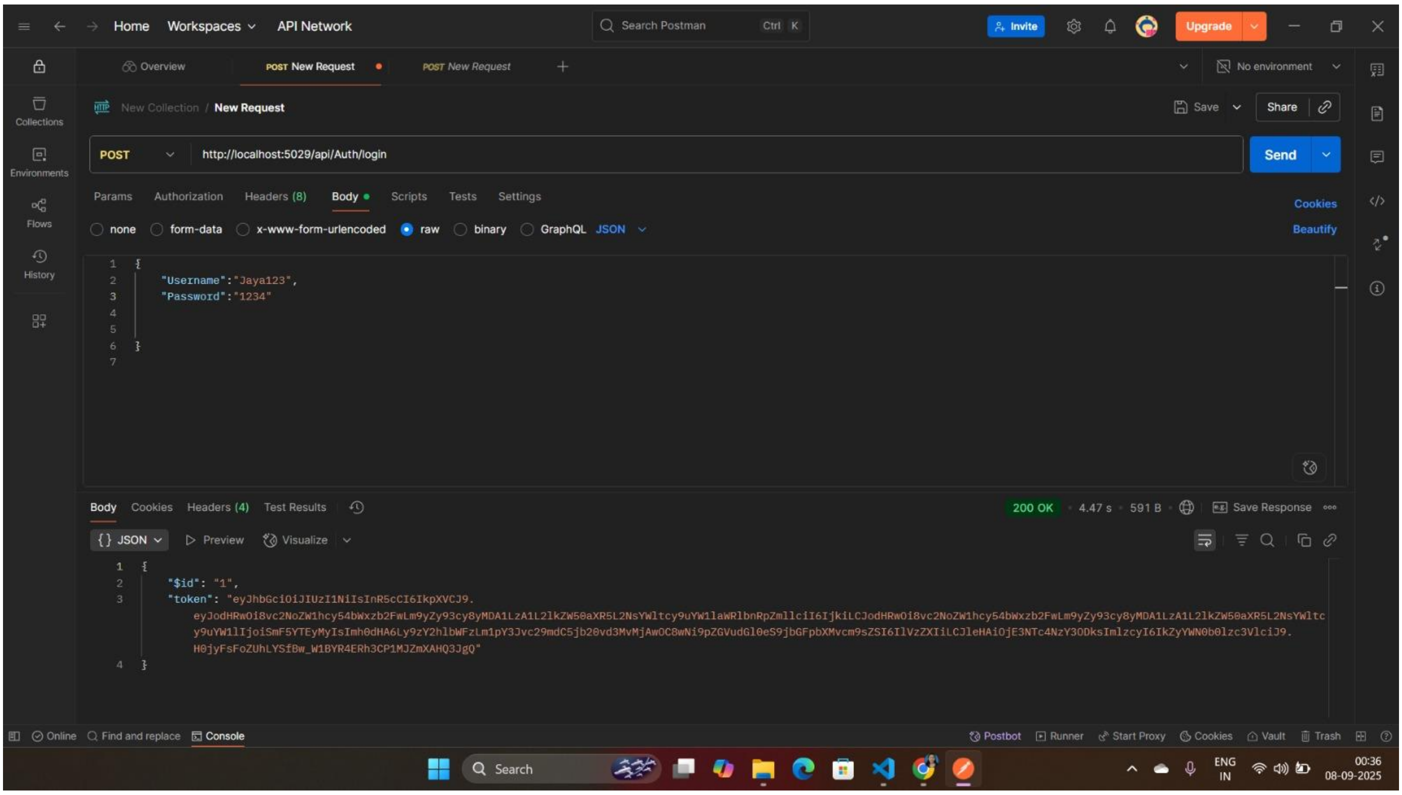


POST http://localhost:5029/api/Auth/register

```
{
    "Username":"Jaya123",
    "Password":"1234",
    "Role":"User"


}
```

```
Registration successful.
```



POST http://localhost:5029/api/Auth/login

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
  "Username":"Jaya123",
  "Password":"1234"


}

```

200 OK · 4.47 s · 591 B

```
{
    "$id": "1",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjkiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltc
        y9uYW1lIjoiSmF5YTEyMyIsImh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vd3MvMjAwOC8wNi9pZGVudGl0eS9jbGFpbXMvcm9sZSI6IlVzZXIiLCJleHAiOjE3NTc4NzY3ODksImlzcyI6IkZyYWN0b0lzc3VlciJ9.
        H0jyFsFoZUhLYSfBw_W1BYR4ERh3CP1MJZmXAHQ3JgQ"
}
```

9. Sprint Plan & Deliverables

Sprint I

- Use case documentation for login, registration, doctor search, appointment booking.
- Database schema design (ERD, SQL scripts).
- Backend setup with controllers and EF Core.
- Static Angular templates for login and doctor search.
  Deliverables: Use case doc, ERD diagram, static frontend templates.

Sprint II

- Angular setup with routing for user and admin.
- Backend implementation: User auth with JWT, CRUD for doctors/users.
- Appointment booking APIs.
- Admin and user UI development.
- Component testing for Angular and Web API.
  Deliverables: Angular UI, secure API endpoints, CRUD operations, testing reports.

Sprint III

- Search functionality by city, specialization, rating.
- Appointment booking with available time slots.
- Admin module for approval/cancellation.
- Ratings module for doctors.
- Swagger API documentation.
- Real-time notifications (optional via SignalR).
  Deliverables: Fully functional booking platform, integrated frontend & backend, Swagger docs.

10. Conclusion

The **Fracto platform** successfully delivers a robust solution for online doctor appointment booking. Patients can search doctors by city, specialization, and availability, while admins manage appointments and confirmations. With Angular as the frontend and ASP.NET Core MVC as the backend, the system ensures scalability, security, and user-friendly interaction.

Key Achievements:

- Seamless booking and cancellation process.
- JWT-based secure authentication.
- Integrated Angular frontend with ASP.NET Core Web API backend.
- Optimized SQL Server database design.
- Swagger-based API documentation.

Future Scope:

- Add **tele-consultation** through video calling.
- Integrate **online payment gateways** for booking fees.
- Send **SMS/email notifications** for appointment reminders.
- Implement **AI-based doctor recommendations**.