

# FRACTO

**By: VISHAKHA PRASAD**

---

## **Table of Contents:**

1. Problem Definition and Objectives
  2. Frontend & Backend Architecture
  3. System Design Diagram
  4. Component Breakdown & API Design
  5. Database Design & Storage Optimization
  6. Testing
  7. Conclusion
- 

## **1. Problem Definition and Objectives**

### **Problem Statement:**

Booking doctor appointments is often inefficient. Patients struggle with visibility of available slots, manual booking processes, and difficulties in rescheduling. Doctors and administrators face challenges in record-keeping, managing cancellations, and providing real-time updates.

Objectives:

- Provide a digital platform for users to register, search doctors, and book/cancel appointments.
  - Develop an admin interface for managing doctors, users, and appointments.
  - Implement JWT-based authentication for security.
  - Support file uploads (profile images for doctors and users).
  - Add real-time notifications using (SignalR).
  - Enable a rating system for users to rate doctors.
- 

## **2. Frontend & Backend Architecture**

### **Technology Stack:**

- Frontend: Angular 16, Angular Material, RxJS, Forms, SignalR Client
- Backend: ASP.NET Core Web API (.NET 9), Entity Framework Core, SignalR
- Database: SQL Server / SQLite (development)
- Authentication: JWT Tokens (with UserId and Role claims)
- File Storage: Local server (wwwroot/uploads)

## Architecture Overview

- **Frontend:**

- Angular components for user and admin roles.
- Routing for login, register, search, booking, and admin dashboards.
- Services for API communication (Auth, Users, Doctors, Appointments, Ratings, Files).

- **Backend:**

- REST APIs with controllers for each module (Auth, Users, Doctors, Appointments, Ratings, Specializations, Files).
- EF Core handles database operations.
- JWT Authentication middleware secures APIs.
- SignalR hub broadcasts real-time events.

- **Database:**

- Normalized schema with foreign key relationships.
  - Stores references to uploaded files.
- 

## 3. System Design Diagram

Flow:

- Users/Admins interact with Angular Frontend.
  - Angular sends requests to ASP.NET Core Web API controllers.
  - Controllers use Entity Framework Core to access SQL Server/SQLite.
  - SignalR Hub provides real-time notifications (appointment confirmations, cancellations).
  - File Upload Controller handles storing and retrieving profile images.
- 

## 4. Component Breakdown & API Design

### Frontend Components

#### Core Module

1. Guards: admin.guard.ts, auth.guard.ts
2. Interceptors: jwt.interceptor.ts
3. Services: auth.service.ts (likely manages user authentication across the app)
4. Models: user.ts (the fundamental user data model)

#### Shared/Common Module

1. Components: navbar.component.ts (a shared navigation bar)

#### Appointment Module

2. Pages: book-appointment.component.ts, my-appointments.component.ts
3. Services: appointment.service.ts

4. Models: appointment.ts

#### **Doctor Module**

1. Pages: doctor-list.component.ts
2. Services: doctor.service.ts, specialization.service.ts
3. Models: doctor.ts
4. Ratings Module

#### **Admin Module**

1. Pages: ratings-page.component.ts
2. Services: rating.service.ts

#### **Authentication Module**

1. Pages: login.component.ts, register.component.ts

#### **Admin Module**

1. Pages: admin-appointments.component.ts, manage-doctors.component.ts

#### **API Design**

##### **AdminUsers**

- POST /api/Appointments
- GET /api/Appointments
- PUT /api/Appointments/cancel/{id}
- GET /api/Appointments/user/{userId}
- GET /api/Appointments/pending
- PUT /api/Appointment/approve/{id}

##### **Auth**

- POST /api/Auth/register
- POST /api/Auth/login

##### **Doctors**

- GET /api/Doctors
- POST /api/Doctors
- GET /api/Doctors/{id}
- PUT /api/Doctors{id}
- DELETE /api/Doctors/{id}
- GET /api/Doctors/{id}/timeslots

##### **Ratings**

- POST /api/Ratings
- GET /api/Ratings/doctors/{doctorsId}/avg
- GET /api/Ratings/doctors/{doctorsId}

##### **Specialisations**

- GET /api/Specializations
- POST /api/Specializations
- GET /api/Specializations/{id}
- PUT /api/Specialization/{id}

- DELETE /api/Specialization/{id}

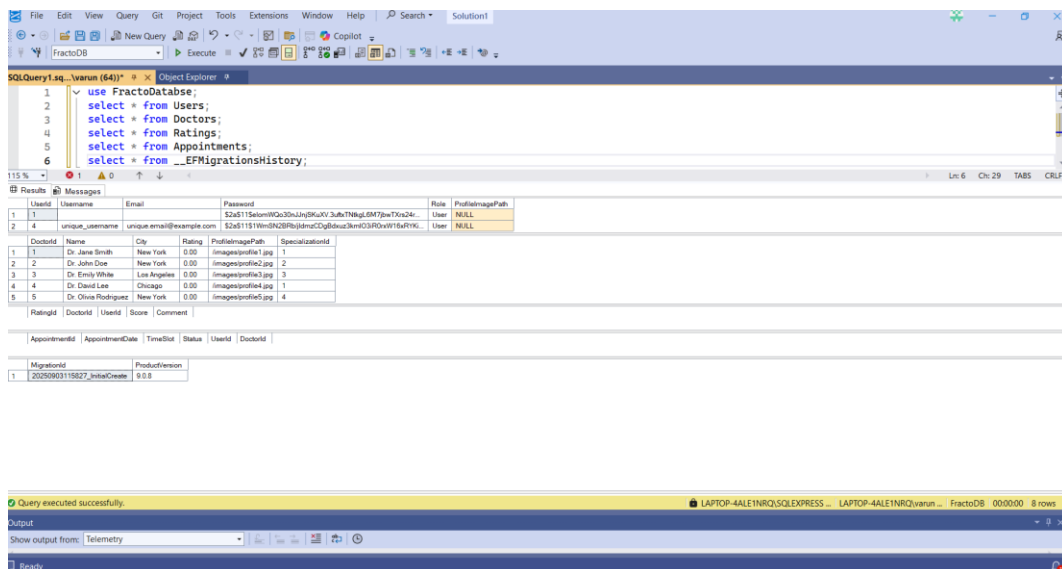
## Users

- GET /api/Users/{id}
- POST /api/Users/{id}/avatar

## 5. Database Design & Storage Optimization

### Entities

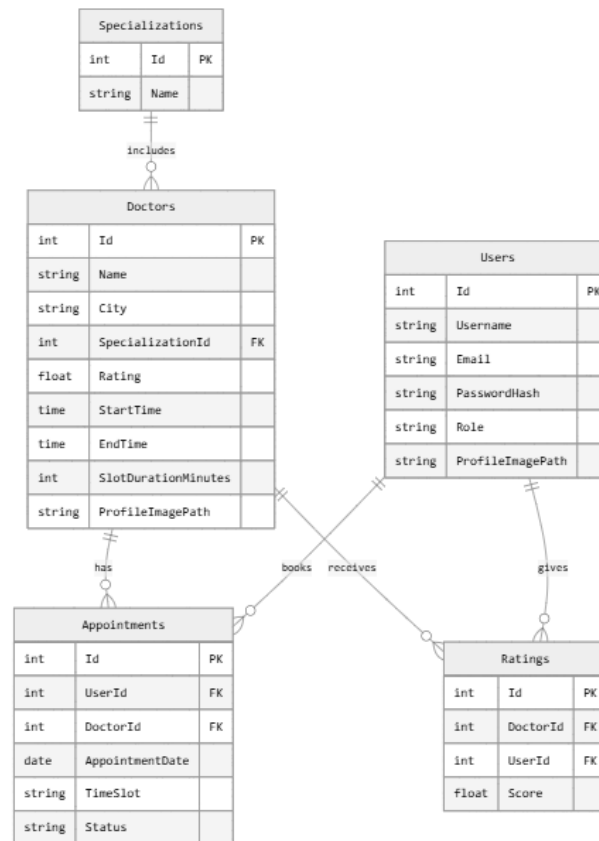
- Users(UserId, Username, Email, PasswordHash, Role, City, ProfileImagePath, CreatedAt)
- Doctors(DoctorId, Name, SpecializationId, City, Rating, ProfileImagePath, CreatedAt)
- Specializations(SpecializationId, SpecializationName)
- Appointments(AppointmentId, UserId, DoctorId, AppointmentDate, TimeSlot, Status)
- Ratings(RatingId, DoctorId, UserId, Rating, Comment, CreatedAt)



### Relationships:

- One-to-Many relationship between Doctors and Appointments (one doctor can have multiple appointments).
- One-to-Many relationship between Users and Appointments (one user can have multiple appointments).
- One-to-Many relationship between Doctors and Ratings (one doctor can have multiple ratings).
- One-to-Many relationship between Users and Ratings (one user can rate multiple doctors).
- One-to-Many relationship between Specializations and Doctors (one specialization can have multiple doctors).

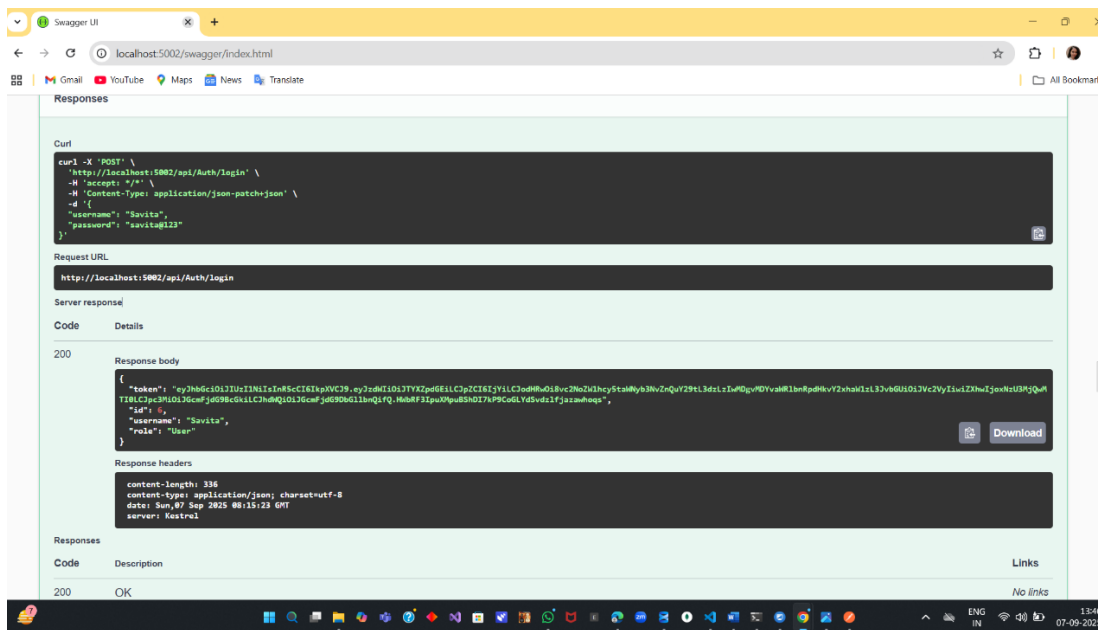
## ER Diagram:-



## 6. Testing Backend Testing

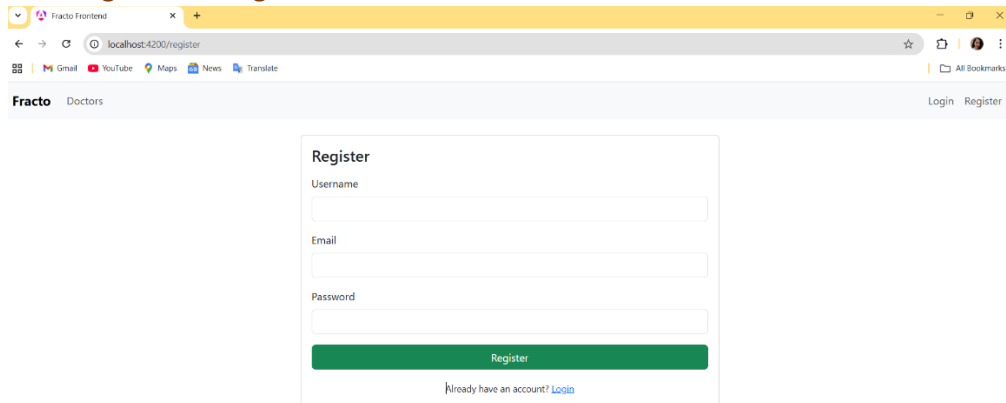
- xUnit + Moq used for unit testing controllers and services.
  - EF Core InMemory provider simulates database.
-

- Postman collections used to test APIs



## 7. Implementation Screenshots

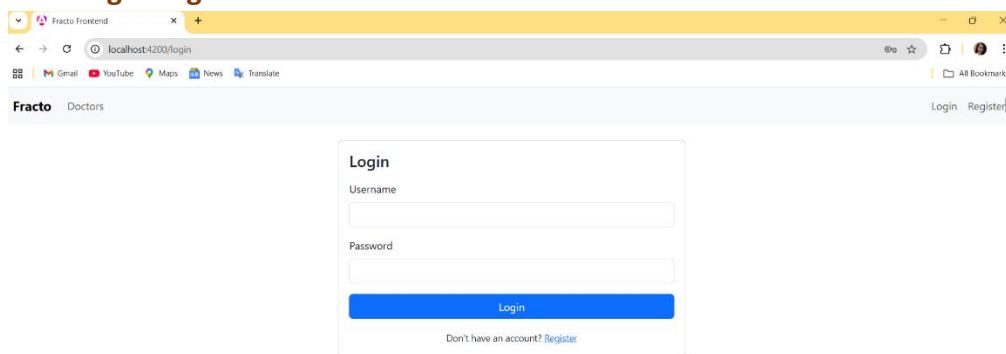
### User Registration Page :-



A screenshot of a web browser displaying the 'Register' page of an application. The browser's address bar shows 'localhost:4200/register'. The page has a header with 'Fracto' and 'Doctors' on the left, and 'Login' and 'Register' on the right. The main content area features a 'Register' form with three input fields: 'Username', 'Email', and 'Password'. Below these fields is a green 'Register' button. At the bottom of the form, there is a link that says 'Already have an account? [Login](#)'.



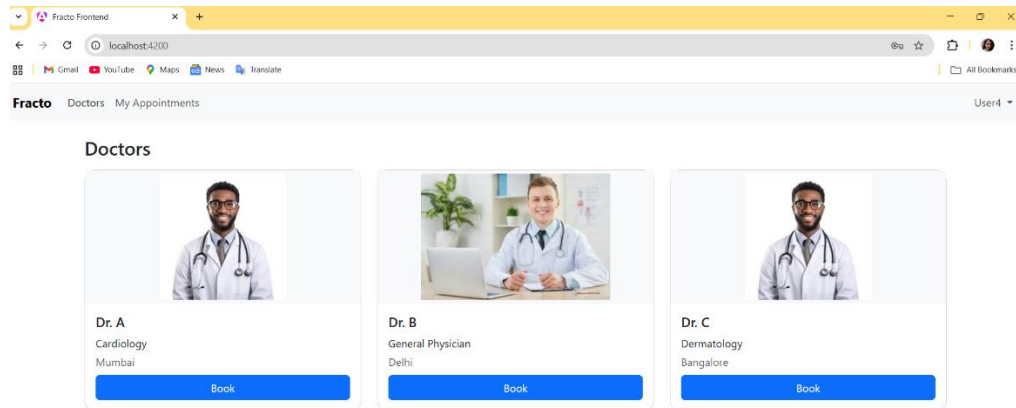
### User Login Page:-



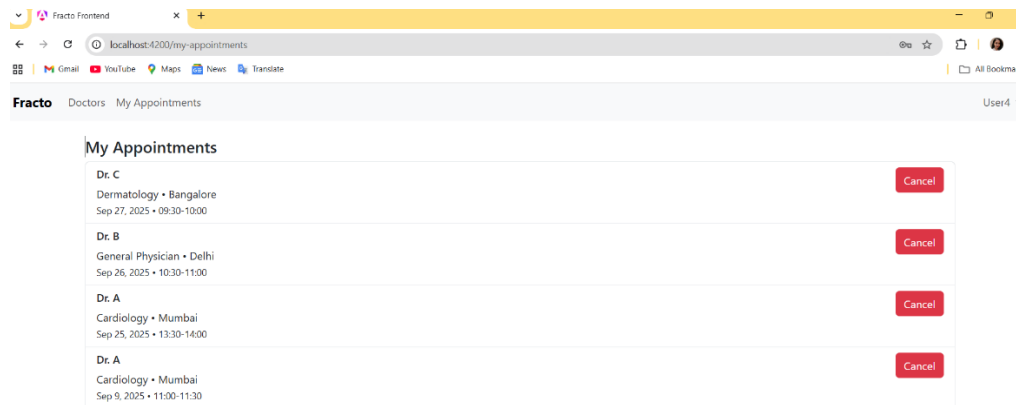
A screenshot of a web browser displaying the 'Login' page of an application. The browser's address bar shows 'localhost:4200/login'. The page has a header with 'Fracto' and 'Doctors' on the left, and 'Login' and 'Register' on the right. The main content area features a 'Login' form with two input fields: 'Username' and 'Password'. Below these fields is a blue 'Login' button. At the bottom of the form, there is a link that says 'Don't have an account? [Register](#)'.



## User Home Page:-



## My Appointment Page:-



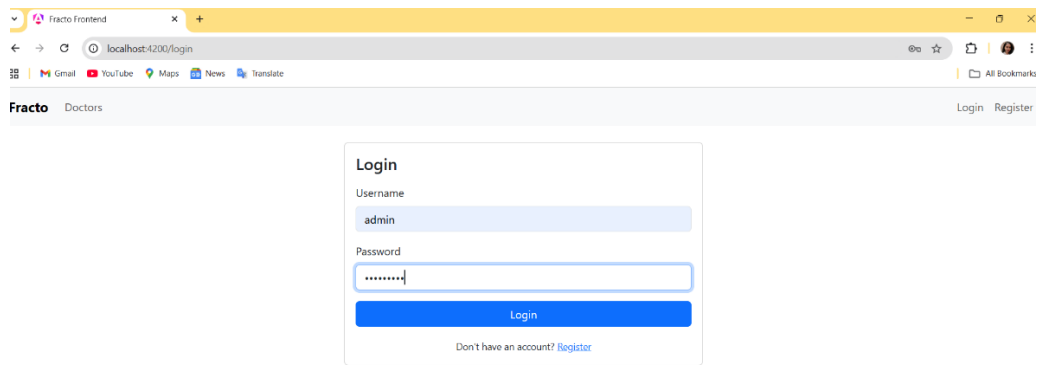


## User Can Book Appointment According To Date and Slot:-

The screenshot displays a web application interface for booking an appointment. The top navigation bar includes the 'Fracto' logo, 'Doctors', 'My Appointments', and a user profile 'User4'. The main content area is divided into two sections. The first section, titled 'Dr. A', lists the doctor's name, specialty (Cardiology), and location (Mumbai). The second section, titled 'Select date & slot', contains a date picker and a list of available time slots. The date picker is set to 'September, 2025' and shows a calendar grid with the 8th of September highlighted. The time slots are listed as follows:

Time Slot
<input type="radio"/> 09:00-09:30
<input type="radio"/> 09:30-10:00
<input type="radio"/> 10:00-10:30
<input type="radio"/> 10:30-11:00
<input checked="" type="radio"/> 11:00-11:30
<input type="radio"/> 11:30-12:00
<input type="radio"/> 12:00-12:30

## Admin Login Page:-



A screenshot of a web browser showing the login page for 'Fracto Frontend'. The browser's address bar displays 'localhost:4200/login'. The page has a header with 'Fracto' and 'Doctors' on the left, and 'Login' and 'Register' on the right. The main content area features a 'Login' form with a title, a 'Username' field containing 'admin', a 'Password' field with masked characters, and a blue 'Login' button. Below the button is a link that says 'Don't have an account? Register'.

Fracto Frontend

localhost:4200/login

Fracto Doctors Login Register

### Login

Username

admin

Password

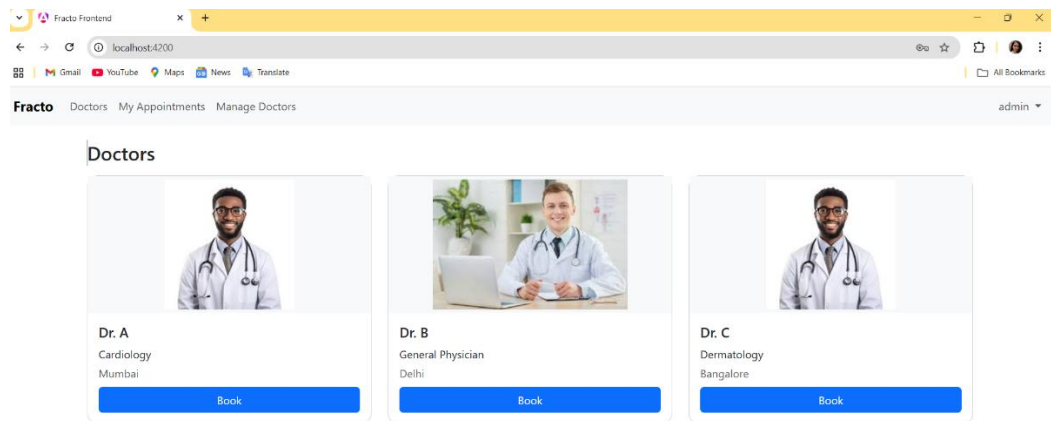
\*\*\*\*\*

Login

Don't have an account? [Register](#)



## Admin Home Page:-




A screenshot of the admin home page in the 'Fracto Frontend' application. The browser address bar shows 'localhost:4200'. The header includes 'Fracto' and navigation links 'Doctors', 'My Appointments', and 'Manage Doctors'. A user dropdown menu shows 'admin'. The main section is titled 'Doctors' and displays three doctor profiles in cards. Each card includes a profile picture, name, specialization, location, and a blue 'Book' button.

Fracto Frontend

localhost:4200


Fracto Doctors My Appointments Manage Doctors admin

### Doctors




**Dr. A**  
Cardiology  
Mumbai

Book



**Dr. B**  
General Physician  
Delhi

Book



**Dr. C**  
Dermatology  
Bangalore

Book



## Admin Can Manage Doctors(using CRUD operations):-

Fracto Frontend x +

localhost:4200/manage-doctors

Fracto Doctors My Appointments Manage Doctors admin

### Manage Doctors (Admin)

**Add Doctor**

Add

Dr. A (Cardiology) Mumbai	Edit Delete
Dr. B (General Physician) Delhi	Edit Delete
Dr. C (Dermatology) Bangalore	Edit Delete



## Adding Doctor:-

Fracto Frontend x +

localhost:4200/manage-doctors

Fracto Doctors My Appointments Manage Doctors admin

### Manage Doctors (Admin)

**Add Doctor**

Add

Dr. A (Cardiology) Mumbai	Edit Delete
Dr. B (General Physician) Delhi	Edit Delete
Dr. C (Dermatology) Bangalore	Edit Delete
Dr. D (Neurology) Kolkata	Edit Delete



Managing Appointments:-

