

Full Stack Development with MERN Report

Docspot: Seamless Appointment Booking for Health

1. Introduction

Title: Docspot: Seamless Appointment Booking for Health

Team members: B. Lohith Pavan Krishna – Full Stack Developer – Frontend & Backend Development including database schema and connections.

2. Project Overview

Purpose:

DocSpot is a web-based healthcare appointment booking system that allows patients to easily schedule appointments with doctors online. The main goal of the system is to replace traditional manual booking methods, reduce waiting time at hospitals or clinics, and make healthcare services more accessible and convenient for users.

Features:

- **User Registration and Login:**
Allows users to create accounts and securely log in to access the system.
- **Doctor Listing and Specialization Filtering:**
Displays available doctors and enables users to filter them based on specialization for easier selection.
- **Appointment Booking System:**
Enables patients to book appointments with doctors quickly through an online interface.
- **Secure Authentication:**
Uses secure login mechanisms (such as JWT and password encryption) to protect user data.
- **Responsive User Interface:**
Provides a clean and mobile-friendly design for smooth usage across different devices.
- **Admin/Doctor Management:**
Allows administrators to manage doctors, users, and appointments efficiently (if implemented).

3. Architecture

Frontend:

The frontend of DocSpot is developed using React.js, which follows a component-based architecture. This approach allows the application to be divided into reusable UI components, making the code more organized and maintainable.

- Axios is used to communicate with the backend APIs.
- React Router handles navigation between different pages such as login, registration, and appointment booking.
- React Hooks (like useState and useEffect) are used for managing application state and handling lifecycle events.

Backend:

The backend is built using Node.js and Express.js, which together create a powerful server-side environment.

- The application follows a RESTful API structure, where different routes handle users, doctors, and appointments.
- Middleware is used for authentication and protecting private routes.
- Environment-based configuration (using .env files) ensures secure handling of sensitive information like database credentials and JWT secrets.

Database:

The project uses MongoDB as a NoSQL database to store application data.

- Mongoose ODM (Object Data Modeling) is used to define schemas and interact with the database efficiently.
- The main collections include:
 - Users – Stores patient and admin information.
 - Doctors – Stores doctor profiles and specialization details.
 - Appointments – Stores booking details and appointment status.

4. Setup Instructions

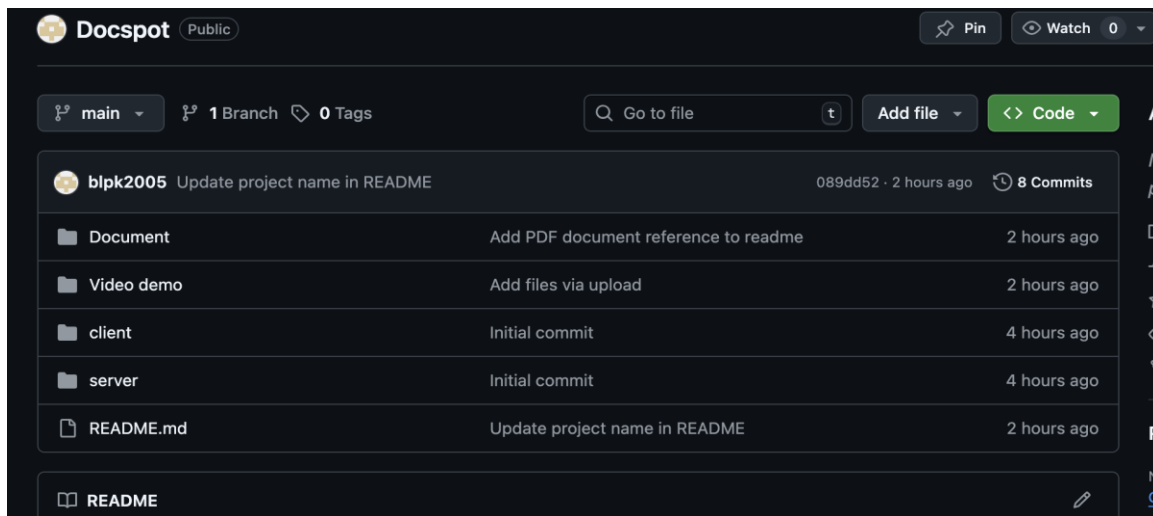
Prerequisites:

Before running the DocSpot application, ensure the following software is installed on your system:

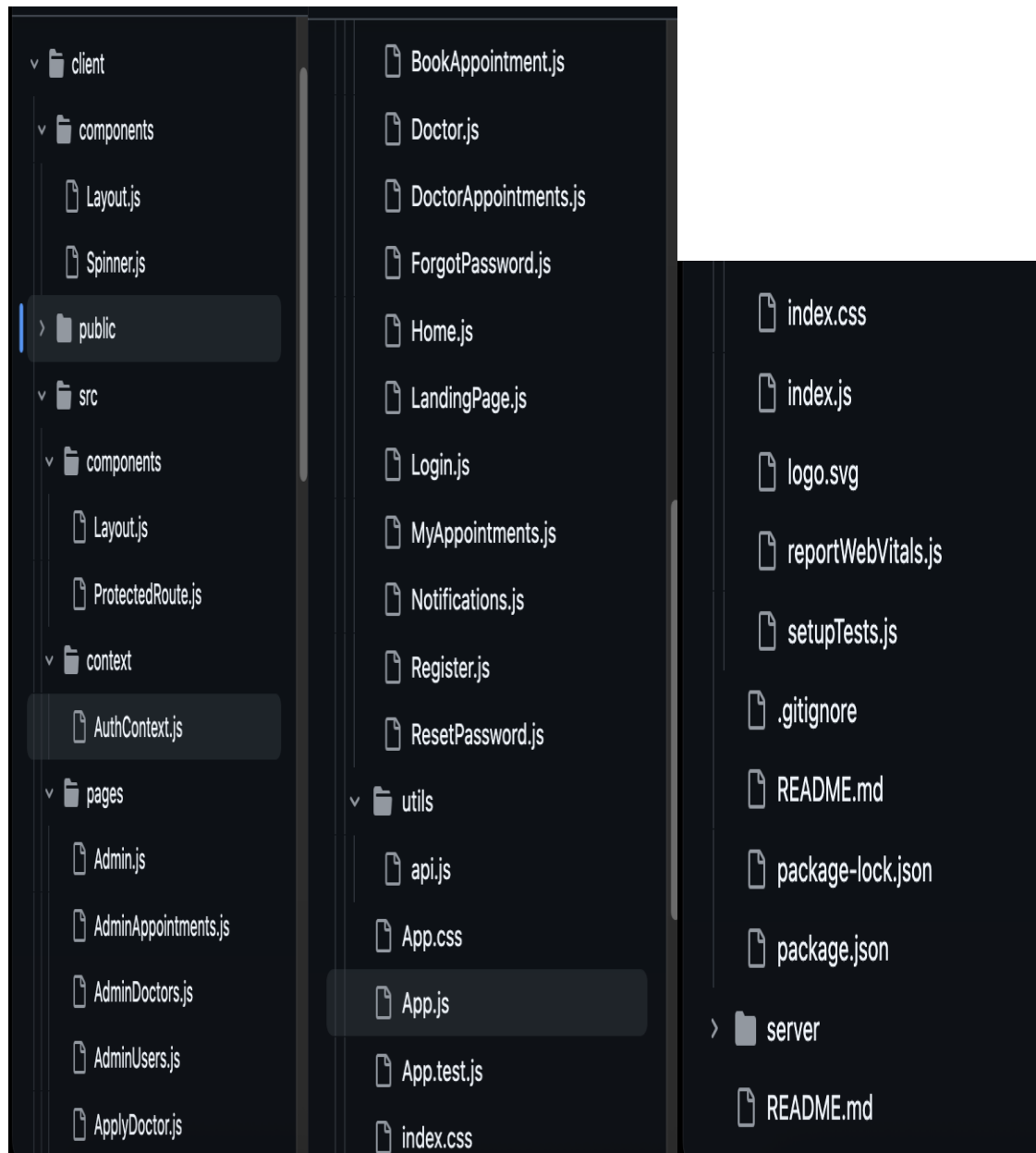
- **Node.js (v16 or above recommended):**
Required to run both the frontend (React) and backend (Node.js/Express) servers.
- **MongoDB (Local Installation or MongoDB Atlas):**
Used as the database to store user, doctor, and appointment data. You can either install MongoDB locally or use a cloud database service like MongoDB Atlas.
- **Git:**
Required to clone the project repository from GitHub and manage version control.

5. Folder Structure

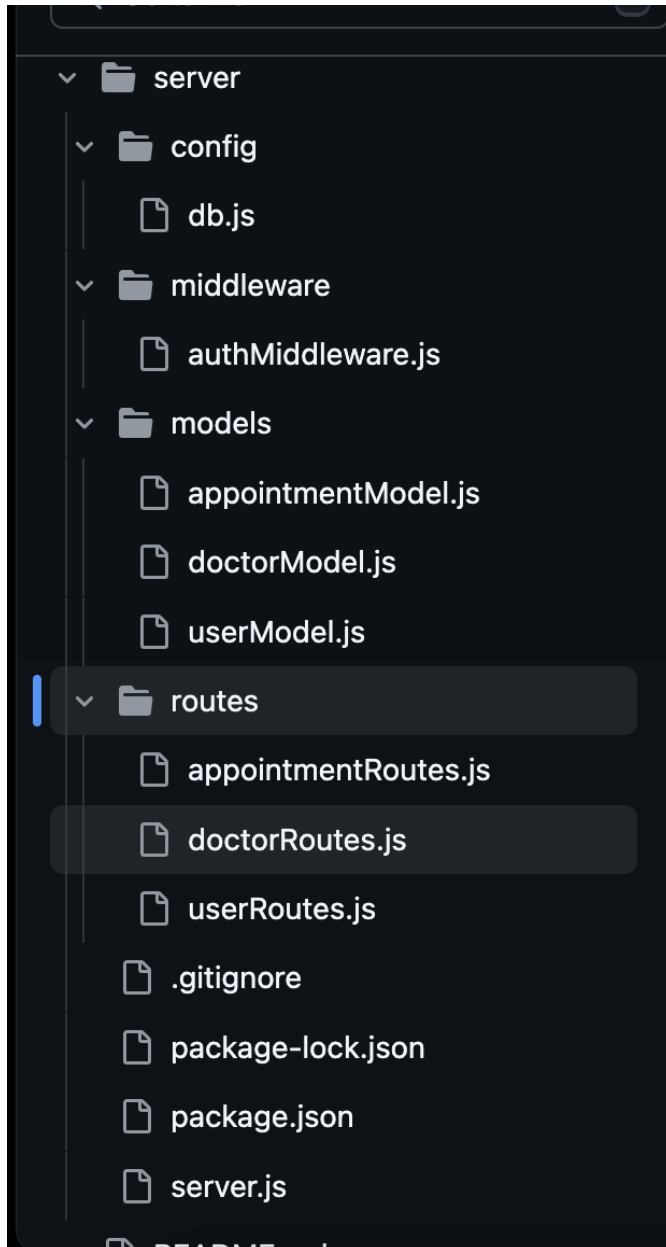
Root Directory:



Client or Frontend:



Server or Backend:



Architecture Explanation

Client Side:

- Handles UI rendering
- Protected routes using ProtectedRoute.js
- Global authentication state using AuthContext.js
- API communication handled via utils/api.js

Server Side:

- db.js → MongoDB connection
- authMiddleware.js → JWT verification
- models/ → Mongoose schemas
- routes/ → API endpoint definitions
- server.js → Express app entry point

6. Running the Application

Start Backend:

```
cd server  
npm start
```

Server runs on: <http://localhost:5000>

Start Frontend:

```
cd client  
npm start
```

Frontend runs on: <http://localhost:3000>

7. API Documentation

Authentication

Method	Endpoint	Description
POST	/api/auth/register	Register new user
POST	/api/auth/login	Login user

Doctors

Method	Endpoint	Description
GET	/api/doctors	Get all doctors
GET	/api/doctors/:id	Get doctor by ID

Appointments

Method	Endpoint	Description
POST	/api/appointments	Book appointment
GET	/api/appointments	Get user appointments

8. Authentication

- JSON Web Tokens (JWT) used
- Token stored in localStorage
- Middleware verifies token for protected routes
- Passwords hashed using bcrypt

9. User Interface

- Clean and responsive design
- Doctor cards with details
- Booking confirmation system
- Navigation bar for easy routing

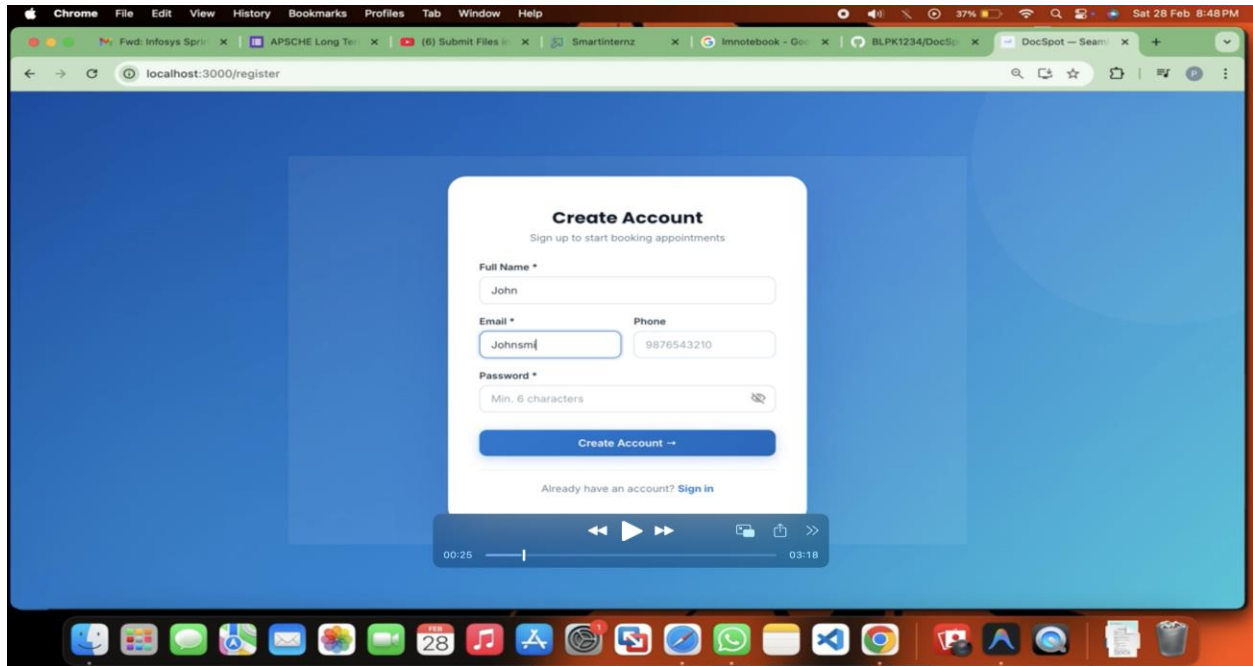
10. Testing

- Manual testing performed
- API tested using Postman
- Form validations implemented on frontend
- Error handling implemented in backend

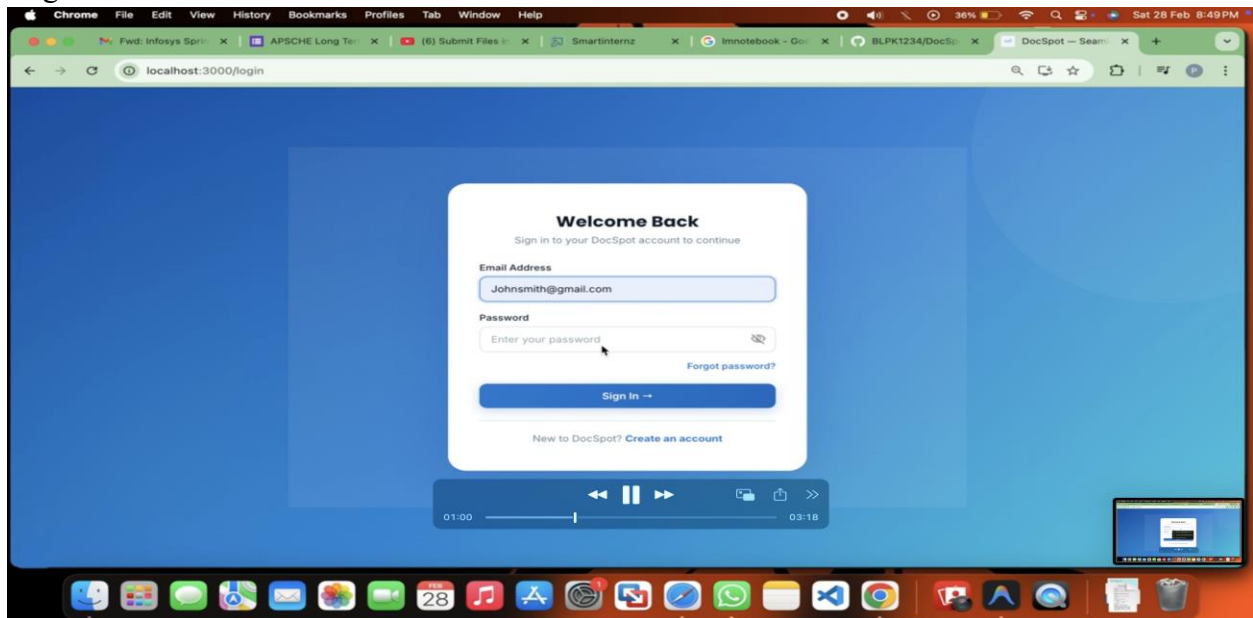
11. Screenshots or Demo

GitHub Repository: <https://github.com/blpk2005/Docspot>

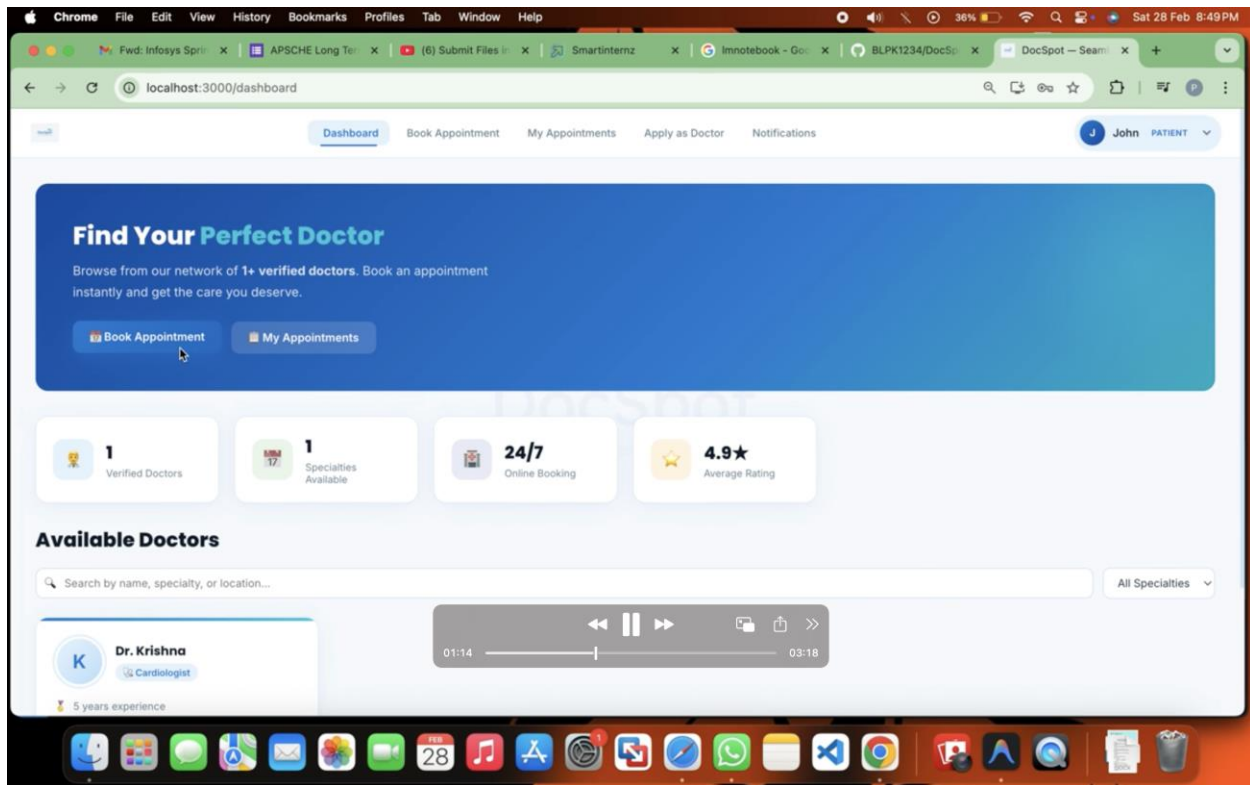
Landing Page:



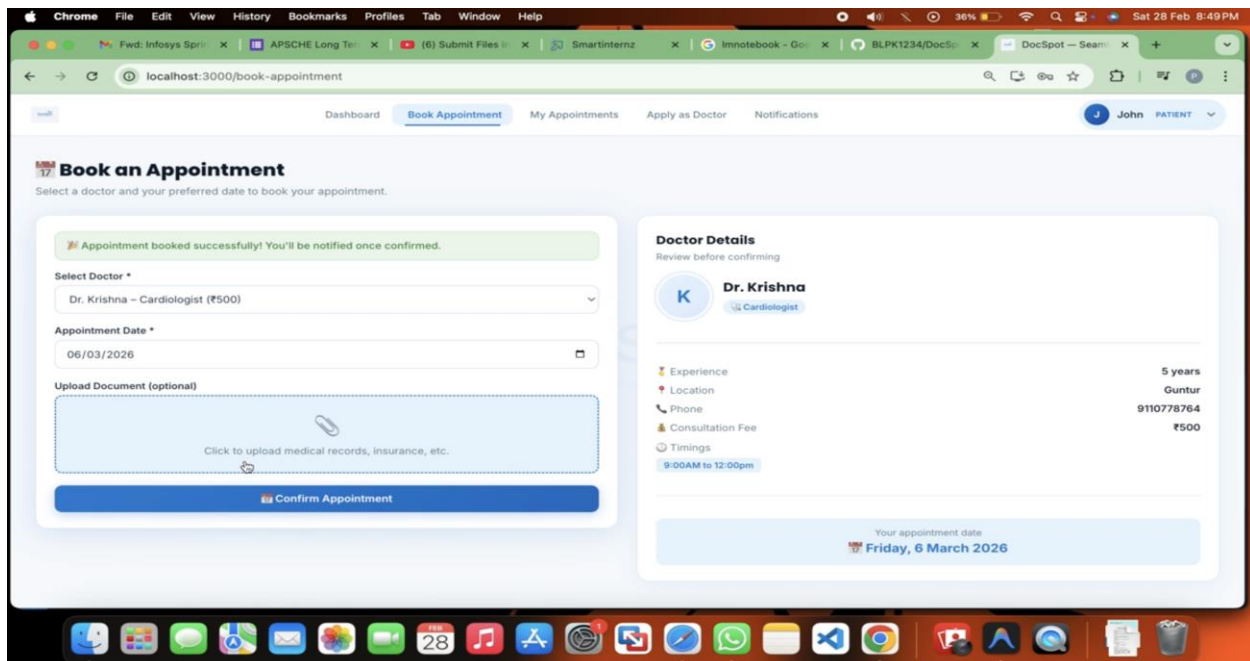
Login:



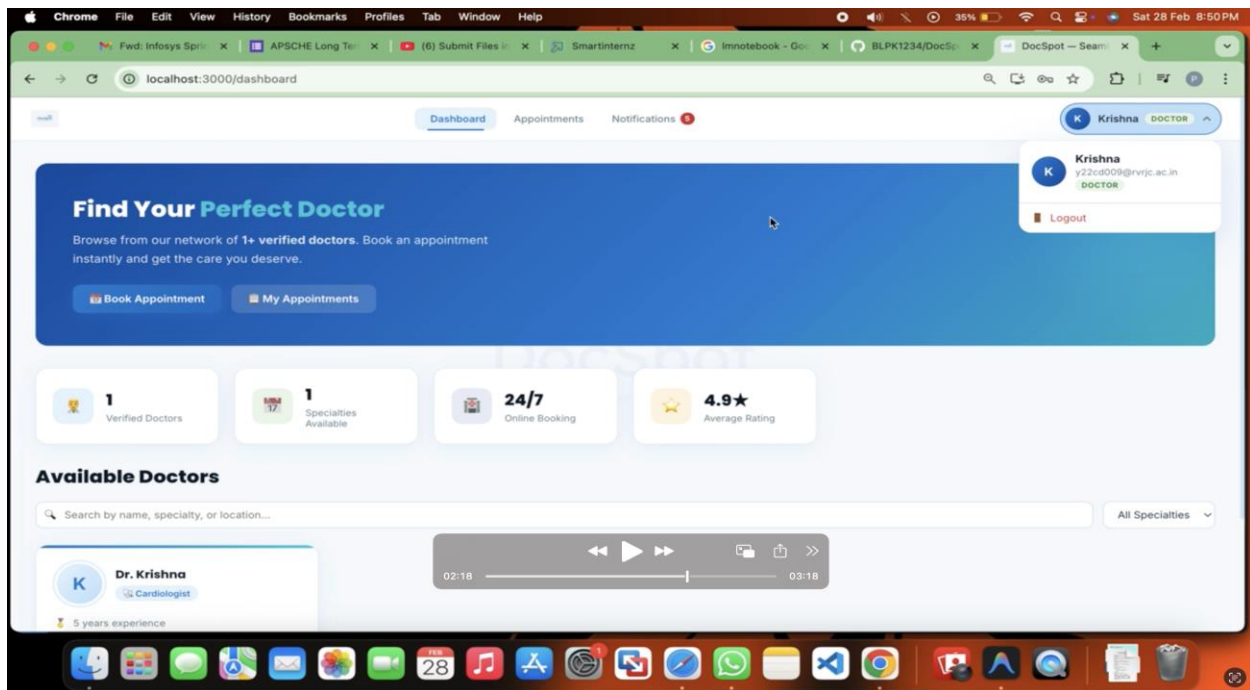
User Dashboard:



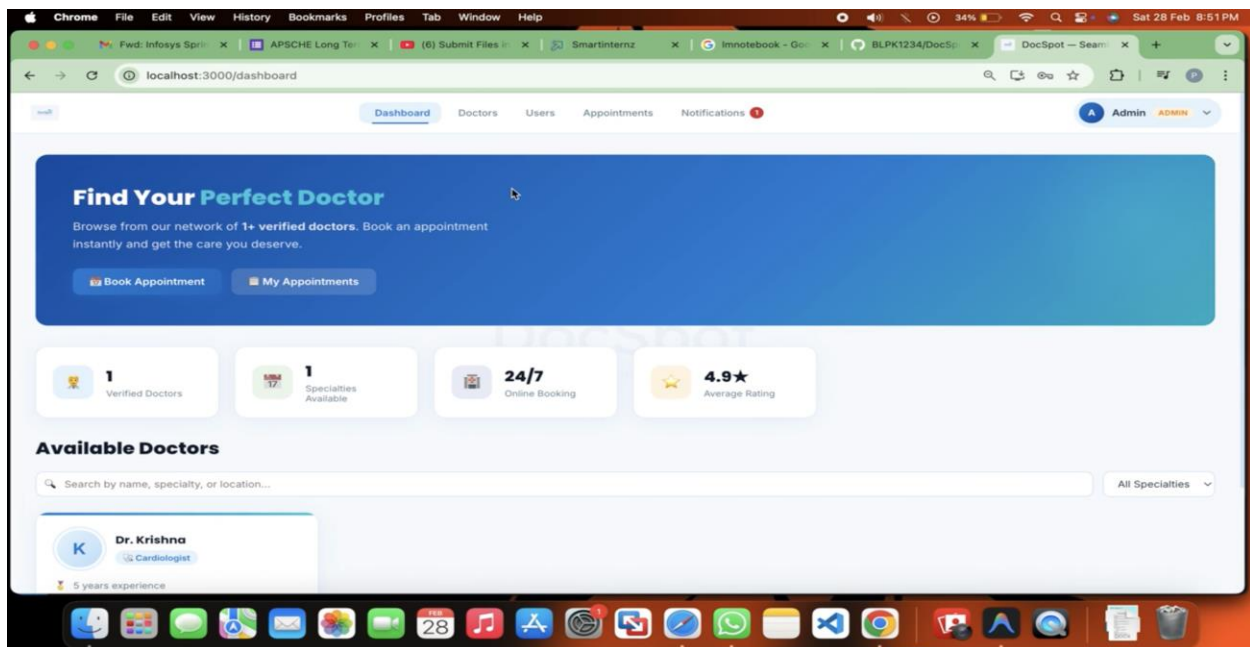
Patient or user booking a appointment:



Doctor dashboard:



Admin Dashboard:



12. Known Issues

- No email notification system
- Limited admin dashboard features
- Appointment time slot validation can be improved