Full Stack Development with MERN

Project Documentation

1. INTRODUCTION

- **Project Title**: BOOK A DOCTOR USING MERN
- **Team Members** [**Team ID**: 5601]
    - **Rudhra Y – Project Manager (Team Lead)**
    - **Sai Arjunaa A - Frontend Developer**
    - **Vignesh S – Backend Developer**
    - **Shreesh Shivalingam – Database Administrator**

2. PROJECT OVERVIEW

- **Purpose:** **MediCareBook** is a comprehensive healthcare application designed to streamline appointment booking and management for patients, doctors, and administrators. The application enhances the efficiency of healthcare services by offering a user-friendly and secure platform. **MediCareBook** is committed to excellence in healthcare technology. We continuously strive to enhance our platform, integrating the latest advancements to improve user experience and deliver superior service. Whether you're booking your first appointment or managing ongoing care, **MediCareBook** is here to support you every step of the way.

- **Features:**

  Role-specific Dashboards:
    - Patients can book appointments and check their statuses.
    - Doctors can view and approve appointment requests, as well as apply to be listed.
    - Admins can oversee all users and appointments.

  Notification System:
    - Patients are notified when doctors approve their appointments.

  Secure Data Handling:
    - All user and appointment details are stored in MongoDB and accessed securely.

User-friendly Design:

- Clean and intuitive interface with clearly defined actions for each role.

3. ARCHITECTURE

- **Frontend:**
  - Developed with React, ensuring a dynamic and responsive user interface.
  - Key components include:
    - **Home Page:** Central hub with navigation to Login, Register, and Appointment booking.
    - **Dashboard Pages:** Tailored for each user type (Patient, Doctor, Admin).
    - **Reusable Forms:** Used for Registration, Login, Appointment Booking, and Doctor Applications.

- **Backend:**
  - Built with **Node.js** and **Express.js**, offering RESTful API endpoints.
  - Key components include:
    - User authentication and role management.
    - Appointment booking and status updates.
    - Secure communication with the database.

- **Database:**
  - Relationships are managed through Mongoose models, ensuring seamless data interaction.
  - **MongoDB** stores all critical data:
    - ***users*:** Holds details about patients, doctors, and admins.
    - ***doctors*:** Stores registered doctor profiles, including specialization, experience, and availability.
    - ***appointments*:** Tracks appointment requests, their statuses, and associated documents.

4. SETUP INSTRUCTIONS

- **Prerequisites:**
  - Node.js
  - MongoDB (local or cloud-based)
  - npm (Node Package Manager)

- **Installation:**

1. Clone the repository:
   - git clone <repository_url>
2. Navigate to the project directory.

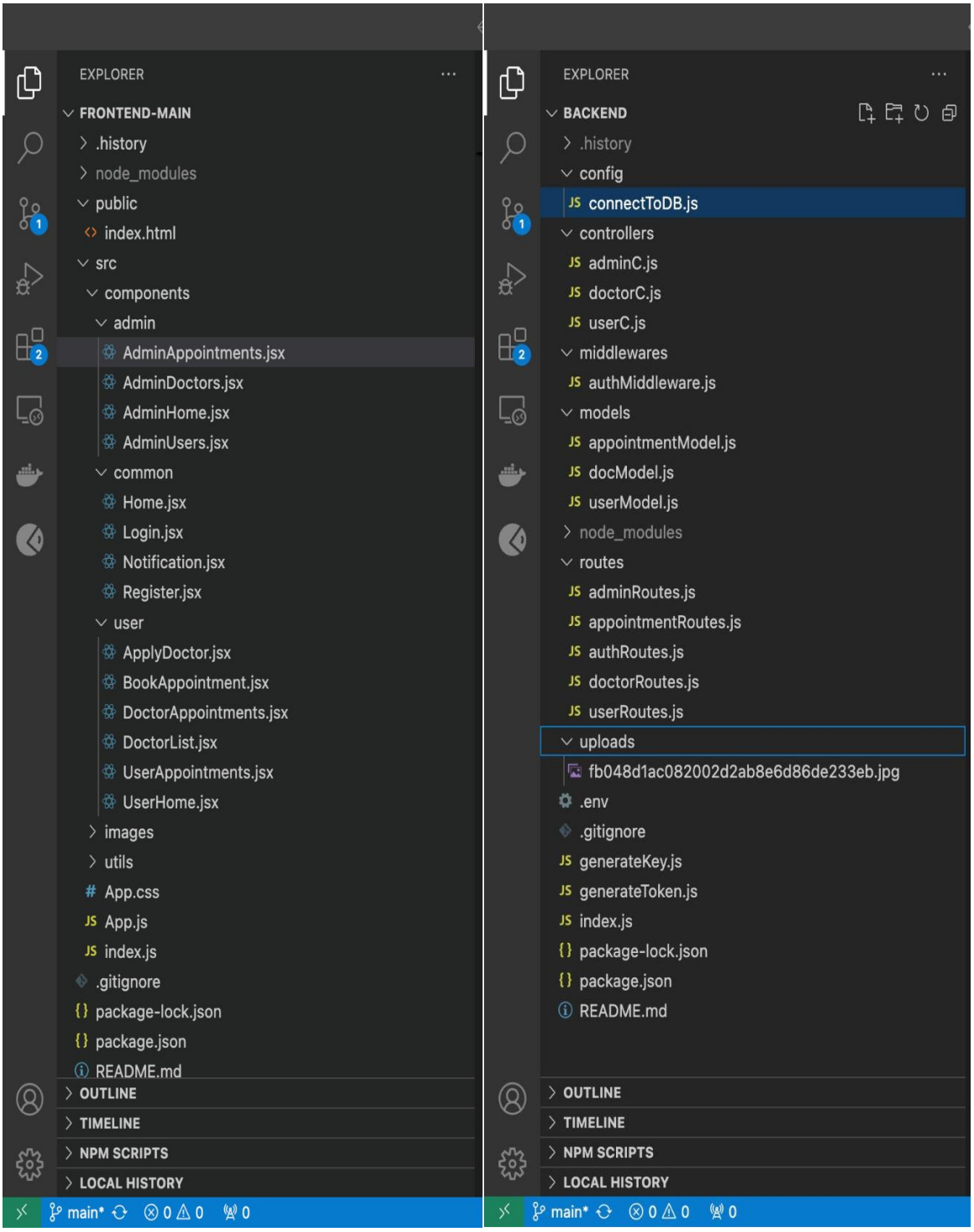
3. Install dependencies for both frontend and backend:
    - *cd frontend-main && npm install*
    - *cd ../backend && npm install*
4. Set up environment variables for database configuration.

5. FOLDER STRUCTURE

- **Frontend:**
    - **node_modules:** Contains all the npm packages required for the frontend project.
    - **public:** Includes the root HTML file (index.html).
    - **src (Source Code):**
        - **Components:**
            - **Admin:** Handles admin-related pages like managing users and appointments.
            - **Common:** Shared components like Home, Login, and Register.
            - **User:** Handles patient-related pages like booking appointments and viewing doctor details.
        - **Images:** Stores static image files used in the application.
        - **App.js:** Defines application routes and integrates all components.
        - **App.css:** Contains global CSS styles for the application.
        - **index.js:** Entry point for rendering the React app.

- **Backend:**
    - **config:**
        - ***connectToDB.js*:** Manages MongoDB connection using Mongoose.
    - **controllers:**
        - ***adminC.js*:** Logic for admin functionalities.
        - ***doctorC.js***: Logic for doctor functionalities.
        - ***userC.js***: Logic for user functionalities.
    - **middlewares:**
        - ***authMiddleware.js*:** Ensures secure authentication using JWT tokens.
    - **routes:**
        - ***adminRoutes.js*:** API routes for admin operations.
        - ***doctorRoutes.js*:** API routes for doctor operations.
        - ***userRoutes.js*:** API routes for patient operations.
    - **schemas:**
        - ***appointmentModel.js*:** Defines schema for appointment data.
        - ***docModel.js*:** Defines schema for doctor data.
        - ***userModel.js*:** Defines schema for user data.
    - **uploads:**
        - Stores uploaded files like medical reports securely.

- **. env:**
  - Contains environment variables such as database URI and JWT secret.
- **index.js:**
  - Entry point for the backend server, which initializes Express, connects to MongoDB, and registers routes.



6. RUNNING THE APPLICATION

- **Frontend:**
  - Navigate to the frontend directory and run the following command:
    - `cd frontend-main`
    - `npm start`
  - The application will launch on a local development server, typically at `http://localhost:3000`.

- **Backend:**
  - Navigate to the backend directory and start the server:
    - `cd backend`
    - `npm start`
  - The backend runs on a different port http://localhost:8000.

7. API DOCUMENTATION

Endpoint	Method	Description	Parameters/Request Body
`/api/register`	POST	Registers a new user.	Name, Email, Password, Phone, Type
`/api/login`	POST	Authenticates an existing user.	Email, Password
`/api/doctors`	GET	Fetches the list of doctors.	N/A
`/api/appointments`	POST	Books a new appointment.	Date, Time, Document

8. AUTHENTICATION

- **JWT Authentication:**

  - Upon login, the user receives a token.
  - Each user receives a JWT token upon successful login.
  - Tokens are used to validate API requests.
  - The token is included in headers for all secure requests to verify identity.
  - Role-based access ensures appropriate functionalities for patients, doctors, and admins.

9. USER INTERFACE

- **Key Screens:**
    - **Home Page:** Includes navigation buttons (**Home**, **Login**, **Register**) and a prominent **Book Appointment** button.
    - **Registration Page:** Separate forms for patients, doctors, and admins, collecting essential details.
    - **Login Page:** Secure login with email and password.
    - **Patient Features:**
        - View doctor profiles.
        - Book appointments by specifying date, time, and uploading medical reports.
        - Gets notified once the appointment is approved by the Doctor.
    - **Doctor Features:**
        - View pending appointment requests.
        - Apply to be a registered doctor with professional details.
        - Approve appointment requests.
    - **Admin Features:**
        - Comprehensive view all users and their roles.
        - Monitor all appointments and manage system data.

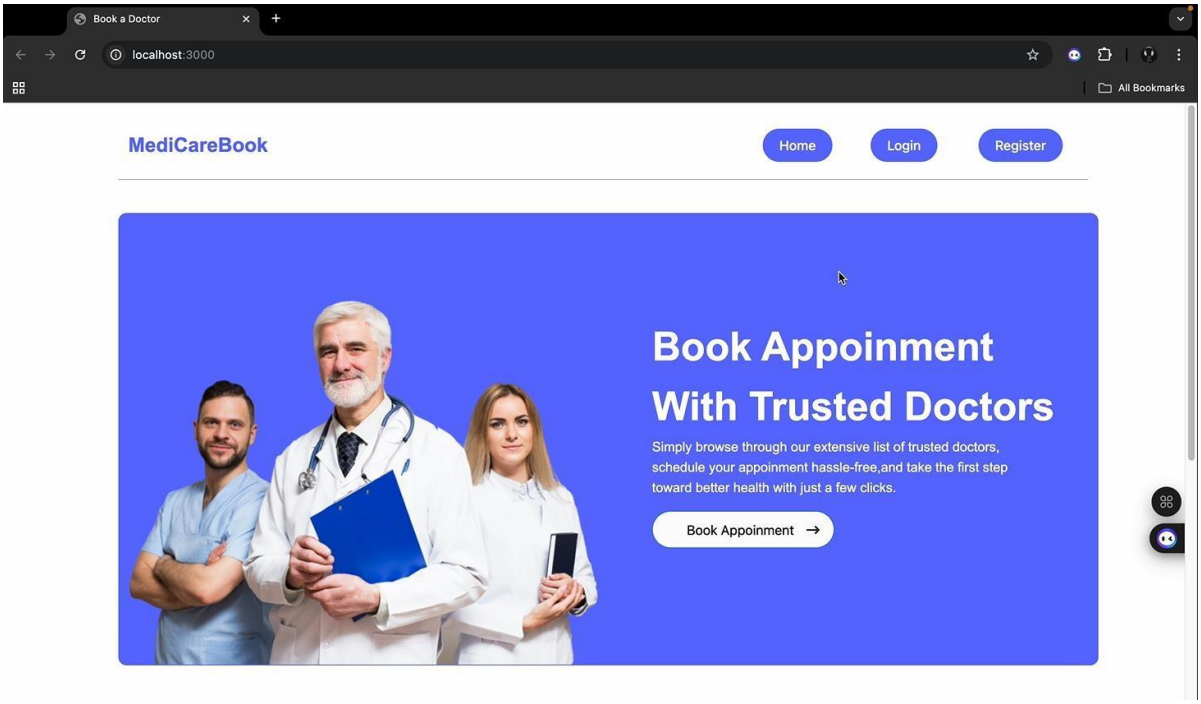
10. TESTING

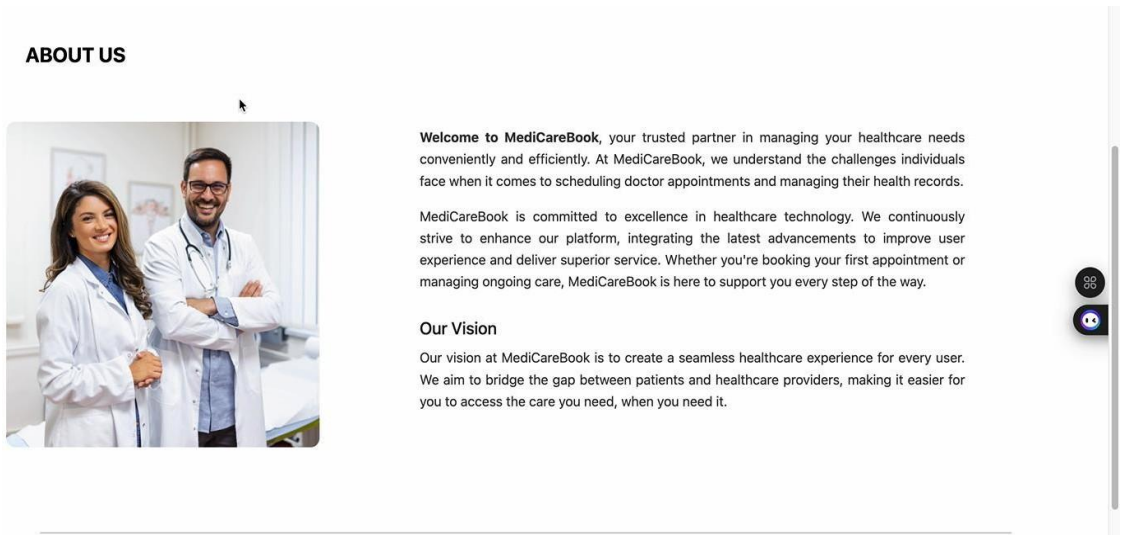
- **Manual Testing:**
    - **Verified critical workflows:**
        - User registration
        - Login flow
        - Role-based access
    - **Verified major workflows:**
        - Appointment booking
        - Approval processes
        - Database enteries for each action

- **Database Validation:**
    - Ensured data integrity in MongoDB
    - Checked MongoDB collections for:
        - Accurate data storage
        - Retrieval

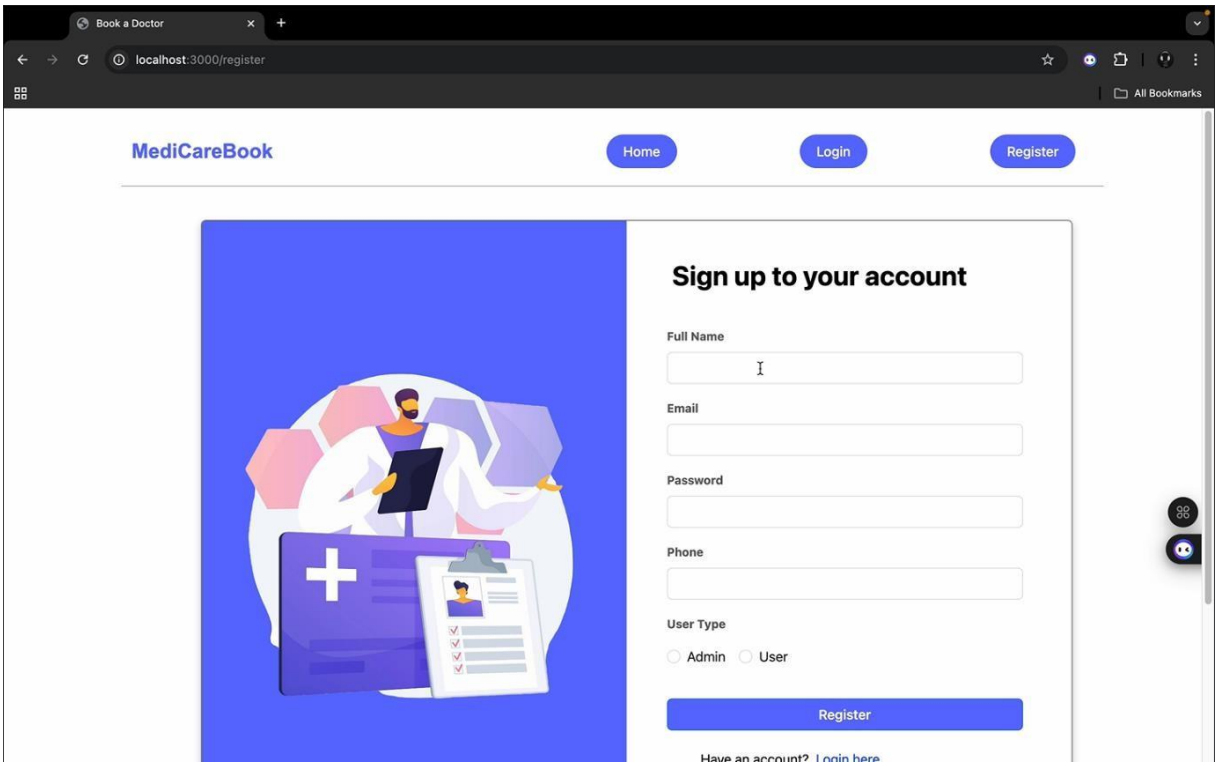
11. SCREENSHOTS OR DEMO

- **Home page**

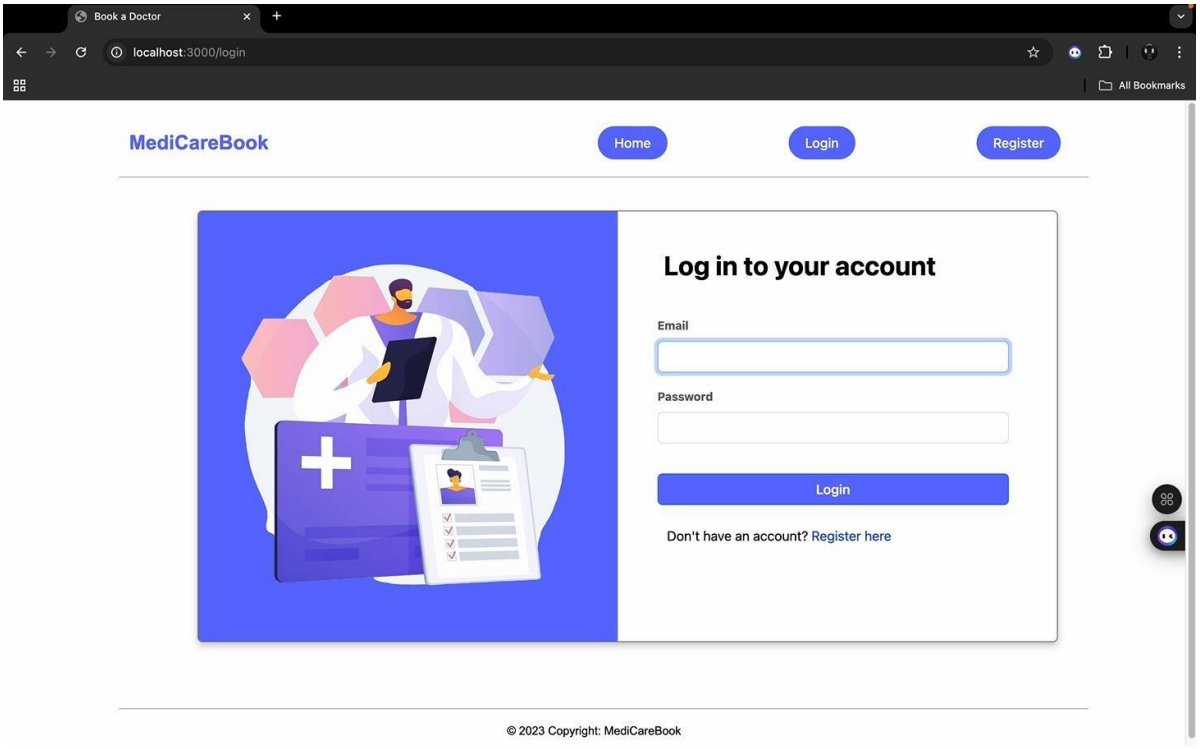




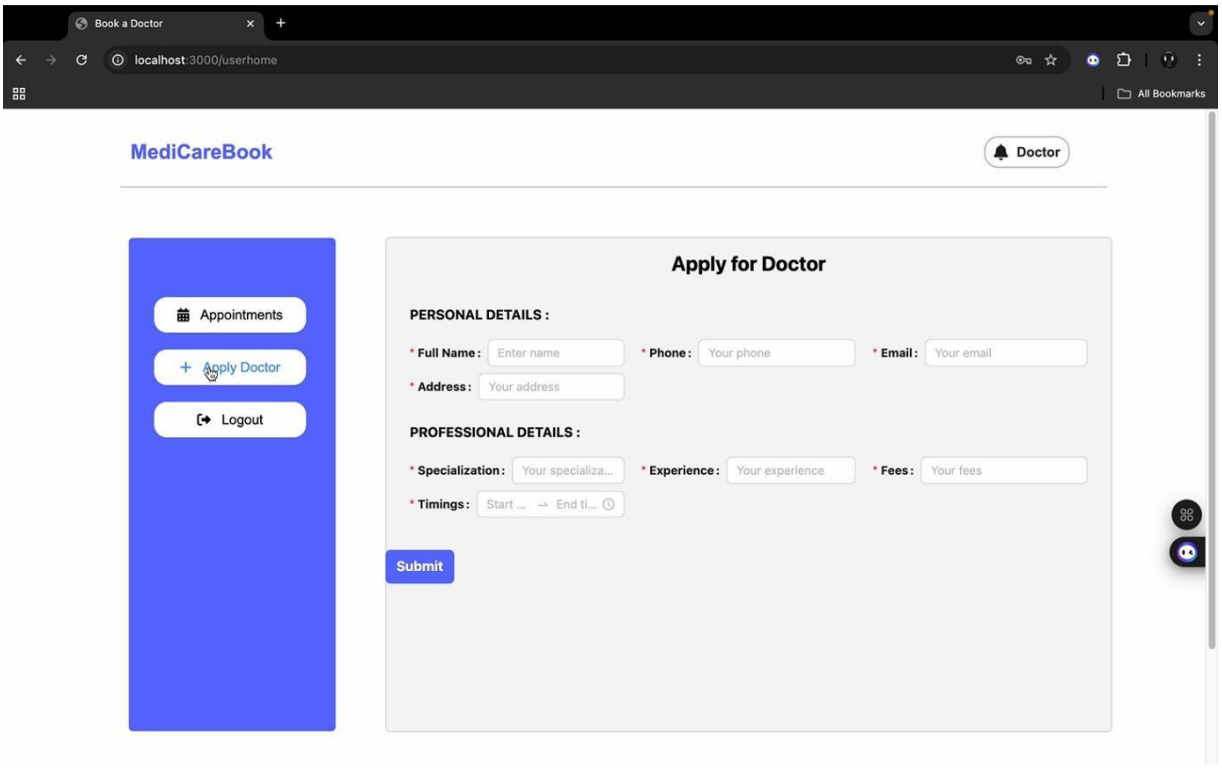
- **Register page**

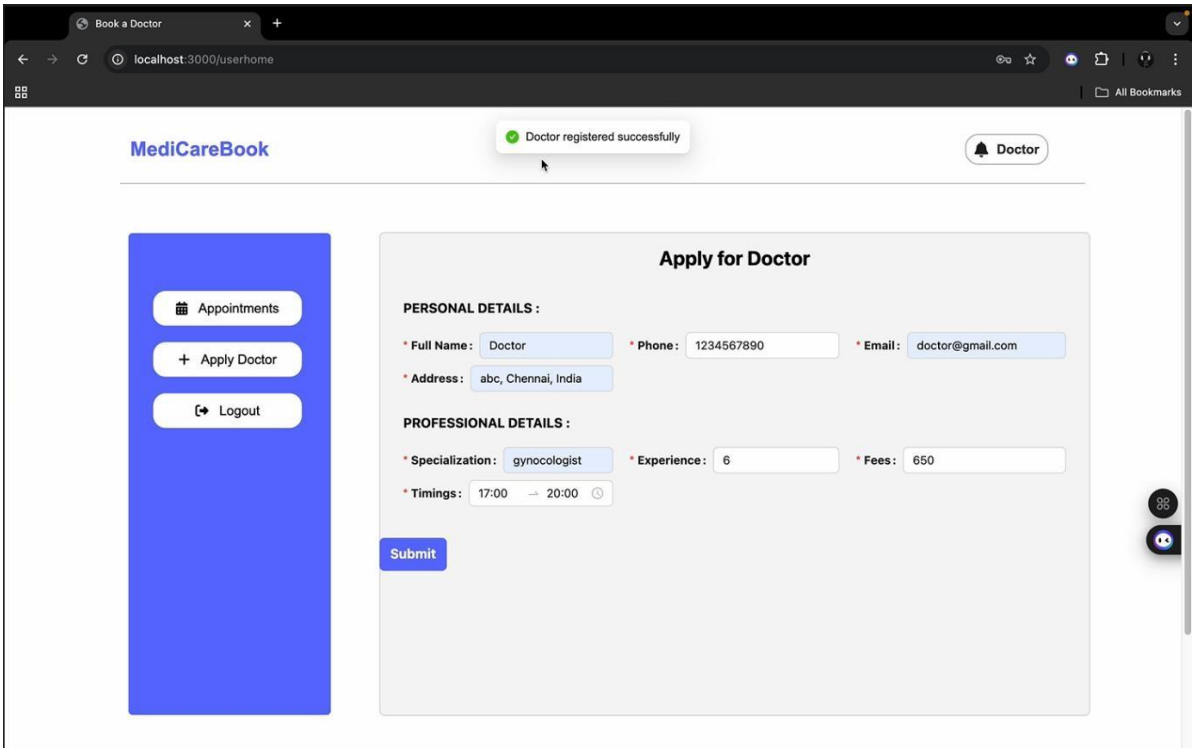


- **Login page**

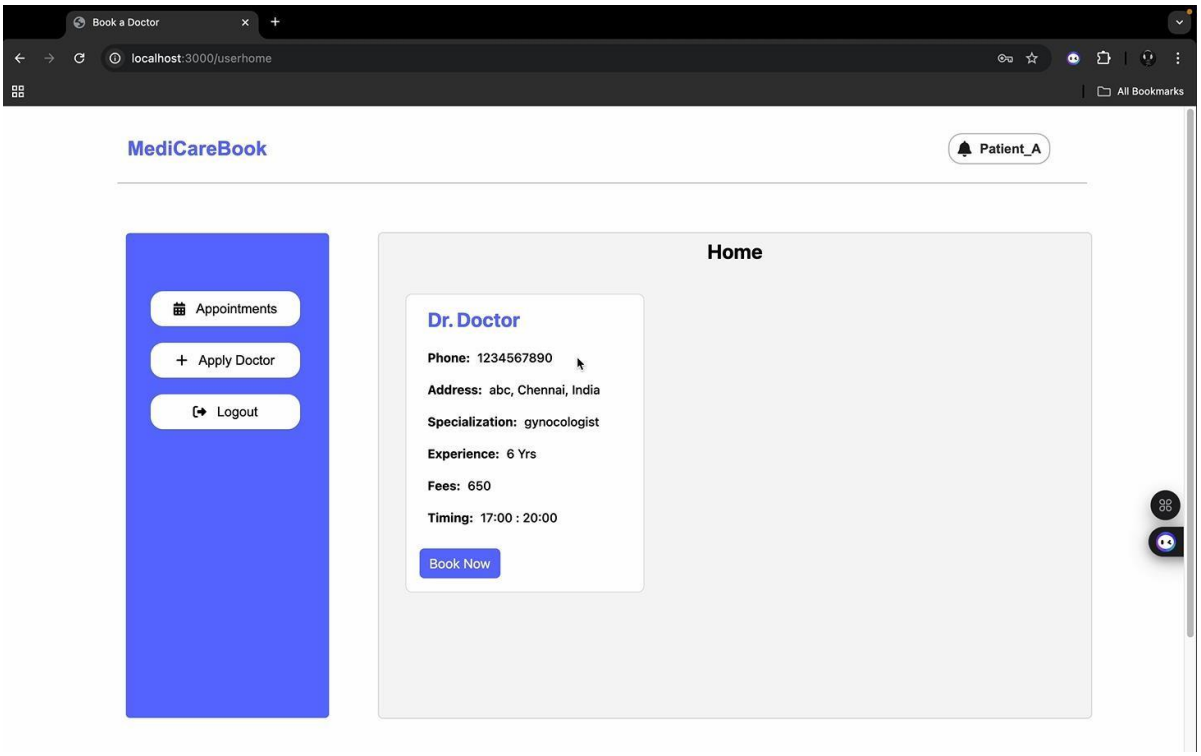


- **Apply for Doctor page**

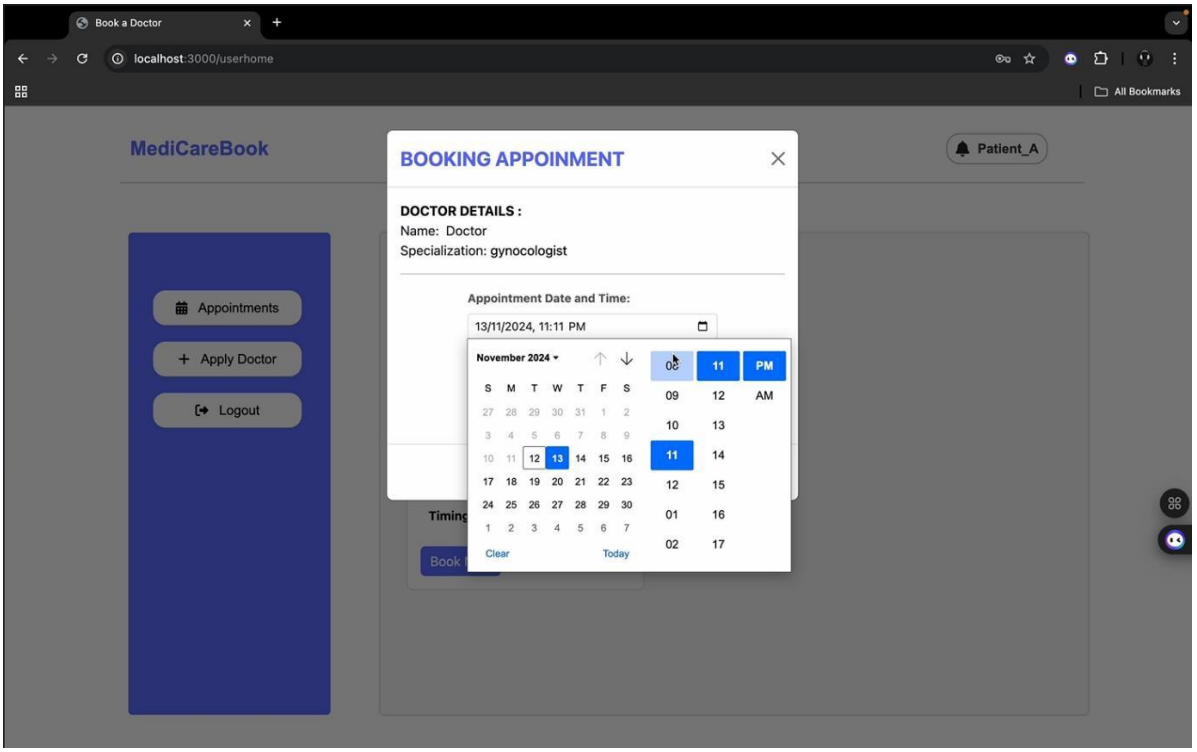




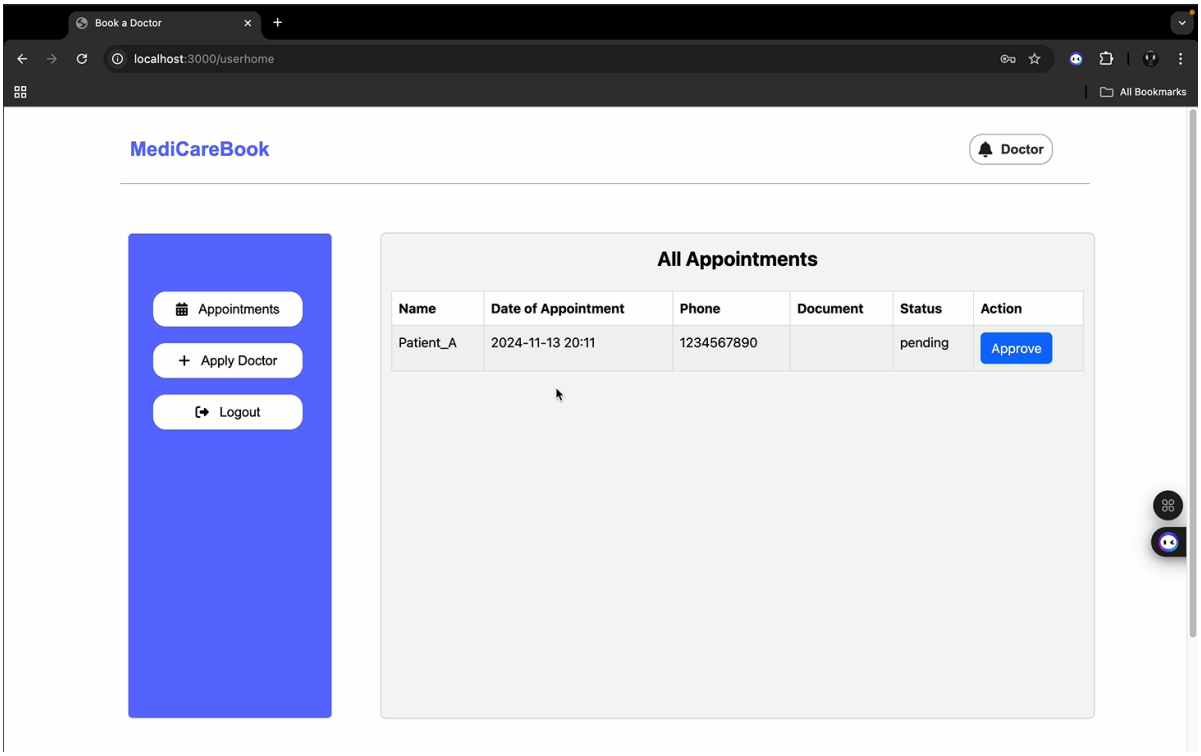
- **Patient dashboard**



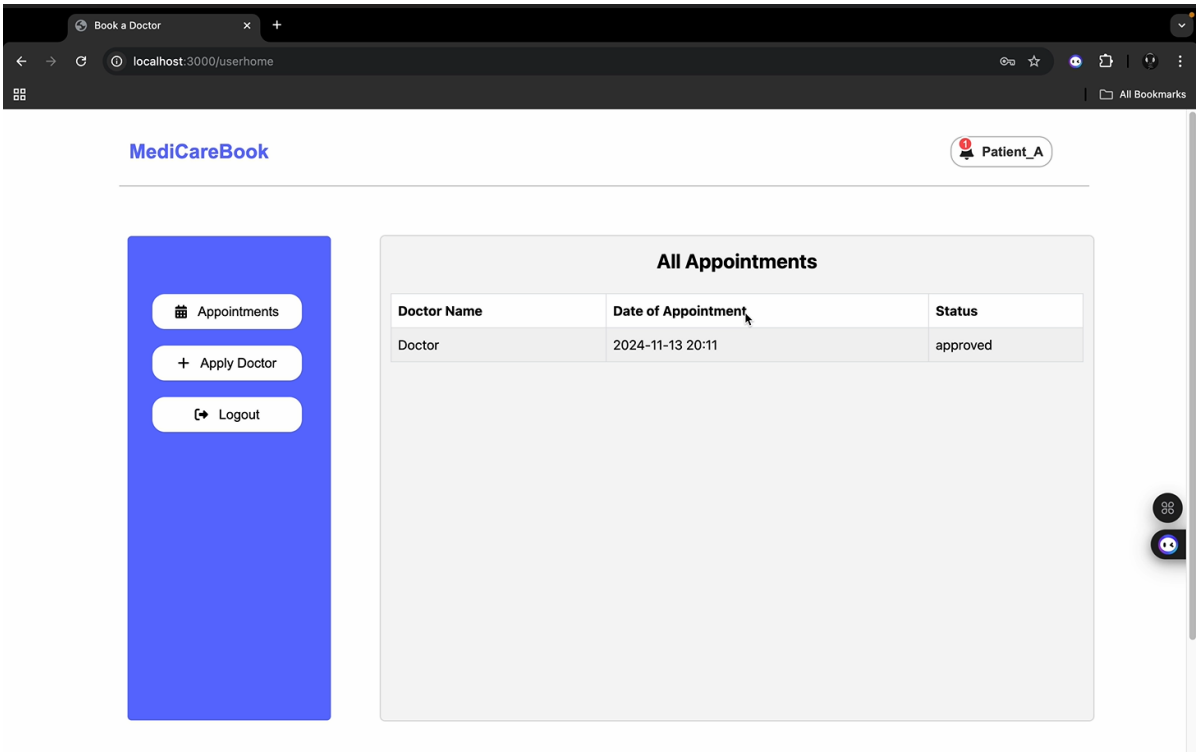
- **Appointment booking page**



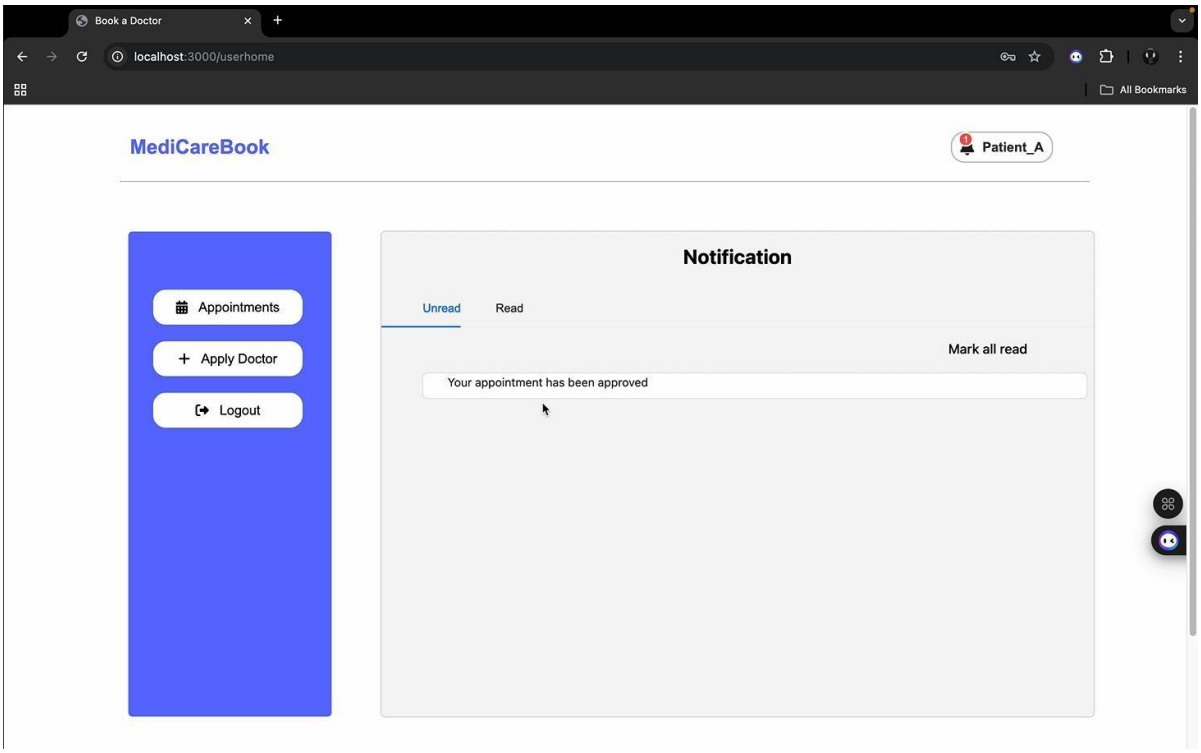
- **Doctor’s dashboard**



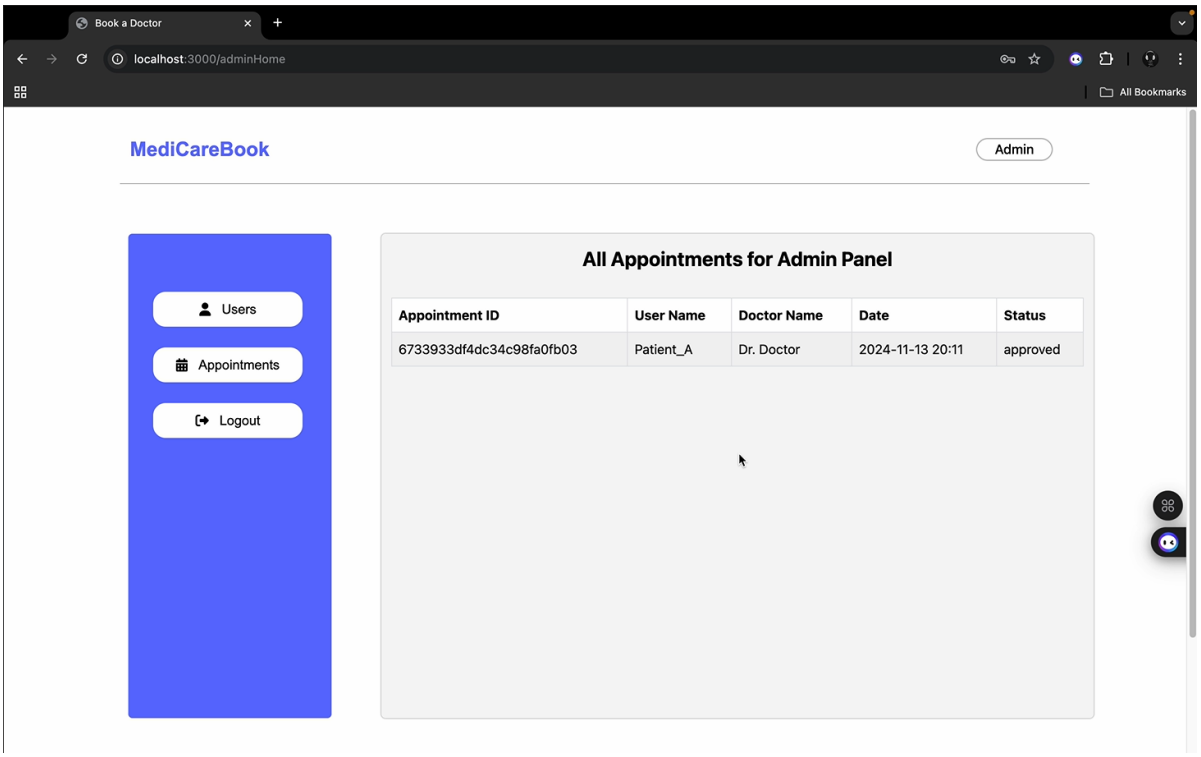
- **Patient’s appointments page**



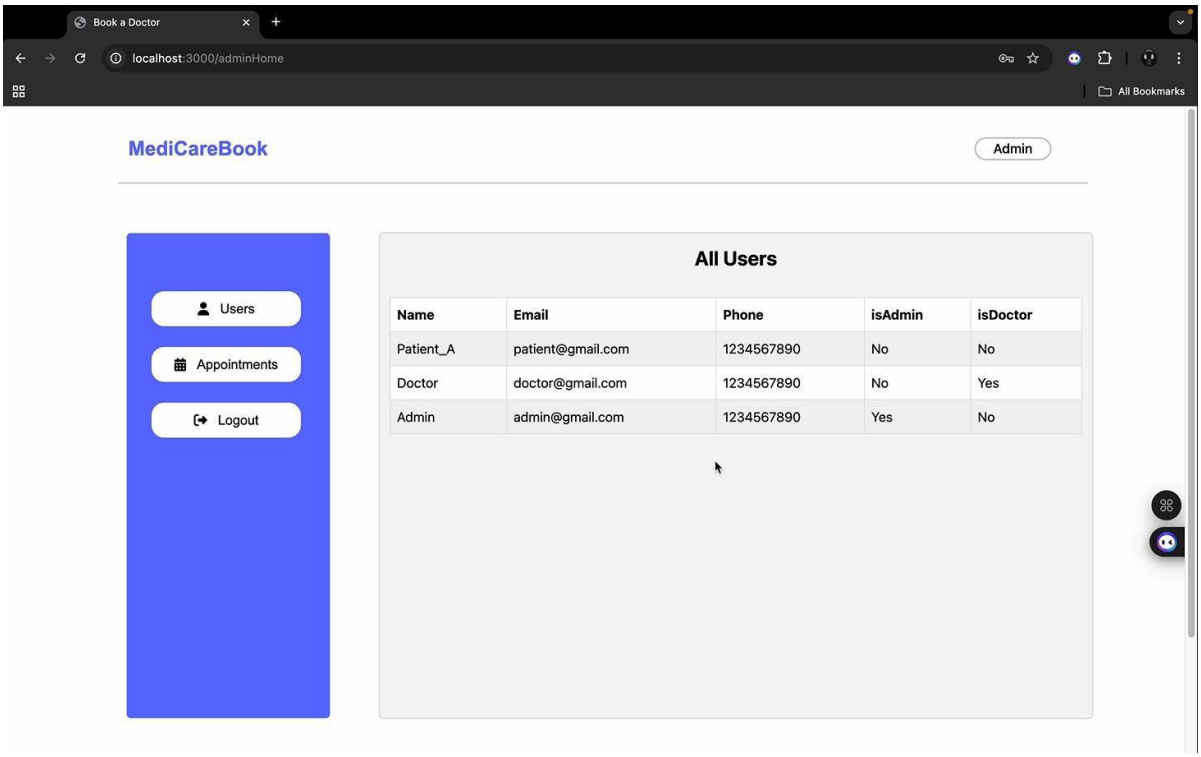
- **Patient's notification page**



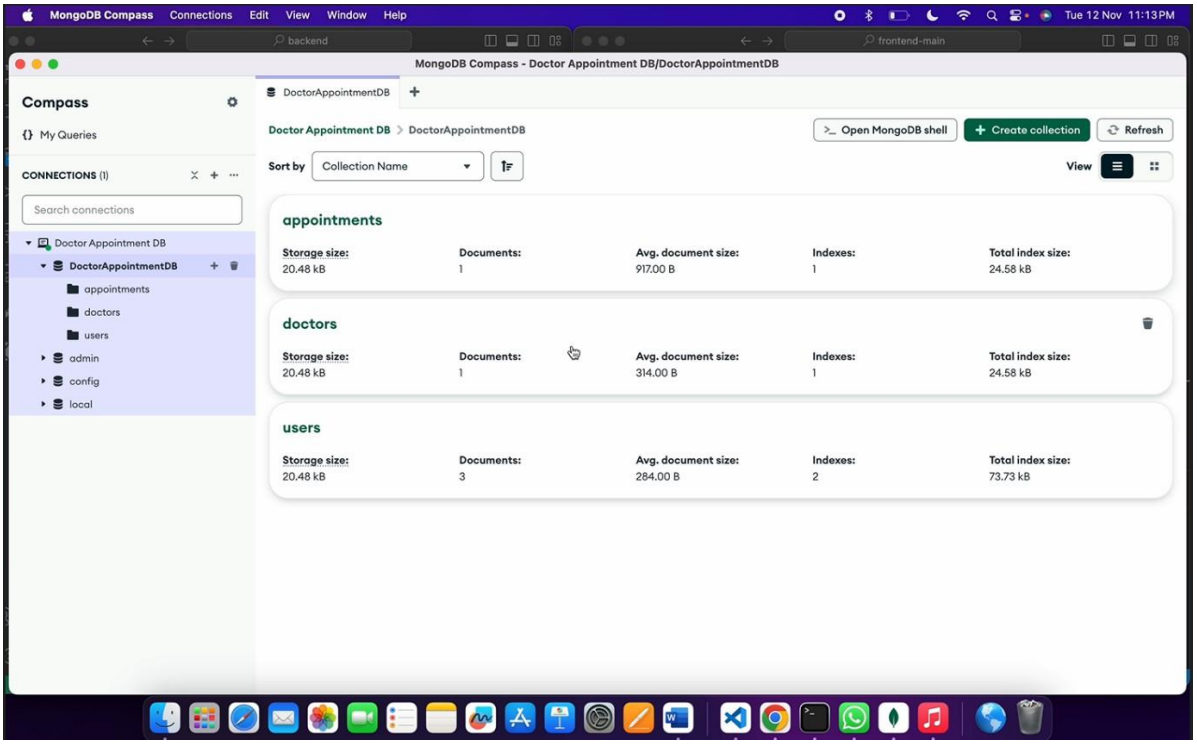
- **Admin panel**



- **Admin's all users page**



- **MongoDB connection**



The demo of the app is available at:

https://drive.google.com/drive/folders/16HFlRyuKeZfp3h4kSSR75J02ElpPZk6E?usp=sharing

12. KNOWN ISSUES

- Occasional delays in notification updates.
- Limited input validation on the frontend.

13. FUTURE ENHANCEMENTS

- **Real-time Features:**
  - Introduce WebSocket for instant notifications about appointments.
- **Advanced Search:**
  - Allow patients to filter doctors by specialization, availability, and location.
- **Online Payments:**
  - Integrate a secure payment gateway for booking consultations.
- **Mobile App Support:**
  - Extend MediCareBook to Android and iOS platforms.

14. CONCLUSION

The **MediCareBook** application provides an efficient solution for booking and managing doctor appointments. By leveraging the MERN stack, the project demonstrates the effective use of modern technologies to address real-world problems. The application is user-friendly, scalable, and adaptable for future improvements, making it a valuable asset for healthcare services.