**Naan Mudhalvan Project Report**

**1**. **Introduction**

* **Project Title:** Book a doctor using MERN
* **Team Members:**

Team ID: NM2024TMID12016 **Vignesh B** - Project Coordinator (Team lead)

**Balamurugan V** - Backend Developer **Udhayakumar U** - Frontend Developer

**Kantha Priya R** - Quality Analyst

**2. Project Overview**

* **Purpose:**  
  The purpose of the Book a Doctor project is to create a seamless platform for booking appointments with doctors based on specialization, location, and availability.
* **Features:**
  + User registration and login for patients and doctors.
  + Doctor profiles and specialization-based search.
  + Appointment scheduling and notification system.
  + Role-based access (Admin, User, Doctor).

## 3. Architecture

* Frontend: React.js is used to create dynamic, user-friendly interfaces. Components like Login, Registration, and Dashboard are designed to ensure smooth navigation.
* Backend: Node.js with Express.js handles API routes, authentication, and data flow between the database and frontend.
* Database: MongoDB is used for data storage with schemas for User, Appointment, and Notification. It ensures structured data management.

## 4. Setup Instructions

* **Prerequisites**:
* - Node.js (v16+).
* - MongoDB (Cloud-based or local instance).
* - npm package manager.
* **Installation**:
* 1. Clone the repository: `git clone [repository-link]`.
* 2. Navigate to the project directory and install dependencies: `npm install`.
* 3. Set up environment variables in a `.env` file (e.g., MongoDB URI, JWT\_SECRET).
* 4. Run `npm start` in both frontend and backend directories to start the project.

## 5. Folder Structure

* **Client**: The React frontend is organized into components like Login, Register, Dashboard, and Appointment.
* **Server**: The Node.js backend has routes for authentication, user management, and appointments. It includes models for database schemas.

## 6. Running the Application

* **Frontend**: Run `npm start` in the client directory.
* **Backend**: Run `npm start` in the server directory.

Ensure MongoDB is running and environment variables are properly configured

## 7. API Documentation

* **Endpoints**:
* - `/api/auth/register`: Registers a new user. [POST]
* - `/api/auth/login`: Authenticates a user. [POST]
* - `/api/appointments`: Manages appointment scheduling. [GET, POST, DELETE]
* - `/api/notifications`: Fetches notifications for a user. [GET]
* Each endpoint includes proper validation and error handling.

## 8. Authentication

* **Authentication**: JWT (JSON Web Tokens) is used for secure authentication.
* **Authorization**: Role-based access control (RBAC) ensures only authorized roles can access specific routes.
* **Sessions**: Tokens are stored in HTTP-only cookies for security.

## 9. User Interface

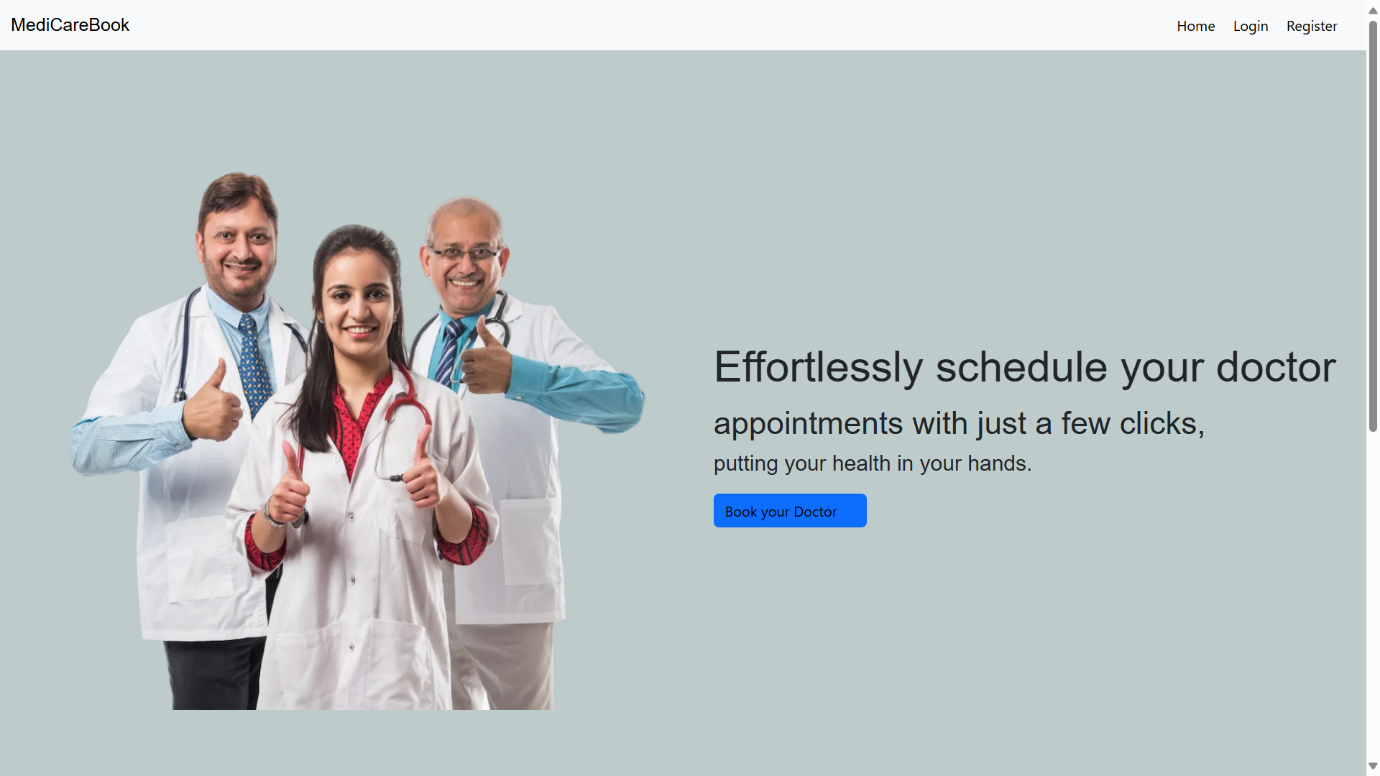
* The user interface includes:
* - Login and Registration pages for user authentication.
* - Dashboard for users to view doctors and book appointments.
* - Notifications panel to track updates.
* - Admin panel for managing doctors and users.

## 10. Testing

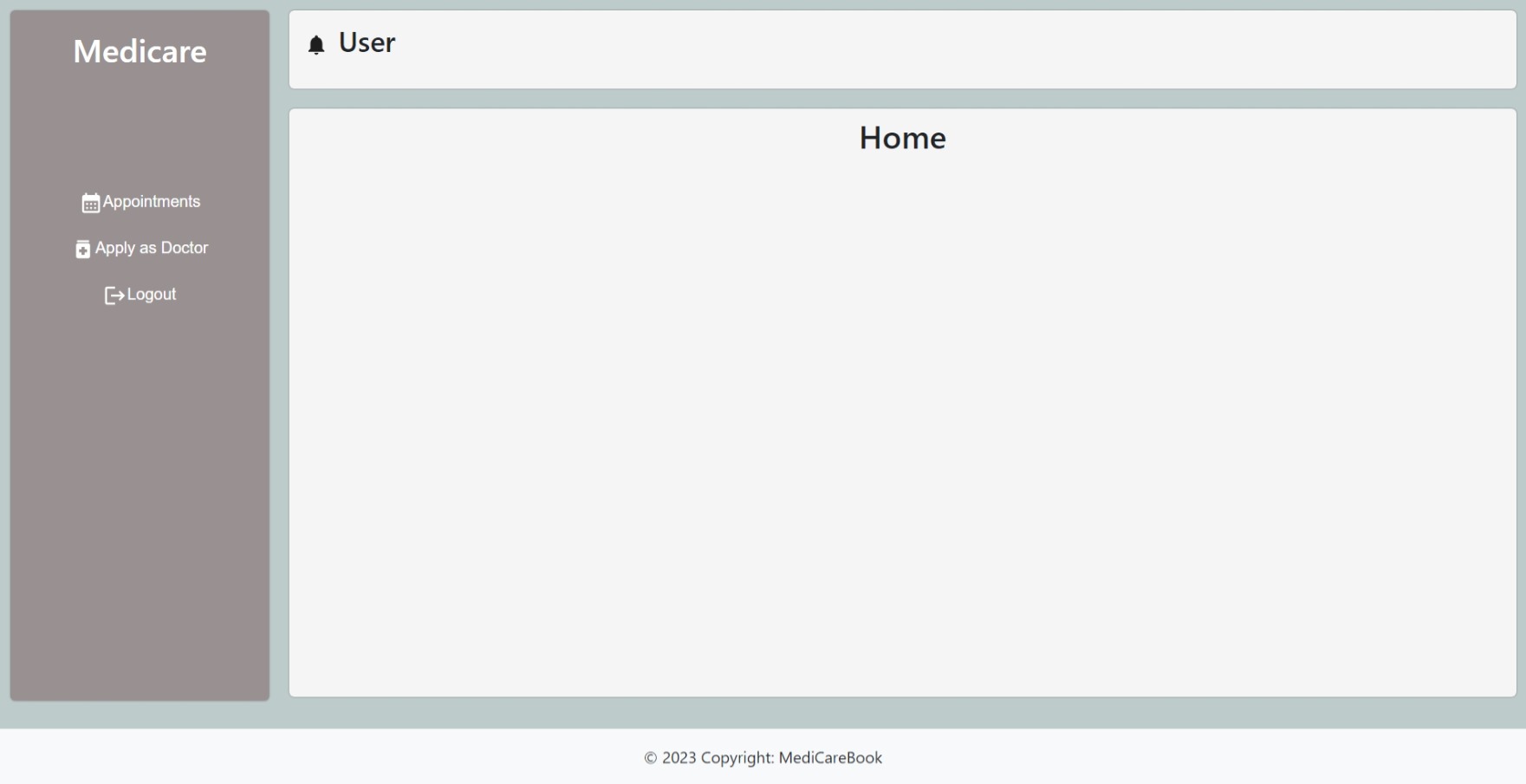
* **Testing Strategy**: Manual and automated testing.
* **Tools Used**: Postman for API testing, Jest for unit testing frontend and backend logic.
* **Focus Areas**: API responses, role-based access control, and frontend responsiveness.

## 11. Screenshots or Demo

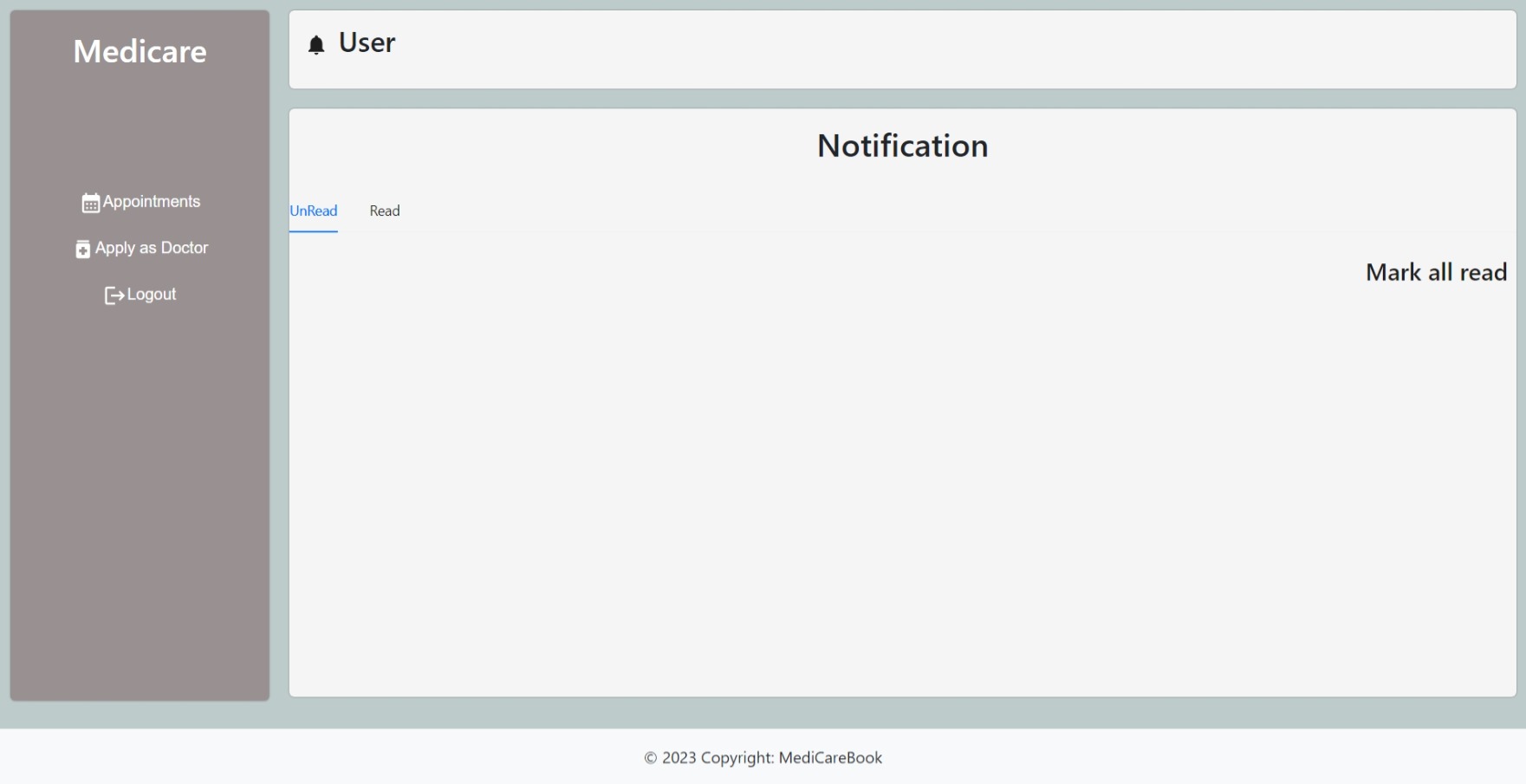
* Include screenshots of:
* - Login page.



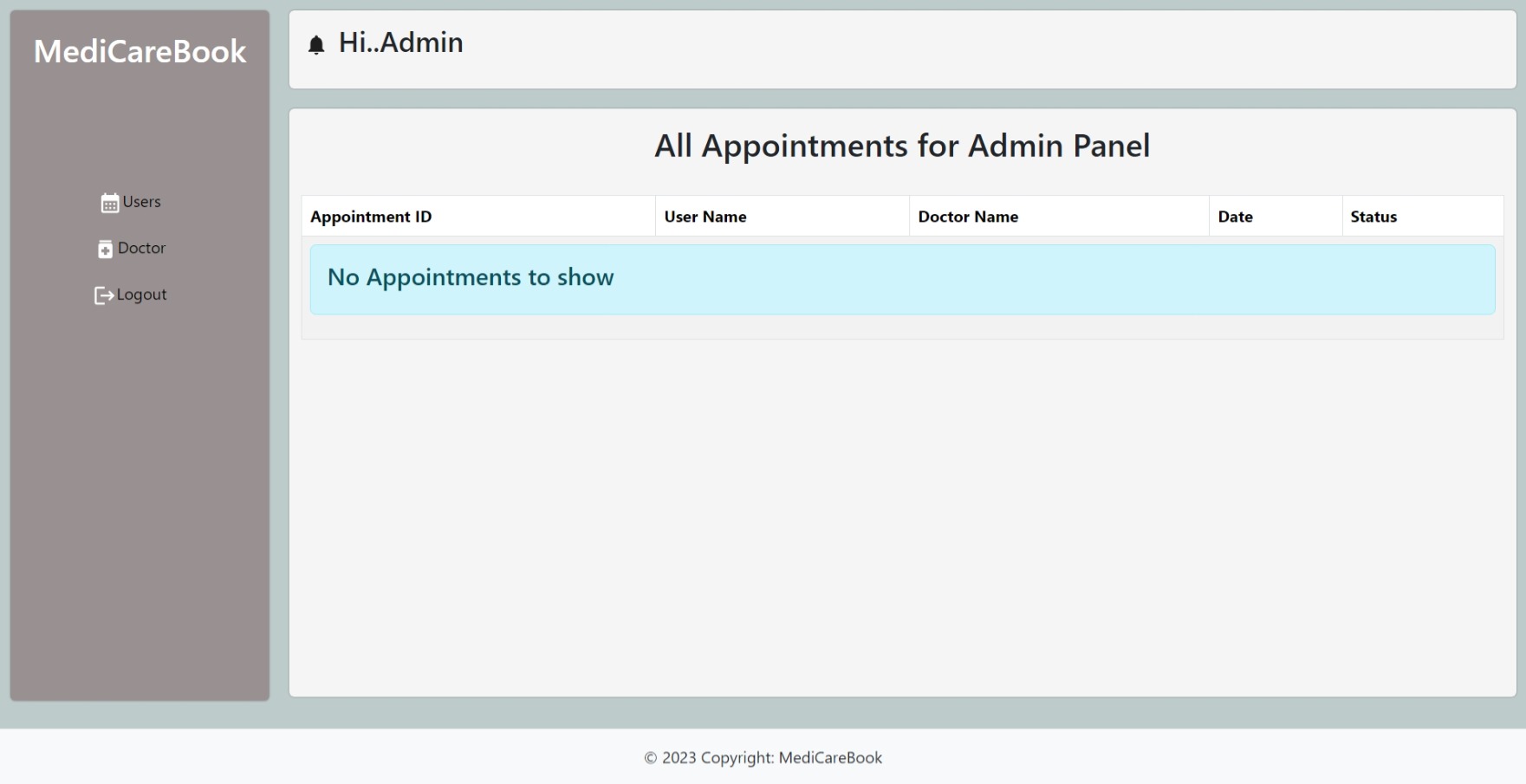
* - User dashboard



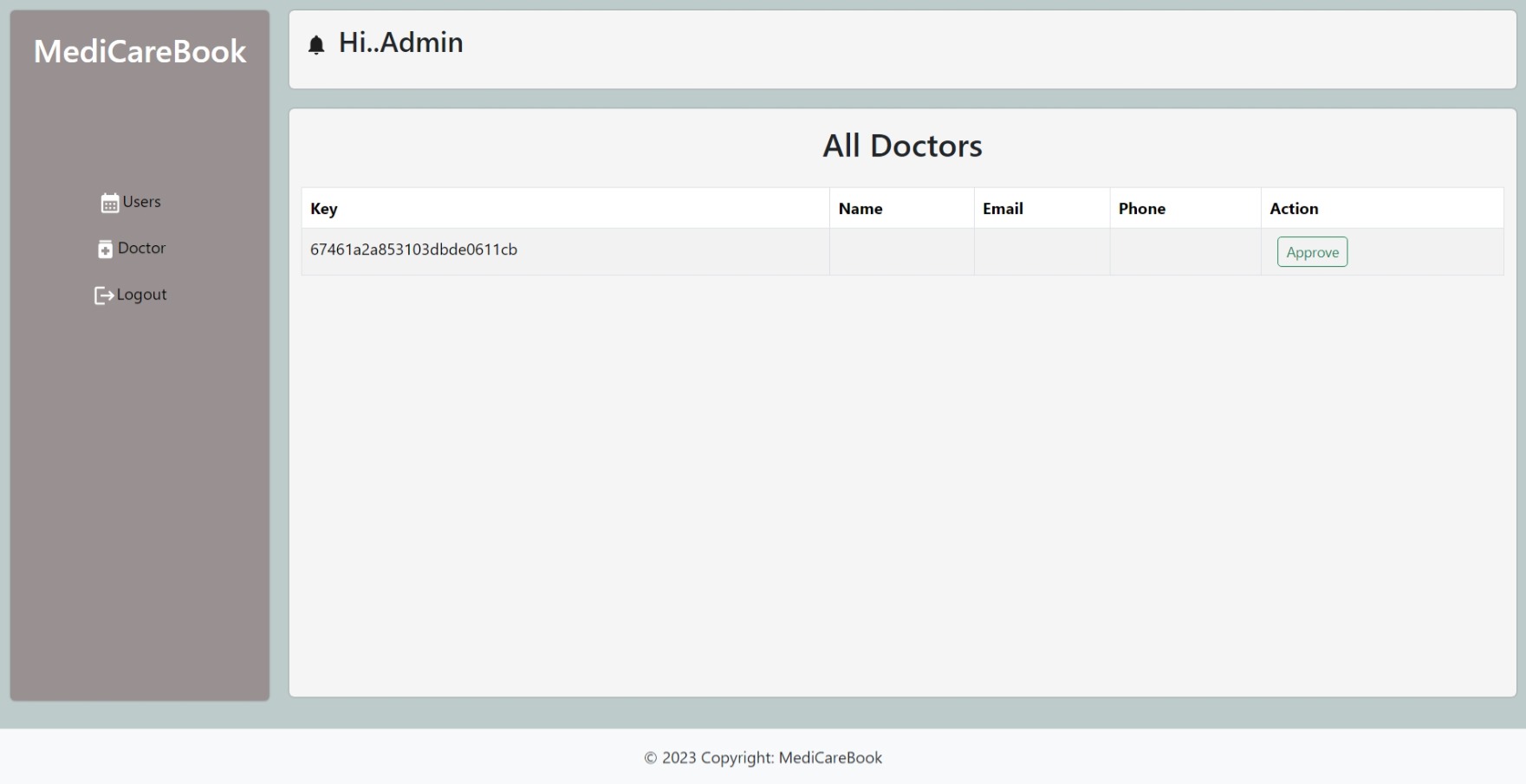
* - Notification system.



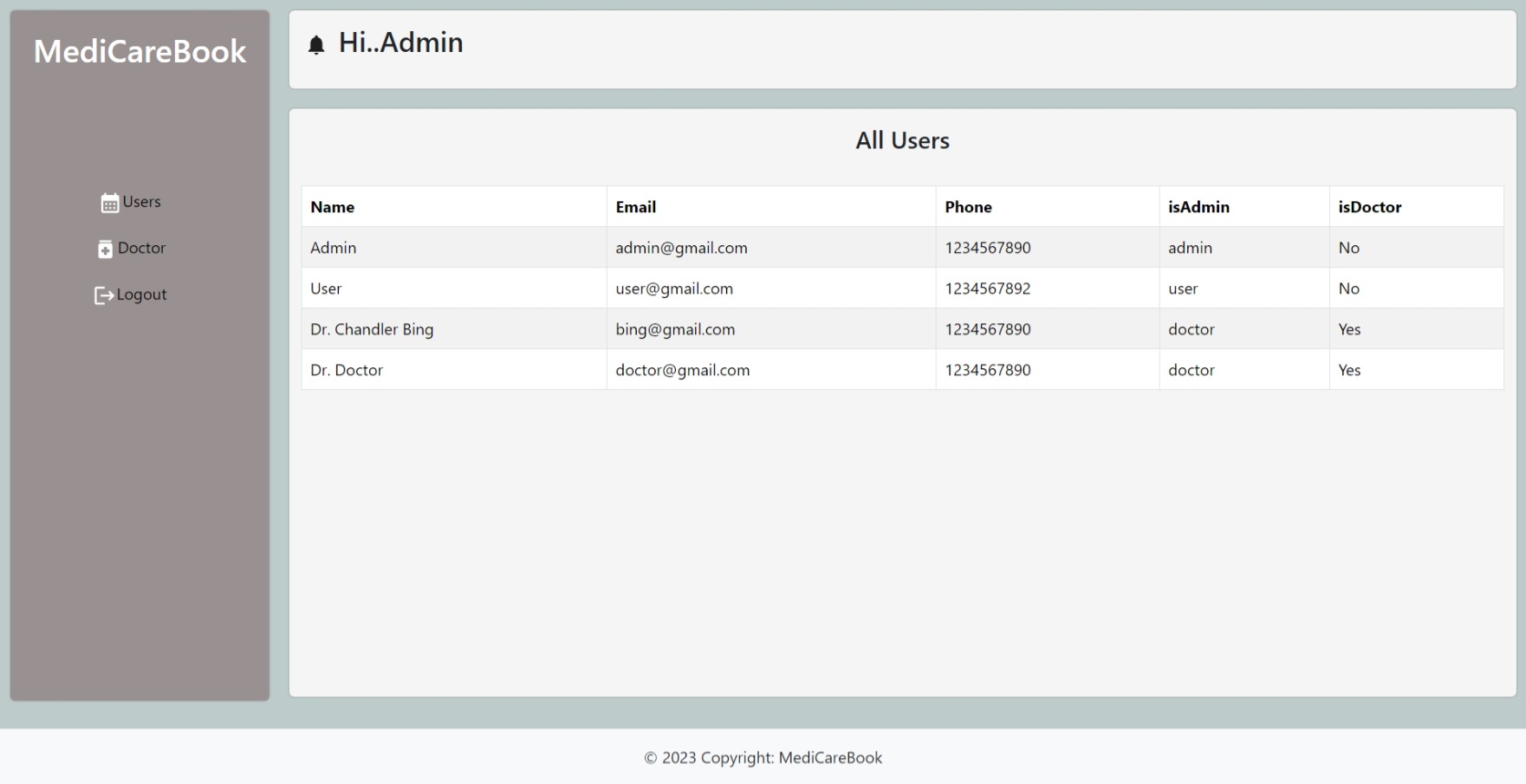
* Admin Home



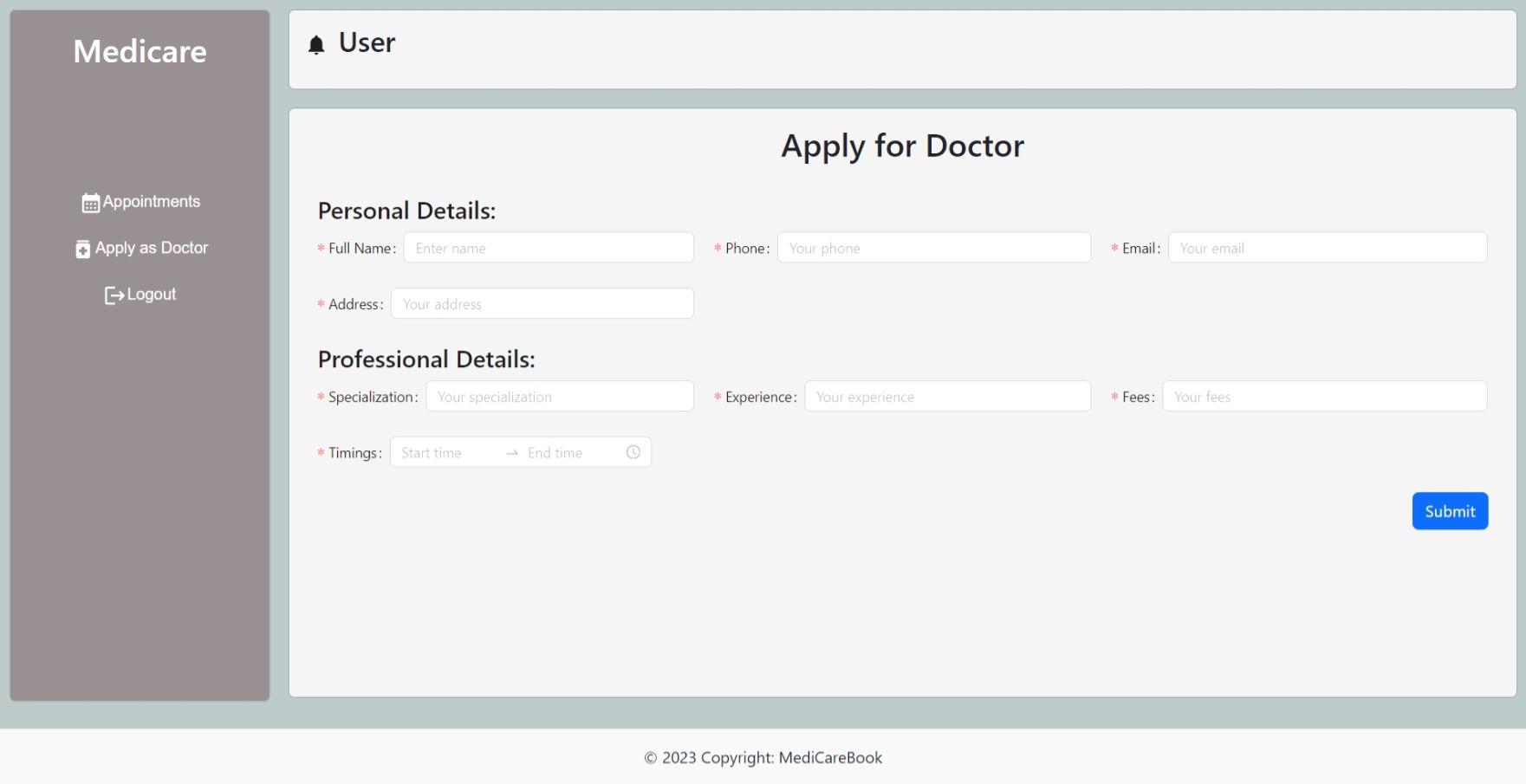
* Doctors List:



* Users list:



* Apply for doctor:



* Demo Link: [Provide a demo or video walkthrough link].

5. **Technology Stack**

* **Frontend:** HTML, CSS, JavaScript, React.js
* **Backend:** Node.js, Express.js
* **Database:** MongoDB Atlas
* **Tools:** Visual Studio Code, GitHub, Postman

6. **Project Workflow**

**Backend Setup**

MongoDB Models:

* **User Schema:** Role-based attributes (user, doctor, admin).
* **Appointment Schema:** Includes doctor, user, and schedule details.
* **Notification Schema:** Tracks notifications for users and doctors.

REST API Routes:

* **Auth Routes:** Register and login functionality.
* **Appointment Routes:** Create, fetch, and update bookings.
* **Notification Routes:** Push and retrieve notifications.

Frontend Setup

* **Landing Page:** Responsive homepage with navigation to login/register.
* **Login/Register Pages:** Authentication forms connected to the backend.
* **User Dashboard:** Displays available doctors and allows booking.
* **Doctor Dashboard:** Lists appointments and notifications.

Database Design

* **User Collection:** Name, email, password, role, etc.
* **Appointment Collection:** User ID, Doctor ID, date, time, etc.
* **Notification Collection:** Message, type, status, etc.

7. **Role-based Functionality**

* **User:** Browse doctors, book appointments, view notifications.
* **Doctor:** Accept/reject appointments, manage availability, receive notifications.
* **Admin:** Monitor platform activity, manage users and doctors.

12. **Challenges Faced**

* **Frontend Issues:** Resolving React component routing errors.
* **Backend Challenges:** Debugging MongoDB connection and schema setup.
* **Integration:** Managing asynchronous API requests effectively.

**13**. **Conclusion**

The Book a Doctor project successfully demonstrates the integration of a full-stack development approach. It showcases role-based user functionality, database management, and seamless frontend-backend communication.

**15**. **Future Enhancements**

* Add video consultation features.
* Implement payment gateway integration for online payments.
* Build a mobile app version for broader accessibility.

**-Doctor Booking System | Final Year Project 2024-**