Full Stack Development with MERN

1.Introduction

Project Title: Book a Doctor using MERN

Team Members

- Abikumar- Technical Lead
- Adhithya S - Back-end Developer
- Parameshwaran R- Front-end Developer
- PARTHIBAN G- Full-stack Developer

2. Project Overview

Purpose:

This project aims to create a web application that enables users to book appointments with doctors online. The platform allows users to view available doctors based on specialization, book appointments, and communicate with doctors through chat. Admins can manage doctors' availability and oversee user interactions.

Features:

- User registration and login with JWT authentication.
- Doctor profile management (create, update, delete).
- Appointment booking and tracking.
- Real-time messaging between users and doctors.
- Admin dashboard for managing doctors, appointments, and users.
- Payment integration for paid consultations.

3. Architecture

Frontend: Built using React.js, the frontend provides a dynamic user interface. It includes components for authentication, doctor browsing, appointment booking, and live chat, leveraging React Router for seamless navigation.

Backend: The backend is developed with Node.js and Express.js. It handles requests, user authentication, doctor profile management, appointment scheduling, and integrates with MongoDB for data storage.

Database: MongoDB stores information about users, doctors, appointments, and chats. Collections are structured to allow fast access and updates, e.g., users, doctors, appointments, messages.

4. Setup Instructions

Prerequisites:

Node.js v14+

MongoDB v4+

(Optional) npm or yarn for package management

Installation:

1. Clone the repository:

```
git clone <repository-url>
```

2. Navigate to both the backend and frontend directories and install dependencies:

For Backend:

```
cd backend

npm install
```

For Frontend:

```
cd ../frontend

npm install
```

5. Folder Structure

Client (Frontend):

- src/components: Reusable React components (e.g., LoginForm, DoctorCard).
- src/pages: Pages for different routes (e.g., Home, DoctorList, Appointment).
- src/utils: Utility functions, API calls, and helper methods.
- src/styles: CSS files for styling.

Server (Backend):

- config: Contains the database connection configuration.
- controllers: Business logic handling (e.g., handling appointments, user registration).
- routers: API route definitions.
- middlewares: For JWT authentication, error handling.
- schemas: MongoDB models for users, doctors, appointments.

6. Running the Application

Frontend:

Start the frontend server:

```
npm start
```

Backend:

Start the backend server:

```
npm start
```

7. API Documentation

- POST /api/auth/login – User login with JWT authentication.

- POST/api/auth/register – User registration.
- GET/api/doctors – Fetch available doctors based on specialization.
- POST/api/appointments – Book an appointment with a selected doctor.

Example Response (Fetch Doctors):

json

```
{
 "status": "success",
 "data": {
  "doctors": [
   {
    "id": "doctor-id",
    "name": "Dr. John Doe",
    "specialization": "Cardiology",
    "rating": 4.5,
    "availableTimes": ["10:00 AM", "02:00 PM"]
   }
  ]
 }
}
```

8. Authentication

Authentication is managed using JWT tokens. After successful login, users receive a token that is stored in local storage. Protected routes (e.g., appointment booking) require a valid JWT token for access, verified using middleware.

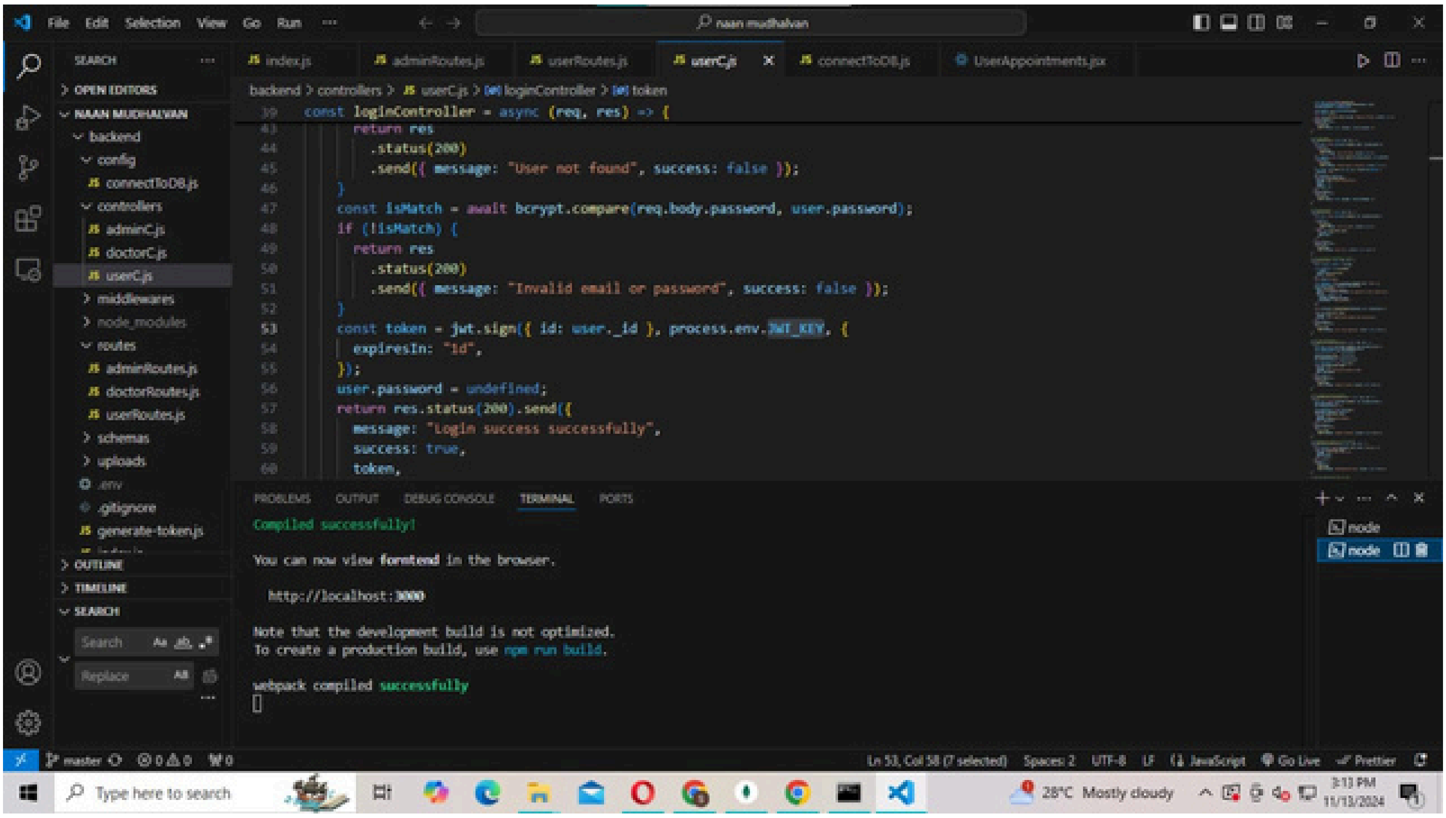
9. User Interface

- Login and Registration Form: For new users to sign up or log in.
- Doctor Catalog: A page where users can browse available doctors by specialization and availability.
- Appointment Booking: A page where users can schedule appointments with doctors.
- Dashboard: Displays user's upcoming appointments and past visit history.
- Real-time Chat: A messaging feature for users to communicate with their doctor before or after the consultation.

10. Testing

Testing is conducted using tools like Jest for unit tests (for React components) and Postman for API testing (validating API responses and functionality)

11.Screenshots



```
const loginController = async (req, res) => {
      return res
        .status(200)
        .send({ message: "User not found", success: false });
    }
    const isMatch = await bcrypt.compare(req.body.password, user.password);
    if (!isMatch) {
      return res
        .status(200)
        .send({ message: "Invalid email or password", success: false });
    }
    const token = jwt.sign({ id: user._id }, process.env.JWT_KEY, {
      expiresIn: "1d",
    });
    user.password = undefined;
    return res.status(200).send({
      message: "Login success successfully",
      success: true,
      token,
```

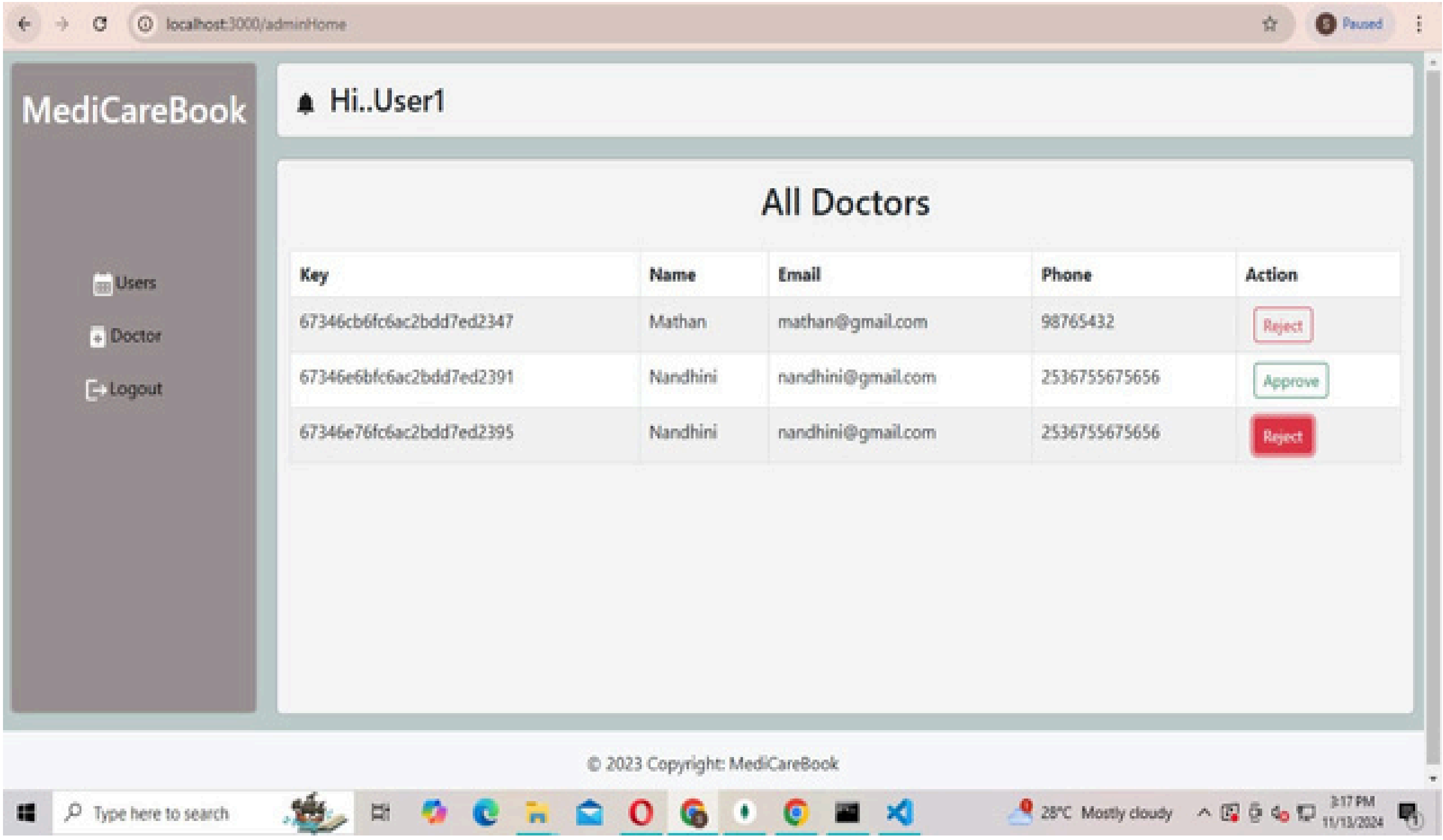
```
Compiled successfully!

You can now view frontend in the browser.

  http://localhost:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

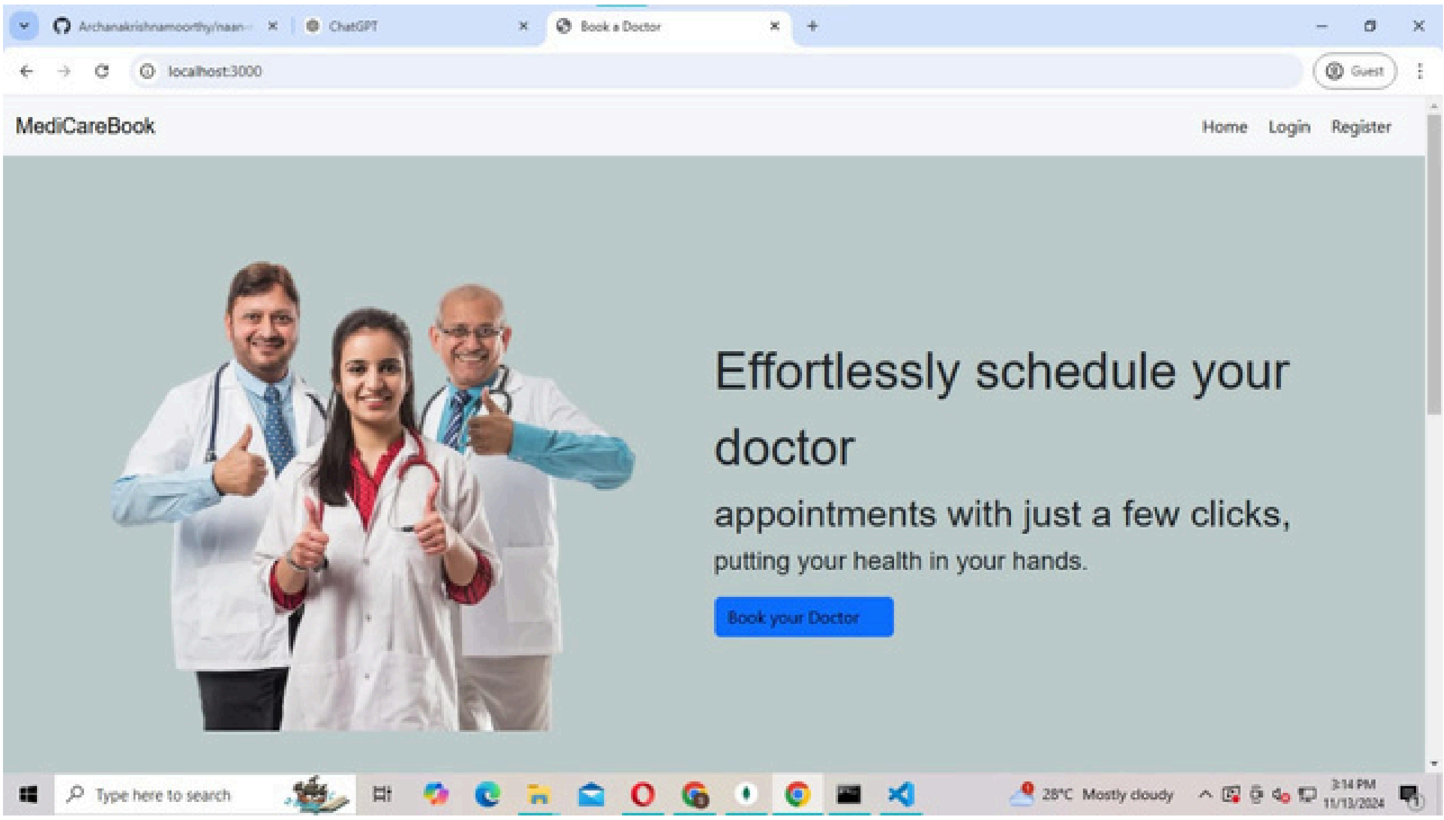


Hi..User1

All Doctors

Key	Name	Email	Phone	Action
67346cb6fc6ac2bdd7ed2347	Mathan	mathan@gmail.com	98765432	Reject
67346e6bfc6ac2bdd7ed2391	Nandhini	nandhini@gmail.com	2536755675656	Approve
67346e76fc6ac2bdd7ed2395	Nandhini	nandhini@gmail.com	2536755675656	Reject

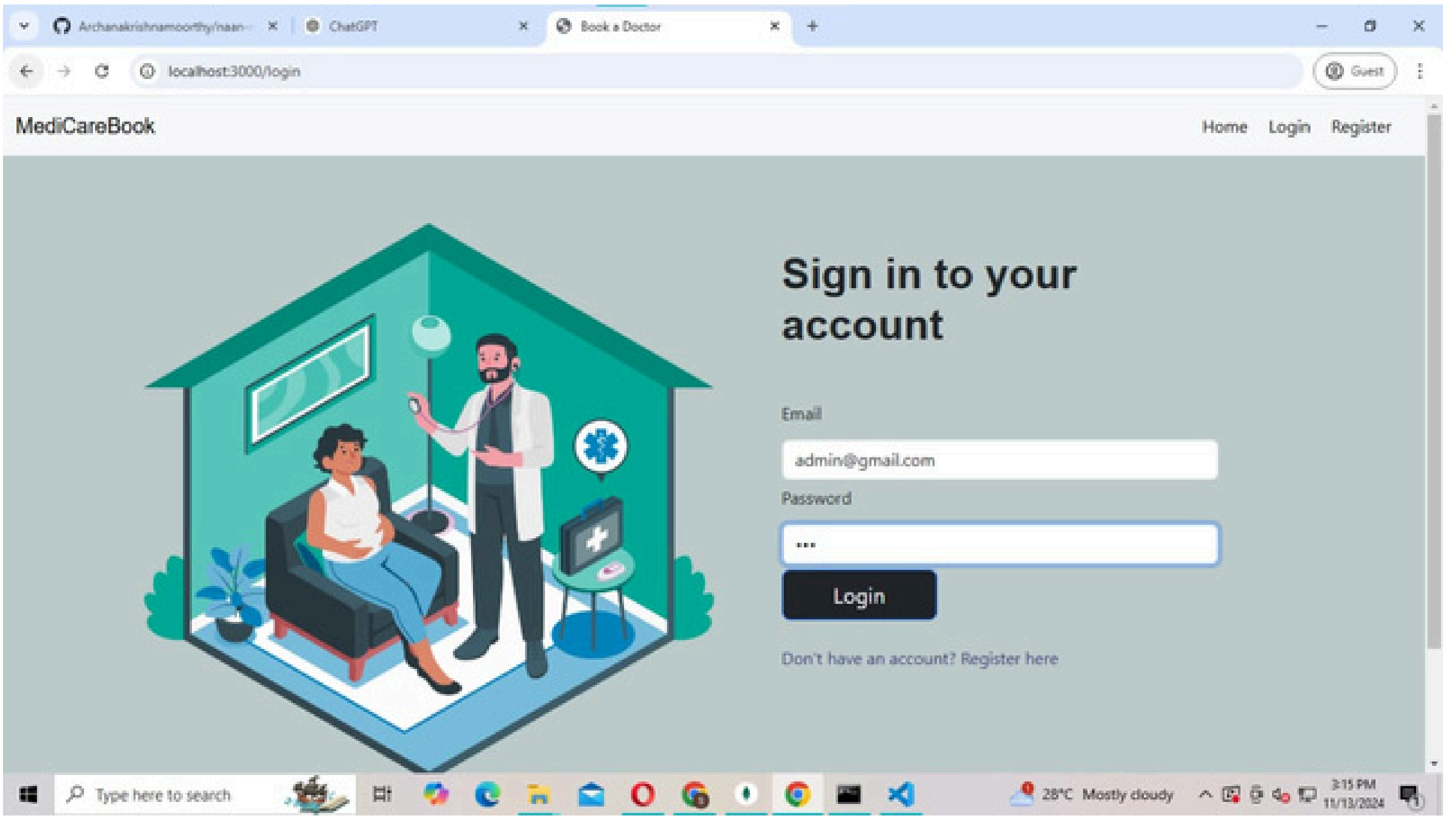
© 2023 Copyright: MediCareBook



Effortlessly schedule your doctor

appointments with just a few clicks,

putting your health in your hands.



MediCareBook

Home Login Register

Sign in to your account

Email

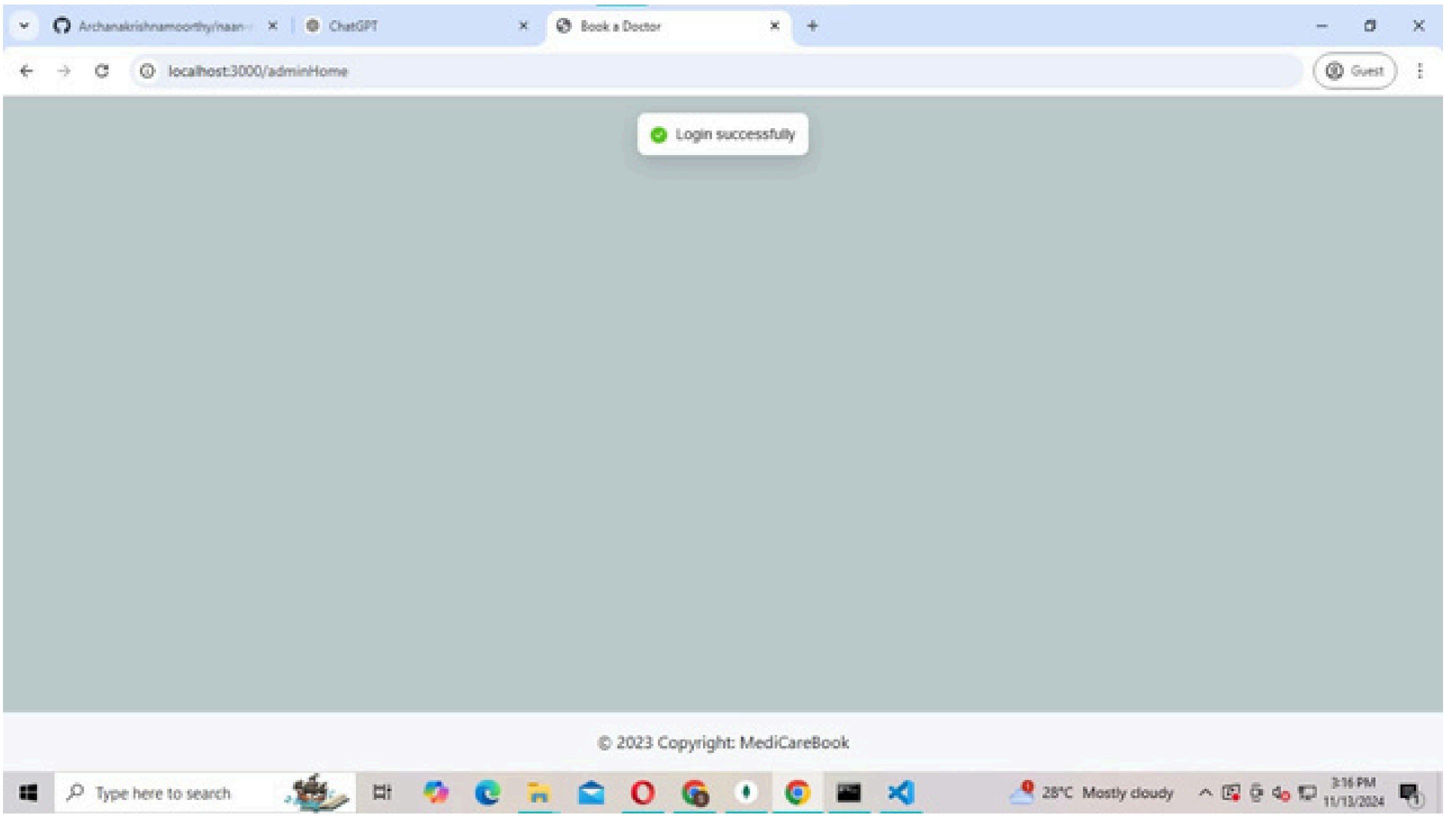
admin@gmail.com

Password

•••

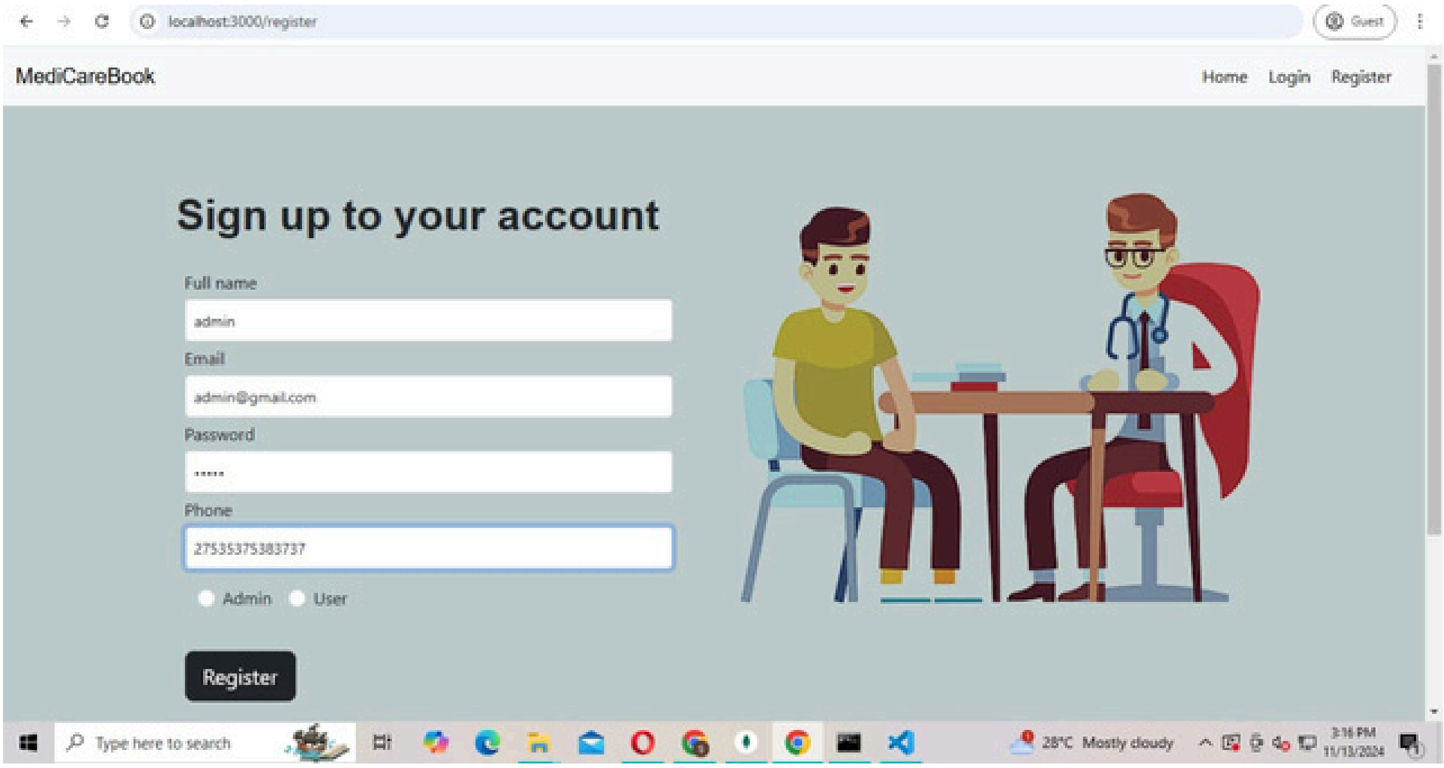
Login

Don't have an account? Register here



Login successfully

© 2023 Copyright: MediCareBook



Sign up to your account

Full name: admin

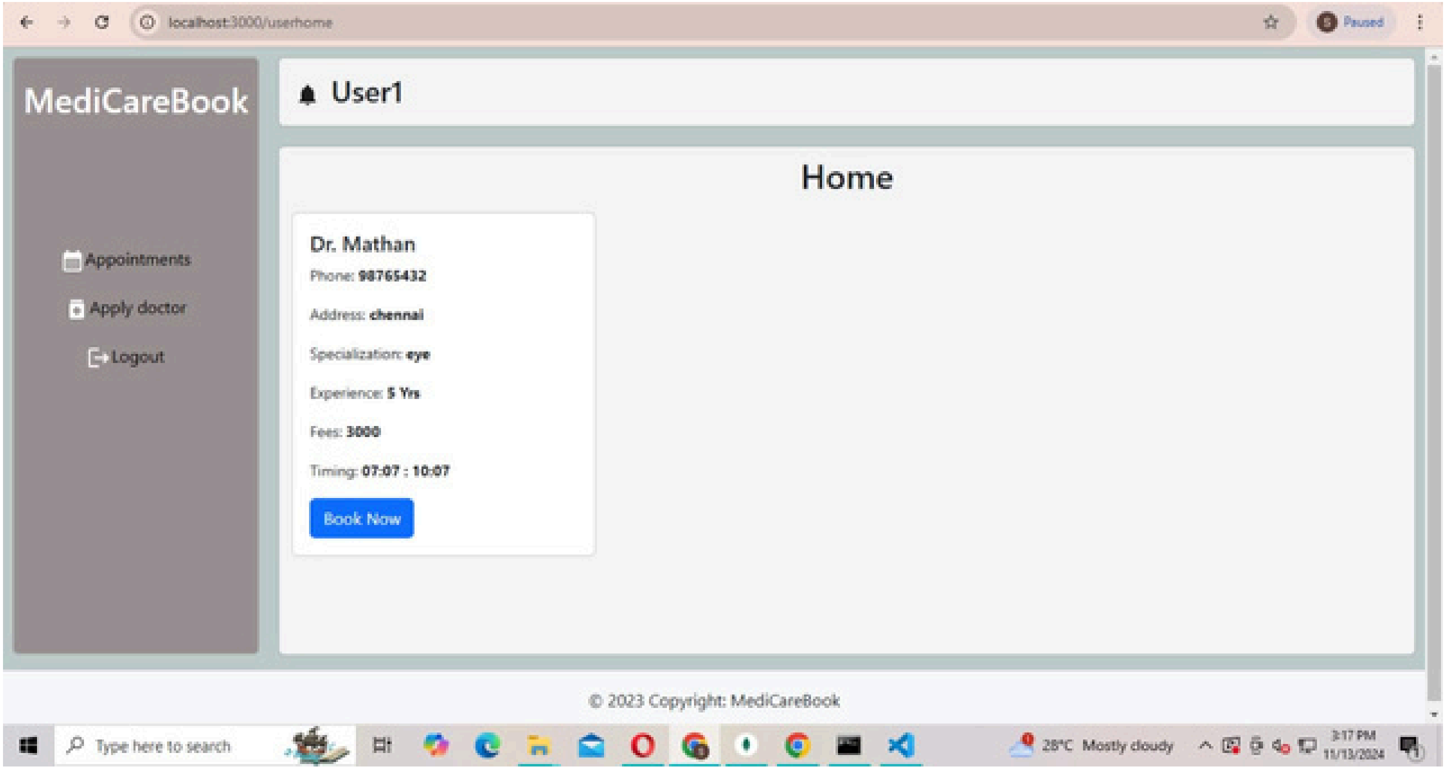
Email: admin@gmail.com

Password: •••••

Phone: 27535375383737

○ Admin ○ User

Register



User1

Home

Dr. Mathan

Phone: **98765432**

Address: **chennai**

Specialization: **eye**

Experience: **5 Yrs**

Fees: **3000**

Timing: **07:07 : 10:07**

Book Now

© 2023 Copyright: MediCareBook

12. Known Issues

- Token expiration may log users out unexpectedly.
- Delays may occur in real-time chat due to network instability.
- Appointment times may not be updated correctly in rare cases.

13. Future Enhancements

- Payment Gateway Integration: Add support for payments for paid consultations.
- Doctor Ratings and Reviews: Allow users to rate doctors after consultations.
- Video Consultation Integration: Enhance real-time chat with video calling features.
- Admin Dashboard Enhancements: Provide more detailed analytics and reporting for appointments and user engagement.