FULL STACK WEB DEVELOPMENT INTERNSHIP REPORT

Submitted by

S. BHARATH

312621243007

In partial fulfillment for the award of

the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

THANGAVELU ENGINEERING COLLEGE

THORIPAKKAM CHENNAI 600097

ANNA UNIVERSITY CHENNAI 600025

NOVEMBER 2024

THANGAVELU ENGINEERING COLLEGE

APPROVED BY AICTE &PROGRAMMES ACCREDITED BY NBA, NEWDELHI, AND AFFILIATED TO ANNA UNIVERSITY, CHENNAI

RAJIV GANDHI SALAI, OMR THORAIPAKKAM, CHENNAI - 600 097.

ANNA UNIVERSITY CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this internship " Full Stack Web Development " by " SmartInternz " is the bonafide work of S.BHARATH who carried out the internship under my supervision.

SIGNATURE	SIGNATURE
MRS. D. BEULAH PRETTY M.E..	MS. C. KANAGALAKSHMI. M.E
HEAD OF THE DEPARTMENT	SUPERVISOR ASSISTANT PROFESSOR
Department of Artificial Intelligence	Department of Artificial Intelligence
And Data Science Thangavelu Engineering College, Thoraipakkam, Chennai-600097	And Data Science Thangavelu Engineering College, Thoraipakkam, Chennai-600097

ACKNOWLEDGEMENT

The success and final outcome of learning the MERN stack required immense guidance and assistance from many people. I feel extremely privileged to have received such support throughout the completion of my course and various projects. All that I have accomplished is due to the valuable supervision and assistance provided to me, and I would like to express my heartfelt gratitude to everyone involved.

I am deeply thankful and fortunate to have received constant encouragement, support, and guidance from all the teaching staff. Their invaluable help has been instrumental in successfully completing my internship and project work.

_____

(Signature of Student)

Name: Bharath.S

Reg no:312621243007

Date: _____

Book a Doctor Application - Project Report

Project Details

- Project Title: Book a Doctor Application
- Organization: Smart Internz
- Batch:5
- TEAM ID: NM2024TMID16427

Team Members and Roles

1. **Project Manager**:
   - Name: BHARATH S
   - Nm ID: BFB2C2A1892FDA48AD6426FA5168575B
   - Role: Oversees project planning, task distribution, and progress tracking.
2. **Frontend Developer**:
   - Name: BUDUGU NANDHINI
   - Nm ID: BC252FD0E0AAE6991854E0F0107AFAF1
   - Role: Develops the user interface using React.js, ensuring responsiveness and user-friendliness.
3. **Backend Developer**:
   - Name**:** DEEPAK S
   - Nm ID: 52C313A87BEB501BA314D12970A95694
   - Role: Manages server-side logic, implements APIs, and ensures seamless data handling.
4. **Database Administrator**:
   - Name: DIKSHWIN ROBINSON R
   - Nm ID: 83DFB864959779D876C4C68B2D58D80F
   - Role: Designs and maintains the MongoDB database, optimizing data storage and retrieval processes.

Table of Contents

Book a Doctor Application - Project Report 3

Team Members and Roles 4

1. Introduction 6

2. Project Overview 7

3. Architecture 8

4. Setup Instructions 9

5. Folder Structure 10

6. Running the Application 11

7. API Documentation 11

8. User Interface 12

9.Future Enhancements 17

1. Introduction

Project Title: Book a Doctor Application
Team Members:

- **Project Manager**: Oversees project planning, task distribution, and progress tracking.
- **Frontend Developer**: Develops the user interface, ensuring responsiveness and user-friendliness.
- **Backend Developer**: Manages server-side logic, implementing APIs for data handling.
- **Database Administrator**: Designs the MongoDB database, optimizing data storage and retrieval processes.

Project Overview:
The "Book a Doctor" application is a web-based platform designed to streamline the healthcare appointment process. By allowing patients to book appointments with doctors based on specialization and availability, it provides a seamless experience for managing healthcare needs. The application also enables doctors to manage their schedules and appointments, and includes an admin interface for platform management and registration approvals.

2. Project Overview

Purpose:
The " Book a Doctor " project aims to simplify the healthcare appointment process by providing a centralized platform where patients can connect with doctors efficiently. By allowing easy access to doctor schedules, this platform improves accessibility and reduces wait times for consultations.

Features:

Patients: Patients can:

- Search for doctors based on specialty and location.
- View doctor availability and book appointments.
- Receive notifications and reminders for upcoming appointments.

Doctors: Doctors can:

- Manage their schedules, adjusting availability as needed.
- Approve or decline patient appointment requests.
- Easily update their profiles and availability in real-time.

Admins: The admin interface enables administrators to:

- Monitor doctor registrations, verify credentials, and approve new profiles.
- Manage user activity to ensure platform reliability.
- Generate detailed reports on platform usage to support continuous improvement.

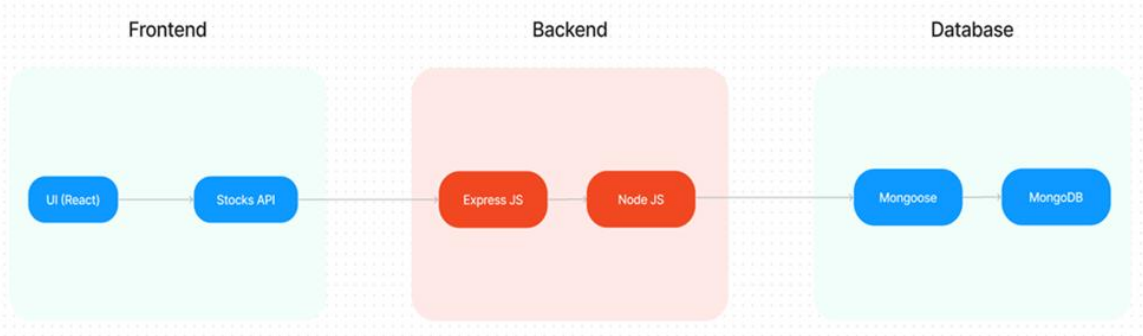
3. Architecture

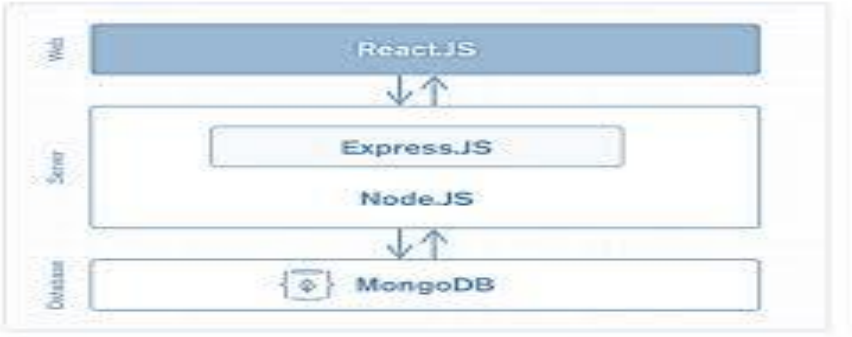
Frontend:
The user interface (UI) is developed with **React.js** and styled with **Material UI** to ensure a responsive and dynamic user experience. Each component is modular and reusable, which supports ease of maintenance and scalability as the application grows.

Backend:
The backend server, developed with **Node.js** and **Express.js**, manages API requests, implements business logic, and processes data efficiently. This server structure supports reliable performance, even under heavy user loads, by handling routing and user interactions smoothly.

Database:
MongoDB serves as the primary database for securely storing user information, appointment records, and system logs. The non-relational structure of MongoDB suits the flexible data requirements of the application, allowing for rapid data retrieval and updates.

Authentication:
JSON Web Tokens (JWT) are used for secure authentication and role-based authorization. This ensures that only verified users can access protected routes, and each role—patient, doctor, or admin—has access to specific features, enhancing security across the platform.





4. Setup Instructions

Prerequisites:

- Install Node.js and ensure it is available in your system's PATH.

- MongoDB should be installed locally or a cloud instance (e.g., MongoDB Atlas) should be configured.

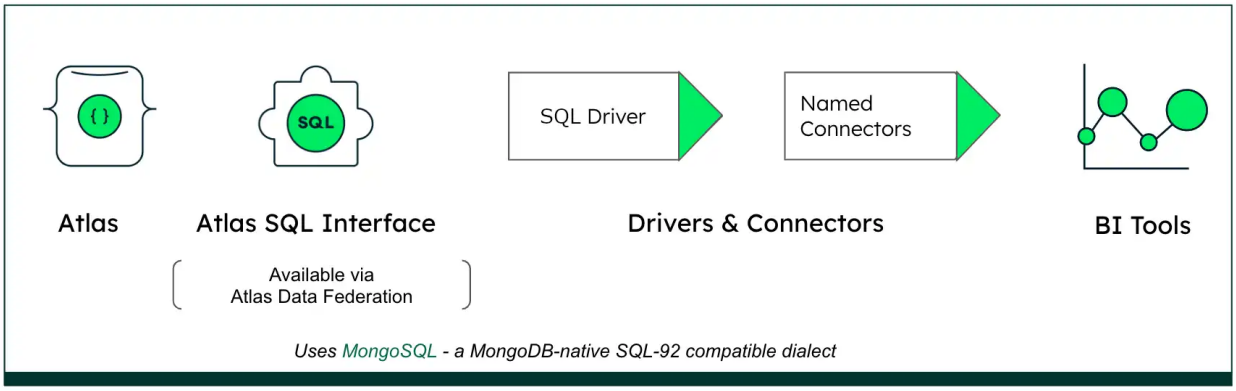
Installation:

1. Clone the repository using Git.

2. Navigate to the project directory and install dependencies using `npm install`.

3. Create a `.env` file based on the provided `.env.example` and set the required environment variables such as database URI.

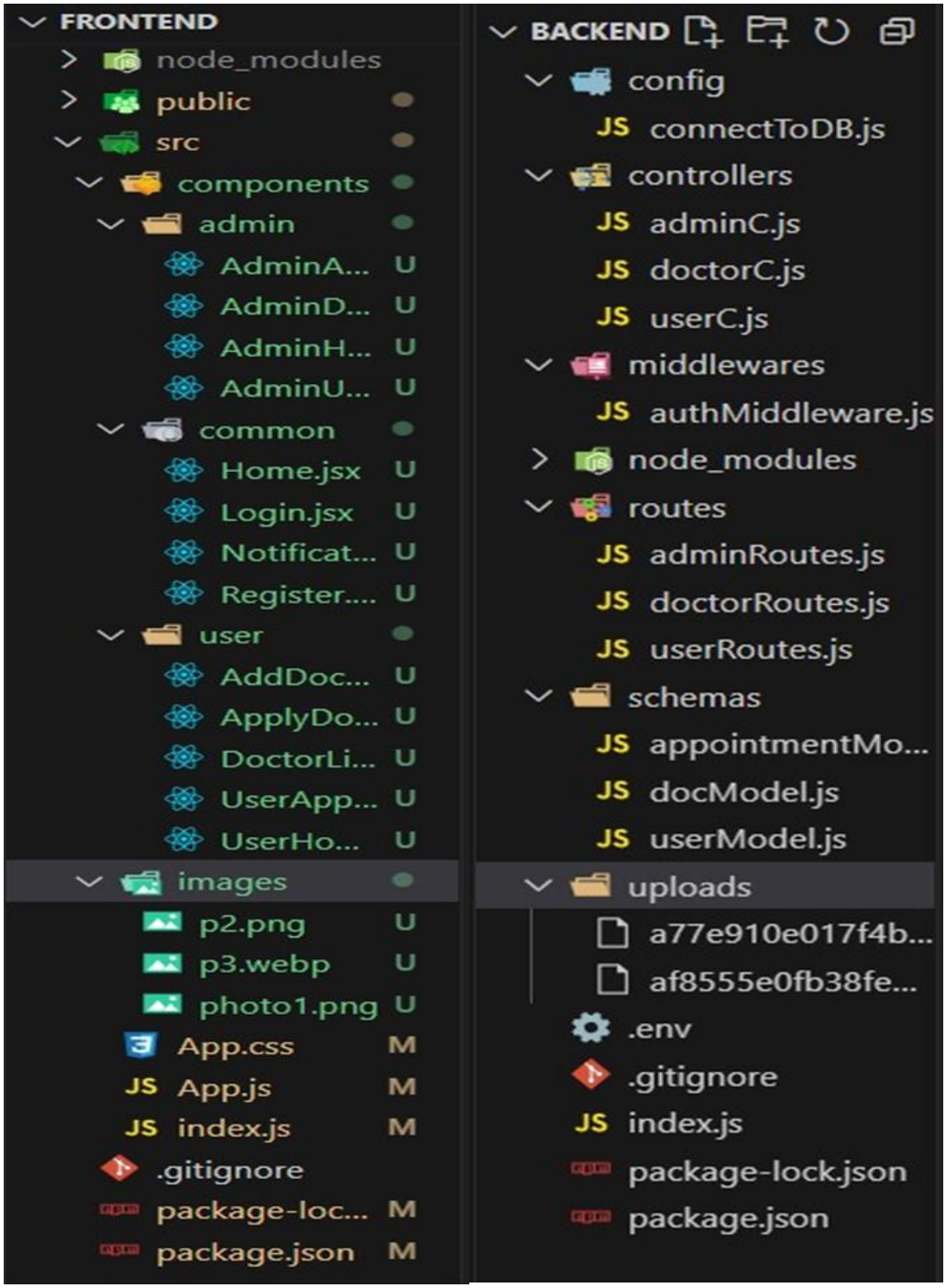
4. Start MongoDB service if running locally, or verify cloud connectivity.



5. Folder Structure

- **Client Folder:** Contains the React frontend structured into components (UI elements), pages (route-specific features), and services (API calls and utilities).

- **Server Folder:** Houses the Node.js backend structured into models (data schemas), routes (API endpoints), controllers (business logic), and middleware (authentication, error handling).



6. Running the Application

Frontend: To start the frontend, navigate to the client folder and execute the command:

`npm start`

The application will open in your default browser on localhost:3000.

Backend: To start the backend server, navigate to the server folder and execute the command:

`npm start`

The API will be available at localhost:8001.

7. API Documentation

The backend exposes several RESTful API endpoints for interaction with the database. Below are key endpoints:

- **POST /auth/login:** Authenticates user credentials and returns a JWT token.

- **GET /appointments:** Fetches appointments for a logged-in user.

- **POST /appointments/book:** Books an appointment with the provided doctor ID and time slot.

- **GET /doctors:** Retrieves a list of available doctors based on filters such as specialization and location.

Authentication

The project implements JWT for secure authentication and role-based access. Upon login, users receive a token that must be included in the Authorization header for protected API routes. Token expiration and refresh mechanisms ensure security and usability.

8. User Interface

The application's user interface consists of:

Home Page: Displays an overview of the platform and allows users to navigate to key features. –

Registration Page: Enables new users to sign up with appropriate role selection (patient, doctor, or admin).

Admin Panel: Offers features to monitor platform activity and approve registrations

- **Doctor Registration Approval**: Admins can review and approve doctor applications to maintain quality and trustworthiness on the platform.
- **User Activity Monitoring**: Tracks platform usage, appointment trends, and user activity logs.
- **Report Generation**: Provides insights into system performance and generates reports on user engagement.
- **Management Tools**: Admins can manage user accounts, resolve disputes, and oversee platform health.

Doctor Search Page: Allows patients to filter and select doctors based on various criteria.

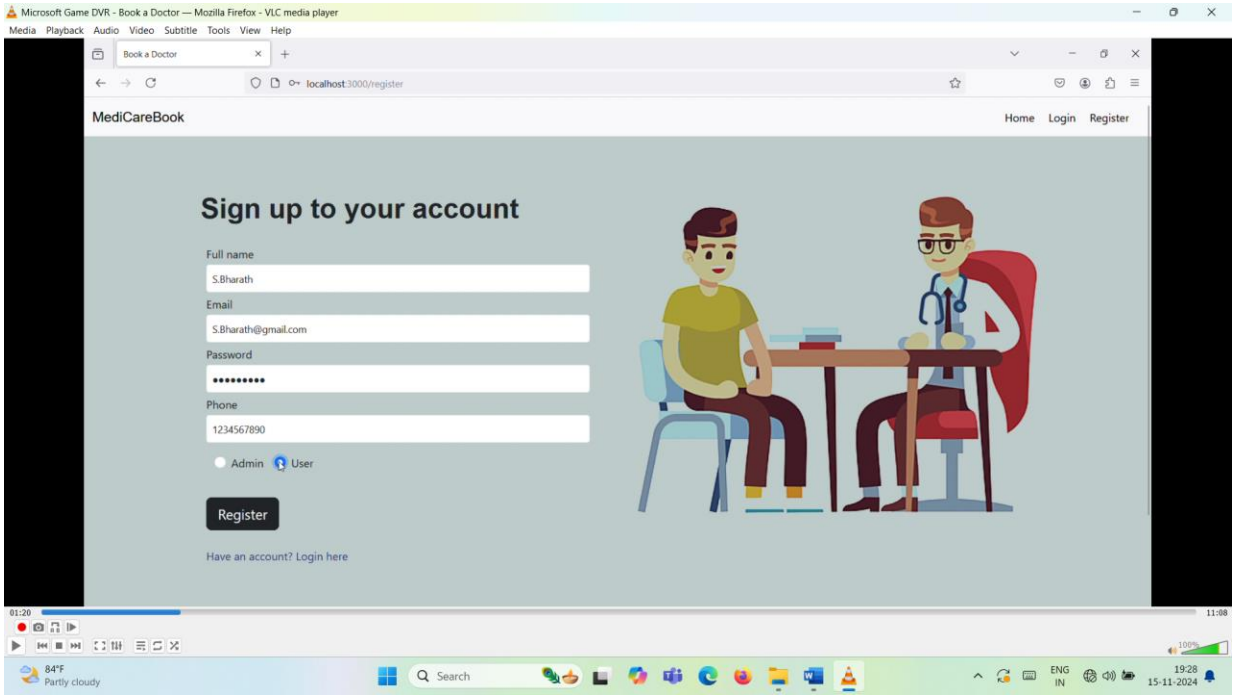
- **Real-Time Availability**: Shows up-to-date availability for booking appointments.
- **Quick Booking**: Allows patients to book appointments directly from the search results page.

Login page Page:

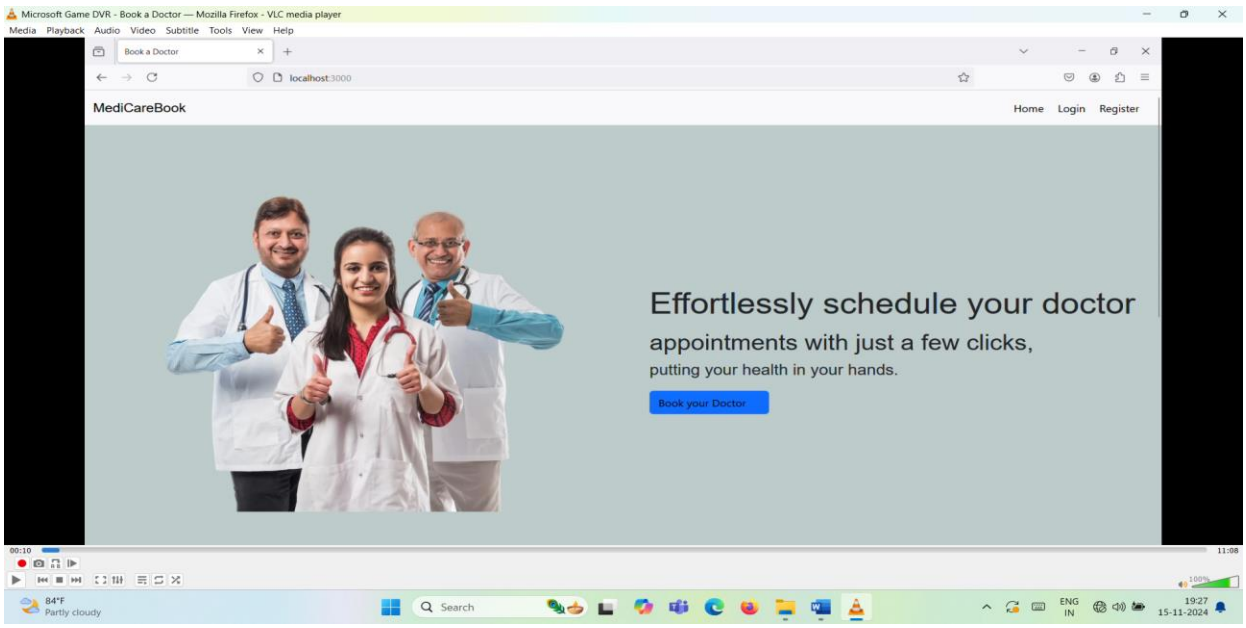
- **Secure Login:** Users enter their credentials (email/username and password) to access the system. Authentication is handled using JWT (JSON Web Tokens) for enhanced security.
- **Role Selection:** Determines whether the user is a patient, doctor, or admin, granting access to the respective dashboard.
- **Error Handling:** Provides user-friendly error messages for invalid credentials or failed login attempts.

Doctor Application Page:

- **Profile Management:** Doctors can update personal details, specialties, and contact information.
- **Schedule Management:** Provides tools for setting availability and managing appointment requests in real-time.
- **Appointment Dashboard**: Displays pending, approved, and completed appointments, with options to accept or decline patient requests.

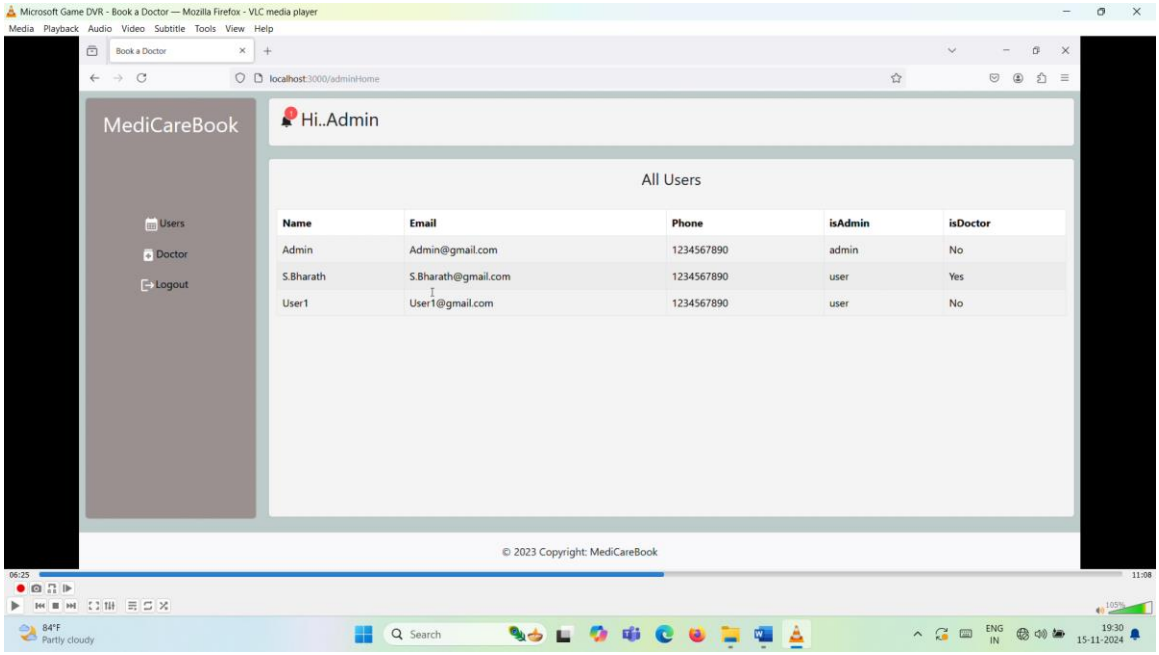


Register Page

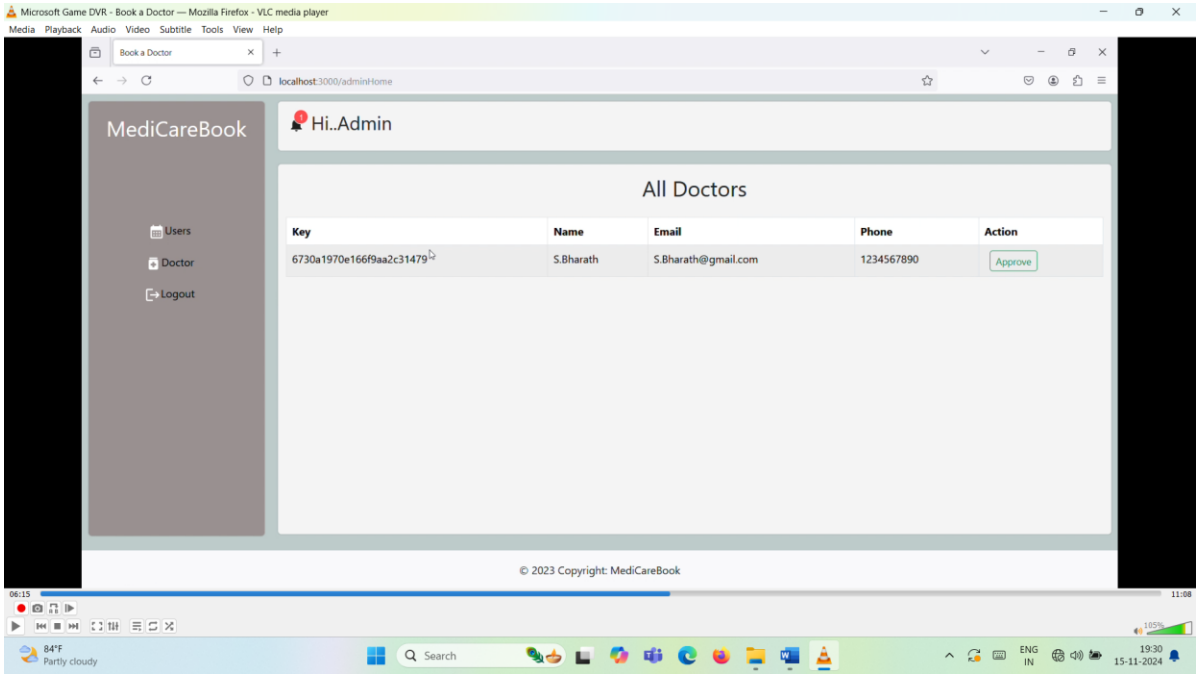


Home Page

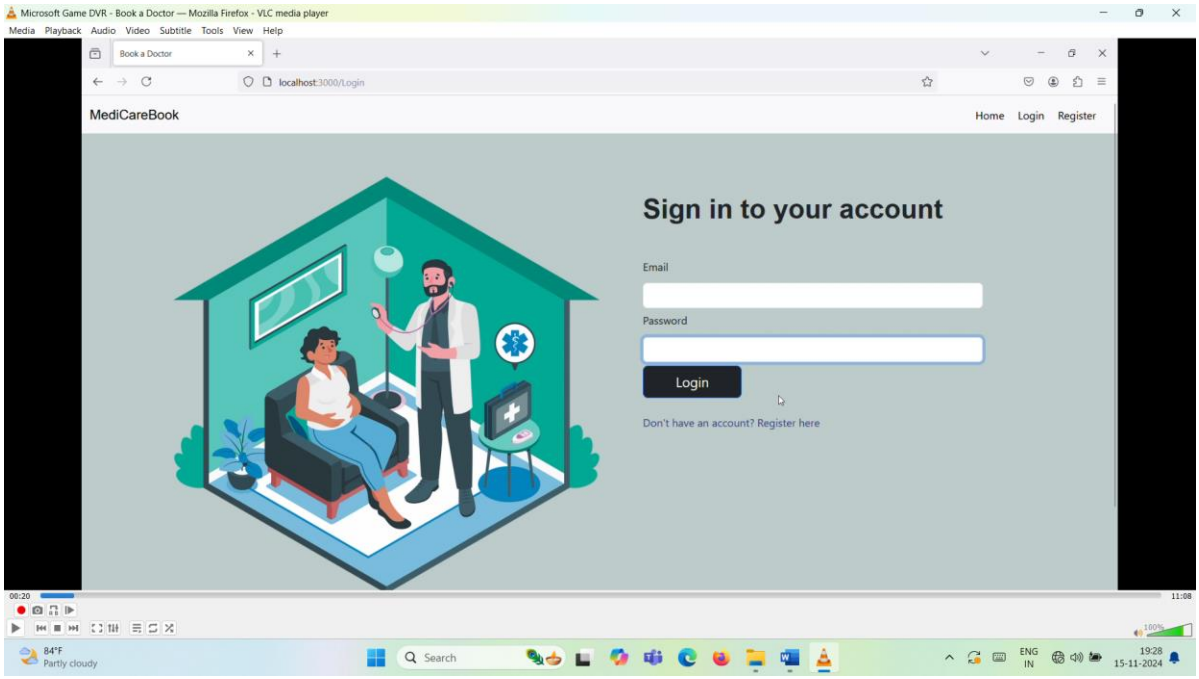
.



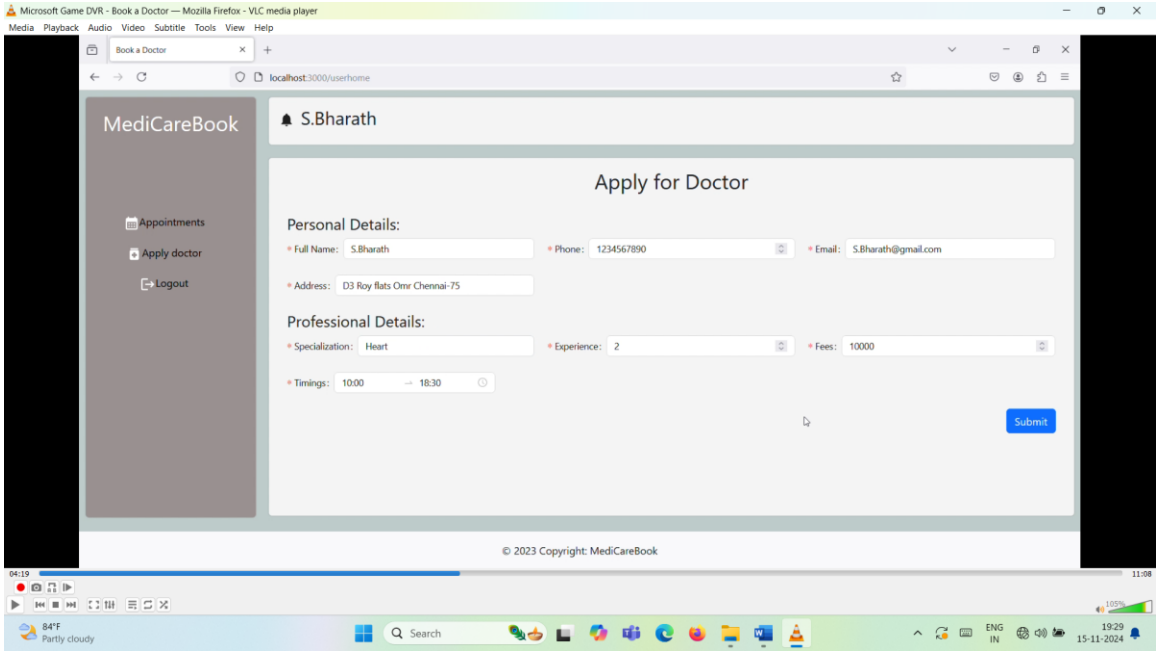
Admin Panel



Doctor Search Page



Login Page



Doctor Application

Testing Types:

1. **Unit Testing:**
    - Focuses on testing individual React components and backend functions.
    - Tools Used: Jest for frontend testing and Mocha for backend validation.
    - Objective: Ensure each module performs as expected in isolation.
2. **Integration Testing:**
    - Simulates real-world user workflows to test interactions between the frontend, backend, and database.
    - Objective: Verify the system's ability to handle complete processes efficiently.

Key Test Scenarios:

- Appointment Scheduling:
  Validates the system's ability to schedule and manage appointments accurately without conflicts.
- Patient-Doctor Communication:
  Ensures that the platform facilitates effective communication, such as messaging or notifications, between patients and doctors.
- Real-Time Availability:
  Confirms that patients can view up-to-date appointment slots and book instantly without delays.
- Additional Features:
  Tests functionalities like video consultations, text-based chats, and prescription access for reliability and ease of use.

Identified Issues:

1. **Real-Time Updates:**
    - Notifications and status updates for appointments are partially implemented and may not reflect changes immediately.
    - Impact: Slight delays in displaying updated appointment statuses**.**
2. **Time Zone Management:**

- Challenges in handling different time zones for global appointments may result in scheduling conflicts.
- Impact: Users across regions might encounter inconsistencies in displayed times.

3. **Security Enhancements:**
   - While data transmission is encrypted, certain areas such as password recovery and API rate limiting could benefit from additional security measures.
   - Impact: Potential vulnerabilities to brute force attacks**.**

9.Future Enhancements

- **Improve patient-doctor communication**

Allow patients to message their healthcare providers securely, which can help build better relationships and make follow-up care more efficient.

- **Reduce missed appointments**

Use automated appointment reminders and confirmations to reduce the risk of patients missing their appointments.

- **Improve record keeping**

Maintain a digital record of patient information and appointment histories to improve the accuracy of record-keeping.

- **Prioritize user-friendly design**

Make it easy for patients to find doctors, schedule appointments, and manage their medical needs.

- **Use cloud-based solutions**

Integrate existing management systems and doctor/patient databases into a single platform for real-time data exchange.

- **Add a blog and health tips**

Use a web platform to share health tips with patients and create a section to cover accepted insurance plans.