



GOJAN

SCHOOL OF BUSINESS AND TECHNOLOGY

Approved by A.I.C.T.E., New Delhi & Affiliated to Anna University, Chennai | NAAC Accredited Institution
An ISO 9001:2015 Certified Institution | Recognized by UGC u/s 2(f) & 12(B) of the UGC Act



BOOK A DOCTOR USING MERN STACK

*In partial fulfilment for the award of the
degree Of*

BACHELOR OF ENGINEERING

**DEPARTEMENT OF COMPUTER
SCIENCE**

Submitted by

NAME : VIGNESH M

DEPARTEMENT : CSE

REGISTER NUMBER: 110521104053

YEAR : IV

SEMESTER : VII



GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY

Approved by A.I.C.T.E., New Delhi & Affiliated to Anna University, Chennai | NAAC Accredited Institution
An ISO 9001:2015 Certified Institution | Recognized by UGC u/s 2(f) & 12(B) of the UGC Act



BONAFIDE CERTIFICATE

Certified that this Project on “ **BOOK A DOCTOR USING MERN STACK**”, is a Bonafide record of work done by,

NAME : **VIGNESH M**

DEPARTEMENT : CSE

REGISTER NUMBER : 110521104053

In the Department of Computer Science and Engineering , Anna University, Chennai.

Internal Guide

Head of the Department

Internal Examiner

External Examiner

TABLE OF CONTENTS

S.NO	TITLE	PG.NO
1	ABSTRACT	4
2	INTRODUCTION	5
3	KEY FEATURES	7
4	ER DIAGRAM	8
5	WORKFLOW	9
6	TECHNICAL STACK	10
7	SETUP INSTRUCTIONS	11
8	FOLDER STRUCTURE	12
9	SAMPLE OUTPUT	19
10	CONCLUSION	24

ABSTRACT

- 1.** A Doctor Booking System built using the MERN (MongoDB, Express.js, React, Node.js) stack is an efficient, scalable, and responsive platform that simplifies the process of scheduling medical appointments. The healthcare industry increasingly relies on digital solutions to enhance patient care and streamline administrative processes.
- 2.** This platform represents a modern solution to the challenges faced in healthcare appointment scheduling, leveraging the MERN stack to provide a responsive, scalable, and maintainable solution suitable for both small practices and large medical institutions.
- 3.** The application aims to reduce the complexity of manual booking processes, improve accessibility to healthcare professionals, and enhance user satisfaction by providing a seamless digital experience.
- 4.** The Doctor Booking System supports features like user authentication, appointment management, search filters, and real-time updates. It also integrates calendar management, allowing both patients and doctors to view and update their schedules in real time.
- 5.** This system enables patients to easily search for doctors based on specialty, location, and availability, while providing doctors with an intuitive interface to manage their appointments and availability.

INTRODUCTION

Booking a doctor's appointment has never been easier. With our convenient online platform, you can quickly and effortlessly schedule your appointments from the comfort of your own home. No more waiting on hold or playing phone tag with busy receptionists. Our user-friendly interface allows you to browse through a wide range of doctors and healthcare providers, making it simple to find the perfect match for your needs

User Registration:

John, who needs to see a doctor for a routine check-up, visits the Book a Doctor app and signs up as a Customer. He provides his email and creates a password.

Browsing Doctors:

Upon logging in, John is presented with a dashboard displaying a list of available doctors and healthcare providers

Booking an Appointment:

John finds a suitable doctor and clicks on "Book Now." A form appears where he selects the desired appointment date and uploads any necessary documents, such as medical records or insurance information.

Appointment Confirmation:

The doctor reviews John's appointment request and availability. Once confirmed, the appointment status changes to "scheduled."

Admin Approval (Background Process):

In the background, the admin reviews new doctor registrations and approves legitimate applicants.

Doctor's Appointment Management:

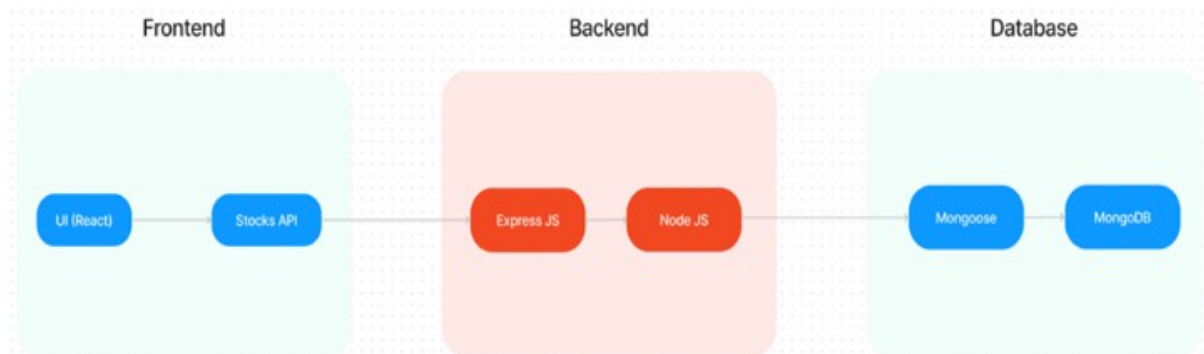
Dr. Smith, an approved doctor on the platform, logs into his account and manages his appointments.

Appointment Consultation:

*On the day of the appointment, John visits the doctor's office for his check-up.

*Dr. Smith provides medical care and advice during the consultation, fulfilling John's healthcare needs.

TECHNICAL ARCHITECTURE:



The technical architecture of our Book a Doctor app follows a client-server model, where the front end serves as the client and the back end acts as the server. The front end encompasses not only the user interface and presentation but also incorporates the Axios library to connect with the backend easily by using RESTful Apis.

Post-Appointment Follow-up:

After the appointment, Dr. Smith updates John's medical records and may prescribe medication or recommend further treatment if necessary.

The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user whether it is an admin, doctor, or ordinary user working on it.

On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.

HTML, CSS, and JavaScript:

Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity:

Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

KEY FEATURES

An **Doctor booking using mern Website** is designed to streamline grievance handling by offering intuitive, efficient, and transparent features. Here are the key functionalities:

1. User Registration and Login:

- Secure account creation using email, phone number, or social media.
- Role-based access for users, administrators, and support doctors.

2. Booking an Appointment:

- Simple form-based interface for filing a appointment.
- Options to select Appointment categories and attach supporting documents for doctor appointment booking.

3. Appointment Confirmation:

- Users can monitor the status of appointment with updates like “Pending,” “In Progress,” or “Resolved.”
- Notifications via email or SMS for status changes.

4. Admin Dashboard:

- Tools for managing appointment prioritizing cases, and assigning tasks to relevant departments.
- Analytics and reporting to identify trends and optimize resolutions.

5. Feedback Mechanism:

- Users can rate their satisfaction and provide suggestions post-resolution.

6. Data Security and Privacy:

- Encryption for user data and secure access protocols to ensure confidentiality.

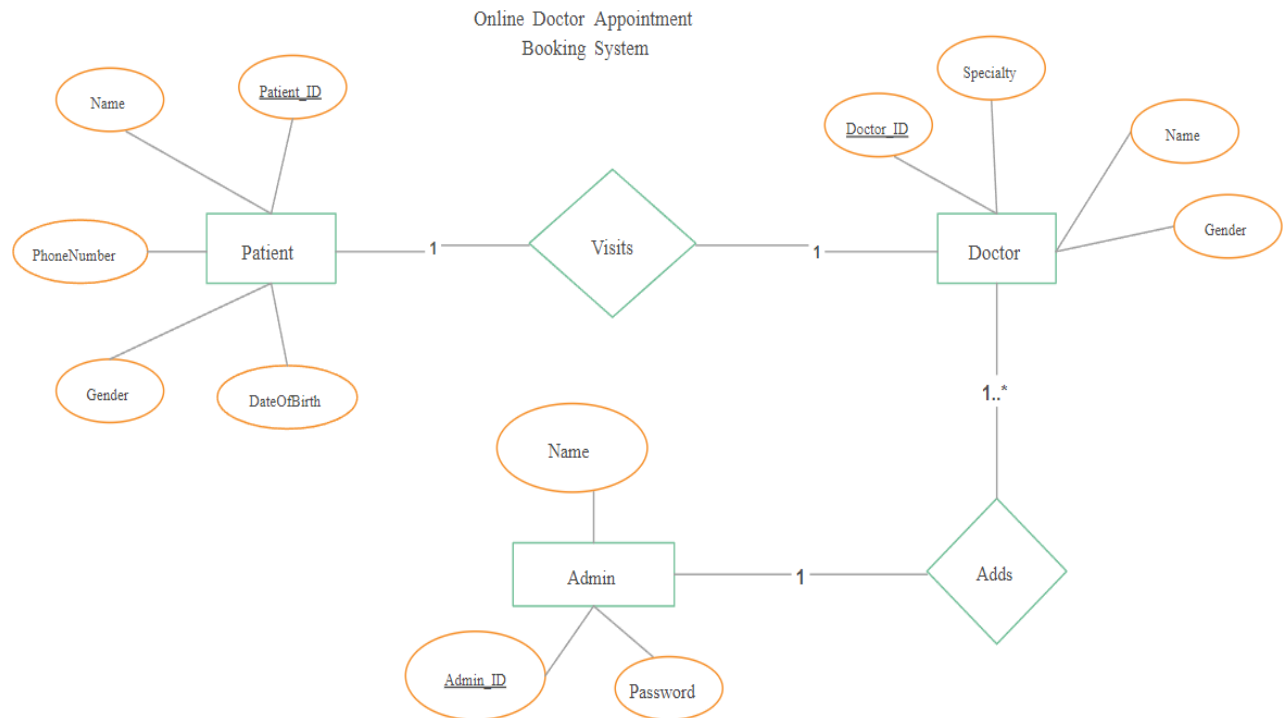
7. Accessibility:

- Mobile-friendly design, multi-language support, and inclusive features for diverse users.

8. Availability Scheduling:

Doctors can set their working hours, define their availability for appointments, and make adjustments for holidays, vacations, or other time.

ER DIAGRAM



Here there is 3 collections namely users, doctors, and appointments which have their own fields in:

***Users:**

- 1.Name.
- 2.Email.
- 3.password
- 4.phone

Doctor:

- 1.Name
- 2.timings
- 3.phone

Appointment:

- 1.id
- 2.doctorInfo
- 3.date

WORKFLOW

1. User Registration/Login:

*Users register by providing their details (name, contact information, etc.) or log in with existing credentials.

*Role-based access ensures users and administrators see relevant dashboards.

2. Appointment Submission:

*Users fill out a structured form specifying the type of appointment, description, and supporting documents (e.g., photos or videos).

*The system validates the inputs and categorizes appointment for efficient routing.

3. Appointment and Tracking ID:

*A unique appointment ID is generated and shared with the user for tracking.

*Users receive acknowledgment via email/SMS, confirming appointment registration.

4. Appointment Processing:

*The appointment is automatically assigned to the relevant department or team based on its category.

*Assigned personnel review, investigate, and update the status of the appointment in the system.

5. Resolution and Updates:

*Users are notified of progress updates, including when the issue is resolved.

*The appointment status changes to "Resolved" upon appointment.

6. Feedback and Closure:

*Users can rate their satisfaction and provide feedback.

*Feedback is logged for performance evaluation.

TECHNICAL STACK

Frontend:

- **HTML, CSS, JavaScript:** For creating the user interface.
- **Frameworks:** React.js, Angular.js, or Vue.js for building interactive and responsive web pages.
- **Bootstrap:** To design mobile-friendly and visually appealing layouts.

Backend:

- **Programming Languages:** javascript, Node.js, MongoDB, or Java (Spring Boot) to handle server-side logic.
- **APIs:** RESTful or GraphQL APIs for communication between frontend and backend.

Database:

- **Relational Databases:** MySQL or PostgreSQL for structured data like user details and complaints.
- **NoSQL Databases:** MongoDB or Firebase for handling unstructured or flexible data.

Authentication and Security:

- **OAuth 2.0/JWT:** For secure user authentication.
- **Encryption:** SSL/TLS to ensure data security.
- **Firewall:** To protect against cyber threats.

Hosting and Deployment:

- **Cloud Services:** AWS, Microsoft Azure, or Google Cloud for hosting and scalability.
- **Web Servers:** Nginx or Apache for serving web content.

SETUP INSTRUCTIONS

Prerequisites:

Ensure you have the following installed on your machine:

1. Node.js (with npm).

- *Download and install Node.js.

- *Ensure both node and npm are installed:

Code:

```
node -v
```

```
npm -v
```

2. MongoDB (Local or Cloud) :

- *Local MongoDB: Install MongoDB on your machine and run it locally.

- *Cloud MongoDB: Create a MongoDB Atlas account at MongoDB Atlas and create a cluster to get a connection string.

- *Git (Optional, for version control)

- *Install Git to clone repositories and manage versions.

- *Verify with git --version.

- *Text Editor (Recommended: Visual Studio Code)

- *Download and install Visual Studio Code.

INSTALLATION STEPS

1. Clone the Repository:

- *If the project is hosted on GitHub (or other Git repositories), clone it to your local machine:

- *GIT clone <https://github.com/VIGNESH-M-2003/DOCTOR-BOOKING-APP.git>

- *cd doctor-booking-system.

2. Backend Setup (Node.js and Express):

- *Navigate to the Backend Directory (if the project is separated into frontend and backend directories)

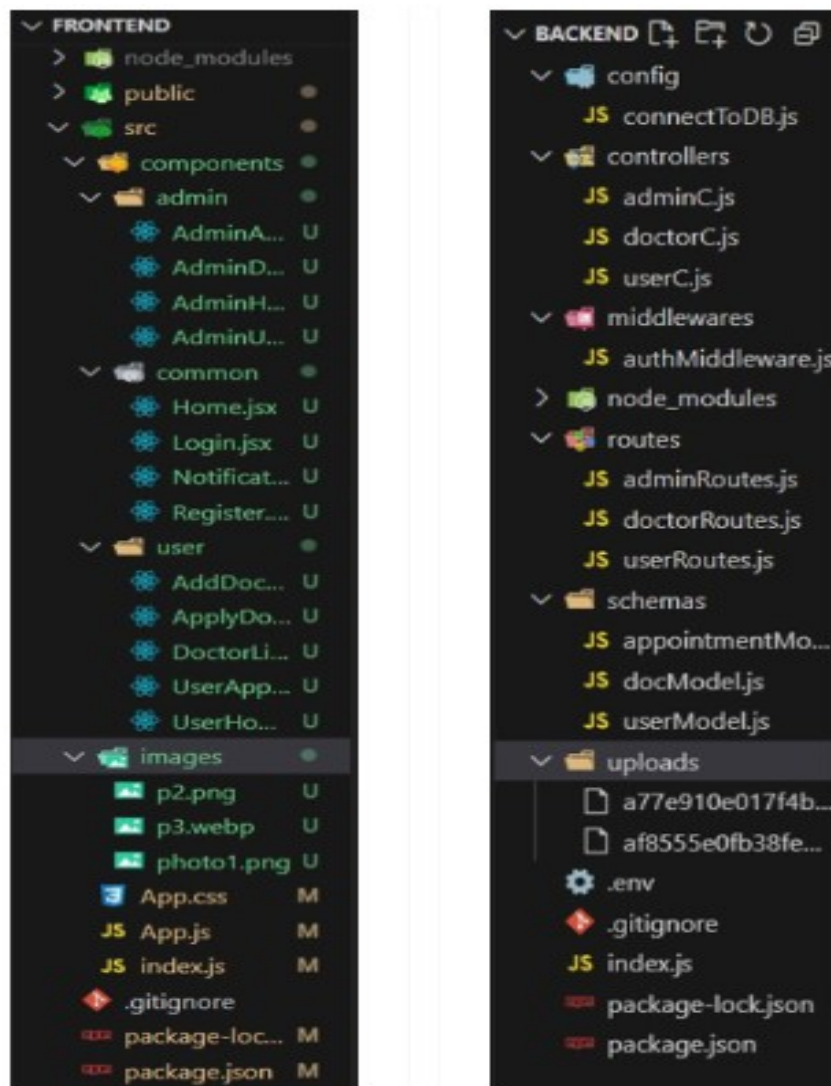
- *Install Backend Dependencies: Install required Node.js packages using npm.

Command:

```
*cd backend.
```

```
*npm install.
```

FOLDER STRUCTURE



BACKEND SAMPLE PROGRAM

Get Doctor by ID Endpoint:

/doctors/:

id Method:

GET Description:

Retrieve details for a specific doctor.

Response:

json

```
{ "success": true,  
  
  "doctor": {  
  
    "id": "doc123",  
  
    "name": "Dr. Smith" ,  
  
    "specialization": "Cardiology",  
  
    "availableSlots": ["2024-11-18T10:00:00", "2024-11-18T11:00:00"]  
  
  }  
  
}
```

} Booking

Book an Appointment

Endpoint:/appointments

Method: POST

Description: Book an appointment

Request Body:

json

```
{  
  
  "userId": "user123",  
  
  "doctorId": "doc123",  
  
  "appointmentDate":  
  
  "2024-11-18T10:00:00"  
  
}
```

- **Install Backend Dependencies:** Install required Node.js packages using npm

Command:

- `cd backend`
- `npm install`

1. Create .env File for Environment Variables:

- Create a .env file in the root of the backend folder to store sensitive information like the MongoDB URI and JWT secret.
- If using MongoDB Atlas, replace the MONGO_URI with your connection string from MongoDB Atlas.

Example .env:

```
MONGO_URI=mongodb://localhost:27017/doctor_booking  
JWT_SECRET=your_jwt_secret_key  
PORT=5000
```

2. Start the Backend Server:

Start the backend server using the following command:

- `npm start`
- The server should now be running on <http://localhost:5000>.

3. Cancel Appointment:

Endpoint: /appointments/:id

Method: DELETE

Description: Cancel a specific appointment.

Response:

json

```
{  
  
  "success": true,  
  
  "message": "Appointment canceled successfully"  
}
```

Middleware:

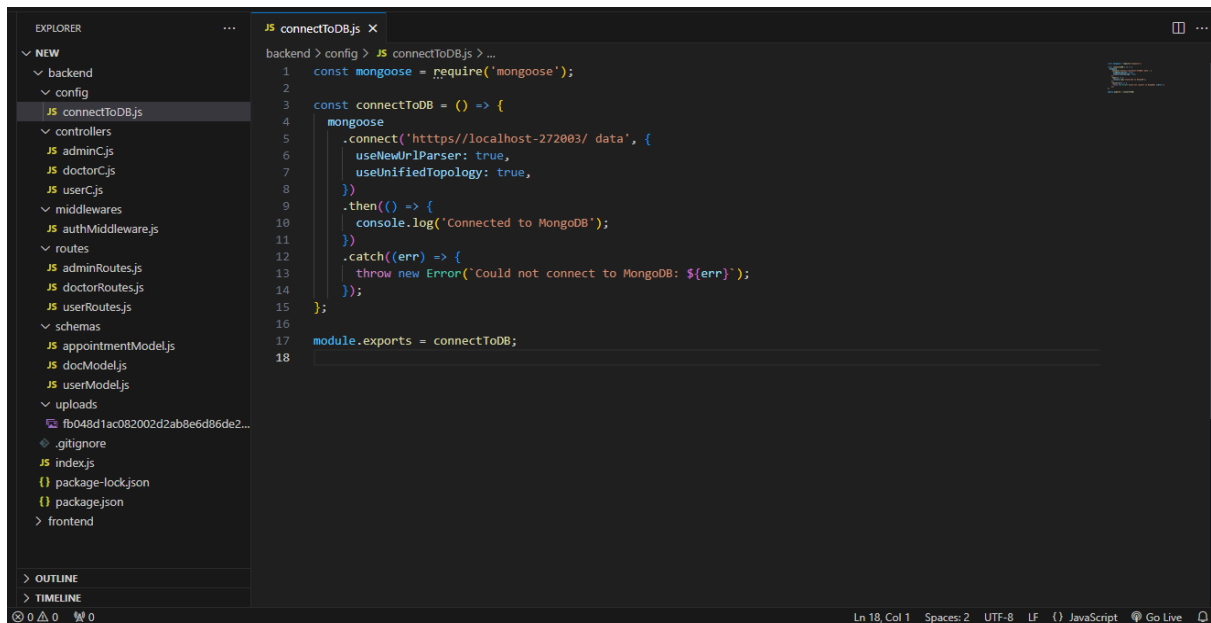
JWT Authentication

Protect routes to ensure only authenticated users can access them.

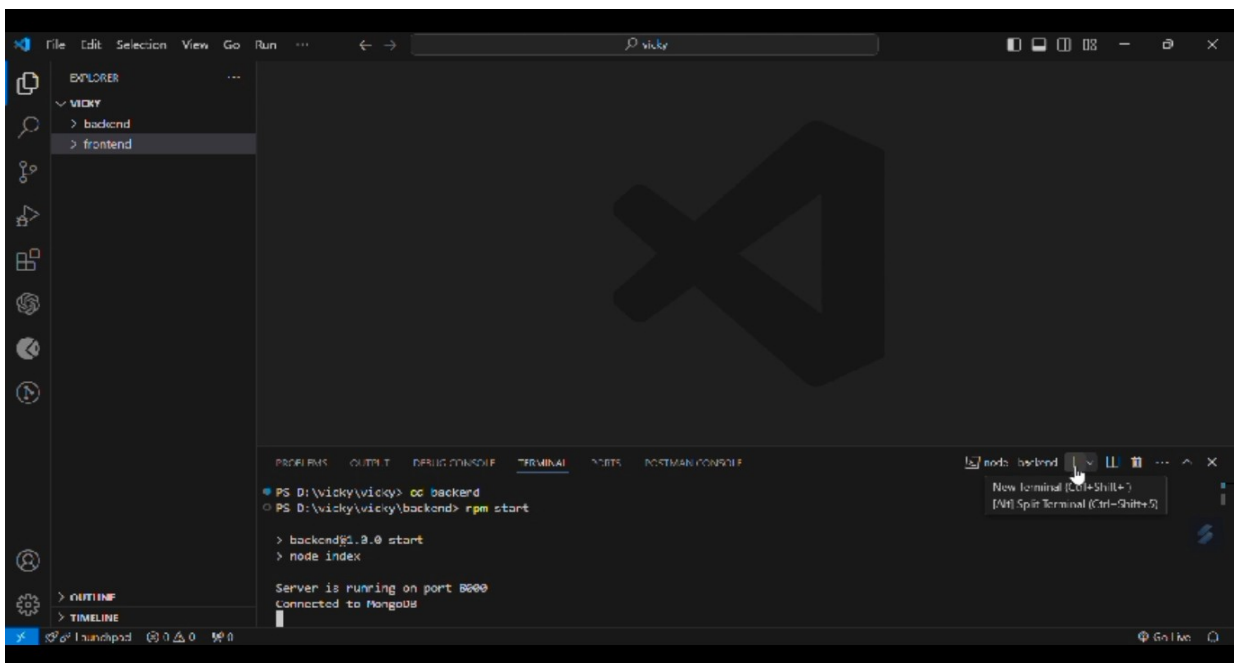
Error Handling

Responses should include an appropriate status code and error message:

- 400: Bad Request
- 401: Unauthorized
- 404: Not Found
- 500: Internal Server Error



```
1  const mongoose = require('mongoose');
2
3  const connectToDB = () => {
4    mongoose
5      .connect('https://localhost-272003/ data', {
6        useNewUrlParser: true,
7        useUnifiedTopology: true,
8      })
9      .then(() => {
10        console.log('Connected to MongoDB');
11      })
12      .catch((err) => {
13        throw new Error('Could not connect to MongoDB: ${err}');
14      });
15  };
16
17  module.exports = connectToDB;
18
```



The MongoDB was connect to the sever backend

Frontend setup

1. Frontend Setup (React.js)

a. Navigate to the Frontend Directory:

cd frontend

b. Install Frontend Dependencies: Install React.js dependencies using npm:

npm install

- c. **Set up Environment Variables** (for frontend API calls): If required, create a .env file in the frontend directory to store the backend API URL or other settings.

Example .env:

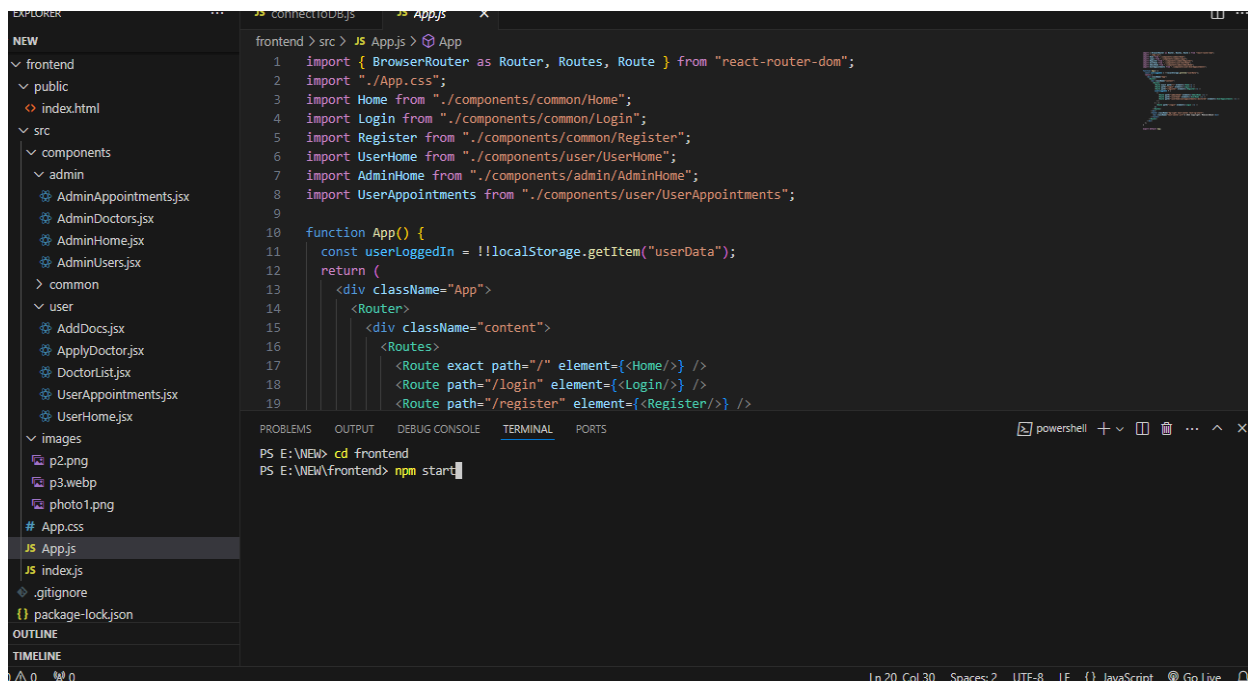
```
REACT_APP_API_URL=http://localhost:5000/api
```

Start the Frontend Development Server:

*Run the react development server using

*npm start

- The frontend should now be running on <http://localhost:3000>.



USER INTERFACE

Designing the user interface (UI) for a doctor booking system involves creating a seamless, intuitive experience for users to interact with features like registration, login, viewing doctors, and booking appointments. Here's an outline of the key components and design considerations:

1. UI Design Overview

The UI will have three primary roles:

- Patients: Register, login, view doctors, and book appointments.
- Doctors: Manage availability and view appointments.
- Admin (optional): Manage users and doctors.

2. Core Pages for Patients

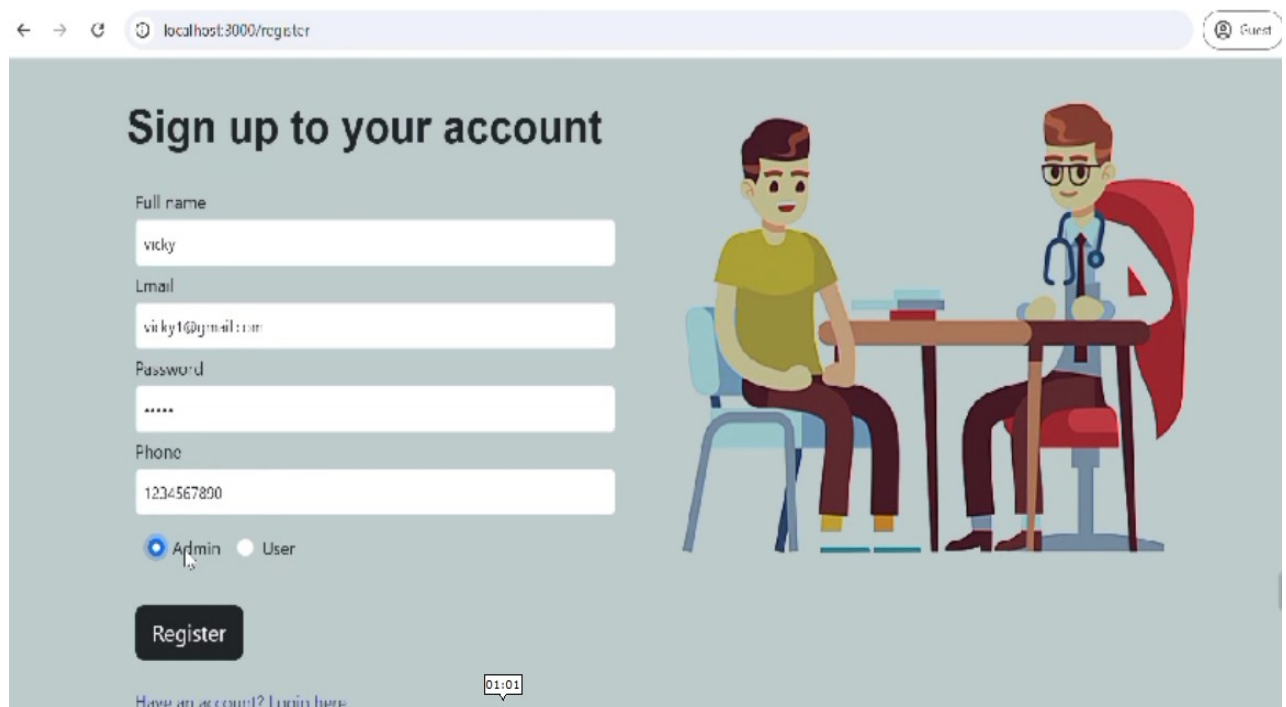
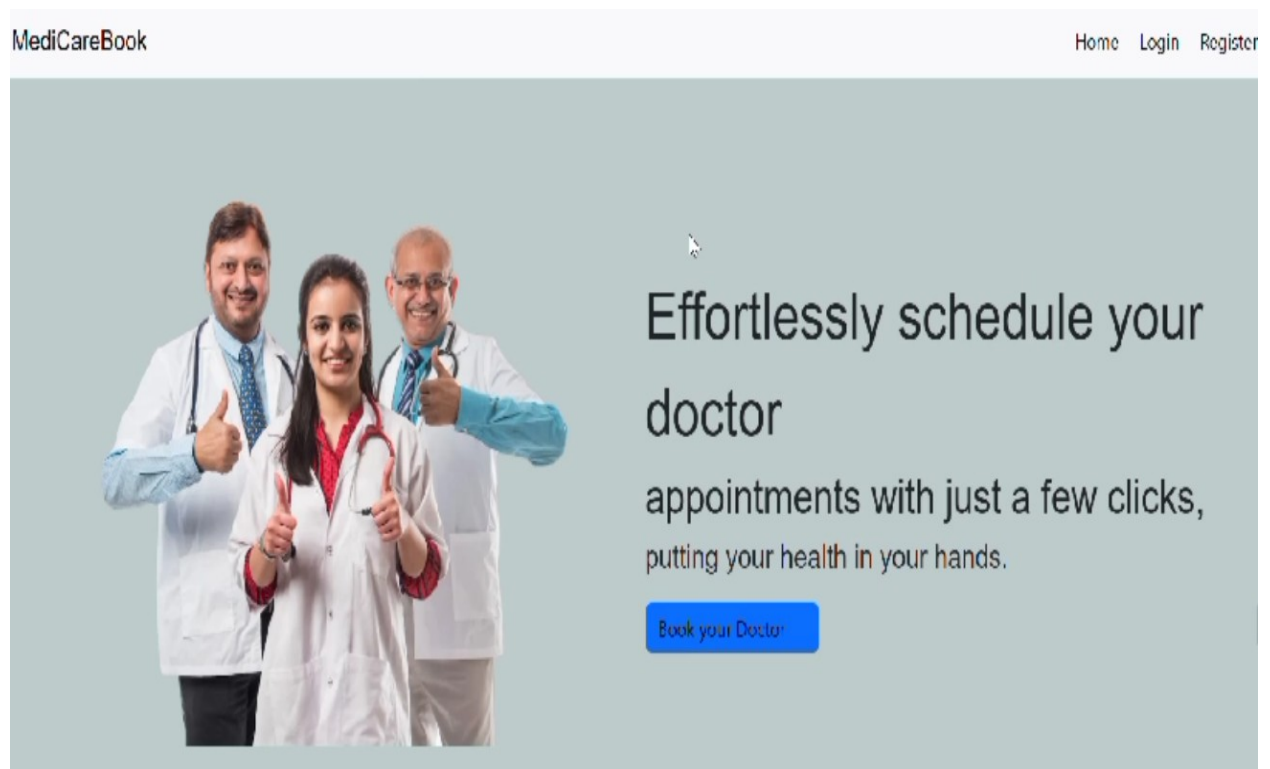
A. Home Page

Purpose: Welcome users and provide quick navigation options.

Features:

- Navigation menu (e.g., Home, Doctors, Services, Contact, Login).
- Call-to-action buttons: "Book a
- " or "Sign In/Register."

SAMPLE OUTPUT



B. Registration Page

Purpose: Allow new users to sign up.

Elements:

Form with fields:

- Full Name
- Email
- Password (with password strength indicator)
- Confirm Password
- "Register" button.

Link to "Already have an account? Login."

B. Login Page

Purpose: Authenticate existing users.

Elements:

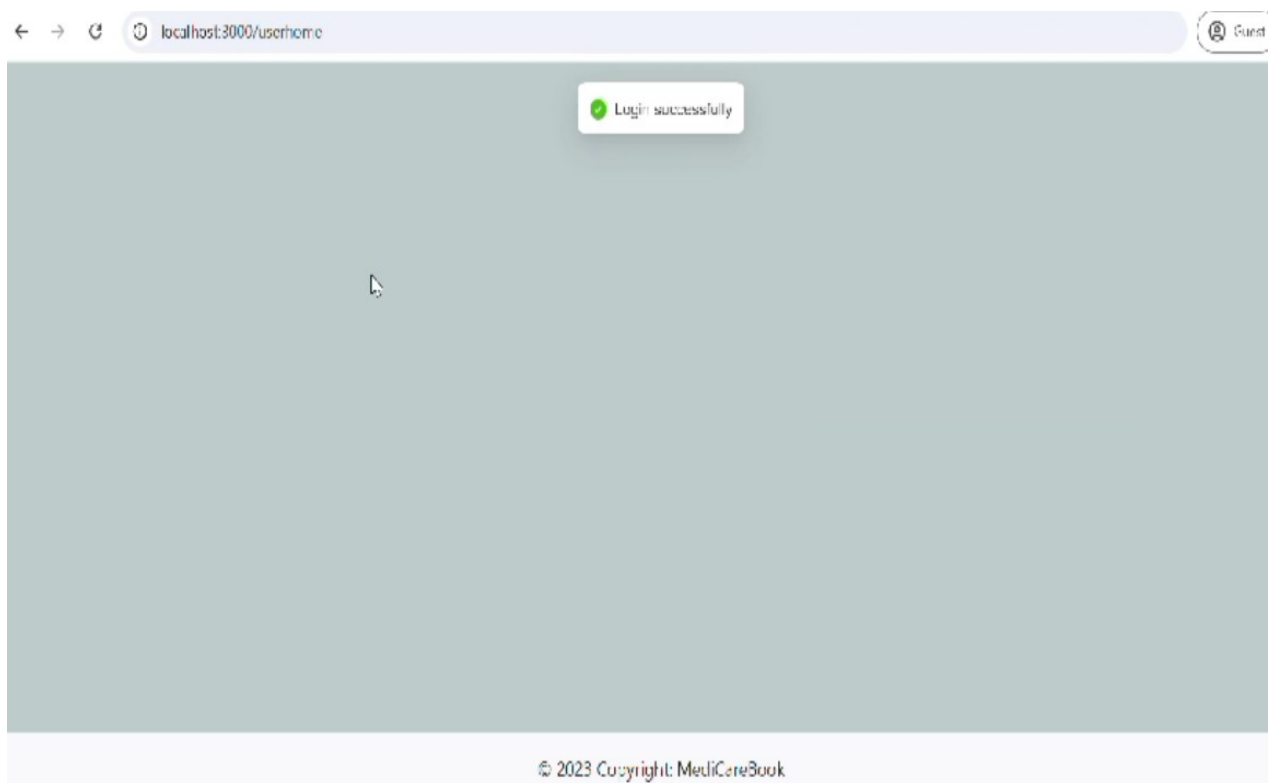
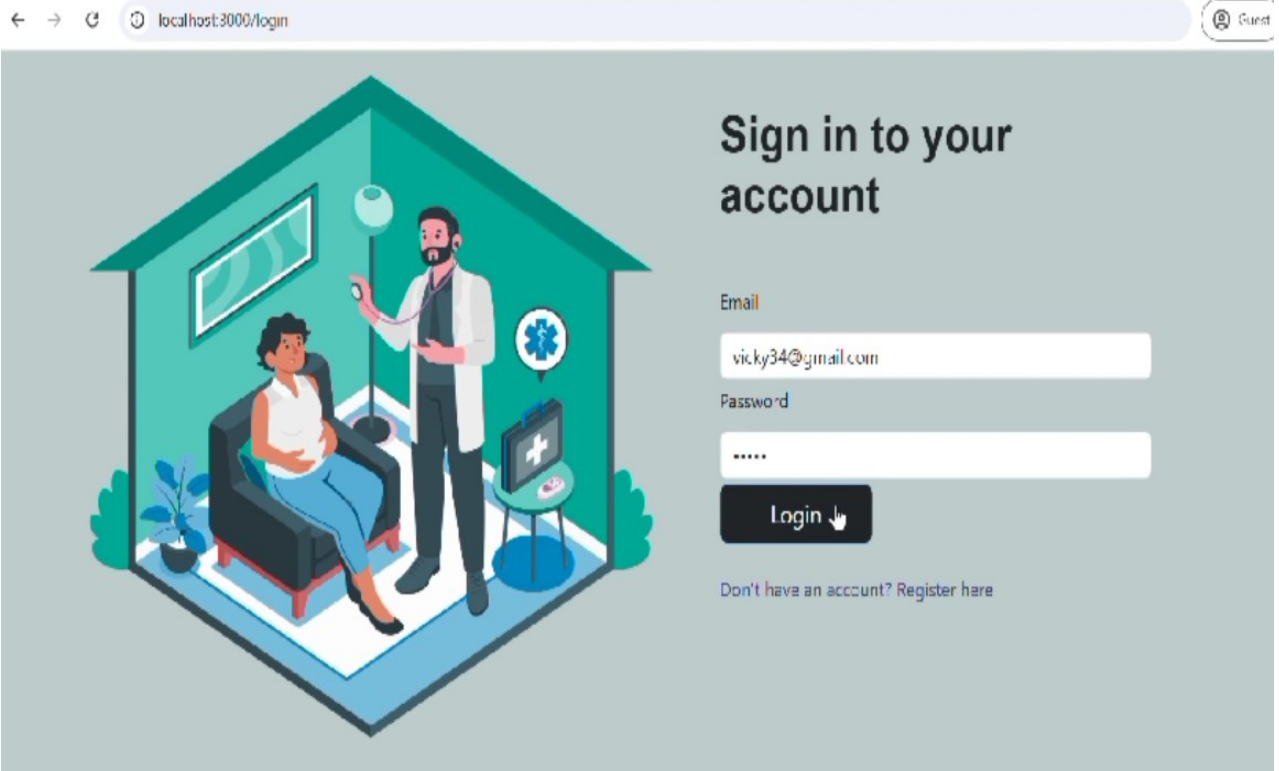
- Email and Password fields.
- "Login" button.
- **Link to "Forgot Password?" and "New here? Register."**

Doctors Listing Page

Features:

- Search bar to find doctors by name, specialization, or location.
- Filters for specialization, availability, and ratings.
- Card-based layout for each doctor:

Doctor's photo, name, specialization, and rating.



Doctor Details Page

Purpose: Provide detailed information about a specific doctor.

Features:

- Doctor's profile:
 - Name, photo, qualifications, specialization, experience.
- Ratings and reviews.
- Available time slots (selectable).
- **"Book Now" button.**

Appointment Booking Page

Purpose: Allow users to book an appointment with a doctor.

Features:

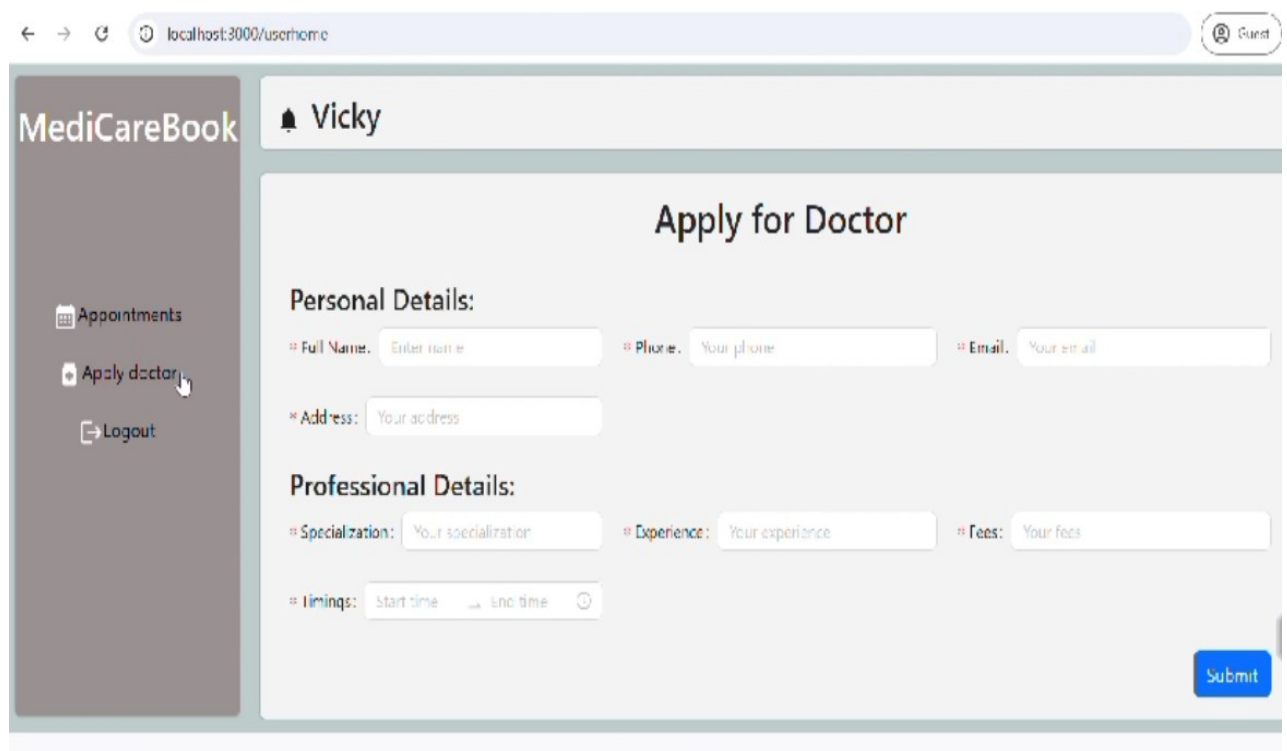
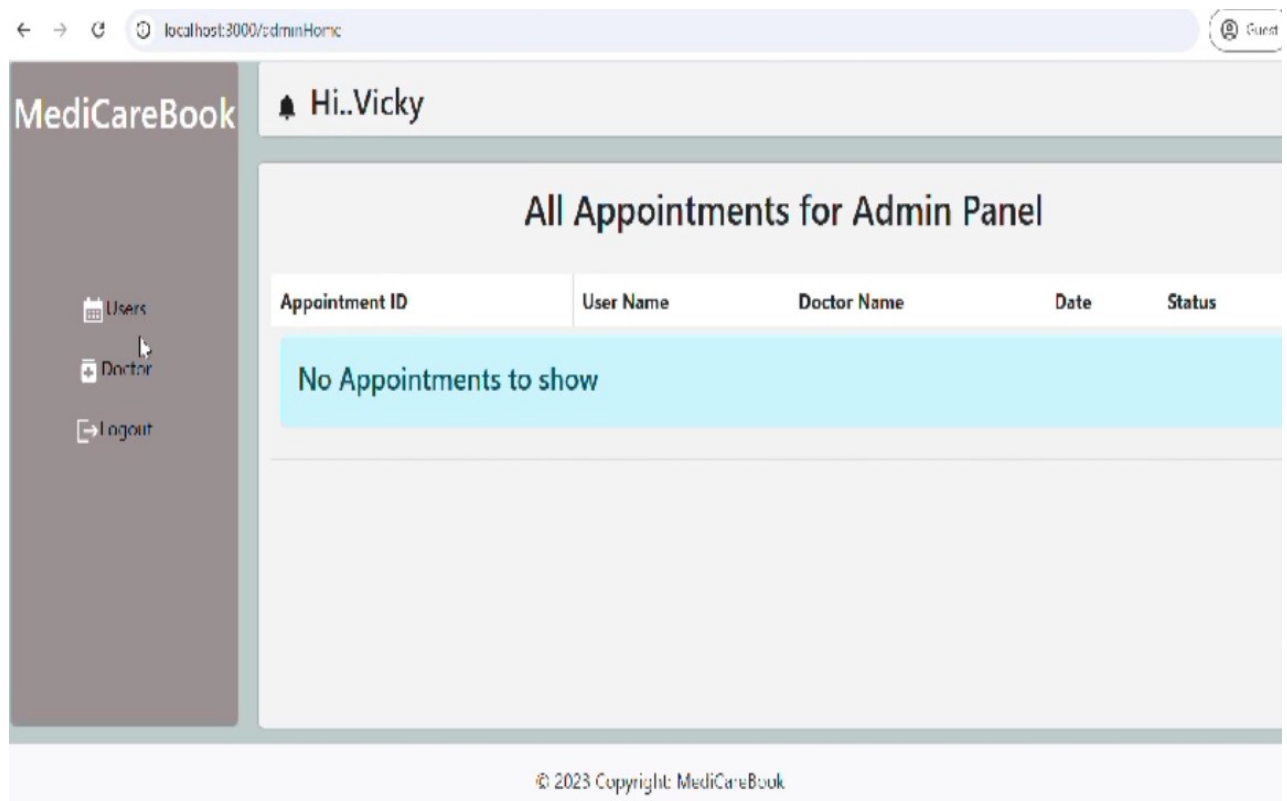
- Selected doctor's details.
- Date picker to select a preferred date.
- **Time slots (based on the doctor's availability).**

Appointment History Page

Purpose: Show upcoming and past appointments.

Features:

- List of booked appointments (upcoming at the top).
- Details: Doctor's name, date, time, and status (confirmed, canceled).
- **Cancel option for upcoming appointments.**



CONCLUSION

An Doctor Booking System built using the mern **Website** is a powerful tool for streamlining grievance- handling processes. It enhances transparency, accountability, and efficiency, creating a better experience for users and organizations alike. While challenges like data security and high traffic exist, proper planning and mitigation strategies ensure a reliable system. As technology advances, integrating AI-driven solutions and real-time analytics can further elevate the platform, making it indispensable for modern service ecosystems. By embracing digital solutions, organizations not only resolve complaints effectively but also foster trust and long-term user satisfaction.