**FULL STACK DEVELOPMENT WITH MERN – PROJECT DOCUMENTATION**

1. **INTRODUCTION**

**Project Title:** DocSpot-Seamless Appointment Booking For Health

**Team Members:**

1. **Team Leader :** Yajjuvarapu Rikshith - Not actively involved in development tasks
2. **Team member :** Yugandhar Veenam - No participation in project execution
3. **Team member :** Yalla Nikhila - sole contributor to the project, handling the complete frontend and backend development, database setup, authentication, real-time chat, testing, and full project documentation.

**2. PROJECT OVERVIEW**

**DocSpot** is a web-based healthcare platform designed to streamline the process of booking doctor appointments. Patients can view available doctors, book appointments by specialty or availability, and receive real-time confirmation. Doctors can manage their availability, appointments, and patient interactions.

**Scenario-Based Flow**

* **User (John) registers → filters doctors → books a slot**
* **Doctor (Dr. Neha) receives booking → confirms appointment → updates consultation status**
* **Admin monitors platform activity and verifies doctors**

**Goals**

* **Simple and transparent appointment booking**
* **Real-time communication and notifications**
* **Multi-role access for patients, doctors, and admins**

**FEATURES**

1. **Patient Registration/Login**
   * **Email/password or OAuth sign-in (Gmail optional)**
   * **Profile creation and session handling**
2. **Doctor Search & Filtering**
   * **Filter by location, specialization, availability**
   * **View profiles and consultation fees**
3. **Appointment Booking**
   * **Choose date/time**
   * **Upload medical history or documents (optional)**
4. **Doctor Dashboard**
   * **View bookings**
   * **Accept/decline appointments**
   * **Add visit notes & summaries**
5. **Admin Control Panel**
   * **Approve new doctor accounts**
   * **Oversee user and system activity**
6. **Notifications**
   * **Email/SMS reminders (via Nodemailer or Twilio)**
   * **Confirmation messages after booking**
7. **Feedback & Rating**
   * **Rate doctor and consultation experience**

**3.ARCHITECTURE**

**Frontend Architecture (React.js)**

The frontend of **ResolveNow** is developed using **React.js**, a powerful JavaScript library for building user interfaces. It follows a **component-based architecture**, allowing reusable and maintainable code. Major UI libraries like **Material UI** and **Bootstrap** are used to ensure responsiveness and modern design.

* **Axios** handles API communication between frontend and backend.
* Components are structured by role: UserDashboard, AdminPanel, AgentView, etc.
* Form validation, conditional rendering, and localStorage are used for session handling and role-specific access.

**Backend Architecture (Node.js + Express.js)**

The backend is built on **Node.js** with the **Express.js** framework, following a modular and RESTful API structure. It handles:

* **Routing** for user, admin, agent, and complaint-related endpoints
* **JWT-based Authentication** and middleware for access control
* **Real-time communication** via **Socket.IO** for user-agent chat
* Middleware like cors, body-parser, and custom error handlers for smooth request handling

**Database Architecture (MongoDB + Mongoose)**

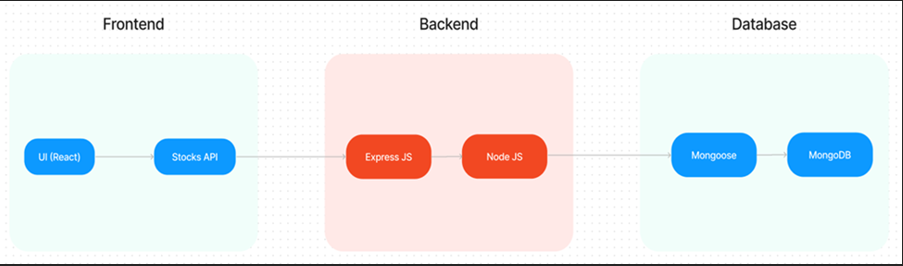
MongoDB is used as the NoSQL database to store application data. Mongoose ORM is used to define clear schemas and interact with the database efficiently.

**Key Schemas:**

* **User Schema:** Stores user details (name, email, password, role – user/admin/agent)
* **Complaint Schema:** Stores complaint info including userId, details, status, timestamps
* **Message Schema:** Stores messages between users and agents with complaintId references
* **Feedback Schema:** Captures user ratings and comments post-resolution

Data is stored in collections and linked using ObjectIds, enabling relational behavior in a document-based model.

**TECHNICAL ARCHITECTURE**

****

**Architecture Type: 3-Tier (Presentation, Logic, Data)**

| **Layer** | **Technology / Role** |
| --- | --- |
| **Presentation Layer** | **React.js, Bootstrap, Material UI – responsive frontend** |
| **Logic Layer** | **Node.js + Express.js – API handling and middleware** |
| **Data Layer** | **MongoDB + Mongoose – user, doctor, and appointment data** |

**Additional Integrations:**

* **JWT + Bcrypt for Authentication**
* **Axios for API communication**
* **Socket.IO/WebRTC (optional for future telehealth)**
* **Nodemailer/Twilio (notifications)**

1. **SETUP INSTRUCTIONS**

### **PRE-REQUISITES**

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

**Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

Download: https://nodejs.org/en/download/

Installation instructions: https://nodejs.org/en/download/package-manager/

**Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

**INSTALLATION**

**npm install express**

**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data. 

Set up a MongoDB database to store your application's data.

Download: https://www.mongodb.com/try/download/community

Installation instructions: https://docs.mongodb.com/manual/installation/

**React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

**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. To Connect the Database with Node JS go through the below provided link:

https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

**Front-end Framework**: Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and boostrap.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

Git: Download and installation instructions can be found at: https://git-scm.com/downloads

**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from <https://code.visualstudio.com/download>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

* Open your terminal or command prompt.
* Navigate to the directory where you want to store the e-commerce app.
* Execute the following command to clone the repository:

**git clone**: <https://github.com/awdhesh-student/complaint-registery.git>

**Install Dependencies:**

• Navigate into the cloned repository directory:

cd complaint-registery

• Install the required dependencies by running the following commands:

cd frontend

npm install

cd ../backend

npm install

Start the Development Server:

• To start the development server, execute the following command:

npm start

• The online complaint registration and management app will be accessible at [http://localhost:3000](http://localhost:3000/)

You have successfully installed and set up the online complaint registration and management app on your local machine. You can now proceed with further customization, development, and testing as needed.

**6. FOLDER STRUCTURE**



**7. APPLICATION FLOW**

**Patient**

* **Register/Login**
* **Browse doctors**
* **Book appointments**
* **Track appointment status**
* **Provide feedback**

**Doctor**

* **Register/Login**
* **Add available slots**
* **Accept/decline appointments**
* **View schedule**

**Admin**

* **Review doctor profiles**
* **Approve/reject doctors**
* **Monitor bookings**
* **Ensure compliance**

**8. API DOCUMENTATION**

**Base URL:** [**http://localhost:8000/api**](http://localhost:8000/api)





* **POST /auth/signup – Register (Patient/Doctor)**
* **POST /auth/login – Login and get JWT**
* **GET /doctors – List all doctors**
* **POST /appointments/book – Book appointment**
* **PATCH /appointments/:id/status – Update status**
* **GET /appointments/user/:id – View patient bookings**
* **POST /feedback – Submit review**

**9. AUTHENTICATION**

* **JWT-based Auth**
  + **Tokens contain user ID and role**
  + **Stored in localStorage**
  + **Used in Authorization: Bearer <token> headers**
* **Role-Based Access Control:**
  + **Patients → Booking + Feedback**
  + **Doctors → Appointment management**
  + **Admin → System control**

**10. TESTING**

**Tools:**

* **Postman (API)**
* **DevTools (Network & Debugging)**
* **React Developer Tools**

**Tested Flows:**

* **Register/Login (Valid & Invalid)**
* **Booking creation & conflict testing**
* **Doctor filtering**
* **Auth-protected routes**
* **Status updates**
* **Feedback submission**

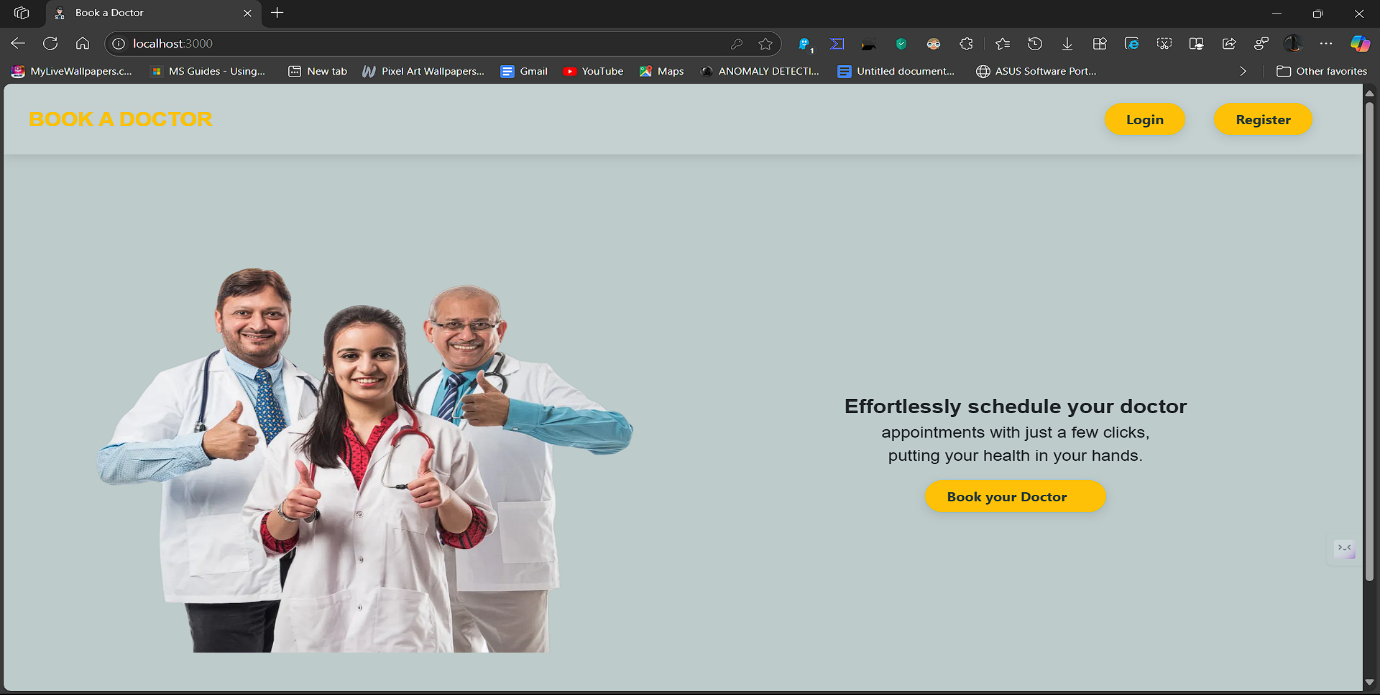
**11. UI SCREENS**

* **Home Page**
* **User Registration/Login**
* **Doctor Dashboard**
* **Patient Dashboard**
* **Admin Panel**
* **Appointment Form**
* **Feedback Page**

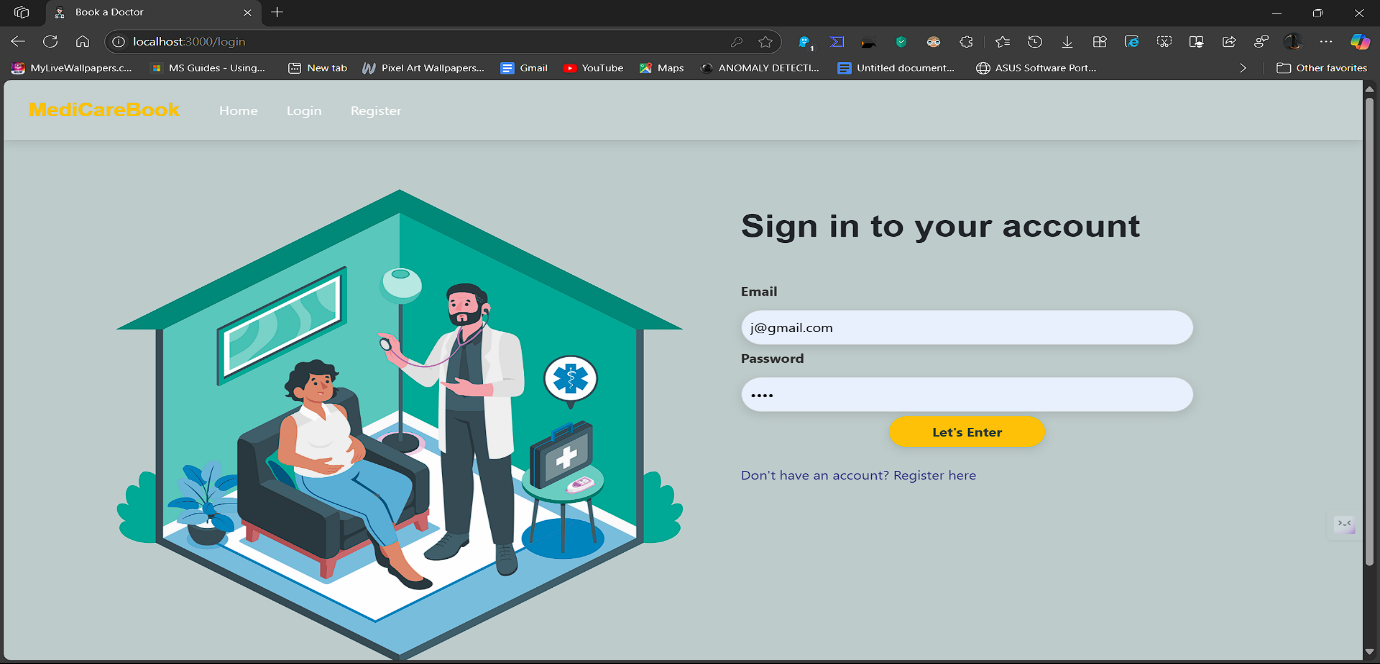
**12. DEMO LINK & SCREENSHOTS**

**Demo Video:** [**https://drive.google.com/file/d/1hoP5YLIrn5QzWkwfvigWaqWdKf-L9Y16/view?usp=drive\_link**](https://drive.google.com/file/d/1hoP5YLIrn5QzWkwfvigWaqWdKf-L9Y16/view?usp=drive_link) **Screenshots:**

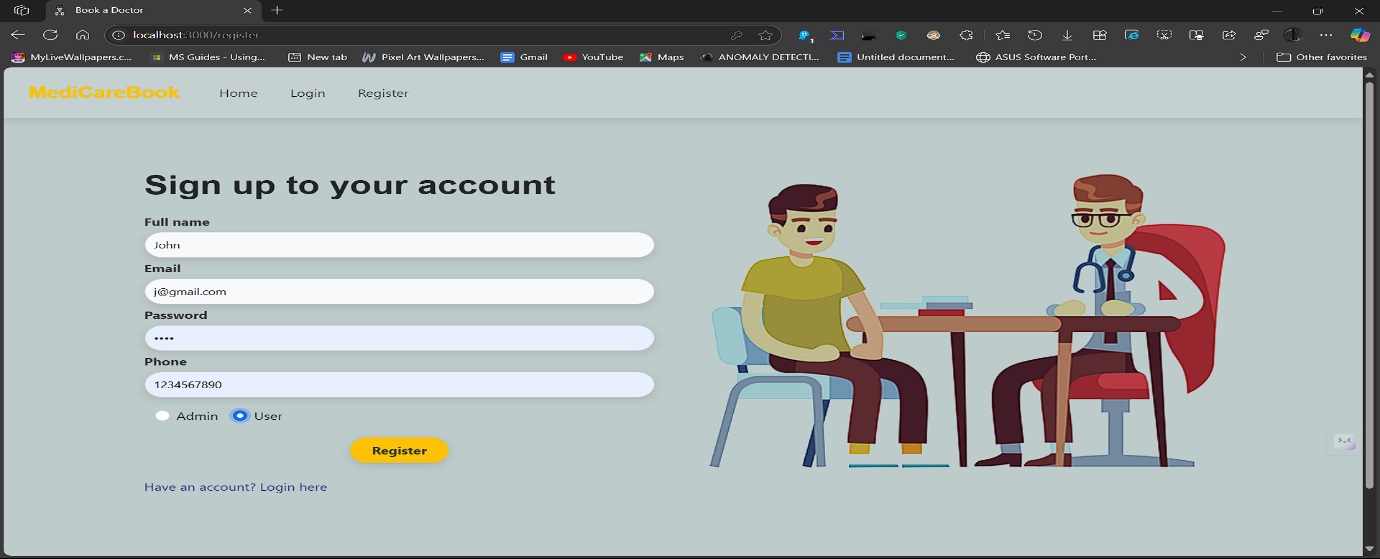
**LANDING PAGE :**

****

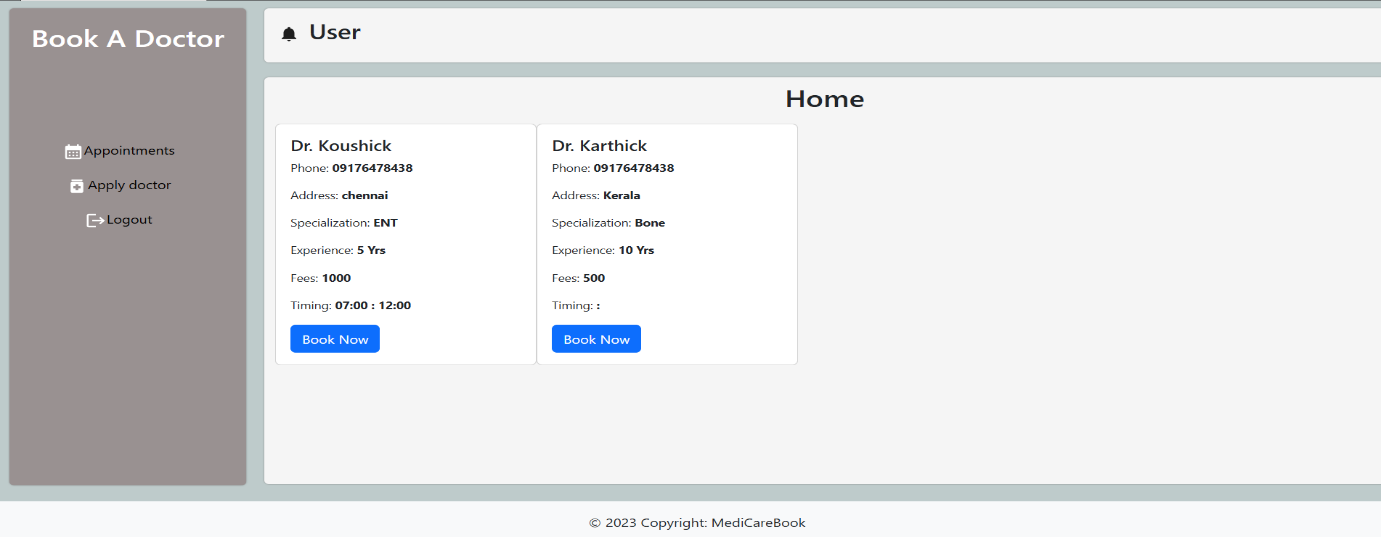
**LOGIN PAGE :**

****

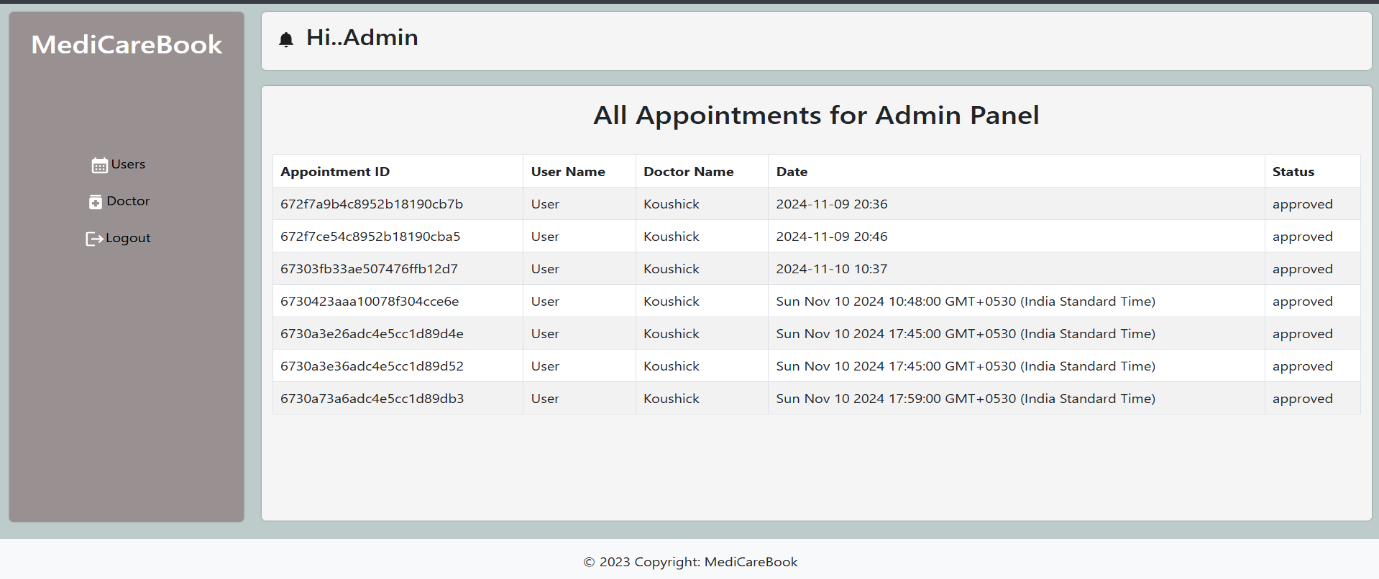
**REGISTRATION PAGE :**

****

**USER PAGE :**

****

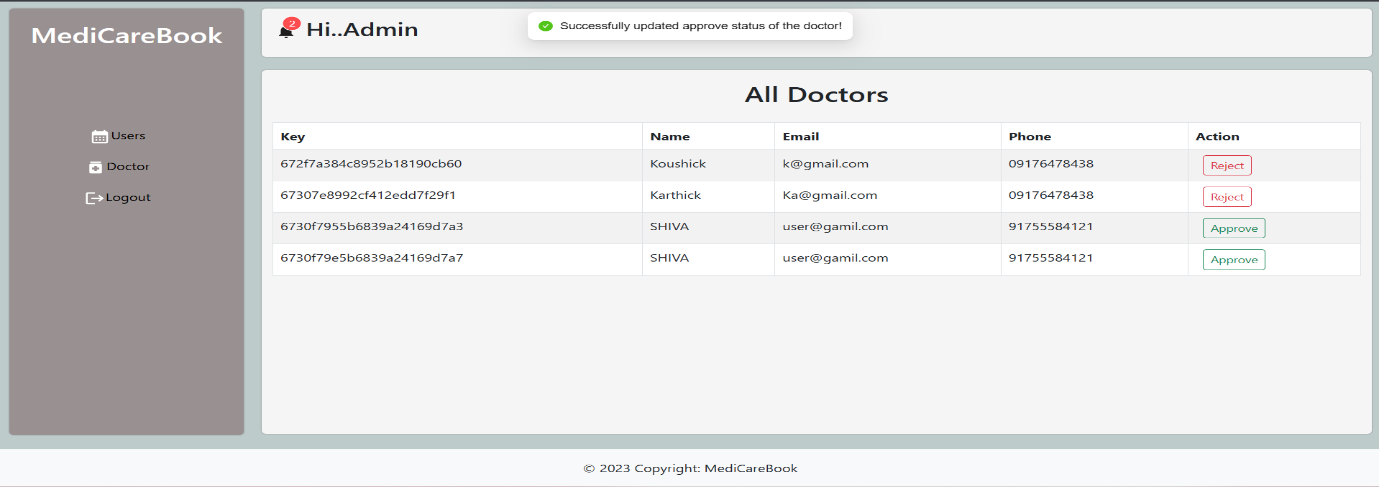
**ADMIN PAGE :**

****

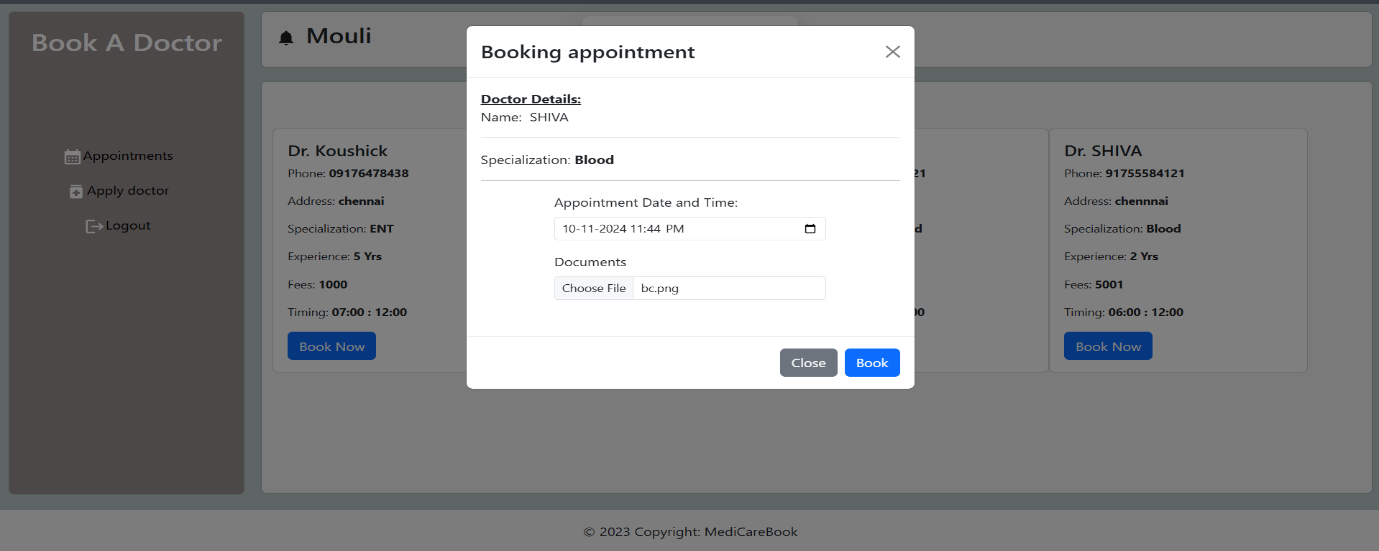
**APPLY AS DOCTOR :**

****

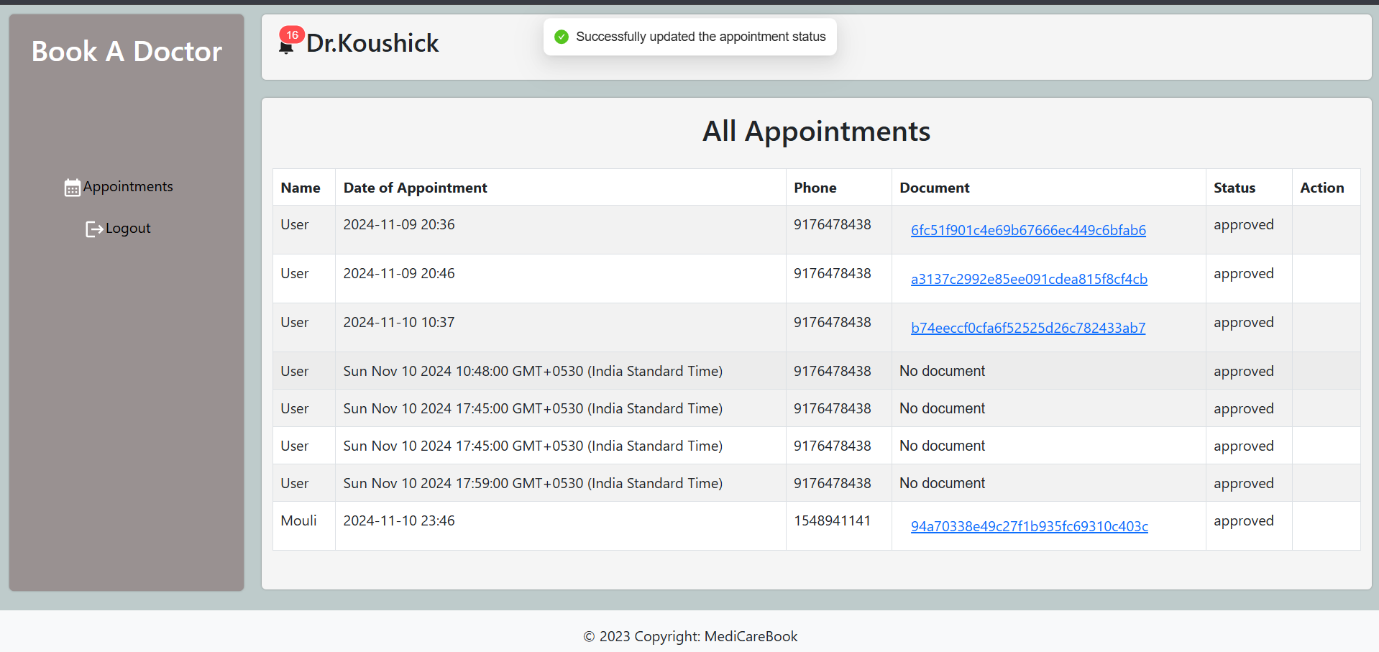
**ADMIN APPROVE DOCTOR :**

****

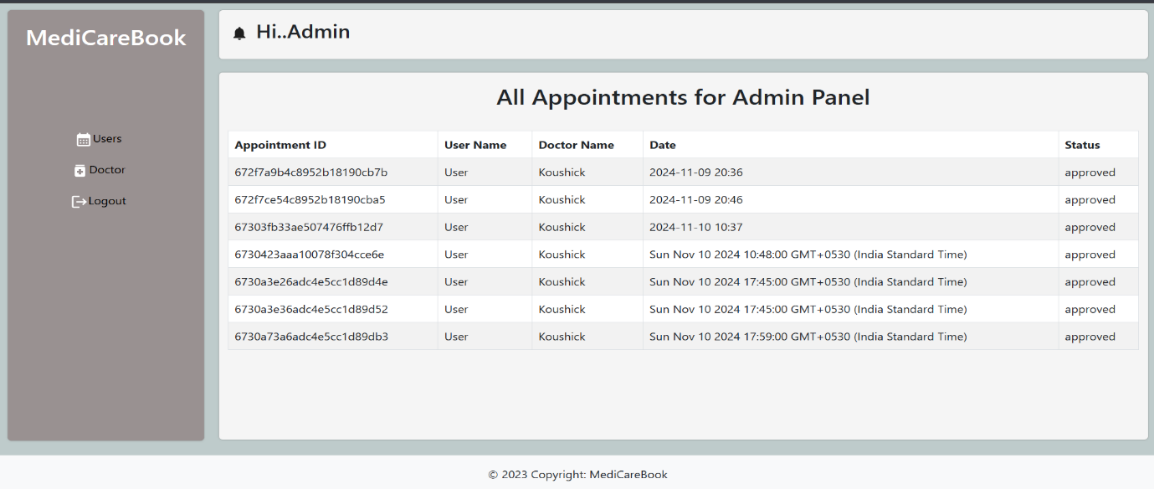
**BOOK DOCTOR :**

****

**DOCTOR APPROVE USER APPOINTMENT :**

****

**ALLHISTORY :**

****

**13. KNOWN ISSUES**

* **No image upload preview yet**
* **Booking conflict not prevented for same slot (WIP)**
* **Mobile UI needs additional polish**
* **No email OTP validation in current version**
* **Feedback form doesn't auto-clear after submission**

**14. FUTURE ENHANCEMENTS**

* **Video consultation with WebRTC**
* **Push notifications (via Firebase or OneSignal)**
* **Google Calendar integration for doctors**
* **Payment Gateway (Stripe/Razorpay)**
* **Export PDF prescription reports**
* **Chat with doctors using Socket.IO**
* **Admin analytics dashboard**