Full Stack Developer (MERN Stack)

UNIT-I : Introduction to FSD and Basics Web Technologies

Intro to Full stack – What is Frontend ? – What is the backend? – Roles & Responsibility for Full Stack Developer – Environment Setup Introduction to HTML & CSS – Basic elements, DOM-create/delete elements – Selectors – Advanced CSS techniques like flexbox and grid

UNIT-II : JavaScript & ES6 Essentials

Introduction to Java script – Variables, datatypes, and operators – Control flow statements (if-else, for, while, switch) – Introduction to ES6 (let, const, template strings) – Arrow function – Spread operator – destructing – Callback – Promise – JavaScript fundamentals: functions, objects, arrays – Manipulating the DOM with JavaScript – Handling events and user interactions with JavaScript

UNIT-III : React JS

Introduction to React and its features – Setting up a React development environment – JSX syntax and its benefits – Creating React components – Creating reusable React components – Using props to pass data between components – Creating conditional rendering and handling events in React – React State, Event Handling & Forms – Understanding state and its importance in React – Setting state and handling events in React – Using forms and controlled components in React – Handling errors and edge cases in React – Understanding the React lifecycle and its phases – Using lifecycle methods to optimize performance – Introduction to React hooks – Implementing custom hooks in React

UNIT-IV : Node JS & Express JS

Introduction to Node.js and its features – Understanding the basics of web servers and HTTP requests – Setting up an Express.js development environment – Building a simple Express.js server – Understanding the principles of RESTful APIs – Building CRUD operations with Express.js – Implementing middleware in Express.js – Understanding the Node.js event loop and asynchronous programming – Using callbacks, promises, and async/await in Node.js – Handling errors and debugging Node.js applications – Implementing security best practices in Node.js

UNIT-V : MongoDB, Integration of Frontend & Backend

Understanding NoSQL databases and MongoDB – Setting up a MongoDB development environment – Building MongoDB schema and models with Mongoose – Using Mongoose to perform CRUD operations in MongoDB – Understanding MongoDB indexing and aggregation – Implementing authentication and authorization with MongoDB - AXIOS- CORS- Integration of frontend with backend

DocSpot: Seamless Appointment Booking for Health

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.

With our advanced booking system, you can say goodbye to the hassle of traditional appointment booking. Our platform offers real-time availability, allowing you to choose from a range of open slots that fit your schedule. Whether you prefer early morning, evening, or weekend appointments, we have options to accommodate your needs.

Scenario-based Case Study:

Scenario: Booking an Appointment with a Doctor

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.

He filters the list based on his preferences, such as specialty, location, or availability.

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.

After submitting the form, John receives a confirmation message indicating that his appointment request has been received.

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

John receives a notification confirming his appointment and providing details such as the date, time, and location.

Appointment Management: As the appointment approaches, John can view and manage his upcoming appointments in the booking history section of his dashboard.

He has the option to cancel or reschedule appointments if needed and can update the status accordingly.

Admin Approval (Background Process): In the background, the admin reviews new doctor registrations and approves legitimate applicants.

Approved doctors are then registered in the app and can start managing their appointments.

Platform Governance: The admin oversees the overall operation of the appointment booking system and ensures compliance with platform policies, terms of service, and privacy regulations.

The admin addresses any issues or disputes to maintain a smooth user experience.

Doctor's Appointment Management: Dr. Smith, an approved doctor on the platform, logs into his account and manages his appointments.

He views his schedule, confirms or reschedules appointments, and updates appointment statuses based on patient interactions.

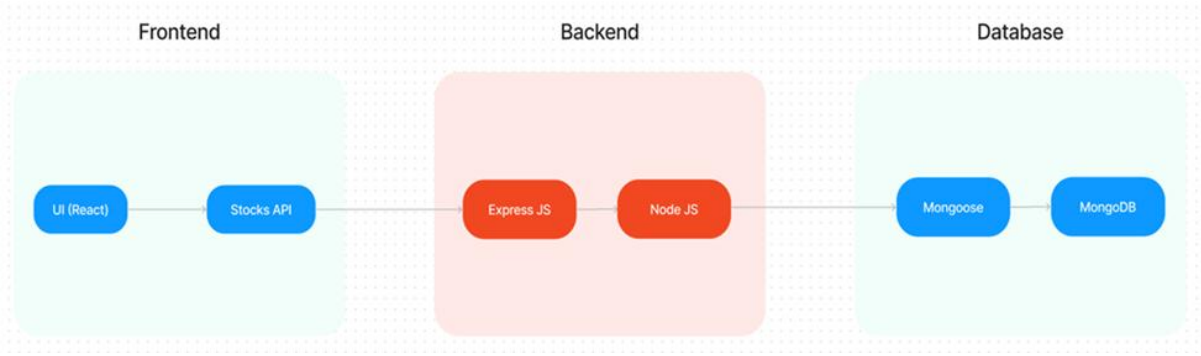
Appointment Consultation: On the day of the appointment, John visits the doctor's office for his check-up.

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

Post-Appointment Follow-up: After the appointment, Dr. Smith updates John's medical records and may prescribe medication or recommend further treatment if necessary.

John receives a visit summary and any follow-up instructions through the app.

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.

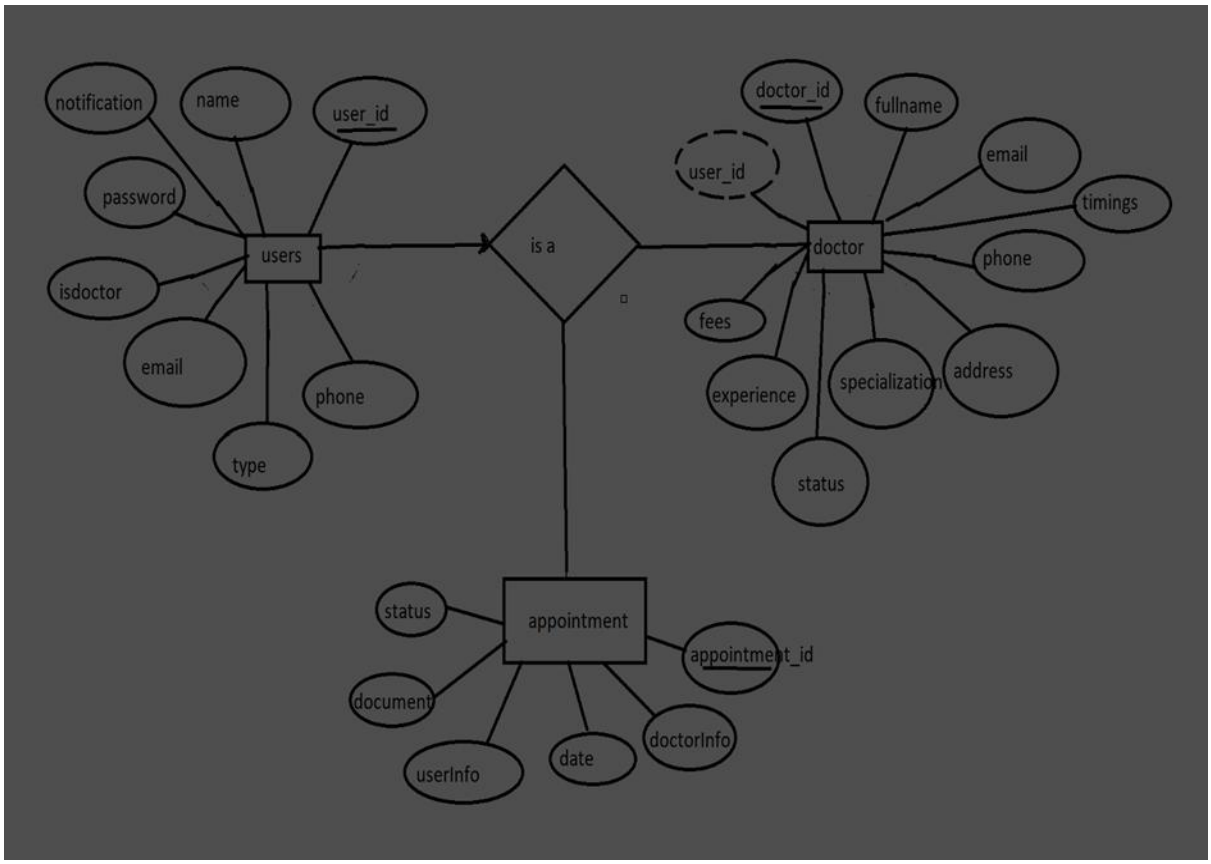
The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user whether it is an admin, doctor, or ordinary user working on it.

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

For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, for booking rooms, adding rooms, etc. It ensures reliable and quick access to the necessary information.

Together, the frontend and backend components, along with Moment, Express.js, and MongoDB, form a comprehensive technical architecture for our Book a Doctor app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive booking of an appointment and many more experiences for all users.

ER DIAGRAM:



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

Users:

1. _id: (MongoDB creates by unique default)
2. name
3. email
4. notification
5. password
6. isdoctor
7. type
8. phone

Doctor:

1. userID: (can be act as foreign key)
2. _id: (MongoDB creates by unique default)
3. fullname
4. email
5. timings
6. phone
7. address
8. specialization
9. status
10. experience
11. fees

Appointment:

1. _id: (MongoDB creates by unique default)
2. doctorInfo
3. date
4. userInfo
5. document
6. status

PRE-REQUISITES

PRE-REQUISITES:

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

npm init

- **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:

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/

- **Moment.js:**

Momentjs is a JavaScript package that makes it simple to parse, validate, manipulate, and display date/time in JavaScript. Moment. js allows you to display dates in a human-readable format based on your location. Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: https://momentjs.com/

- **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

- **Antd:**

Ant Design is a React. js UI library that contains easy-to-use components that are useful for building interactive user interfaces. It is very easy to use as well as integrate. It is one of the smart options to design web applications using react.

Follow the installation guide: https://ant.design/docs/react/introduce

- **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 booking room, status of the booking, and user interfaces for the admin dashboard.

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

Install Dependencies:

• Navigate into the cloned repository directory:

cd book-a-doctor

• Install the required dependencies by running the following commands:

cd frontend

npm install

cd ../backend

npm install

Start the Development Server:

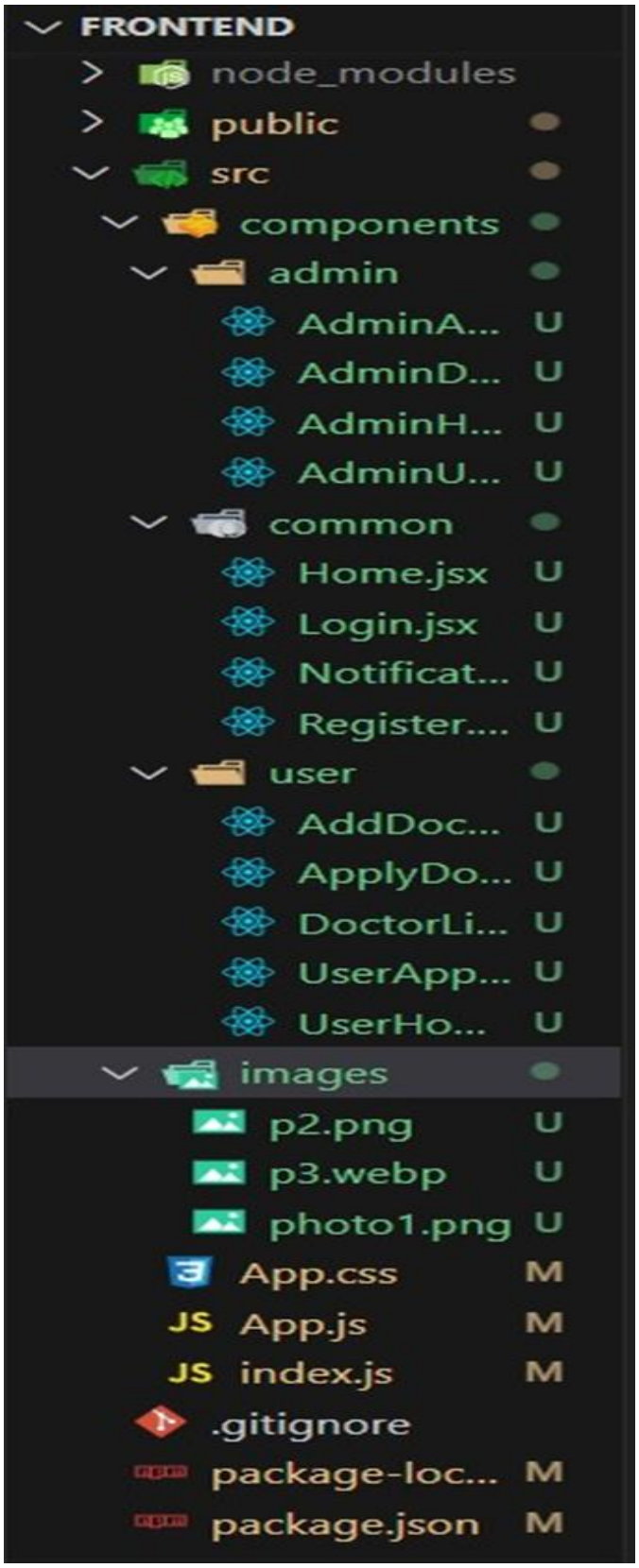
• To start the development server, execute the following command:

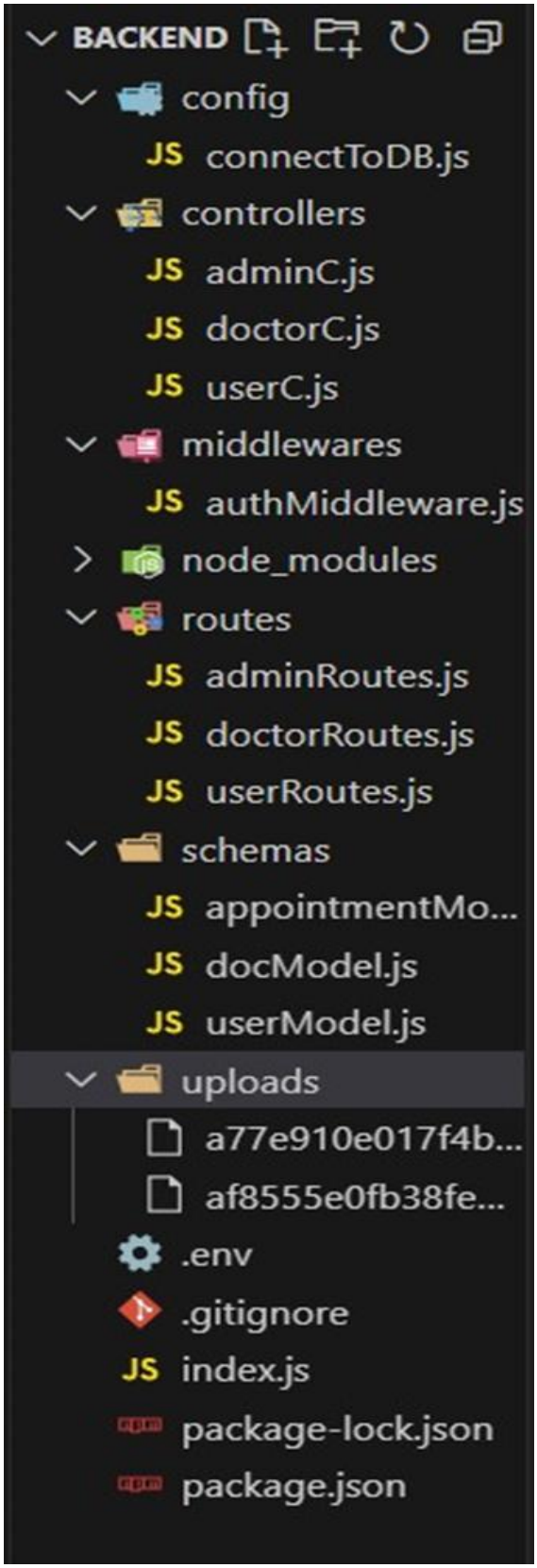
npm start

• The book a doctor app will be accessible at 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.

PROJECT STRUCTURE





The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development

Application Flow: The project has 2 type of user – Customer and Doctor and other will be Admin which takes care to all the user. The roles and responsibilities of these two types of users can be inferred from the API endpoints defined in the code. Here is a summary:

Customer/Ordinary:

1. Create an account and log in to the system using their email and password.

2. They will be shown automatically all the doctors in their dashboard.

3. After clicking on the Book Now, a form will generate in which date of appointment and documents need to send.

4. They can sees the status of their appointment and can get a notification if the appointment is schedule or not.

5. The user can also cancel it`s booking in booking history page and can change the status of booking.

Admin:

1. Manage and monitor the overall operation of the appointment and the type of users and doctors to the application.

2. He monitors the applicant of all doctors and approve them and then doctors are registered in the app.

3. Implement and enforce platform policies, terms of service, and privacy regulations.

Doctor:

1. Gets the approval from the admin for his doctor account.
2. Manages all the appointments that are getting from the users

3. Setup & configuration

4. Let's start with the project development with the help of the given activities

Folder setup

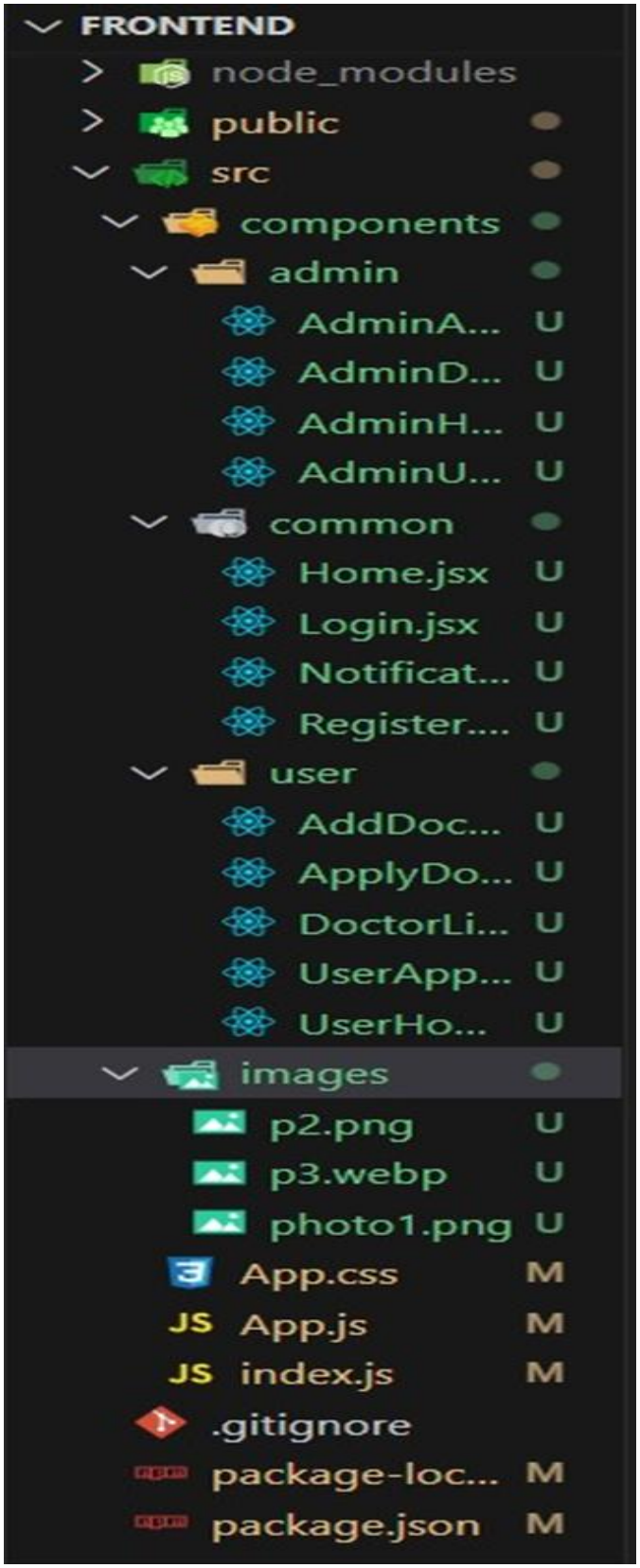
- **Folder setup:**

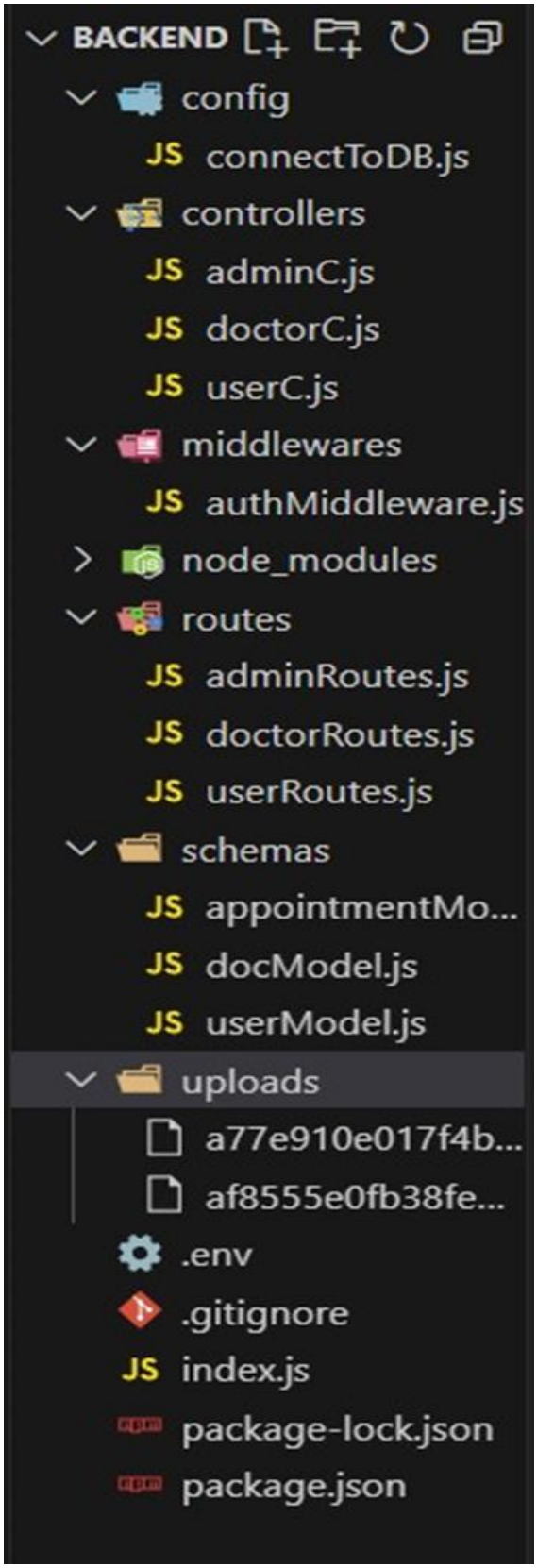
1. Create frontend and
2. Backend folders

. Open the backend folder to install necessary tools

For backend, we use:

- cors
- bcryptjs
- express
- dotenv
- mongoose
- Multer
- Nodemon
- jsonwebtoken


FRONTEND
node_modules
public
src
components
admin
AdminA... U
AdminD... U
AdminH... U
AdminU... U
common
Home.jsx U
Login.jsx U
Notificat... U
Register.... U
user
AddDoc... U
ApplyDo... U
DoctorLi... U
UserApp... U
UserHo... U
images
p2.png U
p3.webp U
photo1.png U
App.css M
App.js M
index.js M
.gitignore
package-loc... M
package.json M



Backend Development

In this milestone explains about Backend Development

Setup express server

- **Setup express server**

1. Create index.js file in the server (backend folder).
2. define port number, mongodb connection string and JWT key in env file to access it.
3. Configure the server by adding cors, body-parser.

- **Add authentication:** for this,

1. You need to make a middleware folder and in that make an authMiddleware.js file for the authentication of the projects and can use in.

Ref: **link**

Database

In This Milestone Explains about Database

Configure MongoDB

- **Configure MongoDB**
-
-

1. Import mongoose.
2. Add database connection from config.js file present in config folder
3. Create a model folder to store all the DB schemas like renter, owner and booking, properties schemas.

ref: link

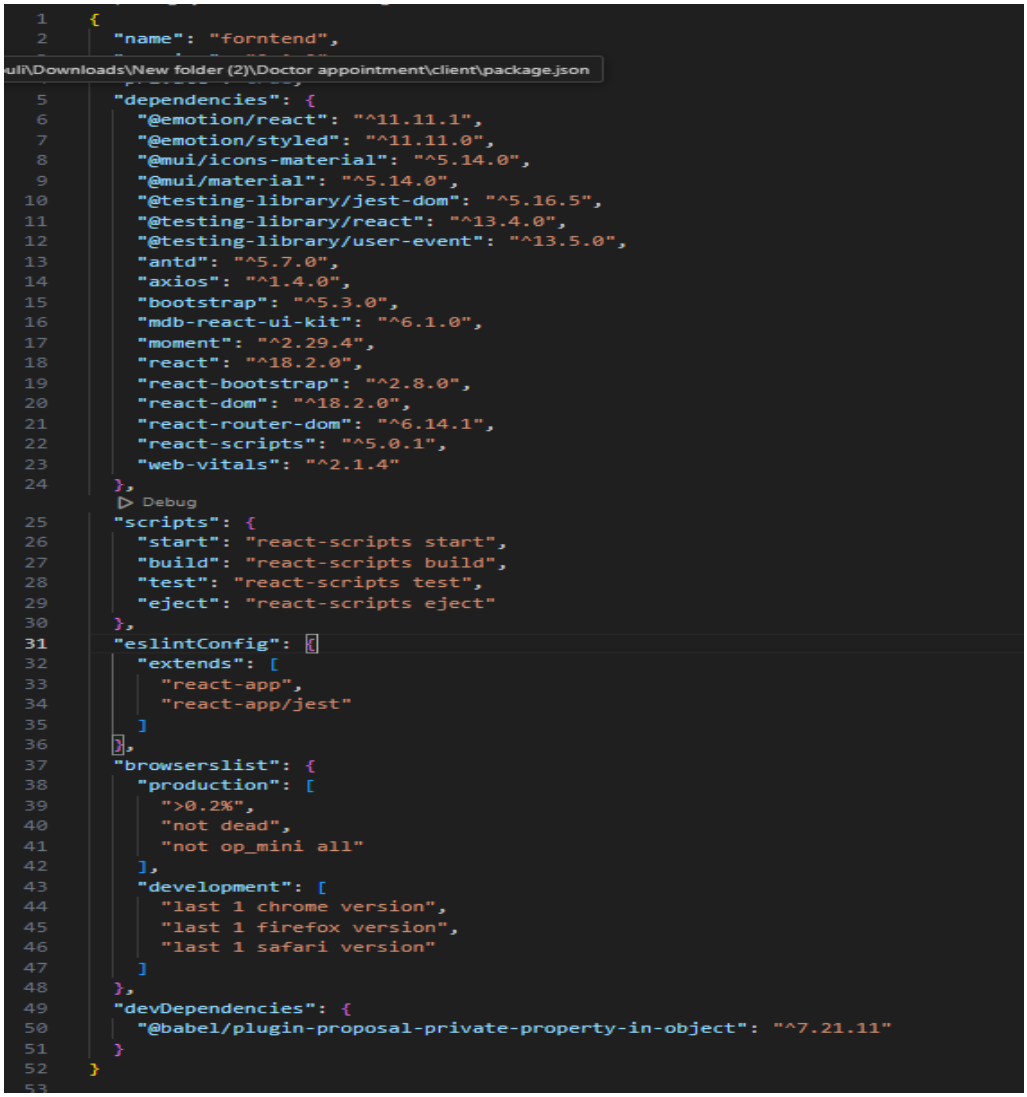
Frontend Development

In this milestone explains about Frontend Development

Installation of required tools

- **Installation of required tools:**
-
- For frontend, we use:

1. React
2. Bootstrap
3. Material UI
4. Axios
5. Antd
6. mdb-react-ui-kit
7. react-bootstrap



ref: [link]

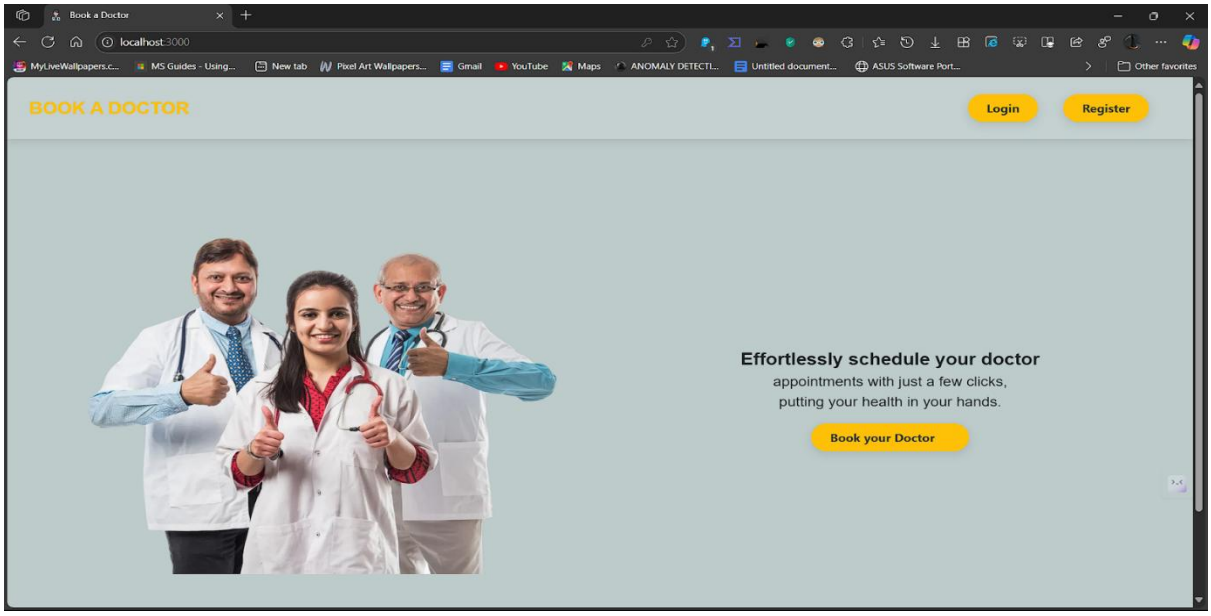
Project Implementation

In this milestone explains about project implementation

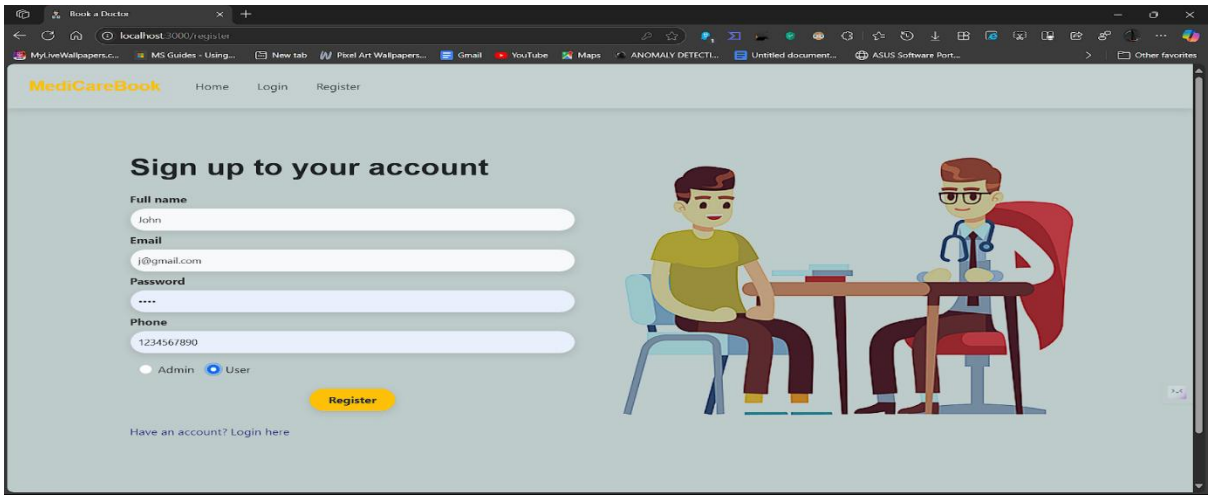
Landing page

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The application's user interface looks a bit like the one provided below.

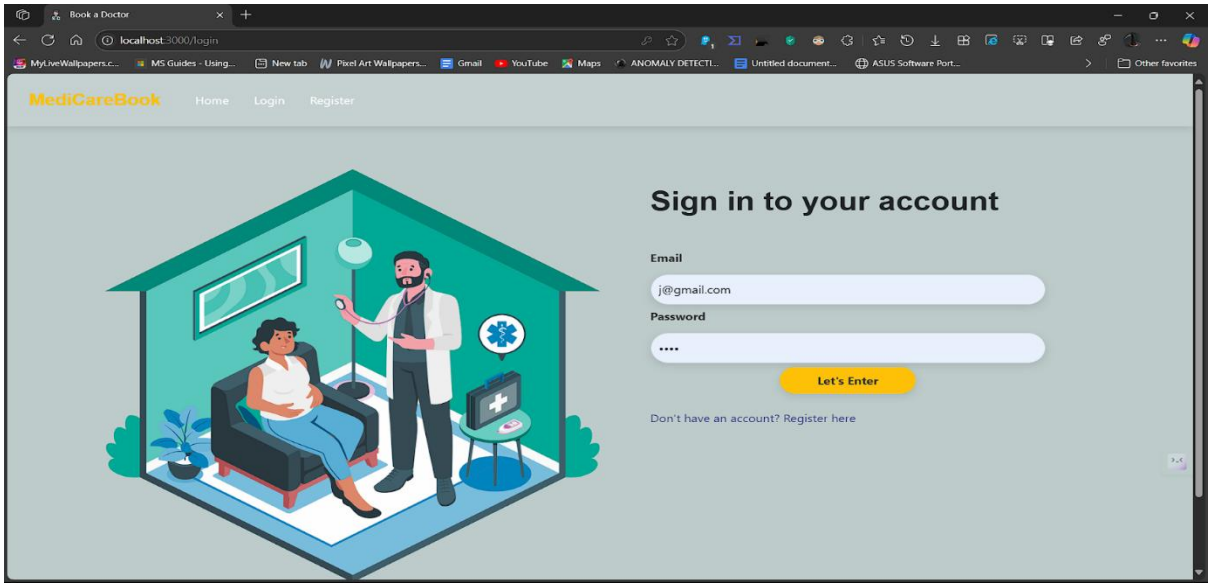
Landing page



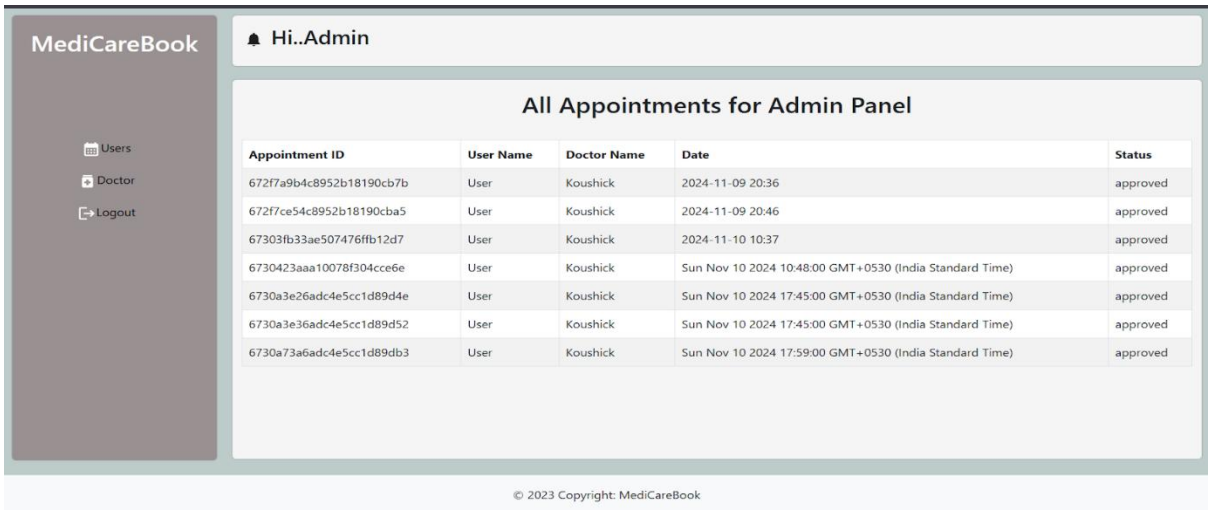
Register page:



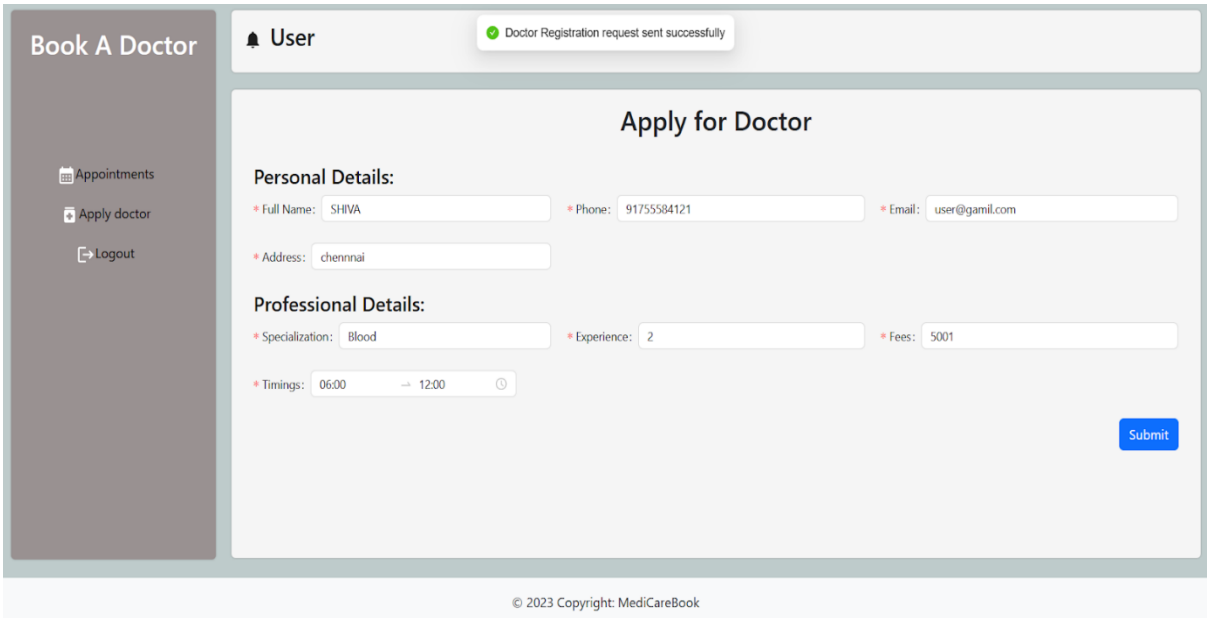
Login page:



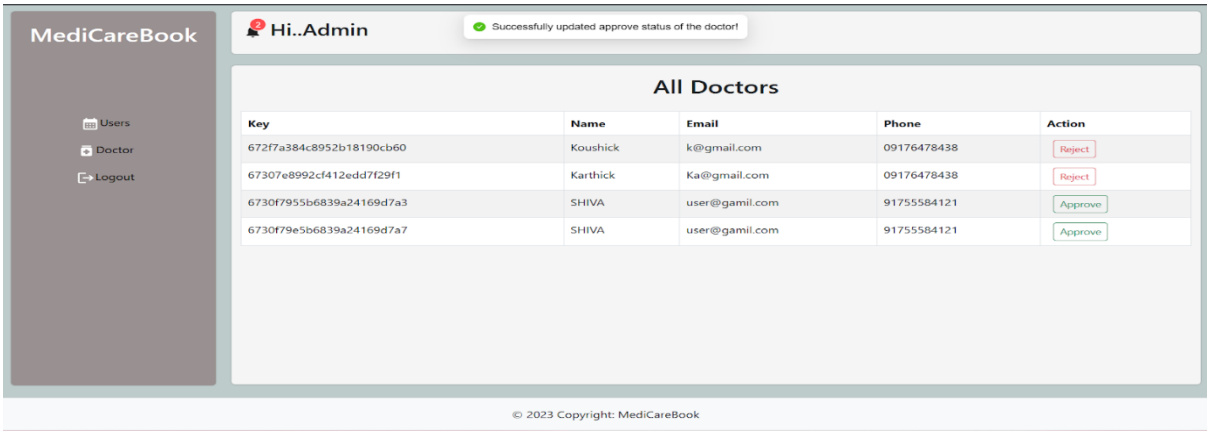
Admin Dashboard:



Doctor dashboard:

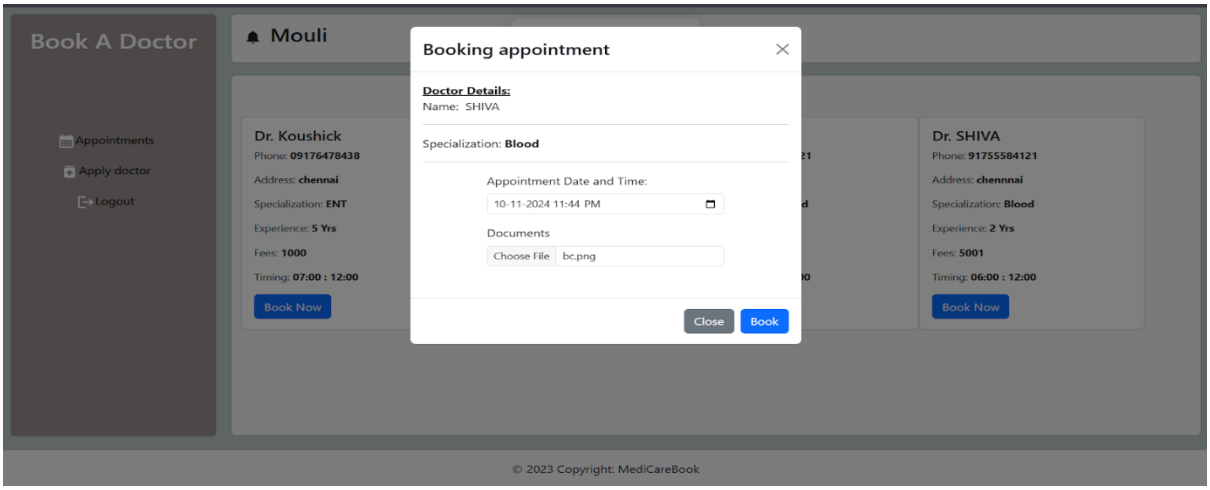


Admin approve doctor:

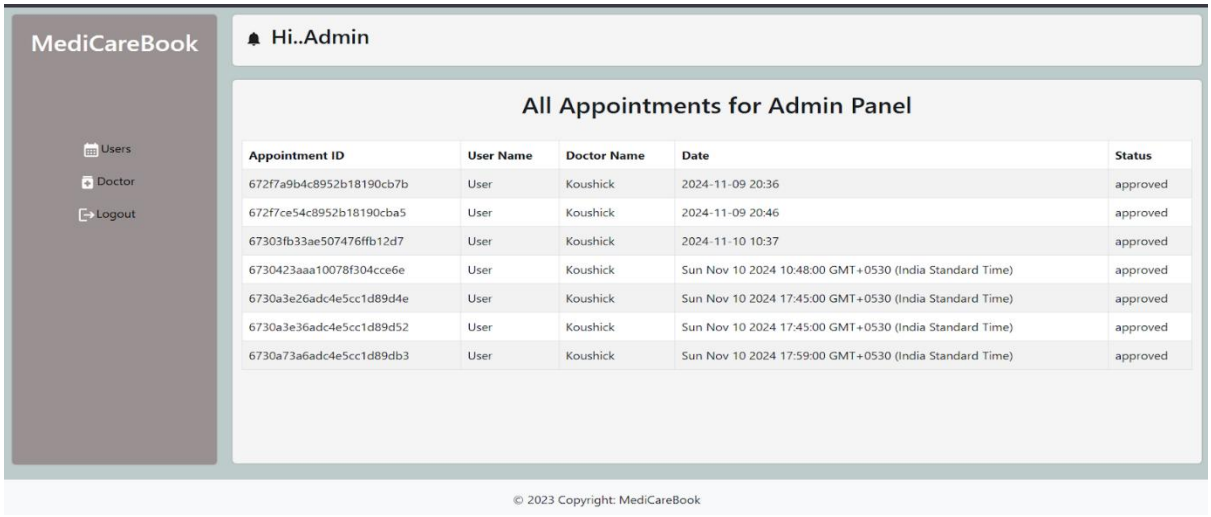


BOOK

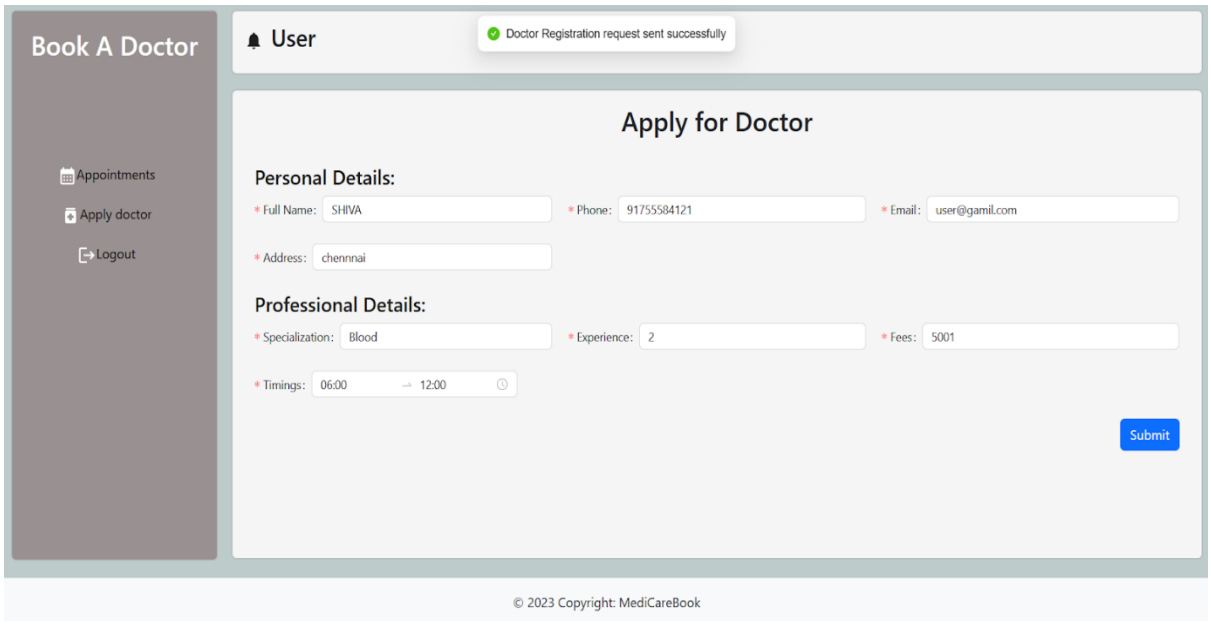
DOCTOR :



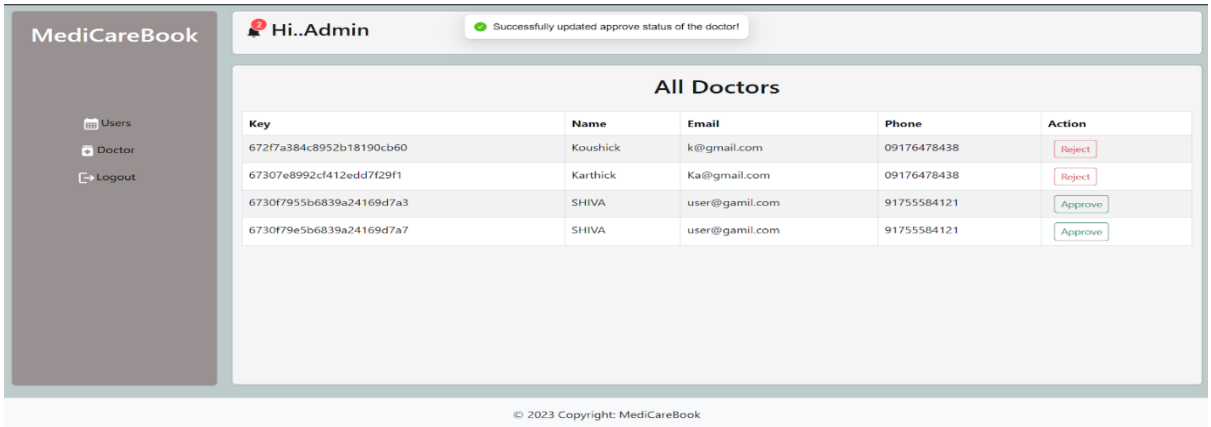
ALL HISTORY :



Doctor dashboard:

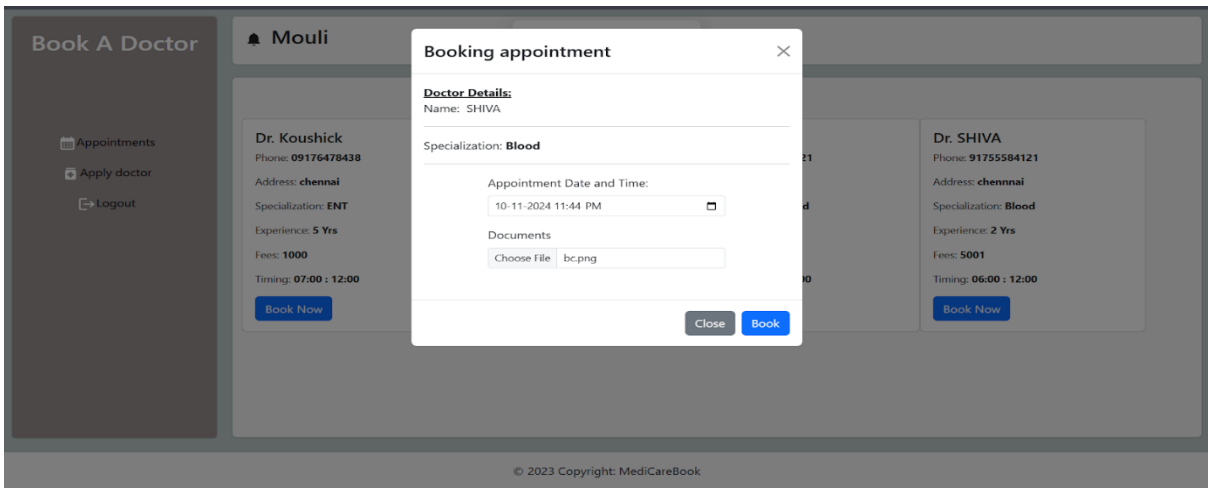


Admin approve doctor:

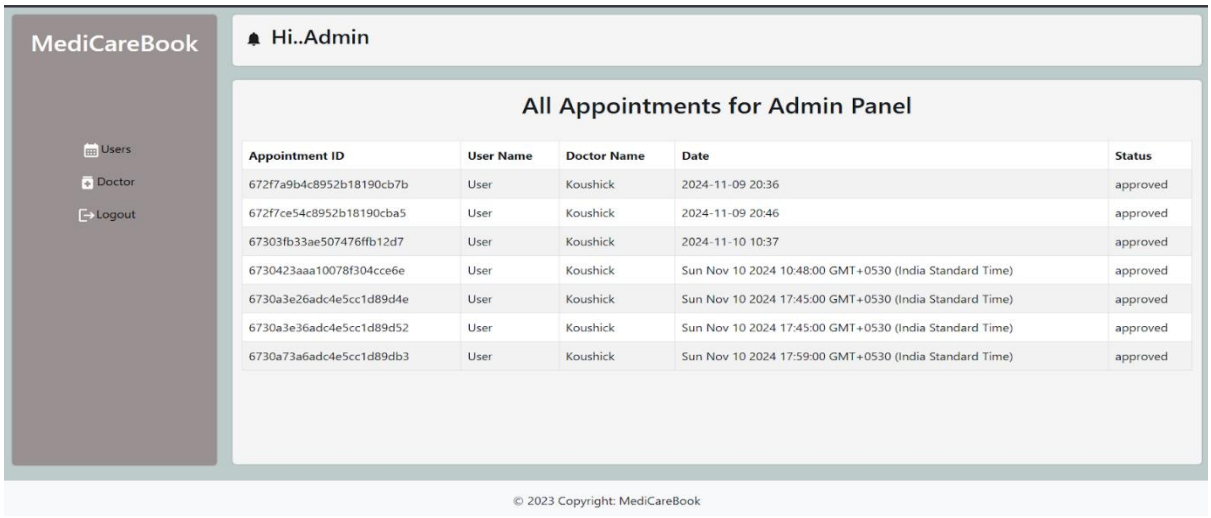


BOOK

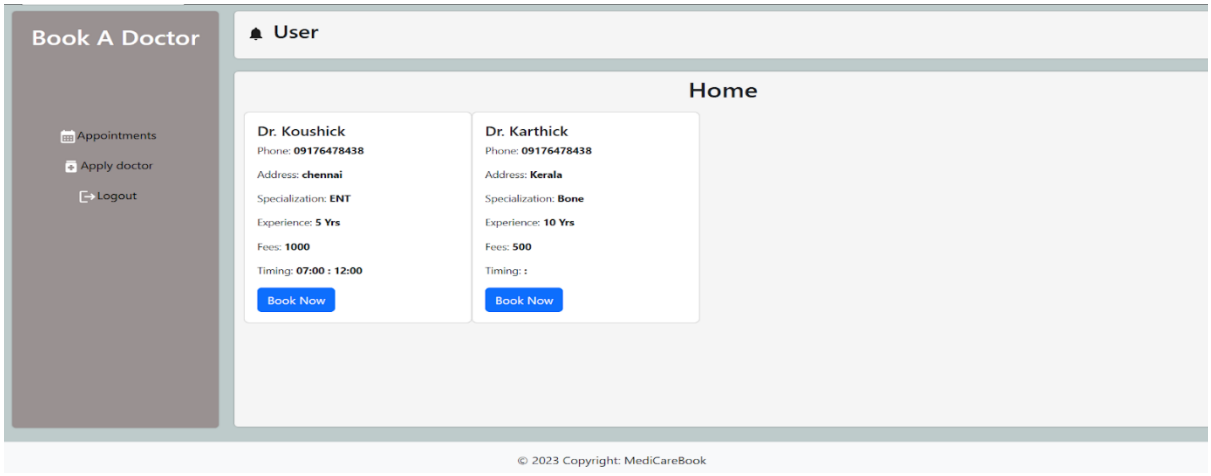
DOCTOR :



ALL HISTORY :



User dashboard:



Note: For code drive, click on link and demo link, click on link

Project FlowDemo Video and reference code link

reference video link:

- https://drive.google.com/drive/folders/1pteT8STdObONWwELNDHRK9biItLuiJ-1?usp=sharing

reference code link:

Doctor Appointment Booking Using MERN Source Code