BOOK A DOCTOR – NM PROJECT

TEAM NO: 16160

TEM MEMBERS:

Mukilarasan V- 582091D35CAAB5681604442FDE8DB22

Sanmitha V S- 5071DB63FC4E7E0B30E4E4B255836F01

Shanjanaa G - 2AA4851B55FCBC48F394F06C38669674

Nijanthanathan - EA49628A69408C4A16B44657849446CE

INTRODUCTION:

The **"Book a Doctor"** website is a comprehensive web-based platform designed to streamline healthcare appointment management. It provides an intuitive solution for patients, doctors, and administrators, ensuring a seamless and efficient experience for everyone involved.

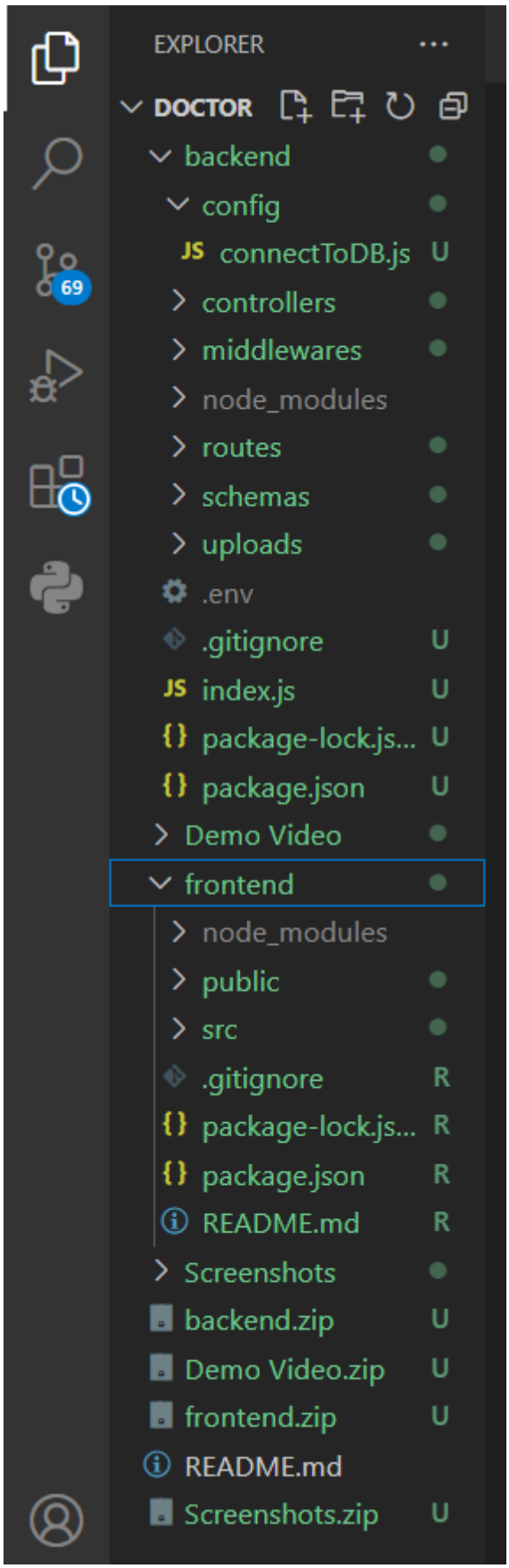
TECHNOLOGIES USED:

- MongoDB - A NoSQL database used to store and manage user profiles, doctor details, appointment data, and other application-related information.
- React JS - A frontend library used to build a dynamic and responsive user interface for seamless interaction with the platform.
- Node JS - It serves as the runtime for executing backend JavaScript code, while npm manages dependencies for the project.
- Express.js - A lightweight web framework for Node.js, used to build the backend API and handle server-side logic.
- HTML - The foundation of the website's structure, used for creating the layout and basic content of the platform.

PLATFORM USED:

VSCode - A powerful code editor used for writing, debugging, and managing the project's code across all technologies, including React, Node.js, and MongoDB.

FOLDER SETUP:



IMPLEMENTAION:

HOME PAGE:

The Home page was coded in React.js and serves as the landing page for the "Book a Doctor" platform. The navigation bar at the top provides links to the Home, Login, and Register pages, ensuring easy navigation across the platform.For routing, React router is used , enabling faster and efficient client side routing.

In the main section of the page, there is a "Book your Doctor" button that redirects to the login page. Additionally, an "About Us" section explains the platform's features in detail, highlighting its benefits like real-time appointment scheduling, detailed doctor profiles, secure online payments, and storage of medical history.

LOGIN PAGE:

The Login page is developed to provide secure access to the platform for users and administrators. The page is built using React.js and integrates various libraries such as React-Bootstrap and MDB-React-UI-Kit . User’s are asked to enter their email and password, which are validated by the backend through an API call.

Upon successful login, the system stores an authentication token and user data in localStorage to maintain the user's session. Based on the user's role (admin or user), the page redirects them to the appropriate dashboard, ensuring personalized access.

SIGN UP PAGE:

The Register page is developed to allow new users to create an account on the platform. The registration form captures essential details, including the user's full name, email, password, phone number, and account type (admin or user).

Upon submission, the form sends the user data to the backend API using Axios for account creation. If the registration is successful, the user is redirected to the Login page, and success or error messages are displayed using Ant Design's message component for real-time feedback. The "Register" button submits the form, triggering the handleSubmit function, which handles API integration and error handling. Additionally, a link at the bottom allows users with an existing account to navigate to the Login page.

ADMIN HOME PAGE:

The Admin Home page is designed as a central hub where an admin can navigate between different sections of the application, including users, doctors, notifications, and appointments.

Features:

Sidebar Navigation - The sidebar contains menu items such as "Users", "Doctor", and "Logout", represented by icons and labels. The user can click on each item to view different sections.

Displaying Admin Data - The getUserData function fetches data about the user from the backend via a POST request to the getuserdata API. The user data is stored in the userdata state.

Notification Management - The NotificationsIcon in the header is a clickable badge that shows the number of unread notifications. The count is dynamically updated based on the userdata.notification array.

The content area dynamically renders components based on the active menu item. For example:

- When "Users" is selected, the AdminUsers component is shown.
- When "Doctor" is selected, the AdminDoctors component is shown.
- If no specific section is selected (other than notifications), the default section AdminAppointments is displayed.

ADMIN USERS PAGE:

The Admin Users page is designed to display a list of all users in the system, including their basic information such as name, email, phone, admin status, and doctor status. It fetches the user data from the backend API and presents it in a table format using React Bootstrap.

Once the data is fetched, it is stored in the users state using setUsers.The user information is rendered in a Bootstrap table with columns: Name, Email, Phone, isAdmin, and isDoctor. If there are no users fetched (i.e., users array is empty), an Alert is shown, indicating that there are "No Users to show".

ADMIN APPOINTS PAGE:

The Admin Appointments page is designed to allow the admin to view all appointments in the system. It fetches appointment data from the backend and displays it in a table format using React Bootstrap. The data is fetched and once the

data is fetched from the server, the appointments are stored in the allAppointments state using setAllAppointments.

ADMIN DOCTORS PAGE:

The Admin Doctors page is designed to allow admins to manage doctor profiles, specifically approving or rejecting doctors based on their application status. Upon accessing this page, the admin can view a list of all doctors and their details such as name, email, phone number, and their current status (pending, approved, or rejected). This list is populated by fetching the doctor data from the backend using an API request.

The getDoctors function is responsible for making a GET request to the backend at the /api/admin/getalldoctors endpoint. This request includes an Authorization token from localStorage to ensure that the request is authenticated and can be processed by the server. Once the data is fetched, it is stored in the doctors state, which triggers a re-render to display the doctors' information in a Bootstrap table.

If the doctor's status is still pending, the admin is presented with two action buttons: Approve and Reject. Clicking on the Approve button sends a POST request to the /api/admin/getapprove endpoint, changing the doctor’s status to approved. Similarly, clicking the Reject button sends a POST request to the /api/admin/getreject endpoint, changing the status to rejected.

USER APPOINTMENT PAGE:

The UserAppointments component allows users and doctors to view and manage appointments in a medical system. This page dynamically adapts based on whether the logged-in user is a doctor or a regular user, displaying different information and actions for each type.

The component begins by defining several state variables: userid to store the user's ID, type to track whether the user is a doctor (boolean), userAppointments to store the appointments of the user, and doctorAppointments to store the appointments of the doctor. The state is updated based on the user's role.

The getUser function retrieves the user's information from localStorage, such as their user ID and whether they are a doctor. This information is used to set the userid and type states.

[Depending on the user's type, the component fetches either user appointments or doctor appointments.]

For doctors, each appointment in the table displays the user’s name, the appointment date, the user's phone number, a downloadable document (if available), the current appointment status, and an action button. If the appointment status is not "approved", the doctor can approve the appointment by clicking the Approve button, which sends a POST request to the backend to update the status of the appointment.

For users, their appointments are displayed with the doctor’s name, the appointment date, and the status. Users can only view the appointments, and no actions are provided.

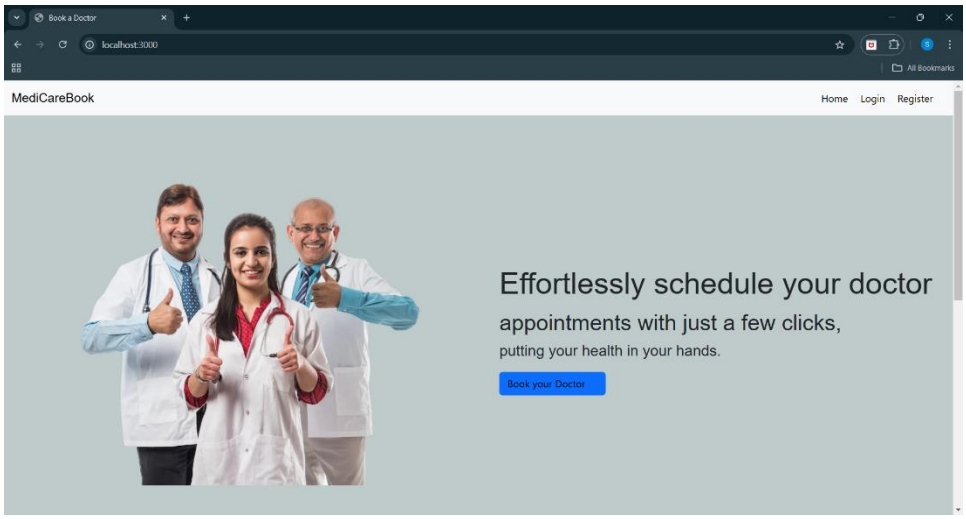
APPLYING FOR DOCTOR PAGE:

The ApplyDoctor component is a form that allows users to apply as doctors by filling in their personal and professional details. It uses React's useState hook to manage the state of the form, specifically a doctor object which includes fields like fullName, email, phone, address, specialization, experience, fees, and timings.

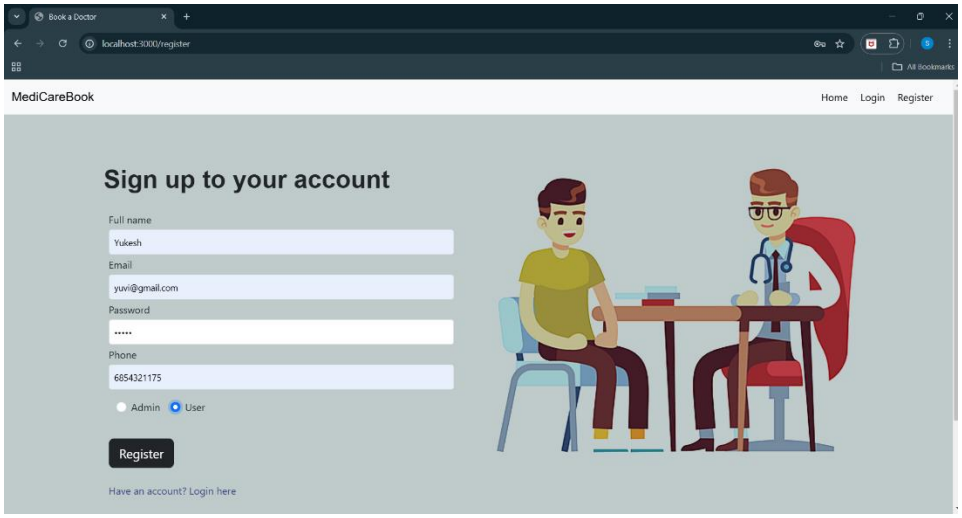
When the user submits the form by clicking the submit button, the handleSubmit function is triggered. This function sends a POST request to the backend API with the form data and the userId (which is passed as a prop). The request includes an authorization token stored in localStorage for user authentication.

RESULTS:

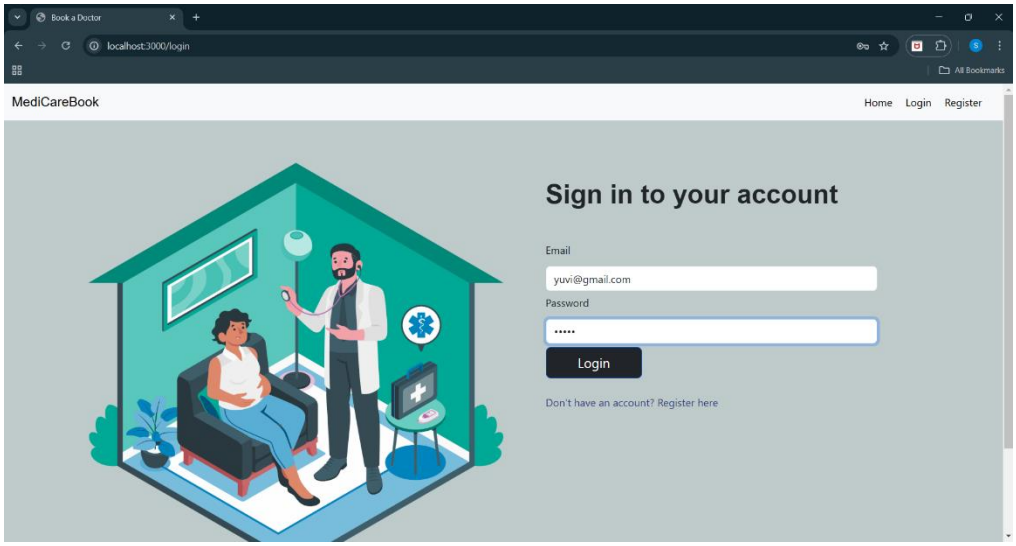
Index page,



New user creation,

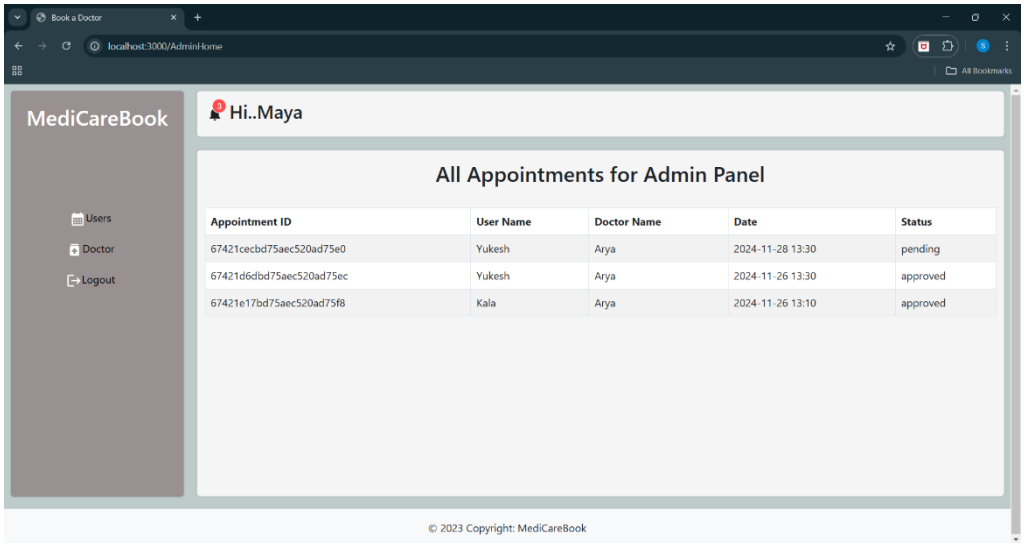


Signing in,

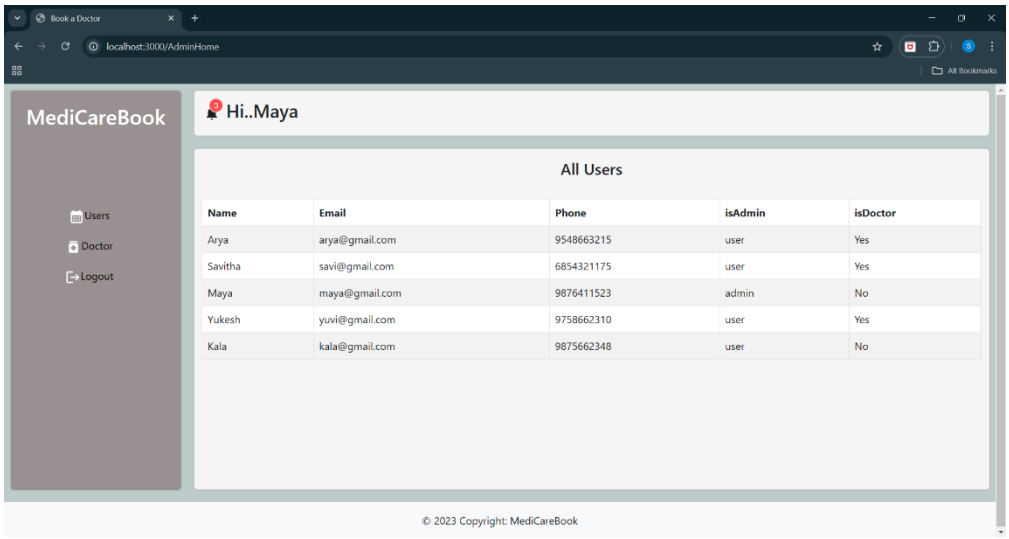


ADMIN PAGES:

Doctor’s approval page,

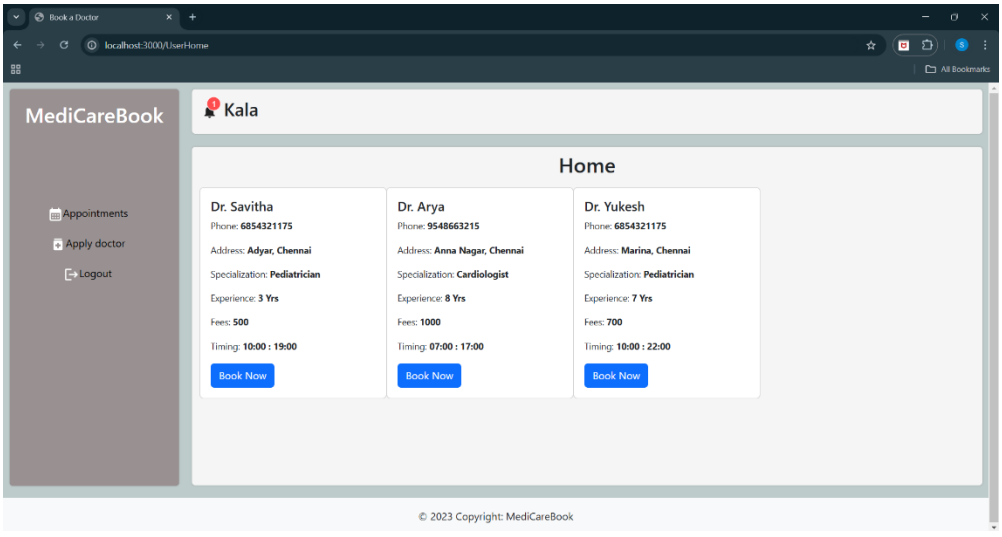


User’s information page,

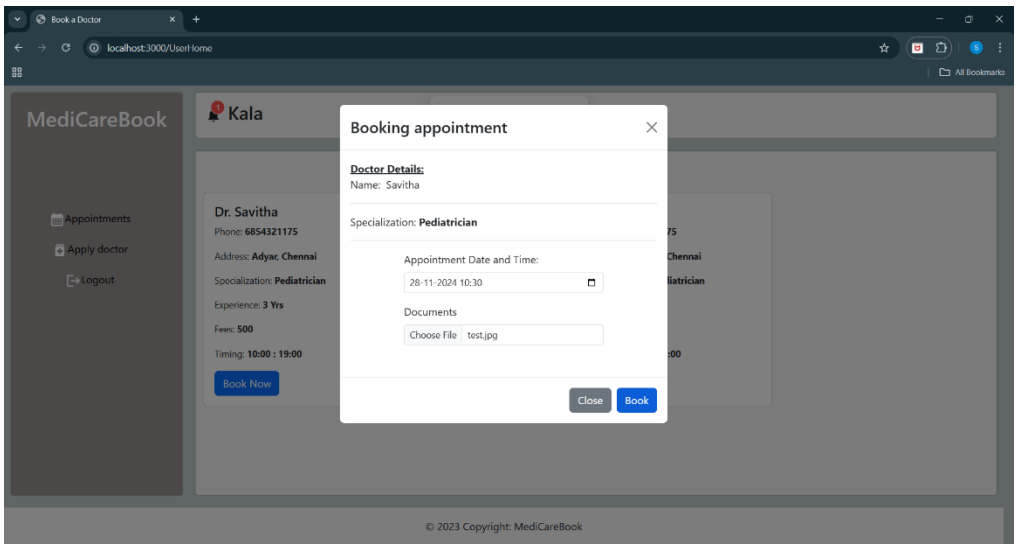


USER PAGES:

Available Doctors dispay,

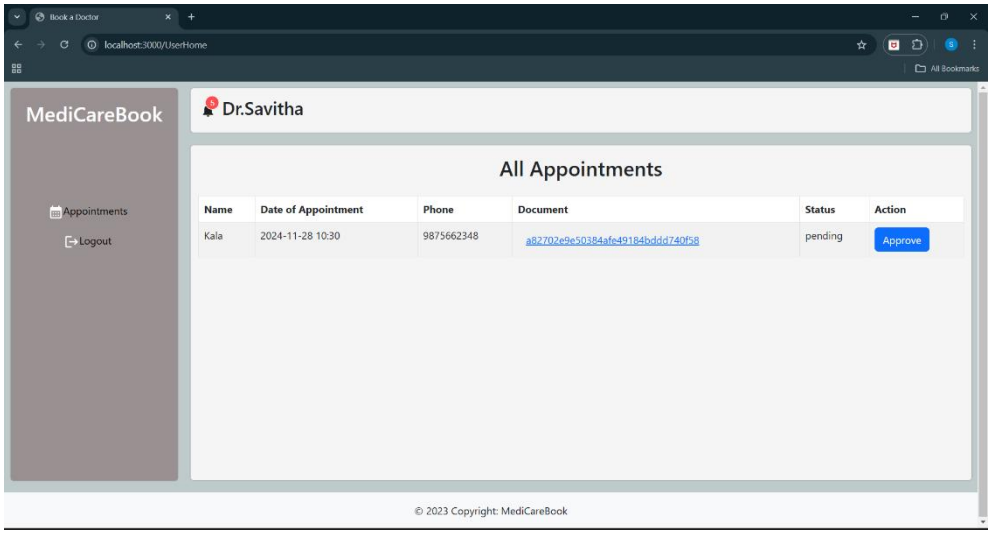


Booking a new appointment,



DOCTOR PAGES:

Appointment display page,



GITHUB LINK: https://github.com/Sanmitha-Sadhishkumar/NM_Project_Book_a_Doctor

DEMO VIDEO LINK: https://drive.google.com/drive/folders/1vE-7yhzPNKjdrXG0y9zAkYl-tdhyCEn1?usp=sharing