

Full Stack Development with MERN

Book a Doctor using MERN Stack

Team Members:

1. Padmaja H
2. Sandhiya S
3. Priyadarshni V
4. Latha Sri SA

1. Project Overview

Purpose:

Booking a doctor's appointment has never been easy. With our convenient online platform, one can quickly and effortlessly schedule your appointments from the comfort of your own home. Our user-friendly interface allows one 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, one can do away with 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 one prefer early morning, evening, or weekend appointments, we have options to accommodate your needs.

Our innovative online platform empowers you to take control of your health and schedule appointments with top-notch healthcare providers, all from the comfort of your home.

Features:

User Features

- **User Registration:** Users can register on the platform by providing basic information like name, email, and password.
- **Doctor Search and Filtering:** Users can search for doctors based on specialization, availability, and other criteria.
- **Appointment Booking:** Users can book appointments with their preferred doctors, selecting a suitable date and time.
- **Appointment Management:** Users can view, edit, and cancel their booked appointments.
- **Notification System:** Users can receive notifications for appointment confirmations, reminders, and updates on their Notification page of their dashboard.
- **Secure Login and Profile Management:** Users can securely log in to their accounts and manage their personal information.

Doctor Features

- **Doctor Registration:** Doctors can register on the platform and provide their credentials and specialization.
- **Appointment Management:** Doctors can view and manage their appointment schedule, including confirming, rescheduling, and cancelling appointments.

Admin Features

- **User and Doctor Management:** Admins can manage user and doctor registrations, approvals, and access controls.
- **Platform Monitoring:** Admins can monitor the platform's performance, user activity, and system health.
- **Data Analytics:** Admins can analyse user behaviour, appointment trends, and other relevant data to improve the platform's functionality.
- **Security and Privacy:** Admins are responsible for ensuring the security and privacy of user data.

General Features

- **Secure Authentication and Authorization:** Implement robust authentication and authorization mechanisms to protect user data and prevent unauthorized access.
- **Push Notifications:** Send timely notifications to users for appointment reminders, cancellations, and other important updates.
- **Data Privacy and Security:** Adhere to data privacy regulations and implement strong security measures to protect user information such as storing hashed passwords in the database.
- **User Experience:** Provide a user-friendly and intuitive interface.

2. Architecture :

Frontend Architecture using React :

The frontend of the Book a Doctor app leverages React to create a dynamic and responsive user interface, delivering an intuitive experience for users, including doctors, administrators, and patients. React's component-based structure ensures efficient state management, allowing the app to maintain a seamless flow between user actions and data updates. Bootstrap and Material UI libraries are integrated to deliver a cohesive, visually appealing, and user-friendly interface across devices. Axios is employed to facilitate secure communication with the backend, enabling RESTful API calls for data retrieval and updates in real-time. Key features such as patient registration, appointment scheduling, and notifications are organized into reusable React components, which streamlines development and enhances maintainability. With Moment.js, date and time formatting in the frontend is simplified, ensuring clear presentation of appointments and availability schedules. This frontend architecture, centered around React's capabilities, provides a scalable and modular framework for enhanced user engagement and efficient data exchange with the server.

Backend Architecture using Node.js and Express.js

The backend of the Book a Doctor app is built on Node.js and the Express.js framework, providing a robust and scalable server environment for handling API requests and managing

application logic. Express.js, known for its efficiency, is used to create RESTful APIs that seamlessly connect the frontend with data stored in MongoDB, a document-oriented NoSQL database. This backend setup ensures efficient storage and retrieval of user data, including profiles, appointment bookings, and other essential information. Express middleware is configured to handle routing, validation, and error handling, ensuring secure and reliable communication between the client and server. MongoDB's flexible schema design allows for quick modifications and scalability, making it ideal for handling diverse data types involved in appointment management. Additionally, the backend architecture is structured to accommodate high traffic loads, ensuring responsiveness and performance even as user demands grow. This architecture effectively handles real-time interactions, making it ideal for applications focused on real-time healthcare service delivery.

Database :

A MongoDB database called the mern_project was created for this project. It has the following collections

- i. appointments
- ii. doctors
- iii. notifications
- iv. users

Schema of these collections

Collection	Attribute	Type	Ref	Required	Validation
Doctor	userId	ObjectId	User	Yes	
	specialization	String		Yes	
	experience	Number		Yes	
	fees	Number		Yes	
	isDoctor	Boolean		Yes (default: false)	
	createdAt	Date		Yes (default: current timestamp)	
	updatedAt	Date		Yes (default: current timestamp)	
Appointment	userId	ObjectId	User	Yes	
	doctorId	ObjectId	User	Yes	
	date	String		Yes	
	time	String		Yes	
	status	String		Yes (default: Pending)	
	createdAt	Date		Yes (default: current timestamp)	
	updatedAt	Date		Yes (default: current timestamp)	
Notification	userId	ObjectId	User	Yes	
	isRead	Boolean		Yes (default: false)	
	content	String		Yes (default: "")	
	createdAt	Date		Yes (default: current timestamp)	
	updatedAt	Date		Yes (default: current timestamp)	
User	firstname	String		Yes	minLength: 3 minLength: 3 unique: true minLength: 5
	lastname	String		Yes	
	email	String		Yes	
	password	String		Yes	
	isAdmin	Boolean		Yes (default: false)	
	isDoctor	Boolean		Yes (default: false)	
	age	Number		No (default: "")	
	gender	String		No (default: "neither")	
	mobile	Number		No (default: "")	
	address	String		No (default: "")	
	status	String		No (default: "pending")	
	pic	String		No (default: image URL)	
	createdAt	Date		Yes (default: current timestamp)	
	updatedAt	Date		Yes (default: current timestamp)	

3. Setup Instructions

Prerequisites:

Backend Dependencies

1. Node.js and npm
2. Express.js: A popular framework for building web applications and APIs.
3. Mongoose: A MongoDB object modeling tool for Node.js.
4. Nodemon: A tool that automatically restarts the Node.js server when file changes are detected.
5. jsonwebtoken: For creating and verifying JSON Web Tokens for authentication.
6. bcrypt: For password hashing.
7. bcryptjs: For password hashing (likely a duplicate or alternative to bcrypt).
8. colors: For colored console output, often used for debugging or logging.
9. cors: For enabling Cross-Origin Resource Sharing (CORS), allowing requests from different origins.
10. dotenv: For loading environment variables from a .env file.
11. moment: A JavaScript library for parsing, validating, manipulating, and formatting dates and times.
12. morgan: HTTP request logger middleware for Node.js.

Frontend Dependencies

Core React Dependencies:

- react: The core React library for building user interfaces.
- react-dom: The DOM renderer for React.
- react-scripts: A configuration and script setup for creating React applications.

State Management:

- react-redux: Connects React components to a Redux store for centralized state management.
- @reduxjs/toolkit: A toolkit for simplifying Redux development.

Routing:

- react-router-dom: For routing and navigation within the React application.
- react-router-hash-link: For smooth scrolling to specific sections of a page using hash links.
- UI Components and Utilities:
- react-icons: For using various icons in the application.
- react-spinners: For displaying loading spinners.
- react-time-picker: For time picker components.
- react-hot-toast: For displaying toast notifications.
- react-countup: For creating animated number counters.
- react-bootstrap: A React implementation of Bootstrap components (optional, not listed in the provided package.json).

Testing:

- @testing-library/react: A testing library for React components.
- @testing-library/jest-dom: Custom matchers for DOM testing.
- @testing-library/user-event: User event simulation for testing.

HTTP Requests:

- axios: A promise-based HTTP client for making requests to APIs.

Other:

- jwt-decode: For decoding JWT tokens.
- web-vitals: For measuring web performance metrics.

Installation:

Step 1: Use Git to clone the project repository to your local machine or Clone the repository by using clone from version control systems option from IDE .

Step 2 : Use npm or yarn to install the required dependencies

Step 3 : Add .env file in root directory for the backend which contains

```
PORT=5000
```

```
MONGO_URI=YOUR_OWN_MONGODB_URL
```

```
JWT_SECRET=YOUR_JWT_SECRET
```

Step 4: Add .env file in client directory for the frontend which contains

REACT_APP_SERVER_DOMAIN=http://127.0.0.1:5000/api

REACT_APP_CLOUDINARY_BASE_URL=https://api.cloudinary.com/v1_1/{CLOUD_NAME}/image/upload

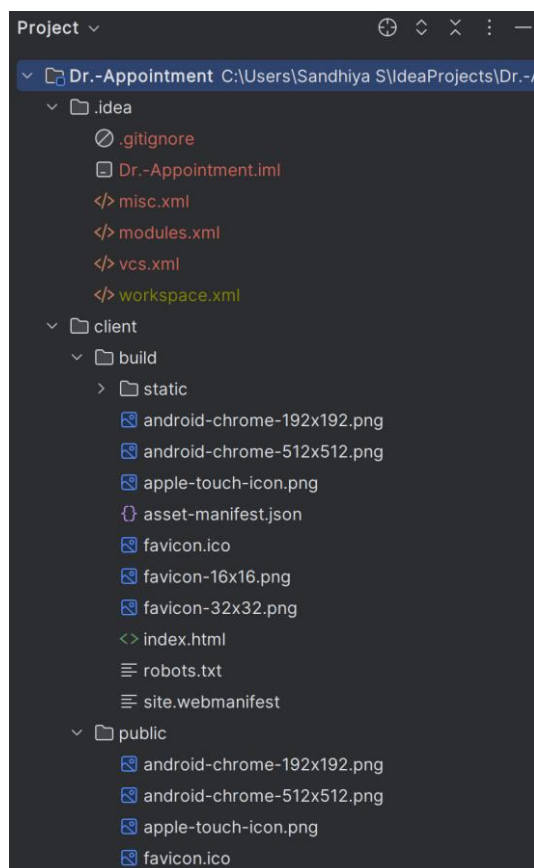
REACT_APP_CLOUDINARY_CLOUD_NAME=YOUR_CLOUDINARY_CLOUD_NAME

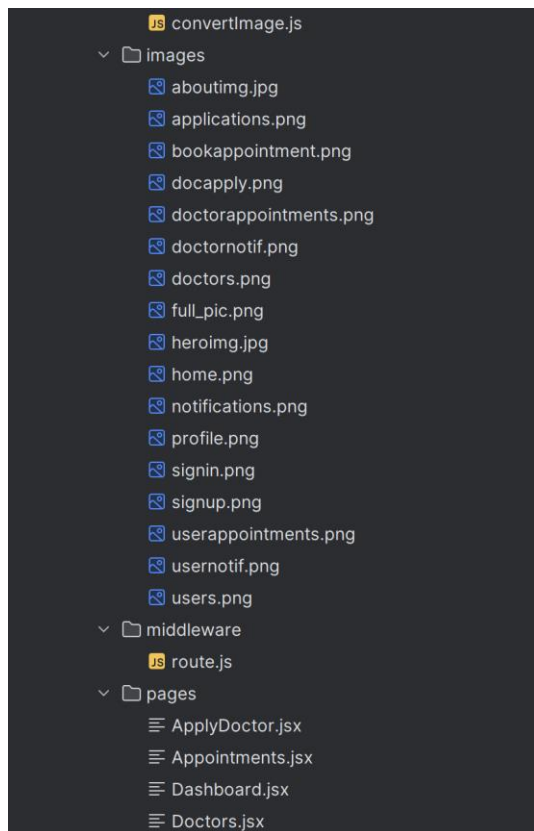
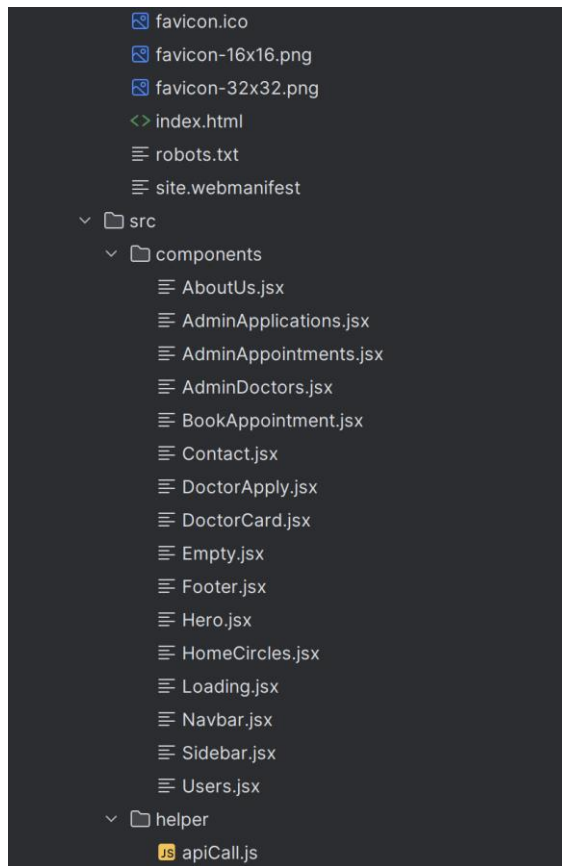
REACT_APP_CLOUDINARY_PRESET=YOUR_CLOUDINARY_PRESET

Replace the **{CLOUD_NAME}** with your own cloudinary cloud name

Step 5: Install MongoDB , NodeJS for your machine and create the database and create required collections In MongoDB

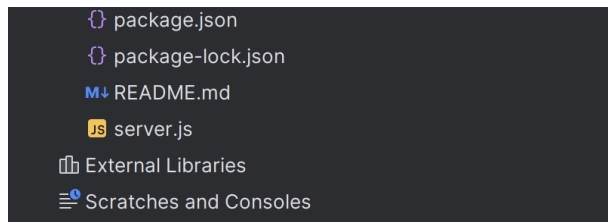
4. Folder Structure





- Error.jsx
- Home.jsx
- Login.jsx
- Notifications.jsx
- Profile.jsx
- Register.jsx
- redux
 - reducers
 - store.js
- styles
 - app.css
 - bookappointment.css
 - contact.css
 - doctorapply.css
 - doctorcard.css
 - doctors.css
 - error.css
 - footer.css
 - hero.css
 - homecircles.css
 - loading.css
 - navbar.css
 - notification.css
 - profile.css
 - register.css
 - sidebar.css

- user.css
- App.js
- index.js
- .gitignore
- package.json
- package-lock.json
- controllers
 - appointmentController.js
 - doctorController.js
 - notificationController.js
 - userController.js
- db
 - conn.js
- middleware
 - auth.js
- models
 - appointmentModel.js
 - doctorModel.js
 - notificationModel.js
 - userModel.js
- routes
 - appointRoutes.js
 - doctorRoutes.js
 - notificationRouter.js
 - userRoutes.js
- .gitignore



5. Running the Application

The `npm run build` command will create an optimized build of the React app in the build folder. This build can be deployed to a production server.

To run the frontend: `npm start` in the client directory.

To run the backend: `npm start` in the server directory.

6. API Documentation:

Backend API Endpoints for Book a Doctor App

- `POST /api/auth/register` - Registers a new user (patient, doctor, or admin) by saving their details in the database.
- `POST /api/auth/login` - Authenticates user credentials, generates a JWT, and returns it for secure session management.
- `GET /api/doctors` - Retrieves a list of all doctors, including details such as specialization, ratings, and availability.
- `GET /api/doctors/`
 - Fetches detailed information about a specific doctor based on their unique ID.

- POST /api/appointments - Creates a new appointment with selected doctor and time, storing it in the database.
- GET /api/appointments/user/
 - Returns all appointments for a specific user, allowing patients to view their bookings.
- PUT /api/appointments/
- /cancel - Allows patients or admins to cancel an appointment based on the appointment ID.
- GET /api/admin/users - Retrieves a list of all users (patients, doctors) for administrative management purposes.
- PUT /api/doctors/
- /availability - Updates a doctor's availability schedule, allowing them to modify available slots.
- DELETE /api/admin/user/
 - Enables admins to delete a user account from the system based on the user ID.

7. Authentication

(i) Authentication and Authorization in the Project:

Authentication and authorization are critical to the security of the Book a Doctor app, ensuring that only authenticated users can access sensitive features based on their roles (patient, doctor, or admin). Authentication is implemented using JWT (JSON Web Tokens), which provides a stateless and scalable

approach to verify user identity. Upon successful login, a JWT token is generated and securely transmitted to the client. The client stores this token, usually in HTTP-only cookies, which minimizes vulnerability to cross-site scripting (XSS) attacks. Authorization, on the other hand, is handled by defining user roles and permissions. When a user requests access to a specific endpoint, the backend verifies the token, checks user roles, and grants or restricts access based on predefined role permissions. This approach to authentication and authorization guarantees a secure, role-based access control framework, essential for managing confidential user information.

(ii) Token and Session Management:

Token management is implemented through JSON Web Tokens (JWT), which allows the Book a Doctor app to handle authentication without server-side sessions, contributing to a stateless and scalable architecture. JWT tokens contain encrypted user information, including role-based identifiers, and are generated upon user login. They are sent to the client, where they are securely stored (often as HTTP-only cookies) to prevent unauthorized access. For each protected route request, the backend verifies the token and extracts user information, ensuring a secure and role-based access system. Unlike traditional session-based authentication, which stores session data on the server, this token-based method reduces server memory load and scales efficiently as the user base grows. Additionally, the token's expiry and refresh capabilities

ensure enhanced security and performance in managing user sessions across all client interactions.

8. User Interface



Fig 1: Home page of the Health Booker

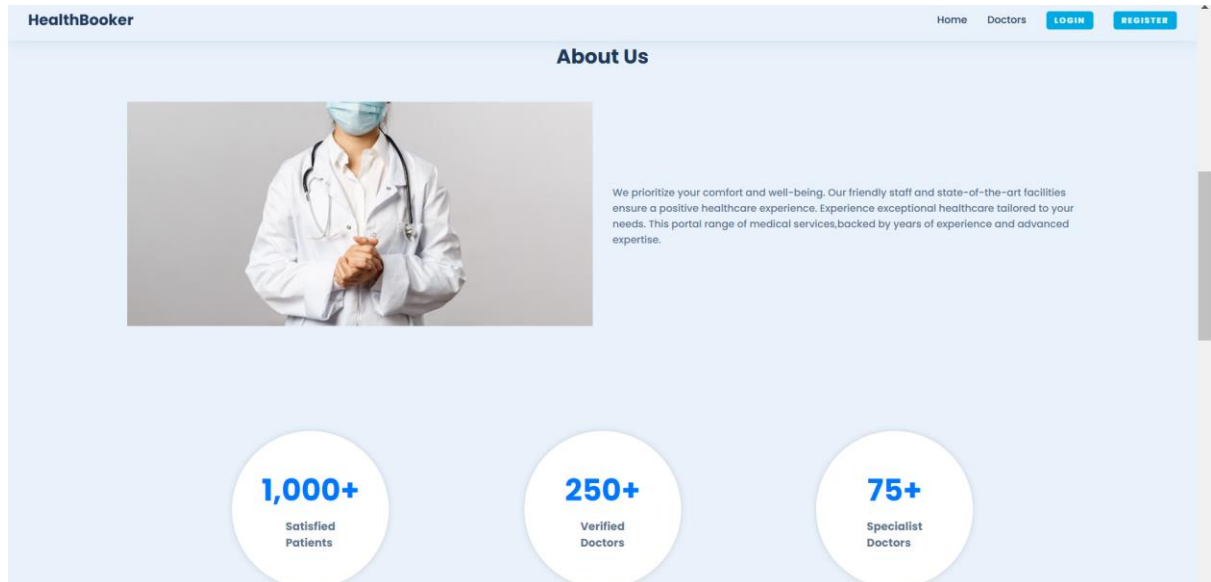
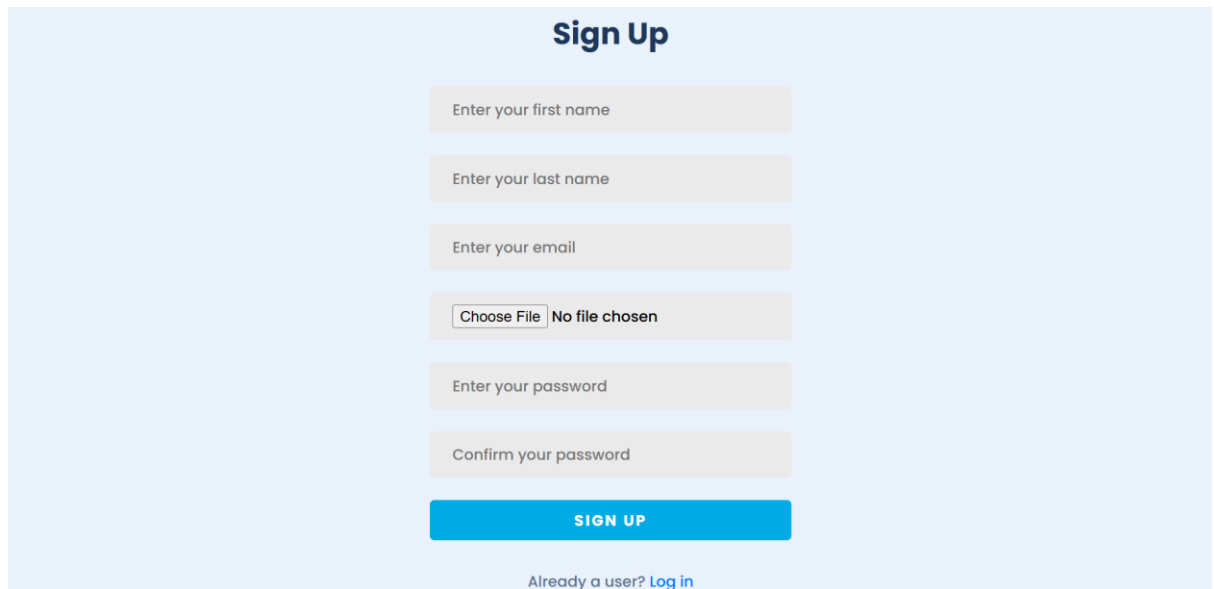


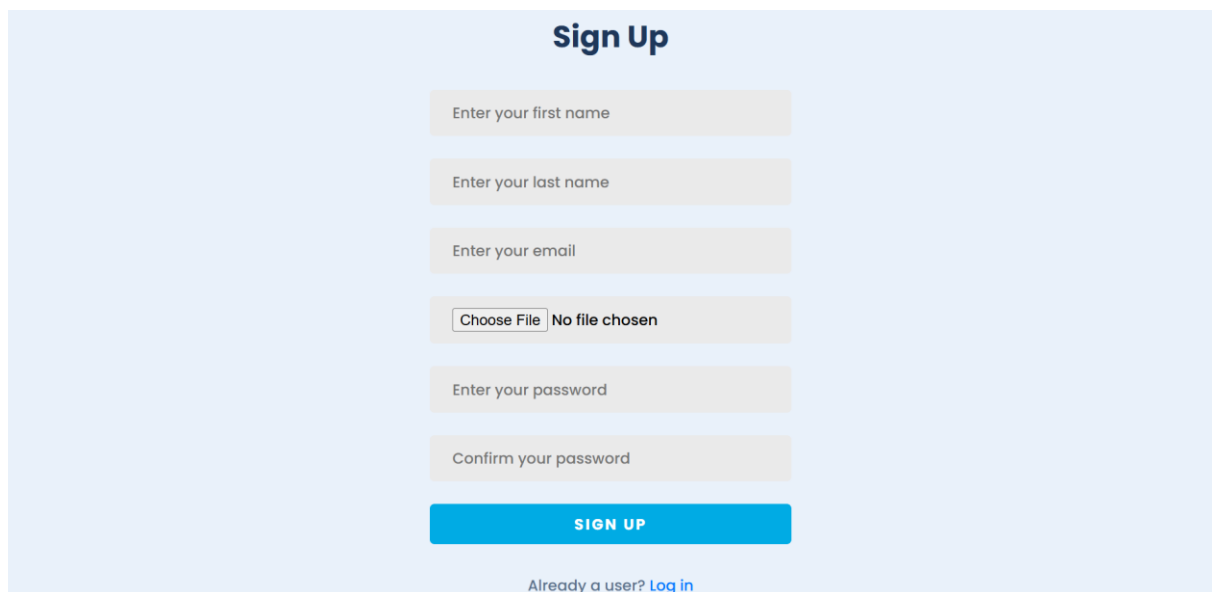
Fig 2: About us Page of the Health Booker



The image shows a 'Sign Up' form on a light blue background. The form is centered and consists of several input fields and a button. At the top, the title 'Sign Up' is displayed in a bold, dark blue font. Below the title, there are six input fields stacked vertically: 'Enter your first name', 'Enter your last name', 'Enter your email', a file upload field with a 'Choose File' button and 'No file chosen' text, 'Enter your password', and 'Confirm your password'. Each input field has a light beige background and rounded corners. Below these fields is a prominent blue button with the text 'SIGN UP' in white, uppercase letters. At the bottom of the form, there is a link that says 'Already a user? Log in' in a smaller, dark blue font.

Fig 3: Register Page of the Health Booker

9. Screenshots or Demo



This image is a duplicate of the one above, showing the 'Sign Up' form for the Health Booker. It features the same layout: a title 'Sign Up', six input fields for personal information and password, a 'SIGN UP' button, and a 'Log in' link for existing users.

Fig 4: New User Registration of Health Booker

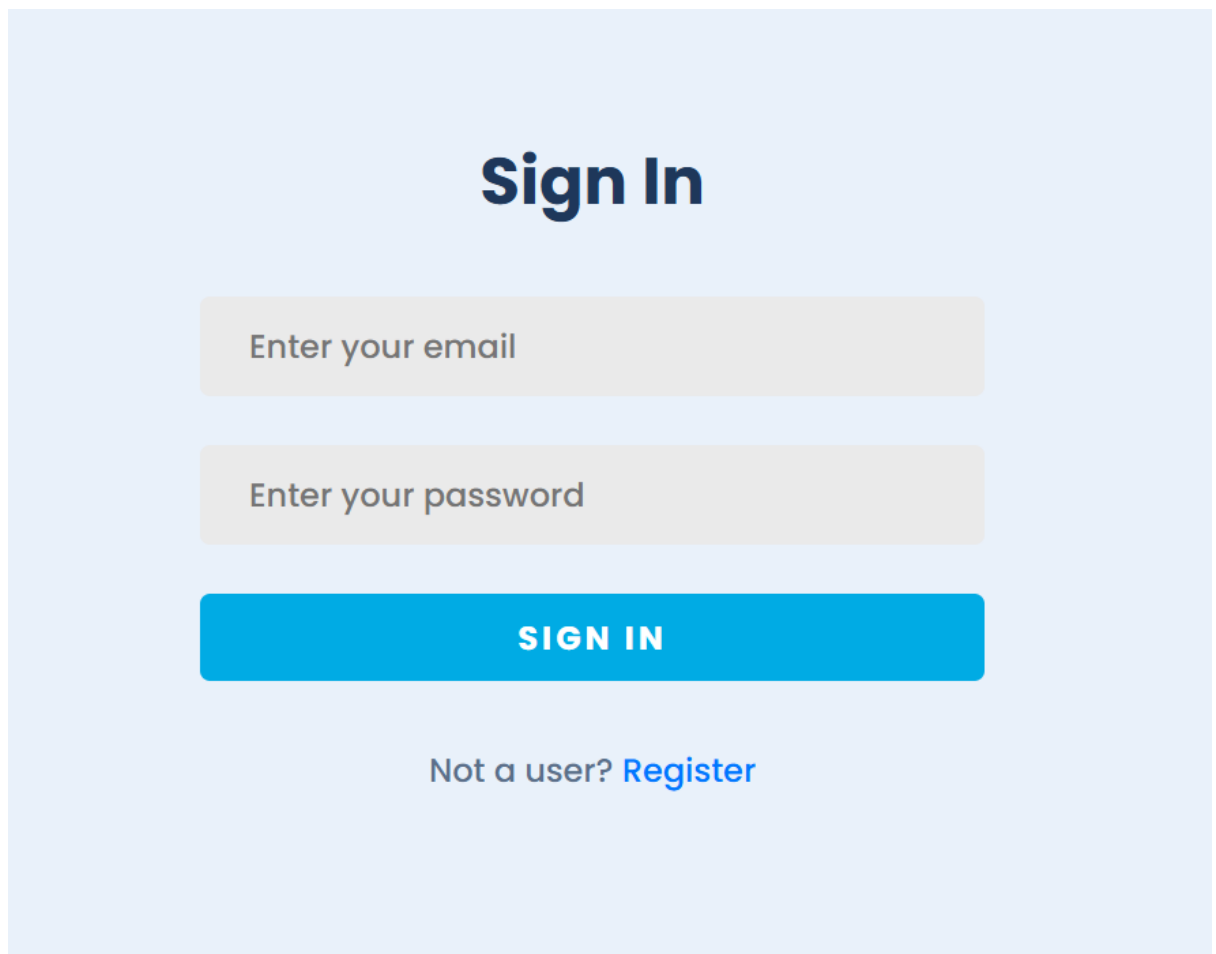


Fig 5: Login Page of the Health Booker

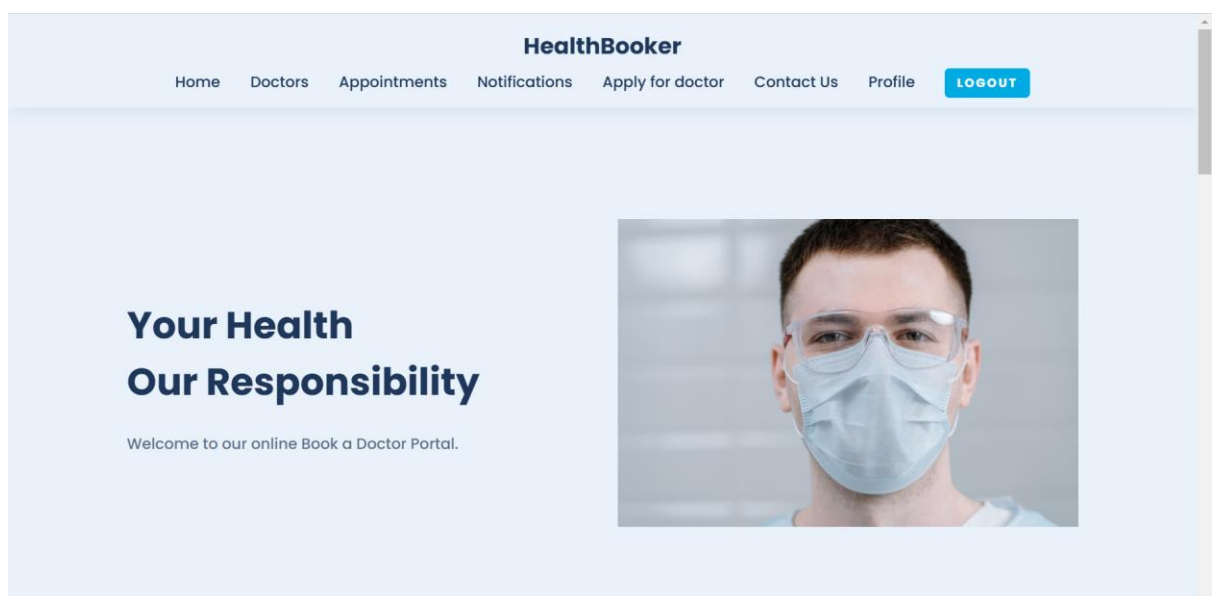


Fig 6: Home Page after successful login

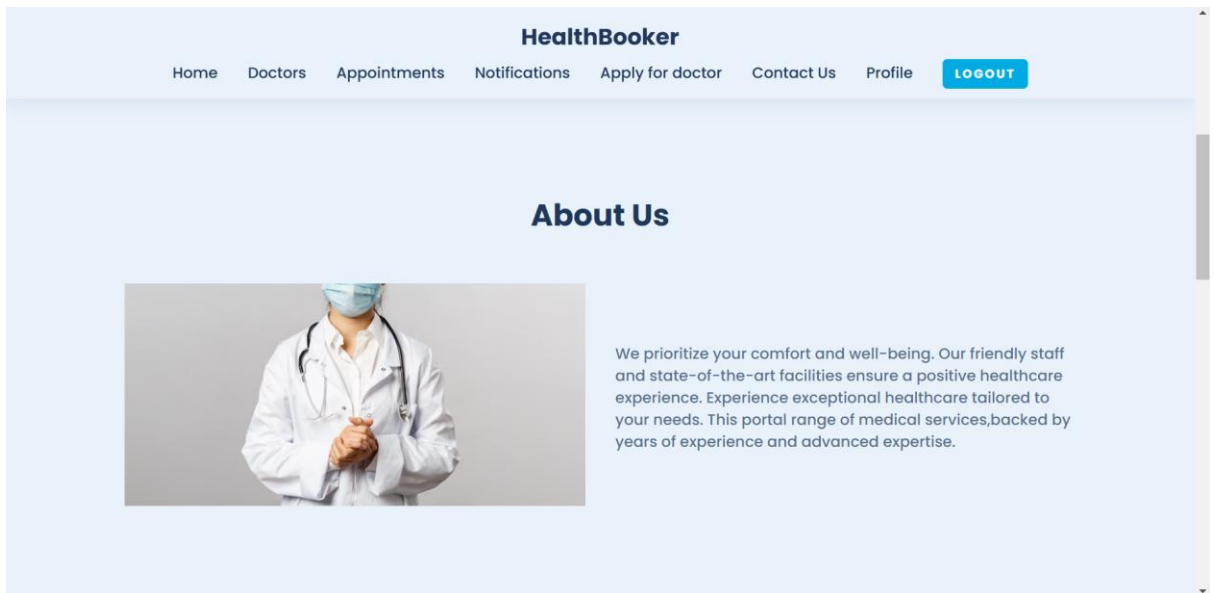


Fig 7: About Us page

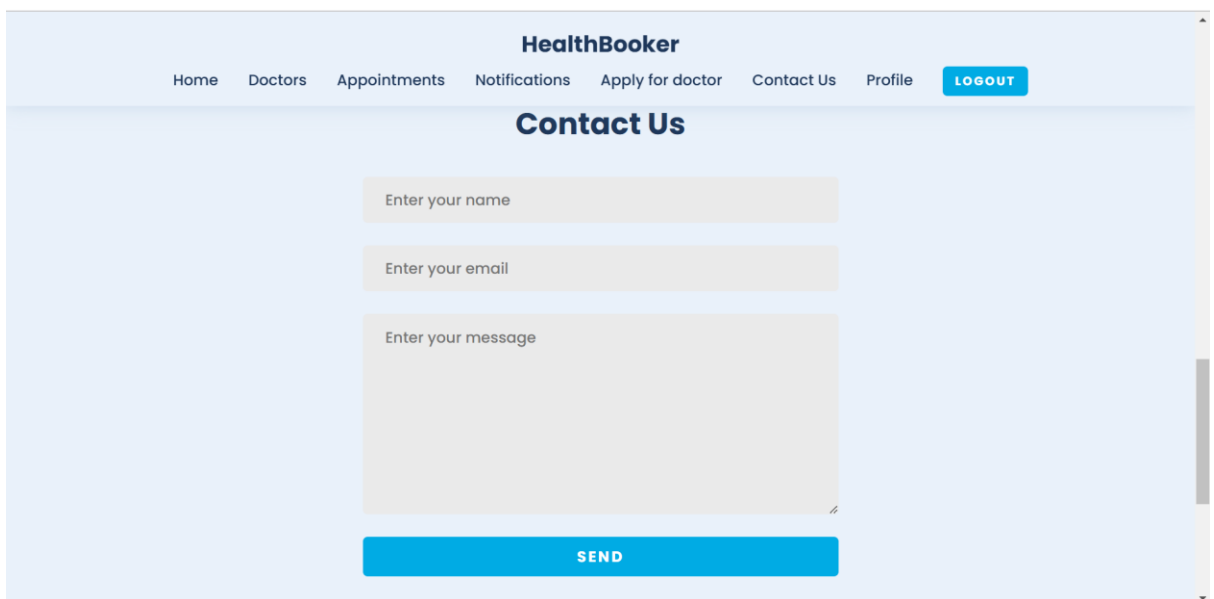


Fig 8: Contact Us Tab

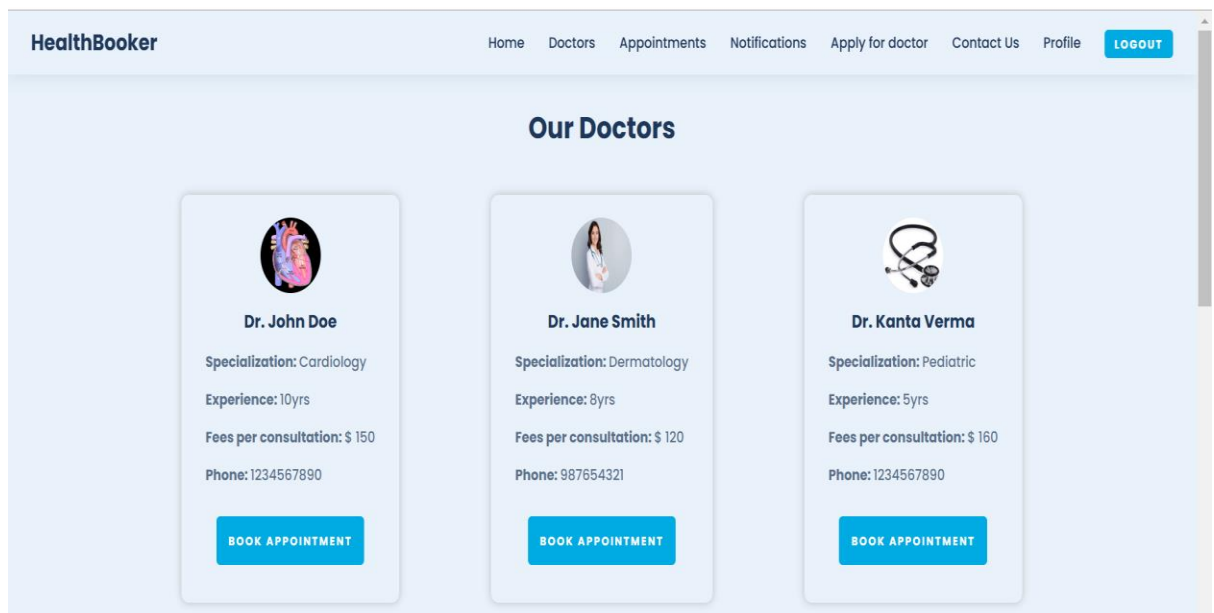


Fig 9: View Doctors page

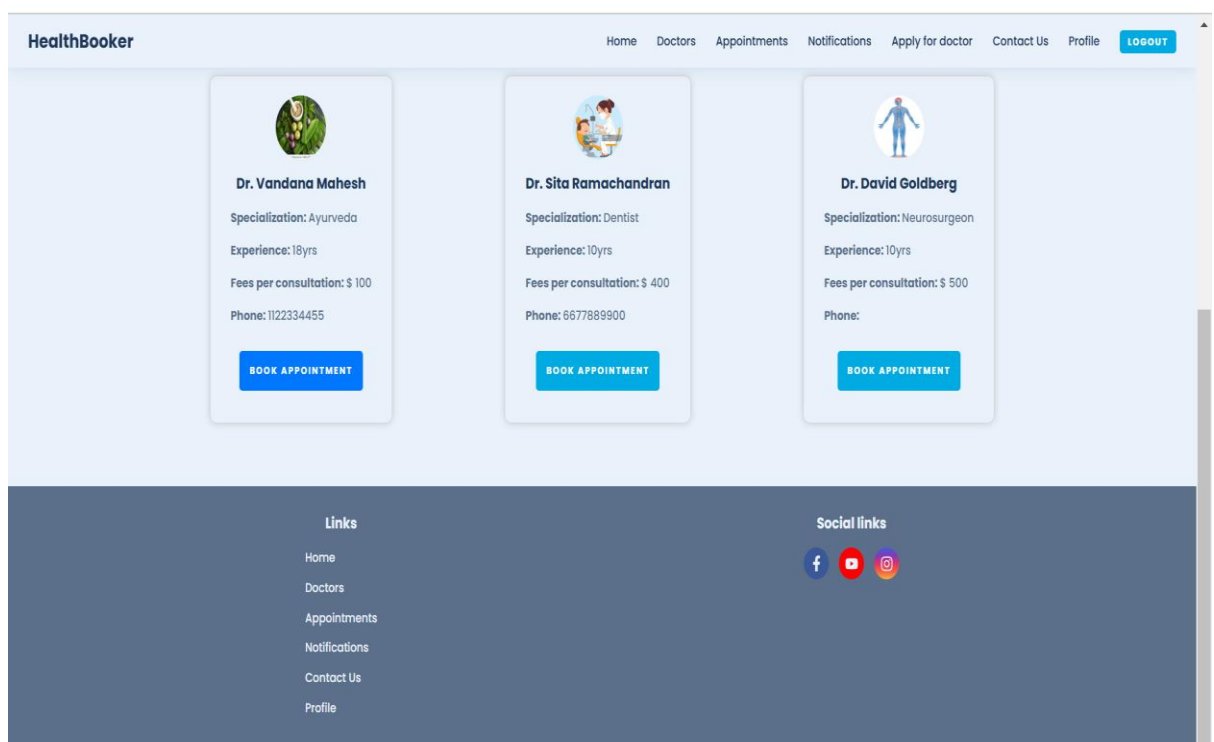


Fig 10 : View More Doctors

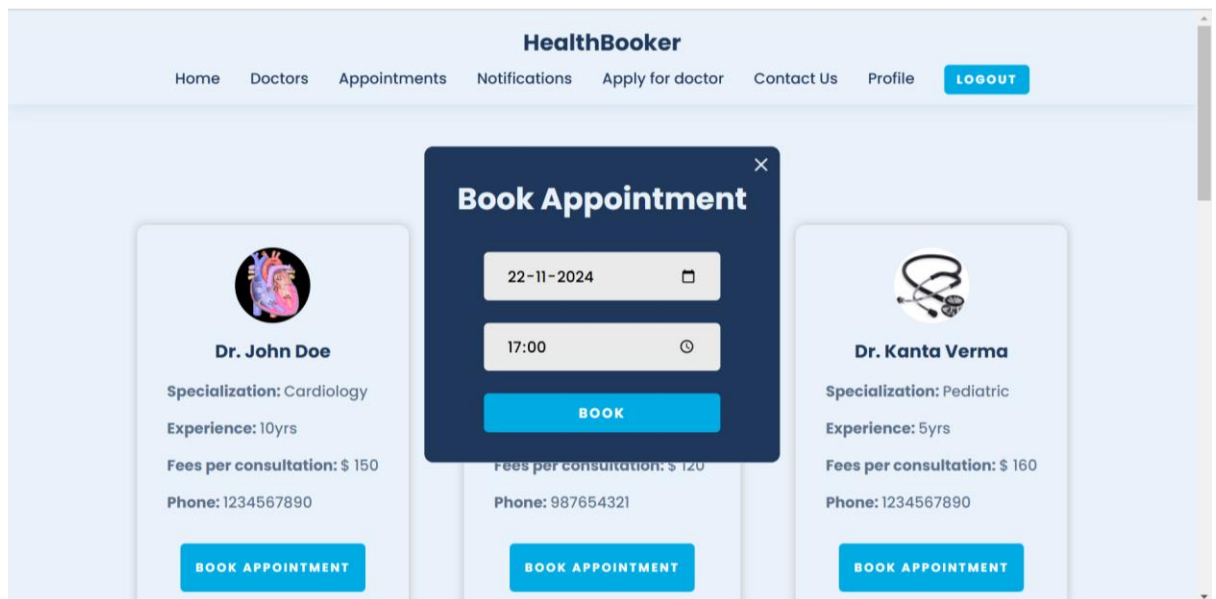


Fig 11 : Book a Appointment with Date and Time

The screenshot shows the 'Your Appointments' page in the HealthBooker application. The page has a light blue header with the title 'Your Appointments'. Below the header, there is a table with 8 columns: S.No, Doctor, Patient, Appointment Date, Appointment Time, Booking Date, Booking Time, and Status. The table contains two rows of data.

S.No	Doctor	Patient	Appointment Date	Appointment Time	Booking Date	Booking Time	Status
1	Sita Ramachandran	Padmaja Harikrishnan	2024-11-22	17:30	2024-11-13	16:38:14	Completed
2	Vandana Mahesh	Padmaja Harikrishnan	2024-11-22	17:00	2024-11-14	16:25:30	Pending

Fig 12 : View the user's scheduled Appointments

HealthBooker

[Home](#)
[Doctors](#)
[Appointments](#)
[Notifications](#)
[Apply for doctor](#)
[Contact Us](#)
[Profile](#)
[LOGOUT](#)

Apply for Doctor

[APPLY](#)

Fig 13 : Apply for a Doctor, if user cannot find a doctor for their requirement

Admin Dashboard










All Users										
S.No	Pic	First Name	Last Name	Email	Mobile No.	Age	Gender	Is Doctor	Remove	
1		Vandana	Mahesh	vandana.mahesh@example.com	1122334455	50	female	Yes	REMOVE	
2		Sita	Ramachandran	sita.ramachandran@example.com	6677889900	38	female	Yes	REMOVE	
3		Jang	Chi	jangchi@example.com	987654321	45	male	Yes	REMOVE	
4		Kanta	Verma	kantaverma@example.com	1234567890	30	female	Yes	REMOVE	
5		Harikrishnan	Rathnaswami	raharikrishnan@gmail.com	8939286424	50	male	No	REMOVE	
6		John	Doe	john doe@example.com	1234567890	30	male	Yes	REMOVE	
7		Jane	Smith	janesmith@example.com	987654321	25	female	No	REMOVE	
8		Padmaja	Harikrishnan	hapadmaja@gmail.com	9176419640	20	female	No	REMOVE	
9		David	Goldberg	david@gmail.com		40	male	Yes	REMOVE	

Fig 14 : All users page








All Doctors										
S.No	Pic	First Name	Last Name	Email	Mobile No.	Experience	Specialization	Fees	Remove	
1		John	Doe	johndoe@example.com	1234567890	10	Cardiology	150	REMOVE	
2		Jane	Smith	janesmith@example.com	987654321	8	Dermatology	120	REMOVE	
3		Kanta	Verma	kantaverma@example.com	1234567890	5	Pediatric	160	REMOVE	
4		Jang	Chi	jangchi@example.com	987654321	8	Orthopedic	300	REMOVE	
5		Vandana	Mahesh	vandana.mahesh@example.com	1122334455	18	Ayurveda	100	REMOVE	
6		Sita	Ramachandran	sita.ramachandran@example.com	6677889900	10	Dentist	400	REMOVE	
7		David	Goldberg	david@gmail.com		10	Neurosurgeon	500	REMOVE	

Fig 15 : All doctors page

All Users								
S.No	Doctor	Patient	Appointment Date	Appointment Time	Booking Date	Booking Time	Status	Action
1	Sita Ramachandran	Padmaja Harikrishnan	2024-11-22	17:30	2024-11-13	16:38:14	Completed	COMPLETE

Fig 16 : Appointment Page

10. Future Enhancements for Book a Doctor Apps

Potential future enhancements for the Book a Doctor app include adding an AI-powered recommendation system to match patients with doctors based on their medical history, location, and preferences, improving the booking experience. Integration with telemedicine features, such as video consultations and in-app chat, would enable remote care, expanding access and convenience for users. Advanced analytics dashboards for doctors and admins could help track appointment trends, patient satisfaction, and resource utilization, supporting informed decision-making. Adding multilingual support would increase accessibility for diverse user groups, while push notifications and SMS reminders

could improve appointment adherence. To enhance security, integrating multi-factor authentication (MFA) and HIPAA compliance checks is crucial for handling sensitive health data. Finally, incorporating a payment gateway for online transactions would streamline billing and provide patients with flexible payment options for a more complete and efficient experience.