FULL STACK DEVELOPMENT WITH MERN

BOOK A DOCTOR

1. INTRODUCTION

Project Title - “Book a Doctor”

Team Members

1. Hariharan R V
2. Deepakkumar V
3. Sathish kumar V M
4. Vimal Prasath S
5. Velmurugan S

2. PROJECT OVERVIEW

Purpose

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.

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 the 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.

3. ARCHITECTURE

Frontend

The frontend is built using **React.js** for the creation of a dynamic and responsive single-page application (SPA). React components are responsible for rendering different parts of the UI, while state management is handled via **Redux** and **React Router**.

- **React**: For building UI components and managing the DOM.
- **React Router**: For navigation and routing between different pages like home, doctor profiles, user dashboard, etc.
- **Redux**: To manage the global state across the app. For example, user authentication status, appointment information, and notifications.
- **Axios**: Used for making HTTP requests to the backend API to fetch and submit data (such as doctor details, appointments, user data).
- **React Bootstrap/Material UI**: For styling and building responsive components.
- **React Icons**: To add icons to enhance the UI.
- **React Spinners**: For showing loading states in the app.
- **React Toastify**: For displaying toast notifications, such as for successful login or appointment booking.

Backend

The backend is built using Node.js with Express.js and connects to a MongoDB database to store user, doctor, appointment, and notification data.

- **Node.js & Express.js:** Node.js provides the runtime for the server, and Express.js is used to build the RESTful API to handle requests from the frontend.
- **MongoDB & Mongoose**: MongoDB is used as the NoSQL database to store and manage data related to users, doctors, appointments, and

notifications. Mongoose is used as an ODM (Object Data Modeling) tool to interact with MongoDB, providing schema-based solutions to validate data.

- **JWT (JSON Web Tokens):** JWT is used for authentication and authorization, allowing secure login and role-based access control.
- **bcrypt.js:** For password hashing and ensuring secure storage of user passwords in the database.

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
	lastname	String		Yes	minLength: 3
	email	String		Yes	unique: true
	password	String		Yes	minLength: 5
	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)	

4. SETUP INSTRUCTIONS

<u>Prerequisites</u>

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.

<u>Frontend Dependencies</u>

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

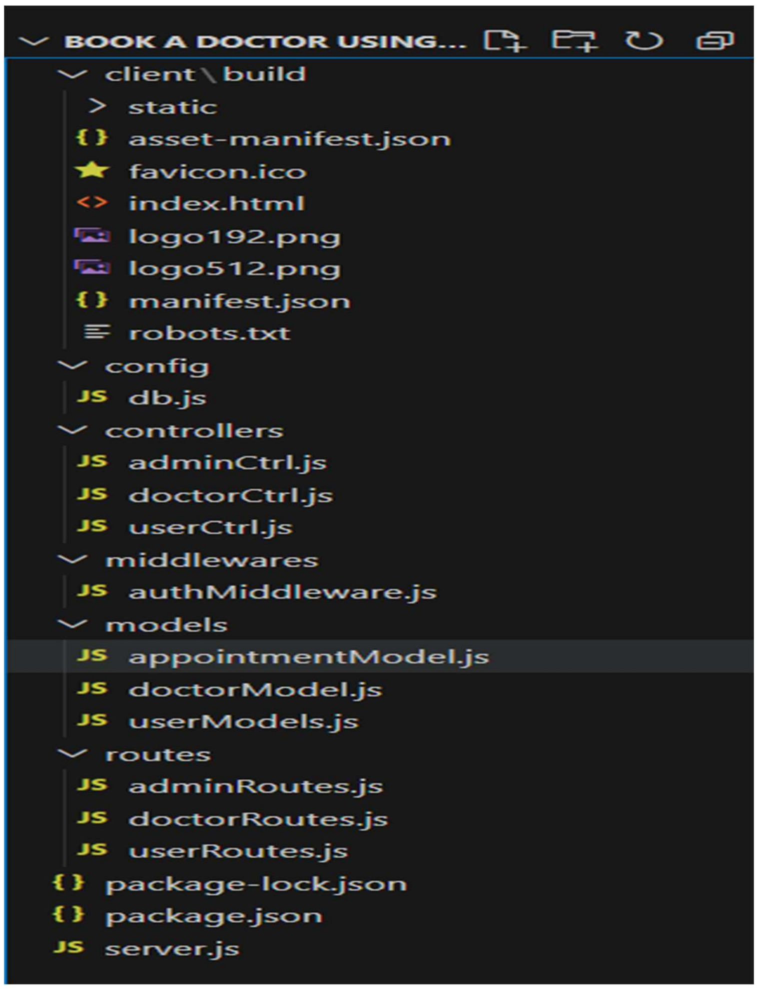
REACT_APP_CLOUDINARY_CLOUD_NAME=YOUR_CLOUDINARY_CLOUD_NAME

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

5. FOLDER STRUCTURE



```
BOOK A DOCTOR USING...
  client\build
    static
    asset-manifest.json
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
  config
    db.js
  controllers
    adminCtrl.js
    doctorCtrl.js
    userCtrl.js
  middlewares
    authMiddleware.js
  models
    appointmentModel.js
    doctorModel.js
    userModels.js
  routes
    adminRoutes.js
    doctorRoutes.js
    userRoutes.js
  package-lock.json
  package.json
  server.js
```

6. RUNNING THE APPLICATION

The npm run build command will create an optimized build of your 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.

7. 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.

8. AUTHENTICATION

(i) Authentication and Authorization in the Project:

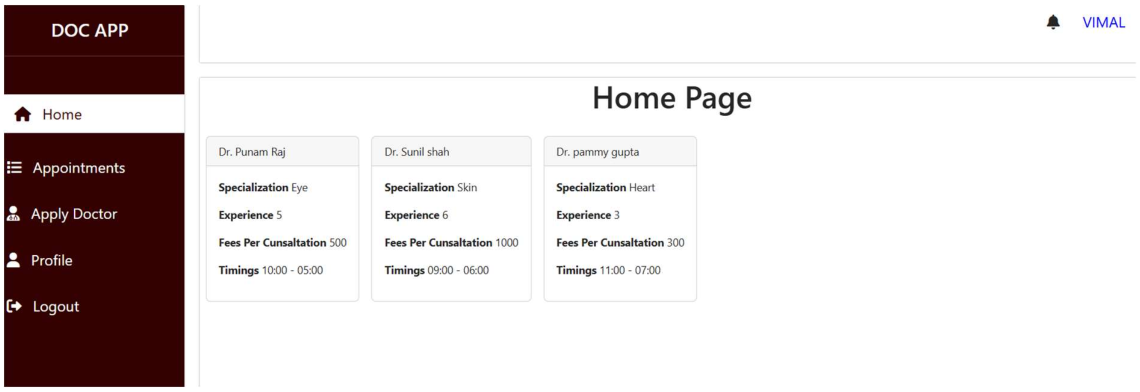
Authentication and authorization are critical to the security of the Book a Doctor app, ensuring that only authenticated users can access sensitive features based on their roles (patient, doctor, or admin). Authentication is implemented using JWT (JSON Web Tokens), which provides a stateless and scalable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

9. USER INTERFACE

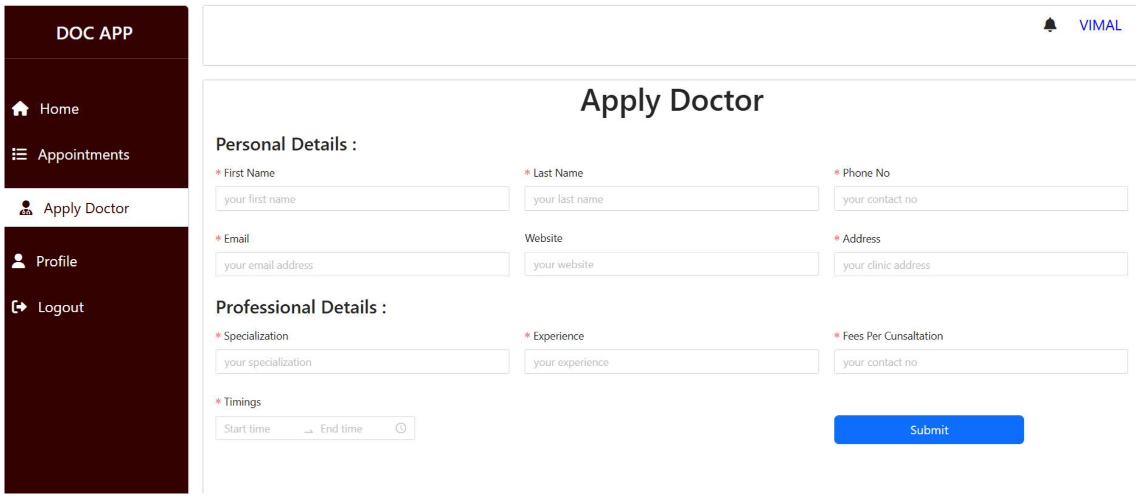
HOME PAGE



Home Page

Dr. Punam Raj	Dr. Sunil shah	Dr. pammy gupta
Specialization Eye	**Specialization** Skin	**Specialization** Heart
Experience 5	**Experience** 6	**Experience** 3
Fees Per Cunsaltation 500	**Fees Per Cunsaltation** 1000	**Fees Per Cunsaltation** 300
Timings 10:00 - 05:00	**Timings** 09:00 - 06:00	**Timings** 11:00 - 07:00

APPLY DOCTOR



Apply Doctor

Personal Details :

* First Name: your first name
* Last Name: your last name
* Phone No: your contact no
* Email: your email address
Website: your website
* Address: your clinic address

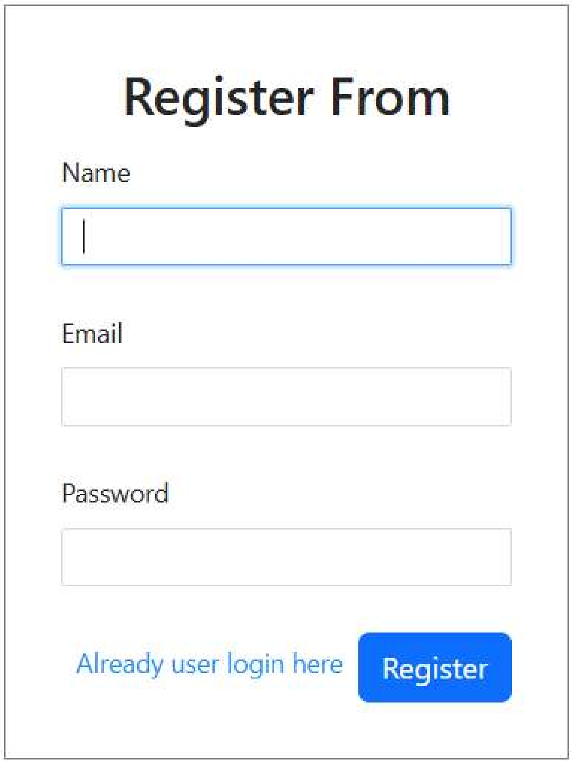
Professional Details :

* Specialization: your specialization
* Experience: your experience
* Fees Per Cunsaltation: your contact no
* Timings: Start time → End time

10. TESTING

11. SCREENSHOTS OR DEMO

SIGN UP PAGE



12. KNOWN ISSUES

- There might be occasional mismatches in available time slots due to concurrency issues between multiple users attempting to book the same time slot. This could be resolved by implementing locks or improving slot booking logic to prevent race conditions.
- While JWT-based authentication is implemented, there could be additional security concerns such as securing the JWT storage and preventing Cross-Site Request Forgery (CSRF) attacks. Integrating **HTTP-only cookies** and **same-site cookie policies** to store tokens would improve security.

13. 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.