Full Stack Development with MERN DocSpot – Project Documentation

# 1. Introduction

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

**Team Members:**- Chejarala Sushma Sri– Full Stack Developer  
- Chittibomma Uma mahesh– Frontend Developer  
- Miriyala Akshaya – Backend Developer  
- Chenikala Nagendra-DataBase Developer

# 2. Project Overview

**Purpose:**

**DocSpot** is an online doctor appointment booking platform built using a full-stack web development approach (HTML/CSS/JavaScript for frontend, Node.js/Express for backend, and SQL for database). The system is designed to provide users with a convenient, accessible, and streamlined way to find doctors and book appointments online.

The platform aims to:

* Allow users to view a list of available doctors based on specialties or locations.
* Enable users to book appointments with selected doctors.
* Store and manage appointment data efficiently using a relational database.
* Simplify the scheduling process for both patients and healthcare providers.

DocSpot ensures a user-friendly experience with a clear interface, while the backend securely handles data operations and scheduling logic.

**Features:**  
- Filter doctors by speciality  
- confirm booking instantly  
- prevent double booking  
- Easy-to-use interface  
- connect frontend to backend using APIs  
- Ready to user login system  
- supports future email alerts

# 3. Architecture

**🖥️ Frontend**

* **Built using HTML, CSS, and JavaScript.**
* **Pages include:**
  + **index.html: Homepage**
  + **doctor-list.html: Displays list of doctors**
  + **book-appointment.html: Form to book an appointment**
* **Communicates with backend using HTTP requests (fetch API/AJAX).**
* **Provides a clean and user-friendly interface.**
* **.**

**🔧 Backend (Server Side)**

* **Built using Node.js and Express.js.**
* **Validates user input and processes booking logic.**
* **Connects to the database and serves data to the frontend.**

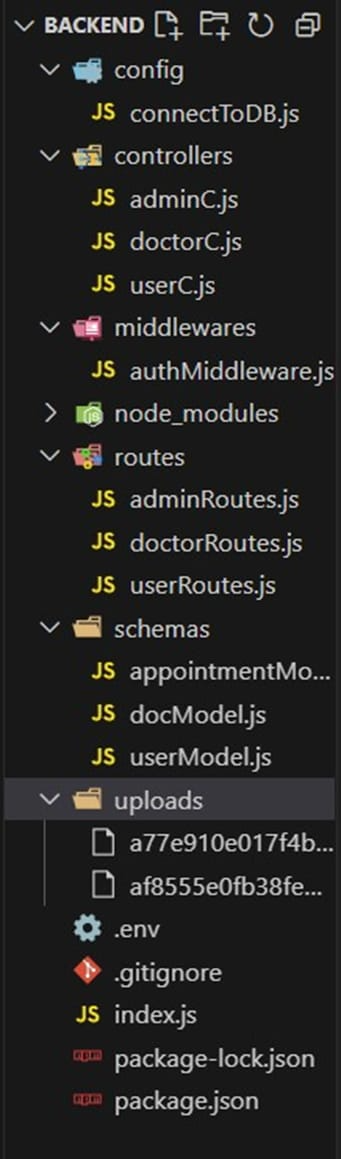
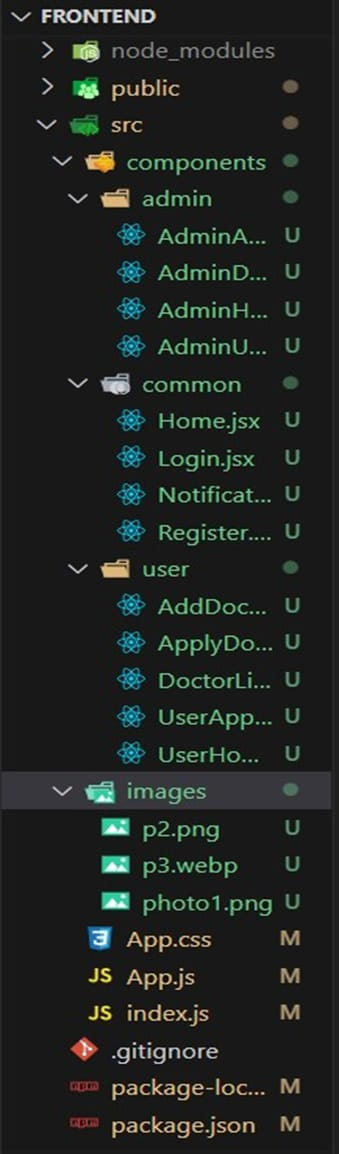
**🗄️ Database (Data Layer)**

* **Managed with MySQL or any SQL-based database.**
* **Structured using docspot.sql file:**
  + **Tables for doctors, appointments, and possibly patients.**
* **Stores all persistent data:**
  + **Doctor profiles**
  + **Appointment records**

# 4. Setup Instructions

**Prerequisites:**  
- Node.js >= 14  
- MongoDB (local or cloud)  
- Git

5.Project Structure



### 6.API Documentation

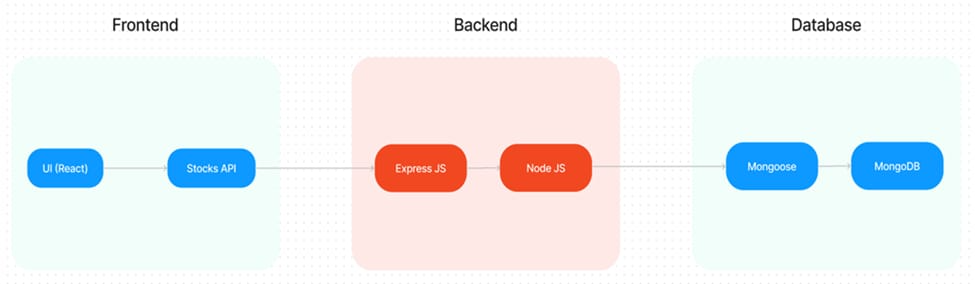
### GET /doctors – Fetch list of all available doctors.

### POST /appointments – Book a new appointment with a doctor.

### GET /appointments – (Optional) Retrieve all booked appointments

### 7.Technical Architecture

The Book a Doctor App features a modern technical architecture based on a client-server model. The frontend utilises Bootstrap and Material UI for a responsive user interface, with Axios handling seamless API communication. The backend is powered by Express.js, offering robust server-side logic, while MongoDB provides scalable data storage for user profiles, appointments, and doctor information. Authentication is secured using JWT for session management and bcrypt for password hashing. Moment.js manages date and time functionalities, ensuring accurate appointment scheduling. The admin interfaces overseas doctor registration, platform governance, and ensures compliance, with Role-based Access Control (RBAC) managing access levels…

 FRONTEND TECHNOLOGIES :

Bootstrap and Material UI: Provide a responsive and modern UI that adapts to various devices, ensuring a user-friendly experience.

Axios: A promise-based HTTP client for making requests to the backend, ensuring smooth data communication between the frontend and server.

BACKEND FRAMEWORK :

Express.js: A lightweight Node.js framework used to handle server-side logic, API routing, and HTTP request/response management, making the backend scalable and easy to maintain.

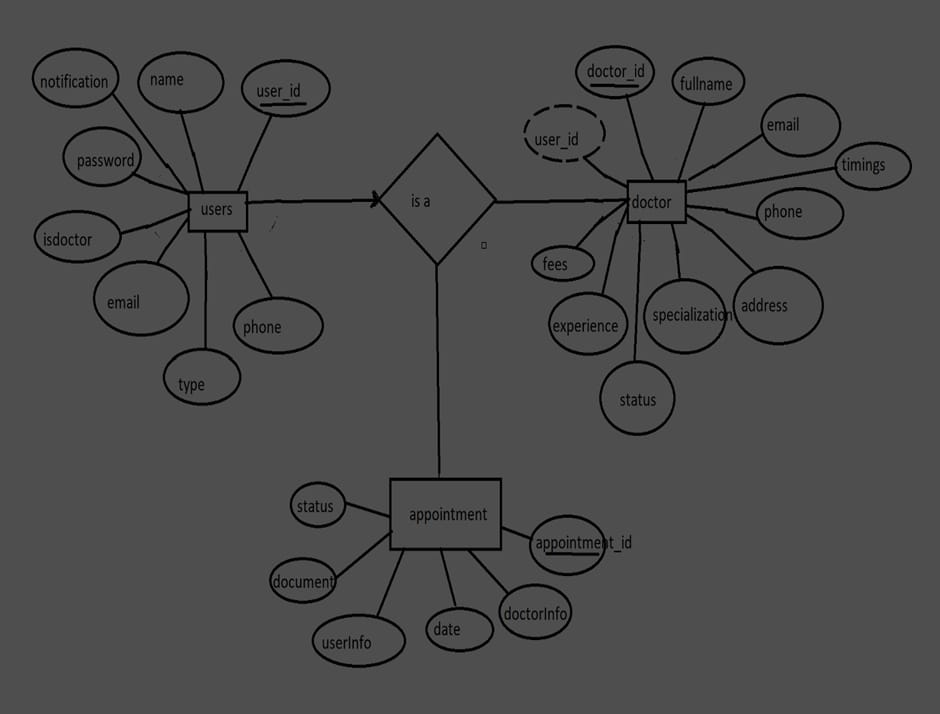
DATABASE AND AUTHENTICATION :

MongoDB: A NoSQL database used for flexible and scalable storage of user data, doctor profiles, and appointment records. It supports fast querying and large data volumes.

JWT (JSON Web Tokens): Used for secure, stateless authentication, allowing users to remain logged in without requiring session storage on the server.

Bcrypt: A library for hashing passwords, ensuring that sensitive data is securely stored in the database.

8.ER DIAGRAM:



The Entity-Relationship (ER) diagram for the Book a Doctor app represents three key entities: Users, Doctors, and Appointments, with their respective attributes and relationships.

The Users collection holds basic user information, including \_id, name, email, notification, password, isdoctor (to differentiate between patients and doctors), type, and phone. The isdoctor field identifies users who are doctors, while others are treated as patients or admins.

The Doctors collection stores information specific to doctors, such as their \_id, userID (acting as a foreign key referencing the Users collection), fullname, email, timings, phone, address, specialisation, status, experience, and fees. The userID` links each doctor to their corresponding user account.

The Appointments collection stores details about appointments, including the \_id, doctorInfo (foreign key referencing the Doctors collection), date, userInfo (foreign key referencing the Users collection), document (medical records or other files), and status (e.g., pending, confirmed). This collection maintains the relationship between users and doctors for each appointment.

The relationships are as follows: one User can be linked to one Doctor (one-to-one), a User can have multiple Appointments (one-to-many), and a Doctor can handle multiple Appointments (one-to-many). The foreign keys userID in the Doctors collection and doctorInfo and userInfo in the Appointments collection establish these connections, enabling the app to manage the interactions between patients and doctors effectively.

9.PRE REQUISITES :

-NODE.JS AND NPM:

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

npm (Node Package Manager) is required to install libraries and manage dependencies.

Download Node.js: Node.js Download

Installation instructions: Installation Guide

Run npm init to set up the project and create a package.json file.

-EXPRESS.JS:

Express.js is a web application framework for Node.js that helps you build APIs and web applications with features like routing and middleware.

Install Express.js to manage backend routing and API endpoints.

Install Express:

Run npm install express

-MONGODB:

MongoDB is a NoSQL database that stores data in a JSON-like format, making it suitable for storing data like user profiles, doctor details, and appointments.

Set up a MongoDB database for your application to store data.

Download MongoDB: MongoDB Download

Installation instructions: MongoDB Installation Guide

-MOMENT.JS:

Moment.js is a JavaScript package for handling date and time operations, allowing easy manipulation and formatting.

Install Moment.js for managing date-related tasks, such as appointment scheduling.

Moment.js Website: Moment.js Documentation

-REACT.JS:

React.js is a popular JavaScript library for building interactive and reusable user interfaces. It enables the development of dynamic web applications.

Install React.js to build the frontend for your application.

React.js Documentation: Create a New React App

-ANTD (ANT DESIGN):

Ant Design is a UI library for React.js, providing a set of reusable components to create user-friendly and visually appealing interfaces.

Install Ant Design for UI components such as forms, tables, and modals.

Ant Design Documentation: Ant Design React

-HTML, CSS, AND JAVASCRIPT:

Basic knowledge of HTML, CSS, and JavaScript is essential to structure, style, and add interactivity to the user interface.

-DATABASE CONNECTIVITY (MONGOOSE):

Use Mongoose, an Object-Document Mapping (ODM) library, to connect your Node.js backend to MongoDB for managing CRUD operations.

Learn Database Connectivity: Node.js + Mongoose + MongoDB

-FRONT-END FRAMEWORKS AND LIBRARIES:

React.js will handle the client-side interface for managing doctor bookings, viewing appointment statuses, and providing an admin dashboard.

You may use Material UI and Bootstrap to enhance the look and feel of the application.

10.PROJECT ROOT STRUCTURE:

Create the Main Folders:

In your project’s root directory, create two main folders: frontend and backend.

plaintext

Copy code

project-root/

├── frontend/

└── backend/

BACKEND SETUP :

Install Necessary Packages in the Backend Folder:

Navigate to the backend folder and install the following essential packages:

plaintext

Copy code

backend/

├── config/

├── controllers/

├── models/

├── routes/

├── middleware/

├── uploads/

├── server.js

└── .env

Packages to Install :

cors: To enable cross-origin requests.

bcryptjs: For securely hashing user passwords.

express: A lightweight framework to handle server-side routing and API management.

dotenv: For loading environment variables.

mongoose: To connect and interact with MongoDB.

multer: To handle file uploads.

nodemon: A utility to auto-restart the server upon code changes (for development).

jsonwebtoken: To manage secure, stateless user authentication.

Installation Commands

Run these commands in the backend folder:

bash

Copy code

npm init -y

npm install cors bcryptjs express dotenv mongoose multer jsonwebtoken

npm install --save-dev nodemon

FRONTEND SETUP :

React Project Initialization:

Navigate to the frontend folder and initialise a new React application:

plaintext

Copy code

frontend/

├── public/

├── src/

│ ├── components/

│ ├── pages/

│ ├── services/

│ └── App.js

├── .env

└── package.json

Setting Up the Frontend Project

In the frontend folder, run the following commands to set up and install any initial dependencies:

bash

11.PROJECT FLOW :

Project Demo :

Before diving into development, you can view a demo of the project to understand its functionality and user interactions.

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

Project Code Repository :

Doctor Appointment Booking Using MERN Source Code

The source code for this project can be accessed and cloned from GitHub, providing a base structure and example code for further customization and understanding.

GitHub Repository: Source Code

Video Tutorials :

For a step-by-step guide on setting up and working with this project, follow the video tutorials below:

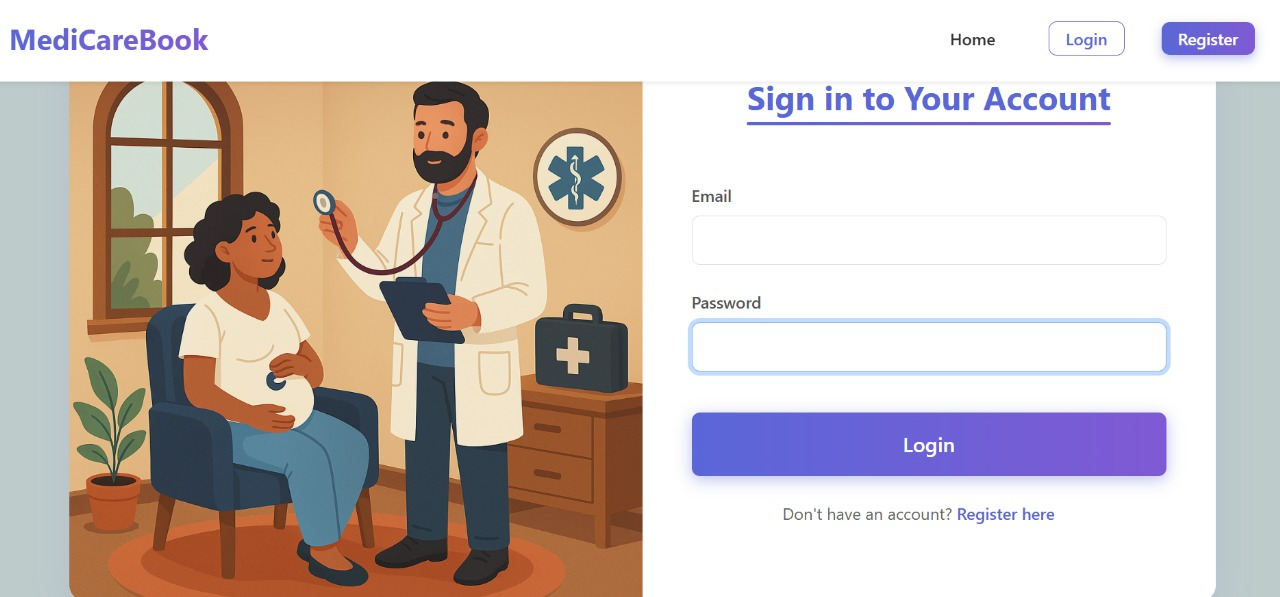
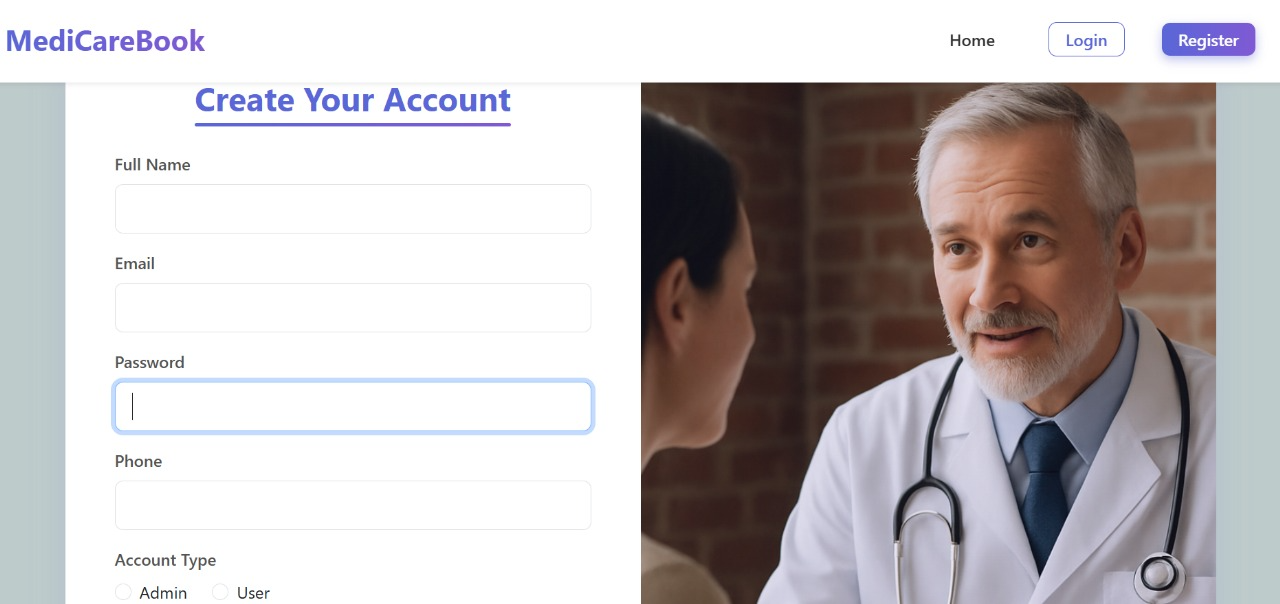
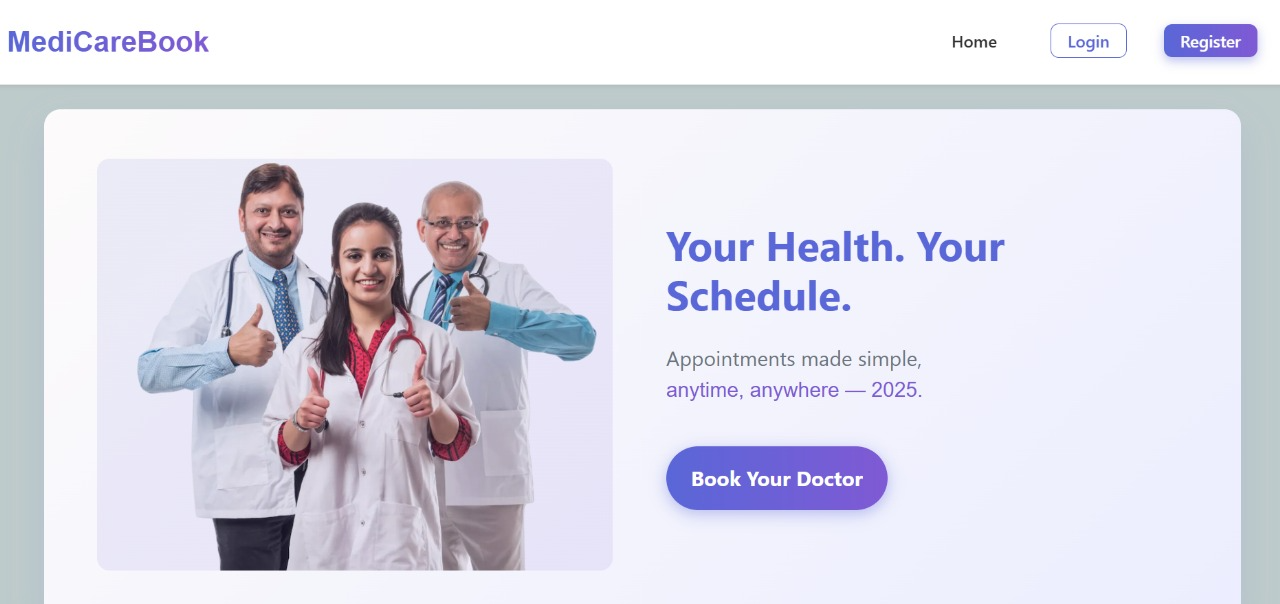
Video Guide 1: Setting Up the Backend

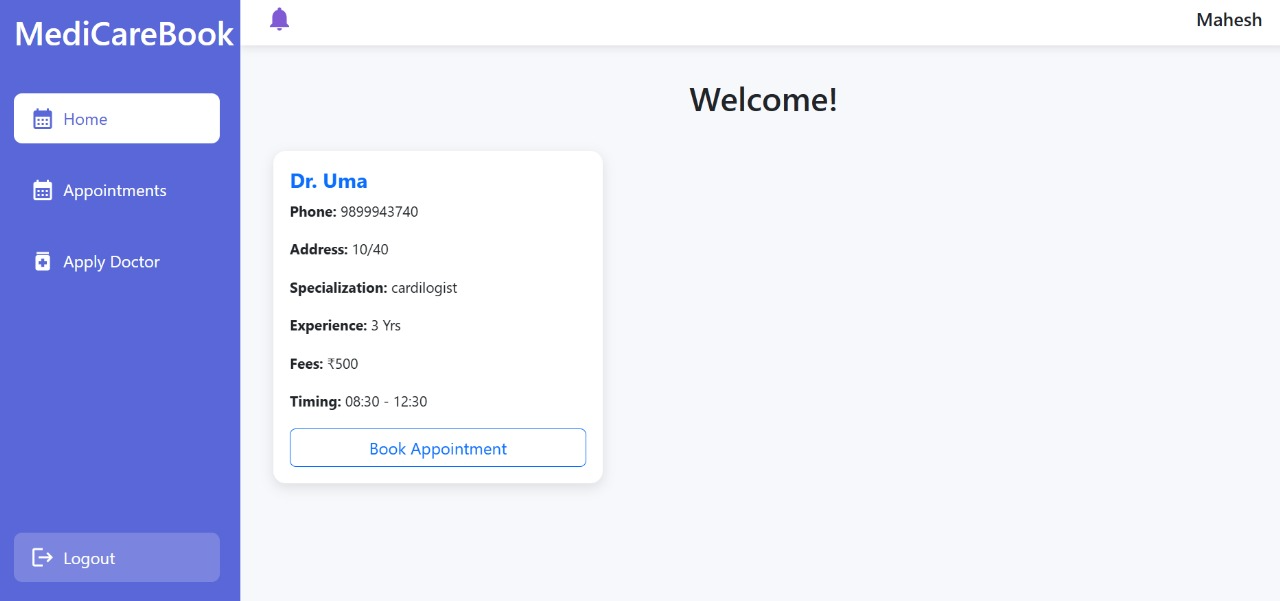
Video Guide 2: Configuring Frontend and API Endpoints

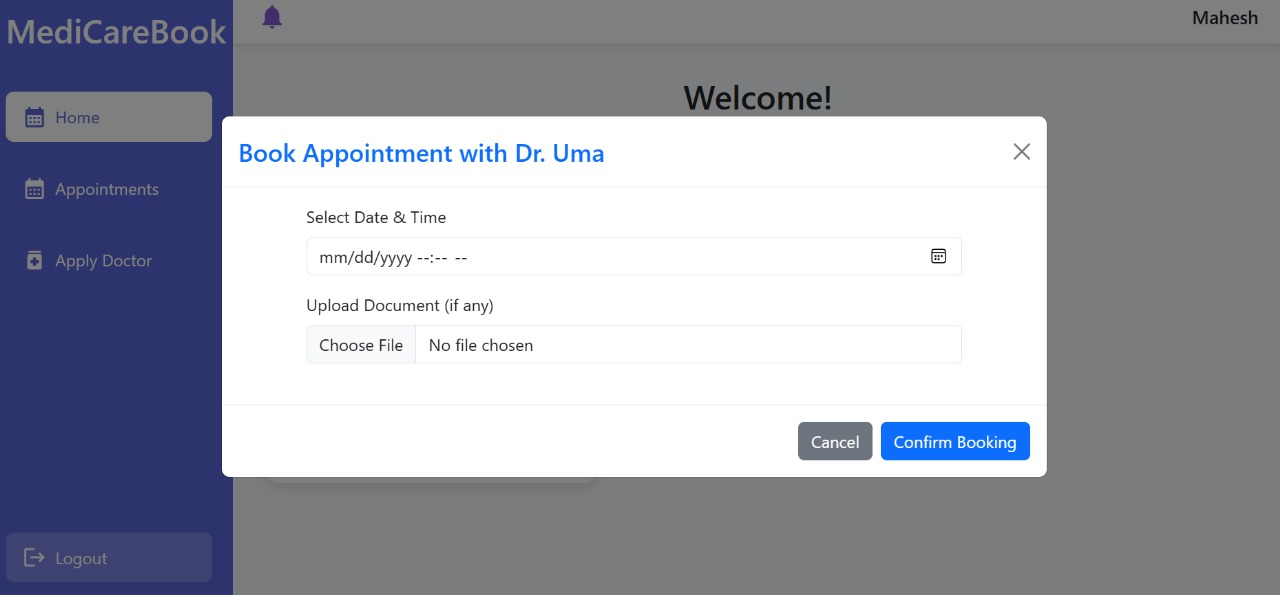
Video Guide 3: Testing and Deployment

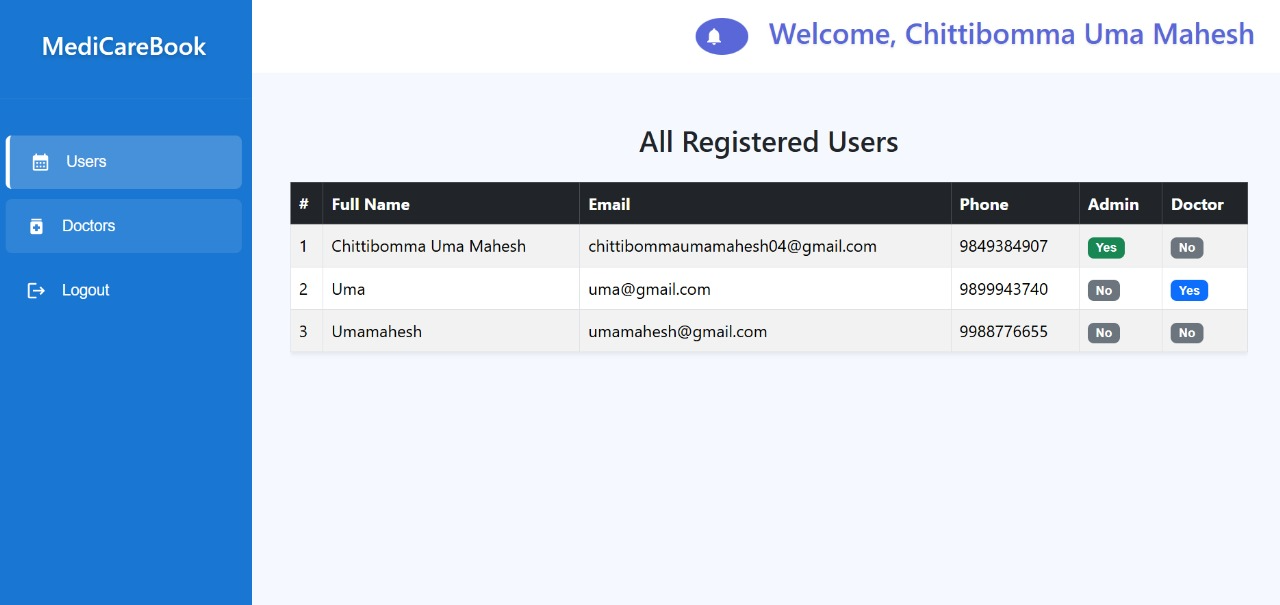
These resources should give you a solid foundation and clear understanding of the project structure and workflow before development begins.

1. SCREENSHOTS OR DEMO









GITHUB: