**FULL STACK DEVELOPMENT WITH MERN PROJECT DOCUMENTATION**

**Team Members:**

**Team ID: LTVIP2025TMID55096**

**Team Leader : Govada Sankara Naga Shyam**

**Team Member: Goli Komal**

**Team Member: Goli Sai Charan Reddy**

**Team Member: Goli Subhash**

1. Introduction

1 **Goli Sai Charan Reddy**. – Frontend Developer  
 Works on the React-based UI, handles component design, page routing, and user interactions.

2. **: Goli Komal**– Backend Developer  
 Builds RESTful APIs using Node.js and Express.js, manages authentication and server logic.

3. **Govada Sankara Naga Shyam**– Database Administrator  
 Designs and manages MongoDB schemas, handles CRUD operations and ensures data consistency.

4. **Goli Subhash** – Project Coordinator & Full Stack Developer  
 Responsible for overall planning, coordination, GitHub management, and integration of frontend and backend.

# 2. Project Overview

Purpose:  
To simplify healthcare booking and management by providing users with real-time appointment slots, secure bookings, and doctor management features.

Key Features:  
- Book doctor appointments  
- User/Admin role-based access  
- Admin dashboard for doctor and booking management  
- Email or dashboard confirmation  
- Secure login with JWT

# 3. Architecture

## Frontend:

Built with React.js using Vite, React Router, Ant Design, and Bootstrap. Pages include Login, Register, User Dashboard, and Admin Dashboard.

## Backend:

Developed using Node.js and Express.js. RESTful APIs handle all CRUD operations related to users, doctors, and bookings.

## Database:

MongoDB stores user data, doctor details, and bookings. Mongoose is used for schema design and validations.

## Frontend (React.js)

Components:  
- Login.jsx, Register.jsx: Auth pages  
- UserDashboard.jsx: Book appointments  
- AdminDashboard.jsx: Manage appointments and doctors

## Backend (Node.js/Express.js)

API Routes:  
POST /api/user/register // User Registration  
POST /api/user/login // User Login  
GET /api/user/profile // User Profile  
POST /api/doctor/book // Book Appointment  
GET /api/admin/requests // Admin approval list

Middleware: JWT authentication, request validation, role-based access.

## Database (MongoDB) Schemas:

User Schema:  
name, email (unique), role, password  
Appointment Schema:  
doctorId, userId, status, date

# 4. Setup Instructions

Prerequisites:  
- Node.js >= 18  
- MongoDB running locally or Atlas  
- npm or yarn

Installation:  
- git clone https://github.com/your-username/DocSpot.git  
- cd DocSpot

Frontend:  
cd client  
npm install  
npm run dev

Backend:  
cd server  
npm install  
node index.js

Environment Variables (.env):  
PORT=5000  
MONGO\_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/  
JWT\_SECRET=your\_jwt\_secret

# 5. Folder Structure

Client:  
/client  
├── /src  
│ ├── /components  
│ ├── /pages  
│ ├── App.jsx  
│ └── main.jsx

Server:  
/server  
├── /controllers  
├── /routes  
├── /models  
├── index.js  
└── .env

# 6. Running the Application

Frontend:  
cd client  
npm run dev

Backend:  
cd server  
node index.js

# 7. API Documentation

- POST /api/user/register: Registers a new user  
- POST /api/user/login: Logs in an existing user  
- GET /api/user/profile: Retrieves logged-in user's profile  
- POST /api/doctor/book: Books a doctor appointment  
- GET /api/admin/requests: Admin fetches pending appointments

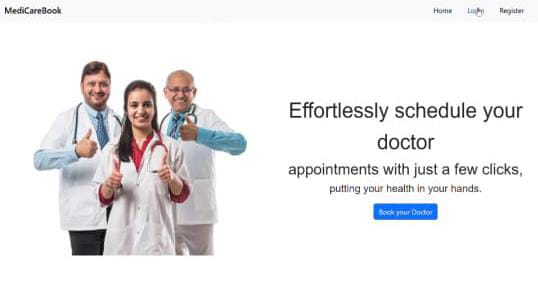
# 8. Authentication

JWT-based authentication  
Tokens stored in localStorage  
Protected routes (admin/user) validated via middleware

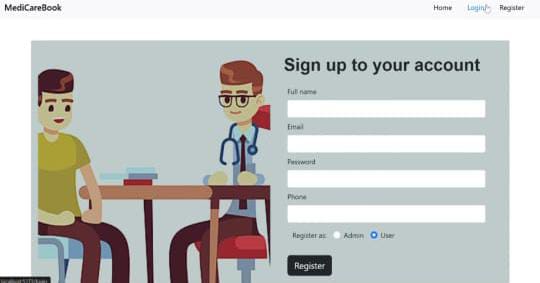
# 9. User Interface

- Login Form: Auth for user/admin  
- Register Page: Role selection and auth setup  
- User Dashboard: Book appointments  
- Admin Dashboard: Approve, reject, or manage appointments

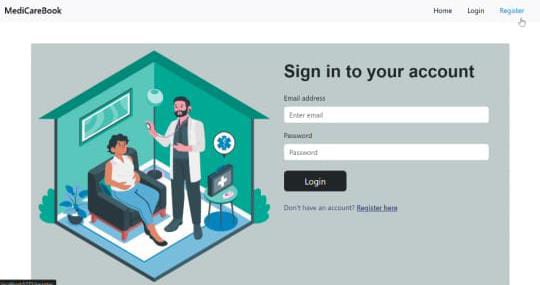
HOME PAGE



LOGIN FORM



REGISTER FORM



ALL APPOINTMENTS FOR ADMIN PANEL



# 10. Testing

Postman used to test all REST API endpoints  
Frontend component testing with Jest

# 11. Demo

Link: To be added

<https://drive.google.com/file/d/1ZHGeFh0eLc2FBF8AGcX6J-LRvsSqY2-7/view?usp=sharing>

# 12. Known Issues

No live chat or notification system yet  
Appointment calendar integration pending

# 13. Future Enhancements

Payment integration via Razorpay/Stripe  
Real-time socket-based notifications  
Video call or chat-based appointments  
Role-based analytics dashboard