

# Full Stack Development with MERN

## Project Documentation format

---

### 1. Introduction

#### Project Title:

**DocSpot – Online Doctor Appointment & Health Management System**

#### Team Members & Roles:

- **Sagarapu Bharghavi (Team Leader)** – Project Management & Backend Development
  - **Marreddi Lalitha Devi** – Frontend Development (React UI)
  - **N Tejaswi** – Database Design & API Integration
  - **Goteti Sesha Venkata Saketh Ram** – Authentication & Security Implementation
  - **Nimmala Durga Mahendra** – Testing & Documentation
- 

### 2. Project Overview

#### Purpose

DocSpot is a web-based healthcare platform that allows patients to book doctor appointments online, manage medical records digitally, and consult doctors easily. The goal is to reduce waiting time and improve healthcare accessibility.

#### Features

- User Registration & Login
  - Doctor Search & Filtering
  - Online Appointment Booking
  - Appointment History
  - Admin Dashboard
  - Secure Medical Records Storage
  - Notifications & Reminders
- 

### 3. Architecture

#### Frontend (React.js)

- Built using React.js
- Uses Axios for API calls
- React Router for navigation
- Responsive UI with Bootstrap / CSS

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

- RESTful APIs using Express.js
- Middleware for authentication
- Handles appointment logic & business rules
- Error handling & validation

#### Database (MongoDB)

- Stores:
    - Users (Patients & Doctors)
    - Appointments
    - Medical Records
  - Uses Mongoose for schema modeling
- 

## 4. Setup Instructions

### Prerequisites

- Node.js (v16 or above)
- MongoDB (Local or Atlas)
- npm

### Installation

1. Clone the repository

git clone <https://github.com/your-project/docspot.git>

2. Install dependencies

cd client

npm install

cd ../server

npm install

3. Create .env file in server folder

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_secret\_key

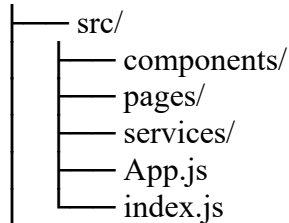
PORT=5000

---

## 5. Folder Structure

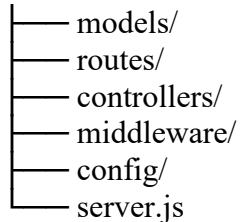
### Client (React Frontend)

client/



### Server (Node Backend)

server/



## 6. Running the Application

### Backend

cd server  
npm start

## Frontend

cd client  
npm start

Frontend runs on: <http://localhost:3000>

Backend runs on: <http://localhost:5000>

---

## 7. API Documentation

### User Routes

- POST /api/auth/register – Register user
- POST /api/auth/login – Login user

### Doctor Routes

- GET /api/doctors – Get all doctors
- GET /api/doctors/:id – Get doctor details

### Appointment Routes

- POST /api/appointments – Book appointment
  - GET /api/appointments/:userId – View appointments
  - DELETE /api/appointments/:id – Cancel appointment
- 

## 8. Authentication

- Uses **JWT (JSON Web Token)** for authentication
  - Passwords hashed using **bcrypt**
  - Protected routes use authentication middleware
  - Role-based access (Patient / Doctor / Admin)
- 

## 9. User Interface

UI includes:

- Home Page
- Login/Register Page
- Doctor Listing Page
- Appointment Booking Page
- Admin Dashboard

(Screenshots to be added here in final document.)

---

## 10. Testing

- Manual testing of all modules
  - API testing using **Postman**
  - Unit testing using **Jest (optional)**
- 

## 11. Screenshots / Demo

- Add screenshots of:

- Login Page
- Doctor Listing
- Appointment Booking
- Dashboard

Or provide demo link (if deployed on Render/Heroku/Vercel).

---

## **12. Known Issues**

- Internet required for real-time booking
- Limited payment gateway integration (future improvement)
- Basic UI (can be enhanced further)

---

## **13. Future Enhancements**

- Video Consultation Integration
  - Online Payment Gateway
  - AI-Based Doctor Recommendation
  - Mobile App Version
  - Insurance Integration
  - Multi-language Support
- .