

# **DOCSPOT: SEAMLESS APPOINTMENT BOOKING FOR HEALTH**

A Project Report Submitted by:  
**Swetha Jivireddi - Full Stack Developer**

## **1) Project Overview**

Healthcare management systems are essential for improving hospital operations and patient services. Traditional appointment booking methods are manual, time-consuming, and inefficient. With the advancement of web technologies, healthcare services can be digitized to enhance accessibility and efficiency. This project focuses on building an online healthcare appointment system using the MERN stack, enabling seamless interaction between patients, doctors, and administrators.

## **2) Project Overview**

### **➤ Purpose**

The purpose of this project is to develop a web-based healthcare appointment management system that allows patients to book appointments with doctors online. Traditional hospital appointment systems are manual and inefficient. This system digitizes the process to improve accessibility, efficiency, and management.

The main goal is to:

- Reduce manual workload
- Provide secure user authentication
- Enable doctor approval system
- Streamline appointment booking process

### **➤ Features**

- User Registration and Login
- Doctor Registration with Professional Details
- Admin Approval System for Doctors
- Browse Approved Doctors
- Book Appointment
- View Appointment Status (Pending / Approved / Rejected)
- Role-Based Access Control (User / Doctor / Admin)
- Secure JWT Authentication
- Responsive UI Design

### **3) Architecture**

The application follows a **three-tier architecture**:

Frontend → Backend → Database

#### ➤ **Frontend Architecture (React.js)**

- Developed using React.js with TypeScript
- Uses Redux Toolkit for state management
- Uses Material UI for UI components
- Uses React Router for navigation
- API calls handled using RTK Query / Axios
- Protected routes implemented for role-based access

Component-Based Structure:

- Pages (Login, Register, Home, Appointments)
- Components (Navbar, Table, Cards, Forms)
- Redux store for managing global state

#### ➤ **Backend Architecture (Node.js + Express.js)**

- Developed using Node.js and Express.js
- RESTful API structure
- MVC architecture pattern
- Controllers handle business logic
- Routes define API endpoints
- Middleware used for:
  - Authentication
  - Authorization
  - Error handling

Security:

- JWT Token Authentication
- Password hashing using bcrypt
- Role-based route protection

#### ➤ **Database Architecture (MongoDB)**

MongoDB is used as a NoSQL database.

## Collections:

### Users Collection

- \_id
- name
- email
- password (hashed)
- role (user / doctor / admin)

### Doctors Collection

- userId (reference to user)
- fullName
- specialization
- experience
- feePerConsultation
- timings
- status (pending / approved / rejected)

### Appointments Collection

- userId
- doctorId
- date
- time
- status

Mongoose is used to define schemas and models.

Flow:

User → Frontend → API → Backend → MongoDB → Response → Frontend

## 4) Setup Instructions

### ➤ Prerequisites

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

## Installation Steps

### ❖ Step 1: Clone Repository

```
git clone <repository-url>  
cd project-folder
```

### ❖ Step 2: Backend Setup

```
cd server  
npm install  
Create .env file:  
PORT=5000
```

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_secret\_key

### ❖ Step 3: Frontend Setup

```
cd client  
npm install
```

## 5) Folder Structure

### ➤ Client (React Frontend)

client/

```
|   └── src/  
|       ├── components/  
|       ├── pages/  
|       ├── redux/  
|       ├── hooks/  
|       ├── utils/  
|       ├── App.tsx  
|       └── main.tsx  
└── package.json
```

Explanation:

- components → Reusable UI components
- pages → Page-level components

- redux → State management
- utils → Helper functions

➤ **Server (Node.js Backend)**

```
server/
  └── controllers/
  └── models/
  └── routes/
  └── middleware/
  └── config/
  └── server.js
  └── package.json
```

Explanation:

- controllers → Business logic
- models → Mongoose schemas
- routes → API routes
- middleware → Auth & error handling

## 6) Running the Application

➤ **Start Backend**

Inside server directory:

```
npm start
```

Server runs on:

```
http://localhost:5000
```

➤ **Start Frontend**

Inside client directory:

```
npm start
```

Frontend runs on:

```
http://localhost:3000
```

## **7) API Documentation**

### **Authentication APIs**

#### **POST /api/users/register**

Registers new user

Request:

```
{  
  "name": "Swetha",  
  "email": "swetha@gmail.com",  
  "password": "123456"  
}
```

Response:

```
{  
  "message": "User registered successfully"  
}
```

---

#### **POST /api/users/login**

Response:

```
{  
  "token": "jwt_token",  
  "role": "user"  
}
```

---

### **Doctor APIs**

#### **POST /api/doctors/apply**

Doctor registration

#### **GET /api/doctors**

Get all approved doctors

---

### **Appointment APIs**

#### **POST /api/appointments/book**

Book appointment

**GET /api/appointments/user**

Get user appointments

## **8) Authentication**

Authentication is implemented using **JWT (JSON Web Tokens)**.

Process:

1. User logs in
2. Server validates credentials
3. JWT token is generated
4. Token is stored in local storage
5. Token is sent in Authorization header
6. Middleware verifies token for protected routes

Authorization:

- Users can book appointments
- Doctors can manage appointments
- Admin can approve doctors

## **9) User Interface**

The UI is built using:

- React.js
- Material UI
- Responsive design

Screens include:

- Login Page
- Registration Page
- Home Page
- Doctor Profile Page
- Appointment Page
- Admin Dashboard

## **10) Testing**

Testing was performed manually.

Test Cases:

- User registration validation
- Login authentication check
- Doctor approval flow
- Appointment booking flow
- Role-based access testing

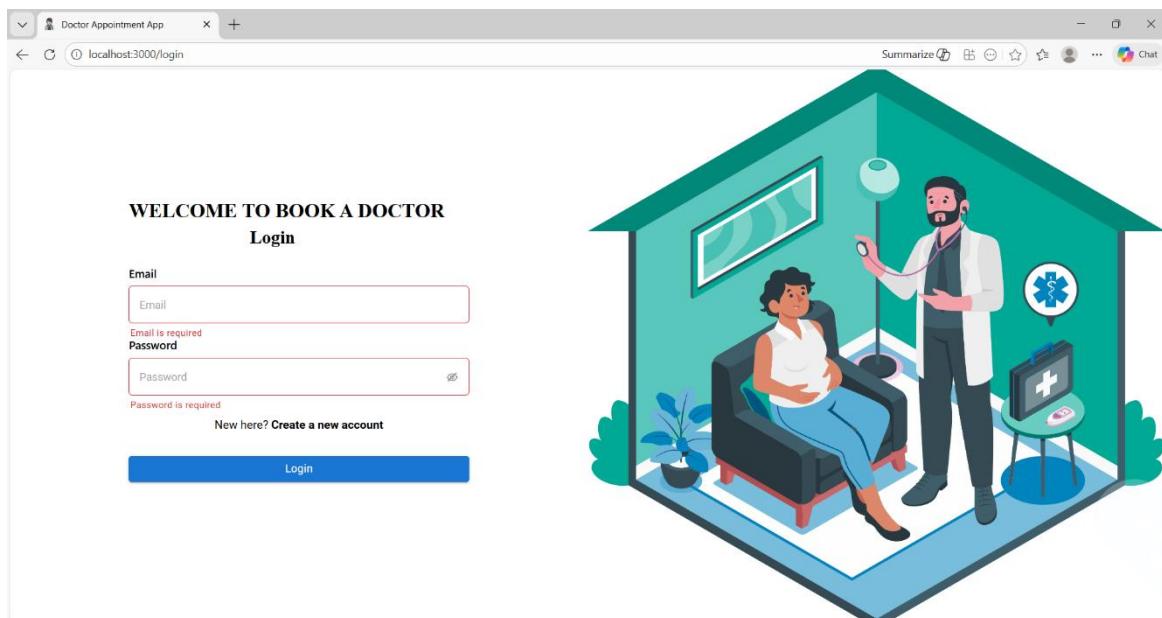
Error handling tested for:

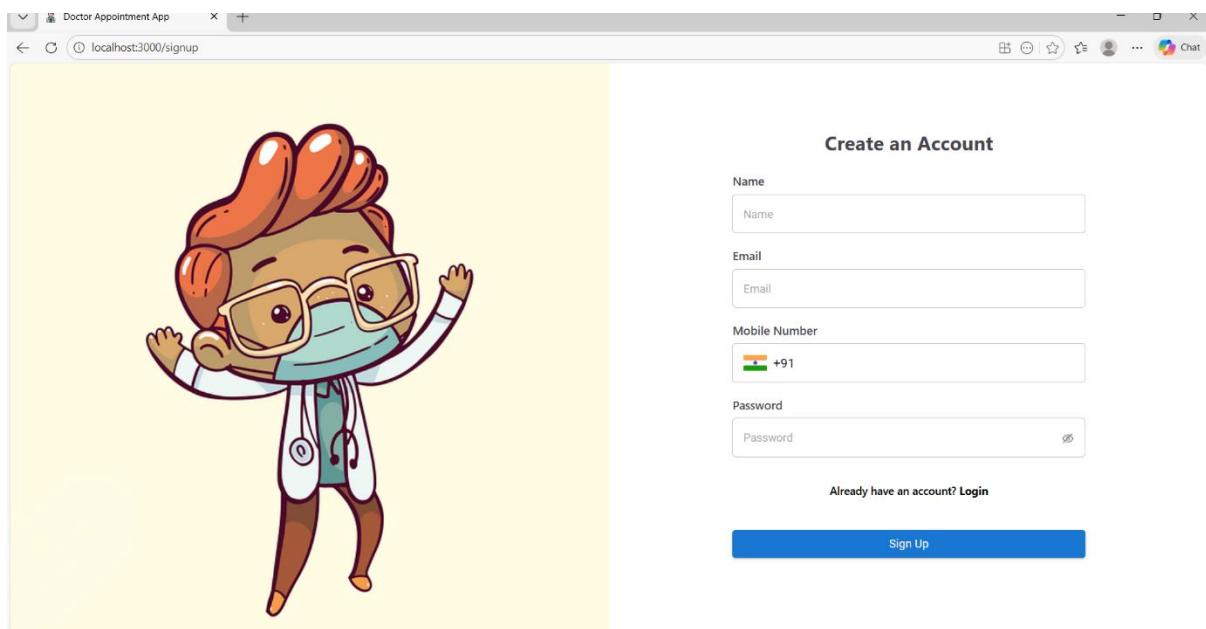
- Invalid credentials
- Unauthorized access
- Invalid form inputs

## **11) Screenshots or Demo**

Include:

- Login Page Screenshot
- Doctor Registration Screenshot
- Admin Approval Screenshot
- Booking Appointment Screenshot
- MongoDB Collections Screenshot





The screenshot shows the "BOOK A DOCTOR" page for a user named "Swetha Jivireddi". The left sidebar has links for Home, Users, Doctors, and Profile. The main content area shows a section titled "Available Doctors" with a single entry for "Dr. Doctor Test (General Physician)". This entry includes details like Phone Number (098765 43210), Address (Visakhapatnam), Fee Per Visit (500), and Timings (9:00 AM to 5:00 PM). There is also a "Select Doctor to add Appointments" button.

The screenshot shows the "BOOK A DOCTOR" page for a user named "Swetha Jivireddi". The left sidebar has links for Home, Users, Doctors, and Profile. The main content area shows a table titled "Users" with four entries:

Name	Email	Date	Roles	Actions
Swetha Jivireddi	swethajivireddi@gmail.com	02/18/2026, 12:26 PM	Owner	<a href="#">Edit</a>
Doctor Test	doctor@gmail.com	02/18/2026, 1:29 PM	Doctor	<a href="#">Edit</a>
Priya	jivireddi028@gmail.com	02/18/2026, 1:37 PM	User	<a href="#">Delete</a>

Doctor Appointment App

localhost:3000/doctors

### BOOK A DOCTOR

#### Doctors

Name	Specialty	Email	Phone Number	Date	Status	Actions
Dr. Doctor Test	General Physician	doctor@gmail.com	098765 43210	02/18/2026, 1:34 PM	Approved	Block

Swetha Jivireddi

Admin

Home

Users

Doctors

Profile

Doctor Appointment App

localhost:3000/notifications

### BOOK A DOCTOR

#### Notifications

Unseen Seen

MARK ALL AS READ

Name: Doctor Test  
Title: New Doctor Request  
Message: Doctor Test has requested to join as a doctor.

Swetha Jivireddi

Admin

Home

Users

Doctors

Profile

Doctor Appointment App

localhost:3000/profile/6995628aa4b41b80fe4ceb4c

### BOOK A DOCTOR

#### Profile Details

Owner

Swetha Jivireddi  
079892 94557

Created At: 02/18/2026, 12:26 PM

Swetha Jivireddi

Admin

Home

Users

Doctors

Profile

Doctor Appointment App

localhost:3000/apply-doctor

## BOOK A DOCTOR

Priya

### Apply For Doctor

1 Basic Information

Prefix	Full Name	Mobile Number
Dr.	Priya	+91 91234-56780
Website	Address	
Website	Address	

2 Professional Information

Specialization	Experience	Fee Per Consultation
Specialization	Experience	Fee Per Consultation
Start Time	End Time	
hh:mm (a p)m	hh:mm (a p)m	

Apply

Doctor Appointment App

localhost:3000/appointments

## BOOK A DOCTOR

Priya

### Appointments

ID	Doctor	Phone	Date	Status
69957372fd6cacb7209d5c5e	Dr. Doctor Test	098765 43210	02/19/2026 10:00 AM	Approved

Doctor Appointment App

localhost:3000/notifications

## BOOK A DOCTOR

Priya

### Notifications

Unseen Seen

MARK ALL AS READ

Name:	Priya
Title:	Appointment Confirmation
Message:	Your appointment status has been approved

```

1 {
2   "name": "client",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@emotion/react": "^11.11.1",
7     "@emotion/styled": "^11.11.0",
8     "@mui/material": "^5.14.18",
9     "@mui/x-date-pickers": "^5.0.10",
10    "@reduxjs/toolkit": "^1.9.7",
11    "@testing-library/jest-dom": "^5.17.0",
12    "@testing-library/react": "^13.4.0",
13    "@testing-library/user-event": "^13.5.0",
14    "@types/jest": "^27.5.2",
15    ...
16  }
17}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Run `npm audit` for details.

> server@1.0.0 start  
> node server.js

Server is running on port 5000  
Connected to MongoDB  
Environment: development  
GET /api/v1/users/verify-user/undefined 401 11.831 ms - 1644  
GET /api/v1/users/verify-user/undefined 401 2.071 ms - 1644

In 1, Col 1 Spaces: 2, UFT-8 LF JSON Go Live Prettier

## 12) Known Issues

- No payment integration
- No real-time notifications
- Basic UI design
- No email confirmation system

## 13) Future Enhancements

- Payment gateway integration
- Email and SMS notifications
- Video consultation
- AI-based doctor recommendation
- Mobile application development
- Dashboard analytics for admin

## 14) CONCLUSION

The Smart Healthcare Appointment & Management System successfully demonstrates the implementation of a full-stack web application using the MERN stack. The system enables secure authentication, role-based access control, and efficient appointment management. It simplifies healthcare appointment booking and reduces manual workload. This project enhanced my knowledge in frontend and backend development, database integration, authentication, and REST API implementation.