FULL STACK DEVELOPMENT WITH MERN

DocSpot: Seamless Appointment Booking for Health

Team ID: LTVIP2026TMIDS56110
Team Size: 4

Team Members and Roles

Name	Role
Balla Sri Divya Satya Joshna	Team Leader / Backend Architect
Dola Gowthami	Member / Frontend Developer
Bhargavi Satya Seetha Naga Lakshmi Nalam	Member / Full Stack Integration
Chillara Venkata Ramakrishna	Member / Database & Testing

1. Introduction

Project Title: DocSpot Seamless Appointment Booking for Health

The Doctor Appointment Booking System is a full-stack web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). It enables patients to book appointments with doctors online, manage schedules, and access healthcare services efficiently. The platform simplifies the traditional manual appointment system by providing a secure and user-friendly digital solution.

2. Project Overview

Purpose

The purpose of the Doctor Appointment Booking System is to provide an easy and efficient way for patients to book doctor appointments online. It helps reduce waiting time and eliminates manual scheduling. The system allows doctors to manage their appointments digitally and enables admins to monitor users and doctors. It ensures secure access through authentication and role-based control. Overall, it improves the efficiency and quality of healthcare services.

Features

- User Registration & Login (JWT Authentication)
- Doctor Listing & Profile View
- Appointment Booking System
- Admin Dashboard
- Doctor Dashboard
- Appointment Status Management
- Secure Password Hashing
- Role-Based Access Control (Admin, Doctor, Patient)

3. Architecture

Frontend (React.js)

The frontend of the Doctor Appointment Booking System is developed as a Single Page Application (SPA) using React.js. It provides a dynamic and responsive user interface that allows patients, doctors and administrators to interact with the system seamlessly. The application is built using reusable components such as Navbar, DoctorCard, AppointmentForm, Dashboard panels to maintain UI consistency across pages.

Backend (Node.js & Express.js)

The backend is developed using Node.js and Express.js following a RESTful API architecture. It acts as the core logic layer of the application, processing client requests, validating user inputs, and managing appointment scheduling. Express middleware is used for authentication, authorization, and error handling. The backend ensures secure communication between the client and the database while maintaining system integrity and performance.

Database (MongoDB)

MongoDB is used as the NoSQL database to store application data such as user details, doctor profiles, and appointment records. The flexible document-based structure of MongoDB allows efficient storage and retrieval of healthcare-related information.

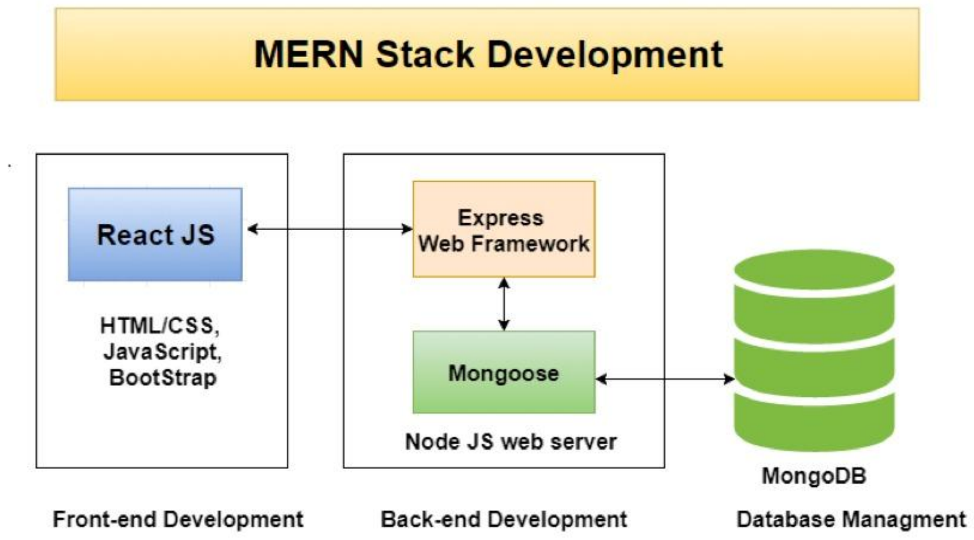


fig: DocSpot architecture

Users (Patient / Doctor / Admin)

↓

Frontend (React.js)

↓

Backend (Node.js + Express.js)

↓

Database (MongoDB)

fig: SystemFlow

4. Setup Instructions

Prerequisites

- Ensure the following are installed on your system:
- Node.js (v14 or higher)
- npm (Node Package Manager)
- MongoDB (Local Community Edition or Atlas URI)
- Git for version control

Installation Steps

1. Clone the repository:

```
git clone https://github.com/gowthamidola123/DocSpot-Seamless-Appointment-Booking-for-Health
```

2. Setup Backend:

```
cd server

npm install

npm start
```

3. Setup Frontend:

```
cd ../client
npm install

npm start
```

5. Folder Structure

Client (React)

```
client/
|── public/
|── src/
|   ├── components/
|   ├── pages/
```

```
│   ├── context/
│   ├── App.js
│   └── index.js
└── package.json
```

Server (Node/Express)

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

6. Running the Application

To start the application locally, run commands in separate terminals:

Frontend

Command: npm start (inside /client directory)

URL: http://localhost:3000

Backend

Command: npm start (inside /server directory)

7. API Documentation

Method	**Endpoint**	**Description**
GET	/api/doctors	Retrieve all all doctors Information
POST	/api/user/register	For New User or Doctor Register
POST	/api/appointments	For User appointment Booking

GET	/api/admin/users	Retrieve user Details
POST	/api/user/login	For User or Admin or Doctor Login
DELETE	/api/admin/doctor/user	For Deleting The doctor or user

8. Authentication

8.1 Authentication Mechanism

The Doctor Appointment Booking System implements a secure, credential-based authentication system to verify user identity and protect sensitive healthcare data. The system ensures that only authorized users can access specific functionalities within the platform.

• **Password Hashing:** User passwords are never stored in plain text. The system uses Bcrypt.js to hash and salt passwords before storing them in the MongoDB database, ensuring strong protection against data breaches.

• **JWT Implementation:** Upon successful login, the server generates a JSON Web Token (JWT). This token acts as a secure digital credential that allows users to access protected routes without repeatedly entering their login details. The token is included in the request header for authentication verification.

8.2 Role-Based Access Control (RBAC)

To maintain system integrity and security, the application distinguishes between different user roles:

• **Patient Role:** Can register, log in, view doctors, book appointments, and track appointment status.

• **Doctor Role:** Can view scheduled appointments and manage availability.

• **Admin Role:** Has full access to manage doctors, users, and appointments within the system.

8.3 Session Management

• **Token Storage:** After login, the JWT token is stored in Local Storage or Context API to maintain the user session.

• **Session Persistence:** The authentication state remains active even after refreshing the page until logout or token expiration.

• **Logout Logic:** During logout, the token is removed from the client-side storage, preventing further access to protected routes.

8.4 Backend Security Middleware - The backend server includes custom middleware to protect restricted API endpoints. Every request to protected routes undergoes the following process:

1. The token is extracted from the request header.
2. The token is verified using the JWT secret key.
3. The user role is checked for proper authorization.
4. If verification fails, the server returns a 401 Unauthorized or 403 Forbidden response.

9. User Interface

The visual identity of DocSpot is designed to be clean, modern, and high-contrast.

Register Page:

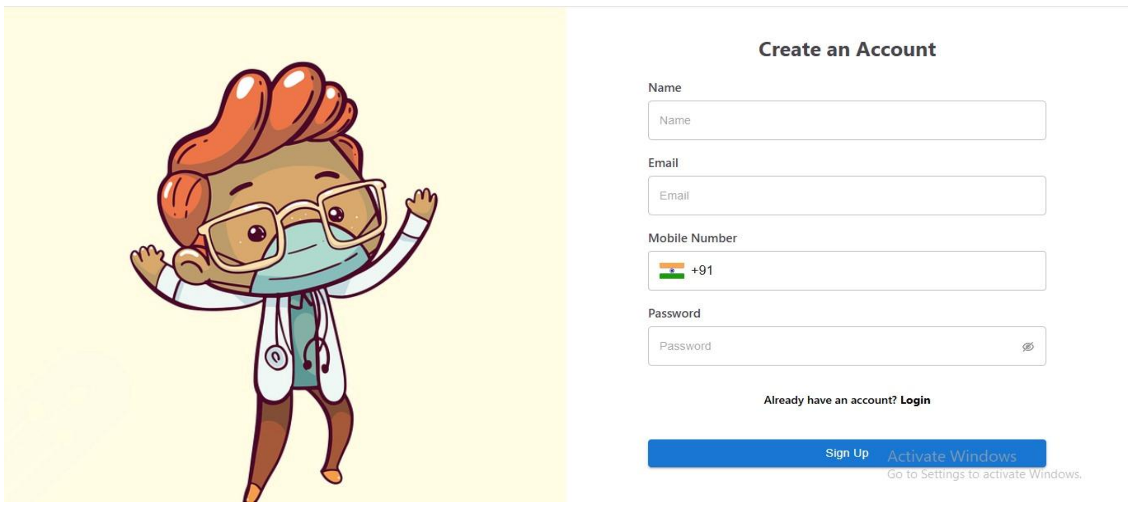
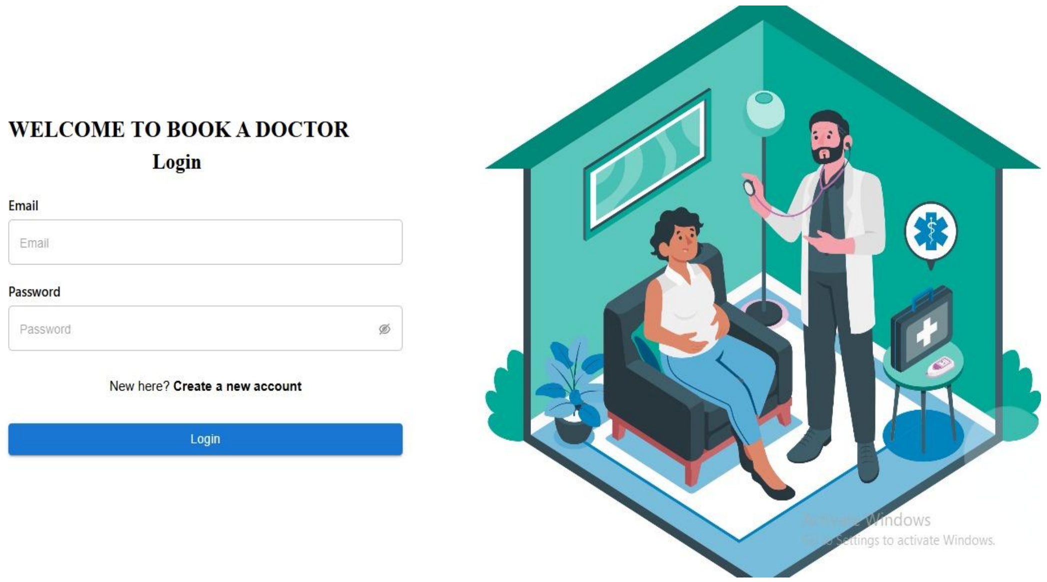


fig: Register user for Credentials

Login Page:


WELCOME TO BOOK A DOCTOR

Login

Email

Email

Password

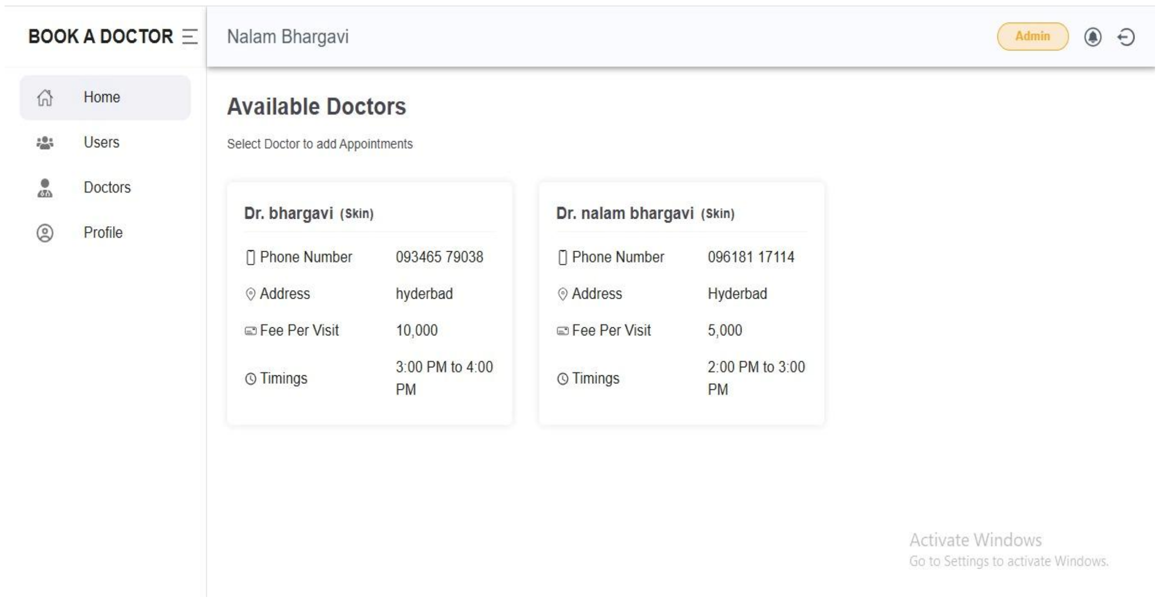
Password

New here? **Create a new account**

Login

fig: Login with credentials

Admin Dashboard:



Available Doctors

Select Doctor to add Appointments

Dr. bhargavi (Skin)

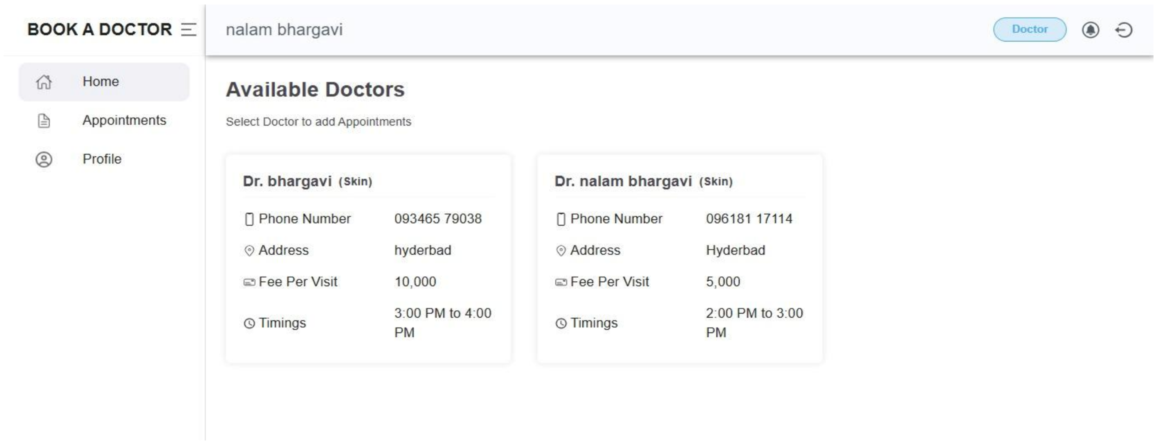
Phone Number	093465 79038
Address	hyderbad
Fee Per Visit	10,000
Timings	3:00 PM to 4:00 PM

Dr. nalam bhargavi (Skin)

Phone Number	096181 17114
Address	Hyderbad
Fee Per Visit	5,000
Timings	2:00 PM to 3:00 PM

fig: Admin dashboard

Doctor Dashboard:



Available Doctors

Select Doctor to add Appointments

Dr. bhargavi (Skin)

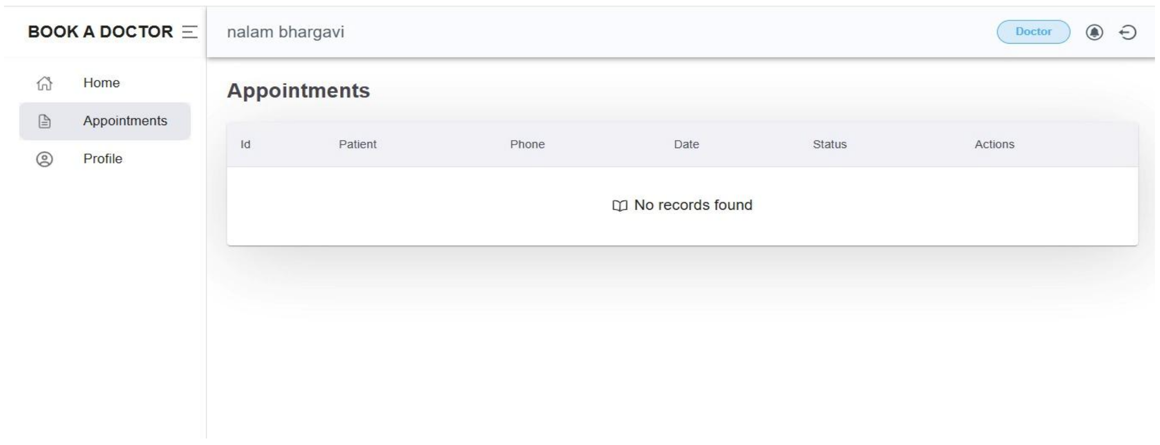
Phone Number	093465 79038
Address	hyderbad
Fee Per Visit	10,000
Timings	3:00 PM to 4:00 PM

Dr. nalam bhargavi (Skin)

Phone Number	096181 17114
Address	Hyderbad
Fee Per Visit	5,000
Timings	2:00 PM to 3:00 PM

fig: Doctor dashboard

Appointment Page :



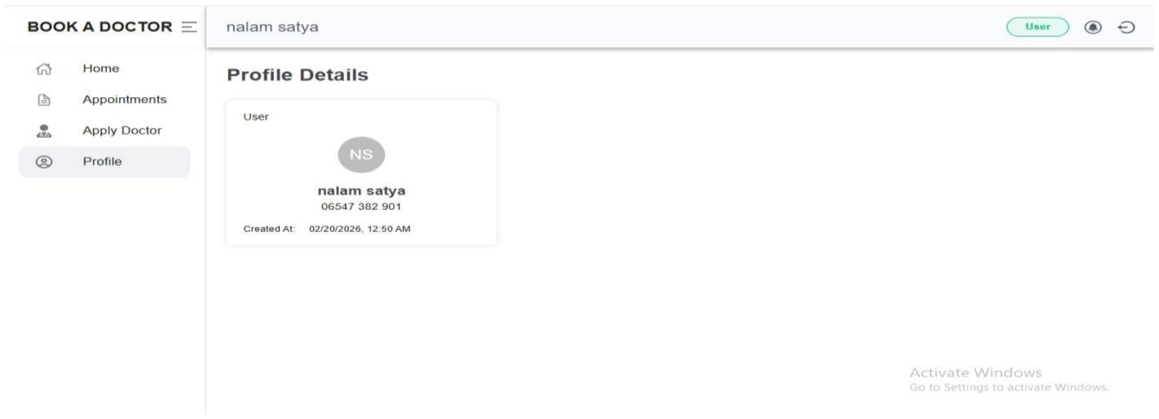
Appointments

Id	Patient	Phone	Date	Status	Actions

No records found

fig: appointment

User Dashboard:



Profile Details

User

nalam satya

06547 382 901

Created At: 02/20/2026, 12:50 AM

fig : user profile

10. Testing

Functional Testing was performed to verify that all major features of the Doctor Appointment Booking System work correctly after extracting and running the ZIP file. The application was tested in a local environment to ensure that each module performs according to the specified requirements.

11. Known Issues

□ The system has not been tested under heavy user load, so performance issues may occur when multiple users attempt to book appointments simultaneously.

□ Browser compatibility issues may arise on older or unsupported web browsers, affecting the user interface layout or functionality.

12. Future Enhancements

1. Integration of an online payment gateway to enable secure payment of consultation fees during appointment booking.
2. Implementation of email and SMS notification services to send appointment confirmations and reminders to users.
3. Development of a doctor rating and review system to help patients choose doctors based on feedback.
4. Introduction of video consultation functionality to support remote medical consultations.

13. Conclusion

The Doctor Appointment Booking System developed using the MERN stack provides an efficient and user-friendly solution for managing medical appointments online. The system simplifies the booking process for patients while enabling doctors and administrators to manage schedules effectively.

This project demonstrates the practical implementation of full-stack development using modern web technologies. With secure authentication and modular architecture, the application serves as a strong foundation for future enhancements and real-world healthcare deployment.