Full Stack Development with MERN

DocSpot: Seamless Appointment Booking for Health

Team Members:

Suma Sree Mandru – Frontend Development & UI Design

Mohammed Farhan – Backend Development & API Development

Nalam Sathvika – Database Design & Implementation

Trinadh Kurapati – Testing & System Validation

Introduction

In today's fast-paced digital world, accessing healthcare services efficiently has become a necessity. Traditional appointment booking methods often involve long waiting times, manual scheduling errors, and difficulty in managing patient records. To address these challenges, DocSpot: Seamless Appointment Booking for Health is developed as a full-stack web application using the MERN stack (MongoDB, Express.js, React.js, and Node.js).

DocSpot provides a centralized and user-friendly platform where patients can easily browse doctors, book appointments, upload medical documents, and track appointment statuses in real time. The system also enables doctors to manage their schedules efficiently and allows administrators to monitor and control overall platform operations.

The application follows a client-server architecture, ensuring smooth communication between the frontend and backend through RESTful APIs. With secure authentication and role-based access control, the system ensures data privacy and authorized access for patients, doctors, and administrators.

The primary objective of this project is to digitalize and simplify the healthcare appointment booking process, making it more accessible, reliable, and efficient for all users.

Project Overview

Purpose:

The purpose of DocSpot is to provide a digital solution for managing doctor appointments efficiently and securely. Traditional appointment booking methods often involve manual processes, long waiting times, scheduling conflicts, and lack of transparency. This project eliminates these inefficiencies by offering an automated, user-friendly, and centralized web-based platform.

The goals of the project are:

- To simplify the appointment booking process for patients
- To provide doctors with an organized system to manage schedules
- To enable administrators to monitor and control platform activities
- To ensure secure authentication and role-based access control
- To provide real-time updates on appointment status
- To create a scalable and maintainable healthcare management system using the MERN stack

Features:

Patient Features

- User registration and login with secure authentication
- Browse and view available doctors
- Filter doctors based on specialization and availability
- Book appointments by selecting date and time
- Upload medical documents during booking
- Receive confirmation and status updates
- View booking history
- Cancel or reschedule appointments

Doctor Features

- Register and apply for doctor account
- Await admin approval
- View appointment requests
- Confirm, reschedule, or cancel appointments
- Manage schedule and update appointment status
- Update consultation details

Admin Features

- Monitor overall platform activities
- Approve or reject doctor applications
- Manage users and doctors
- Maintain system integrity
- Handle operational issues

Technical Features

- RESTful API communication
- JWT-based authentication
- Secure password hashing using bcrypt
- MongoDB for scalable data storage
- Responsive UI using React, Bootstrap, and Material UI
- Dynamic data updates without page reload

Architecture

The application follows a Client-Server Architecture using the MERN stack.

Frontend Architecture

The frontend is developed using React.js with a component-based architecture.

Architecture Design

- Reusable components such as Navbar, Forms, Dashboards, and Appointment Cards
- React Router for navigation
- Axios for API communication
- Bootstrap and Material UI for responsive design
- Role-based dashboard rendering

Frontend Flow

User Interaction → Axios API Call → Backend Processing → Response → UI Update

Key Frontend Modules

- Authentication Pages
- User Dashboard
- Doctor Dashboard
- Admin Panel
- Appointment Booking Form
- Booking History Page

Backend Architecture

The backend is built using Node.js and Express.js following the MVC pattern.

Structure

- Routes – API endpoints
- Controllers – Business logic
- Models – Database schemas
- Middleware – Authentication and authorization

Backend Flow

Client Request → Route → JWT Middleware → Controller → Database → Response

Backend Features

- RESTful APIs
- JWT authentication
- Password hashing
- Role-based authorization
- Error handling middleware
- File upload handling

Database Architecture

MongoDB is used as the database with Mongoose as ODM.

Users Collection

Fields:

- _id
- name
- email
- password (hashed)
- role
- timestamps

Doctors Collection

Fields:

- userId
- specialization

- experience
- availability
- status
- timestamps

Appointments Collection

Fields:

- userId
- doctorId
- appointmentDate
- appointmentTime
- status
- documents
- timestamps

MongoDB ensures scalability, flexible schema design, and efficient data retrieval.

Setup Instructions

Prerequisites

1. Software Requirements

- **Node.js** (v16 or higher)
- **npm** (comes with Node.js)
- **MongoDB Atlas account** (for cloud database connection)
- **Git** (for cloning the repository)
- **Code Editor** (e.g., VS Code)

2. Optional Tools

- Postman (for API testing)
- MongoDB Compass (for database visualization)

Installation

Step 1: Clone the Repository

Open terminal or command prompt and run:

```
git clone <repository-link>
```

Navigate into the project directory:

```
cd docspot
```

Step 2: Install Backend Dependencies

Navigate to the server folder:

```
cd server
```

Install required dependencies:

```
npm install
```

Step 3: Configure Environment Variables

Inside the server folder, create a `.env` file and add the following:

```
PORT=5000
MONGO_URI=your_mongodb_connection_string
JWT_SECRET=your_secret_key
```

Replace:

- `your_mongodb_connection_string` with your MongoDB Atlas connection string.
- `your_secret_key` with a secure random string.

Step 4: Install Frontend Dependencies

Open a new terminal and navigate to the client folder:

```
cd client
```

Install frontend dependencies:

```
npm install
```

Step 5: Start the Application

Start Backend Server:

```
cd server
npm start
```

Backend runs on:

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

Start Frontend Application:

```
cd client
npm start
```

Frontend runs on:

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

After completing these steps, the application will be successfully running on your local system.

Folder Structure

Client Folder Structure

```
client/
|
├── public/
|   └── index.html
|
├── src/
```

```
│   ├── components/
│   ├── pages/
│   ├── redux/ (or context/)
│   ├── hooks/
│   ├── utils/
│   ├── assets/
│   ├── App.js
│   └── index.js
│
└── package.json
```

Server Folder Structure

```
server/
│
├── config/
├── controllers/
├── models/
├── routes/
├── middleware/
├── utils/
├── uploads/ (if file upload is used)
├── server.js
└── package.json
```

Running the Application

Starting the Backend Server

1. Open a terminal.
2. Navigate to the server directory:

   ```
   cd server
   ```

3. Start the backend server:

   ```
   npm start
   ```

If configured correctly, the backend will run on:

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

Starting the Frontend Server

1. Open a new terminal window.
2. Navigate to the client directory:

```
cd client
```

3. Start the React application:

```
npm start
```

The frontend application will run on:

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

API Documentation

Base URL:

http://localhost:5000/api

Authentication APIs

User registration and login with JWT token generation

Doctor APIs

Doctor application, listing approved doctors, admin approval

Appointment APIs

Book appointment, get appointments, update appointment status

Admin APIs

View and manage doctor applications

All protected routes require Authorization: Bearer <JWT_TOKEN>

Authentication

Authentication is implemented using JWT and bcrypt.

- Passwords are hashed before storing
- JWT generated on login

- Token stored on client side
- Token attached to request headers
- Middleware verifies token
- Role-based access control enforced

Security measures include:

- Encrypted passwords
- Secure JWT signing
- Environment variable protection
- HTTP 401/403 for unauthorized access

User Interface

The UI is responsive, modern, and role-based.

Includes:

- Login Page
- Registration Page
- Patient Dashboard
- Appointment Booking Form
- Booking History
- Doctor Dashboard
- Admin Panel

Designed using React, Bootstrap, and Material UI.

Testing

Manual and functional testing was performed.

Frontend Testing

- Form validation
- Navigation testing
- Role-based rendering
- Appointment workflow

Backend Testing

- API response validation
- JWT verification
- CRUD operations
- Error handling

Database Testing

- Data storage validation
- Password hashing verification
- Relationship consistency

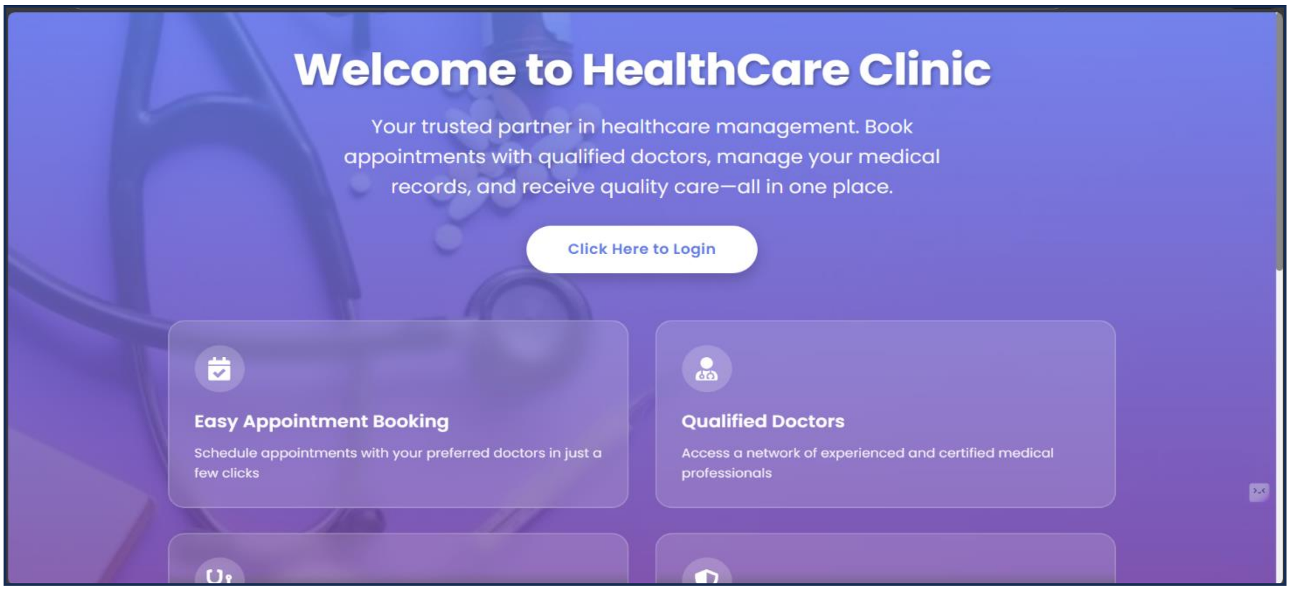
Integration Testing

- Axios API communication
- Real-time UI updates
- Data consistency

All core functionalities were validated successfully.

Screenshots

Dashboard



Welcome to HealthCare Clinic

Your trusted partner in healthcare management. Book appointments with qualified doctors, manage your medical records, and receive quality care—all in one place.

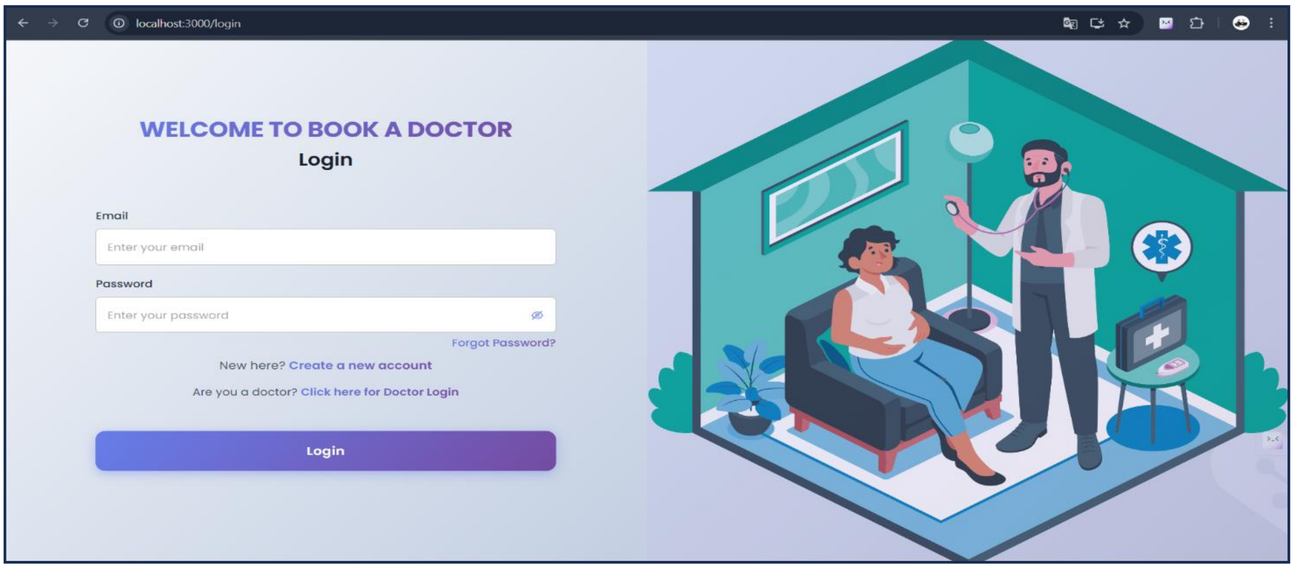
Easy Appointment Booking

Schedule appointments with your preferred doctors in just a few clicks

Qualified Doctors

Access a network of experienced and certified medical professionals

Login Page



localhost:3000/login

WELCOME TO BOOK A DOCTOR

Login

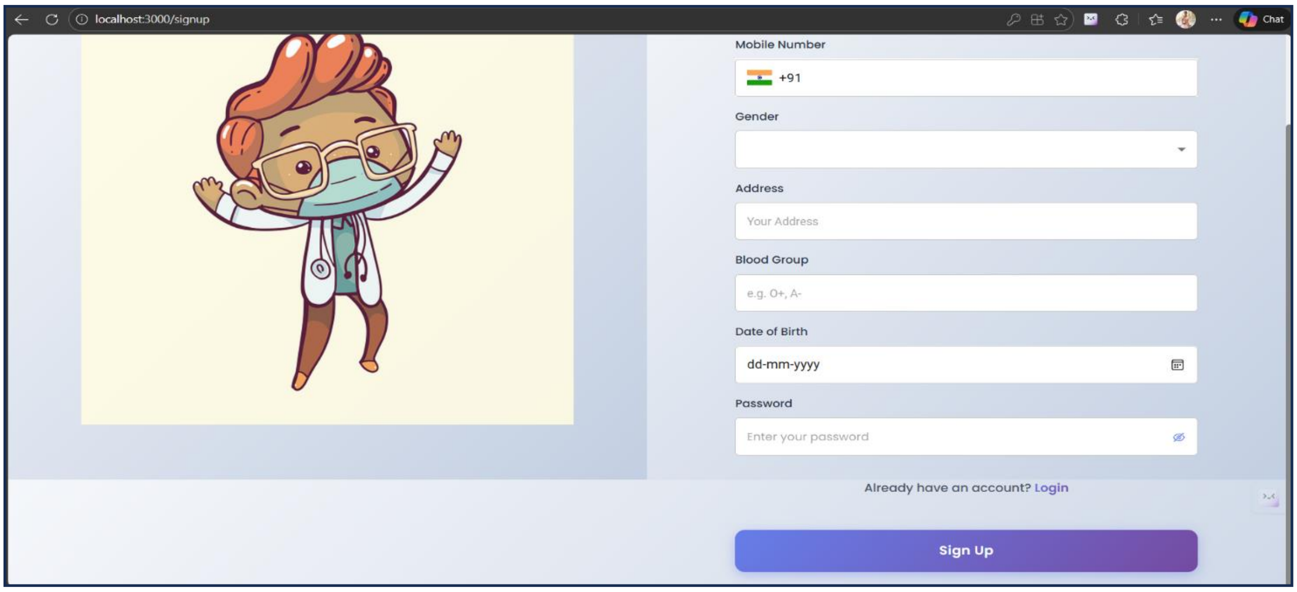
Email: Enter your email

Password: Enter your password

New here? Create a new account

Are you a doctor? Click here for Doctor Login

Registration Page



localhost:3000/signup

Mobile Number: +91

Gender:

Address: Your Address

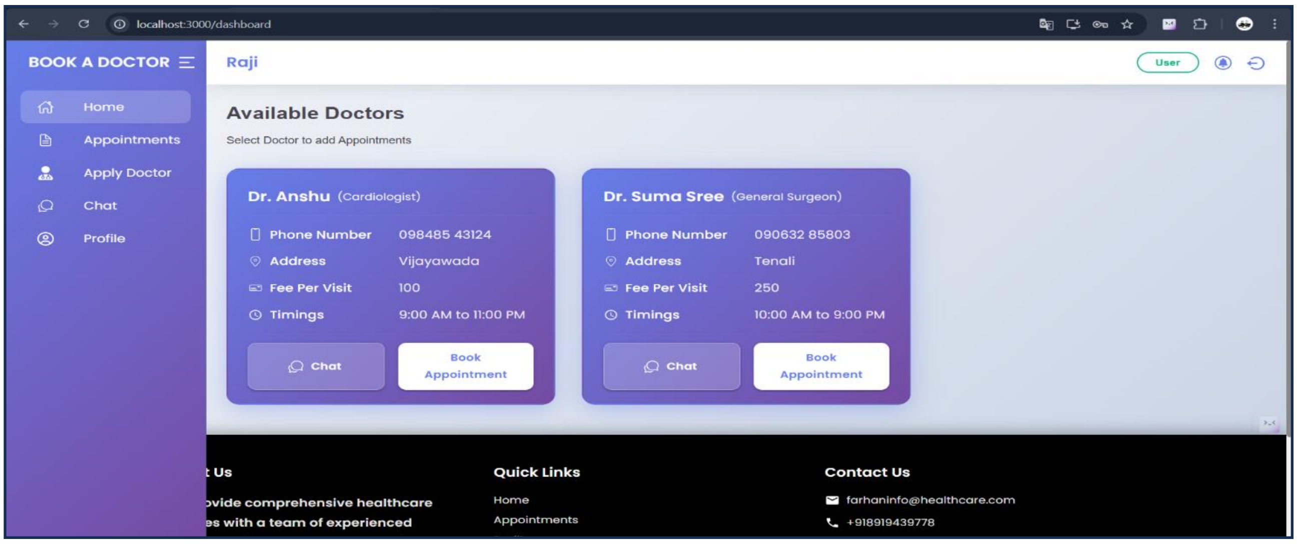
Blood Group: e.g. O+, A-

Date of Birth: dd-mm-yyyy

Password: Enter your password

Already have an account? Login

Patient Dashboard



Available Doctors

Select Doctor to add Appointments

Dr. Anshu (Cardiologist)

Phone Number	098485 43124
Address	Vijayawada
Fee Per Visit	100
Timings	9:00 AM to 11:00 PM

Chat | Book Appointment

Dr. Suma Sree (General Surgeon)

Phone Number	090632 85803
Address	Tenali
Fee Per Visit	250
Timings	10:00 AM to 9:00 PM

Chat | Book Appointment

t Us

ovide comprehensive healthcare es with a team of experienced

Quick Links

Home

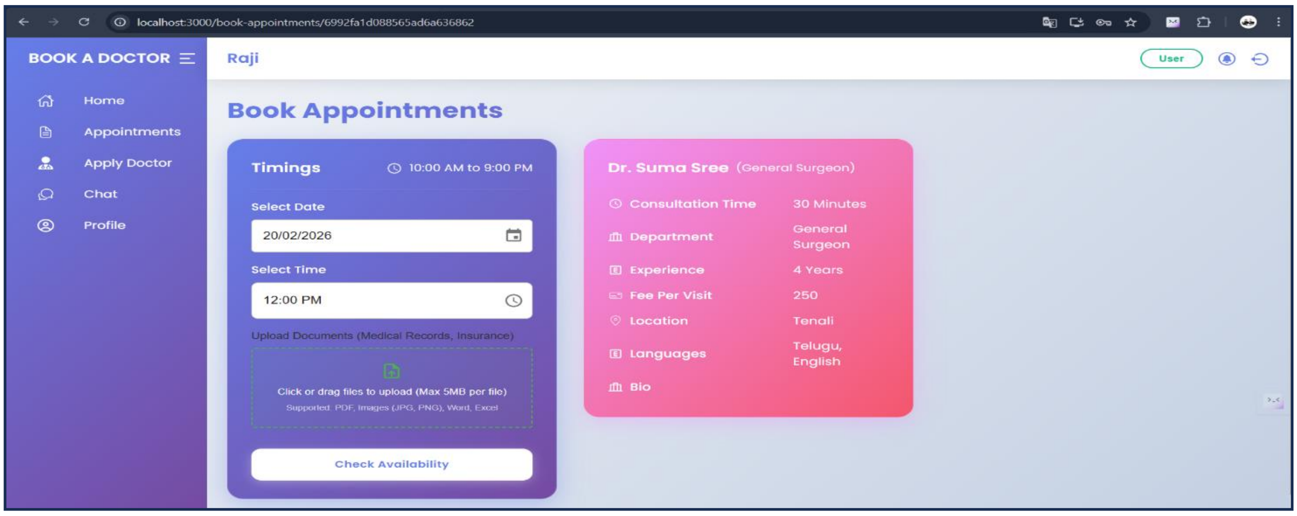
Appointments

Contact Us

farhaninfo@healthcare.com

+918919439778

Appointment Booking Form



Book Appointments

Timings 10:00 AM to 9:00 PM

Select Date: 20/02/2026

Select Time: 12:00 PM

Upload Documents (Medical Records, Insurance)

Click or drag files to upload (Max 5MB per file)

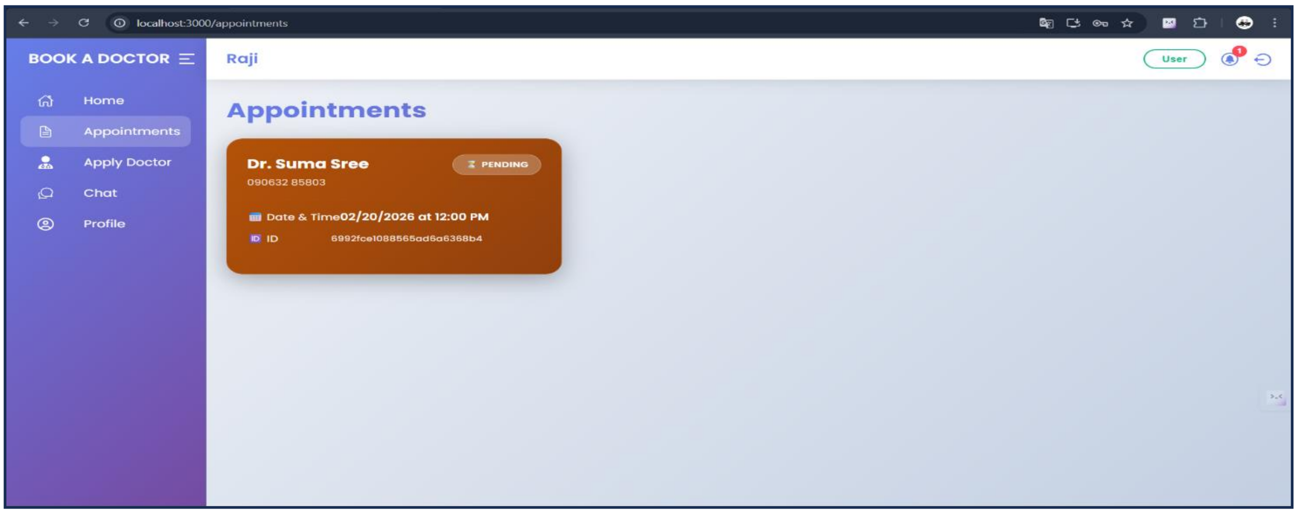
Supported: PDF, Images (JPG, PNG), Word, Excel

Check Availability

Dr. Suma Sree (General Surgeon)

Consultation Time	30 Minutes
Department	General Surgeon
Experience	4 Years
Fee Per Visit	250
Location	Tenali
Languages	Telugu, English
Bio	

Booking History



Appointments

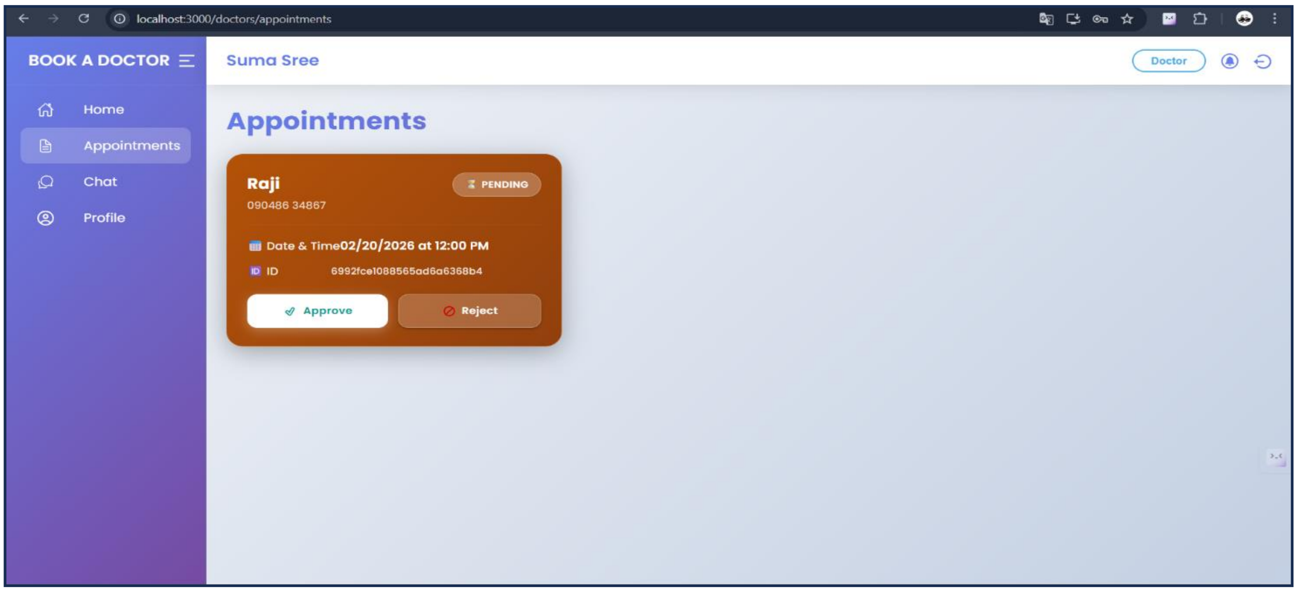
Dr. Suma Sree PENDING

090632 85803

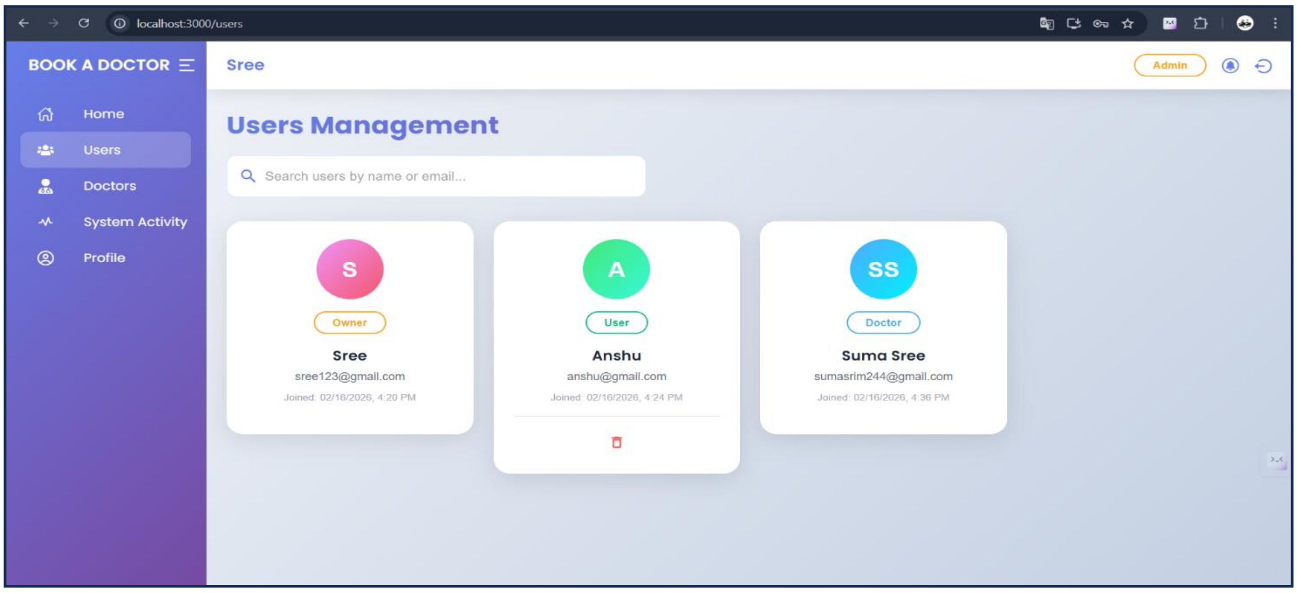
Date & Time02/20/2026 at 12:00 PM

ID 6992fce1088565ad6a6368b4

Doctor Dashboard



Admin Panel



Known Issues

No Real-Time Push Notifications

The system currently does not support real-time push notifications. Users must manually check their dashboard to view appointment updates such as confirmation, cancellation, or rescheduling. Instant notification mechanisms like WebSockets or push services are not implemented.

No Email or SMS Alerts
Appointment confirmations, reminders, and status updates are not sent via email or SMS. All notifications are limited to in-application updates, which may reduce communication effectiveness if users do not log in frequently.

No Online Payment Integration
The platform does not include payment gateway integration. Patients cannot pay consultation fees online during the booking process, and payments must be handled offline at the clinic.

Basic Error Messages
Some backend validation and system errors return generic messages. More descriptive and user-friendly error handling could improve user experience and debugging clarity.

No Load Balancing or Caching
The application does not implement advanced performance optimization techniques such as load balancing or caching. Under heavy user traffic, performance may degrade due to lack of scalability mechanisms.

Manual Testing Only
The project relies solely on manual testing. Automated testing frameworks such as Jest, Mocha, or integration testing tools have not been implemented, which may limit long-term maintainability and regression testing efficiency.

No Token Refresh Mechanism
JWT authentication is implemented without refresh tokens. When a token expires, users are required to log in again manually, which may interrupt the user experience.

Limited File Validation
If document upload functionality is used, file validation is basic. Advanced validation such as strict file type checking, file size limits, and secure cloud storage integration is not fully implemented.

Minor Mobile UI Adjustments Needed
Although the application is responsive, certain layouts and spacing elements may require further optimization for smaller screen devices to improve visual alignment and user interaction on mobile platforms.

Future Enhancements

Online Payment Integration

Integrate secure payment gateways (e.g., credit/debit cards, UPI, net banking) to allow patients to pay consultation fees online during appointment booking.

Benefits:

- Reduces no-shows
- Provides secure transaction records
- Improves convenience for users

Email and SMS Notifications

Implement automated notifications for:

- Appointment confirmation
- Reminder before appointment
- Cancellation updates
- Doctor approval status

This can be achieved using services like email APIs or SMS gateways.

Video Consultation Feature

Add telemedicine support through:

- Video calling integration
- Secure online consultation rooms
- Digital prescription sharing

This would enable remote healthcare access.

Rating and Review System

Allow patients to:

- Rate doctors
- Provide feedback
- View reviews before booking

This improves transparency and trust.

Advanced Filtering and Search

Enhance doctor search functionality by adding:

- Location-based filtering
- Availability calendar view
- Experience-based filtering
- Fee range filtering

Admin Analytics Dashboard

Develop a detailed analytics panel for administrators showing:

- Number of active users
- Appointment statistics
- Doctor performance metrics
- Monthly booking trends

Mobile Application

Develop a cross-platform mobile application using:

- React Native or Flutter

This would improve accessibility and user engagement.

Cloud File Storage Integration

Integrate cloud storage services for document uploads to:

- Improve scalability
- Enhance security
- Reduce server storage dependency

Automated Testing Implementation

Introduce automated testing using:

- Jest (Frontend testing)
- Mocha/Chai (Backend testing)

This would improve code reliability and maintainability.

Token Refresh Mechanism

Implement refresh tokens to:

- Improve user session management
- Prevent frequent login interruptions

These enhancements would transform DocSpot into a more scalable, production-ready healthcare platform capable of handling large user bases and real-world medical service requirements.