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

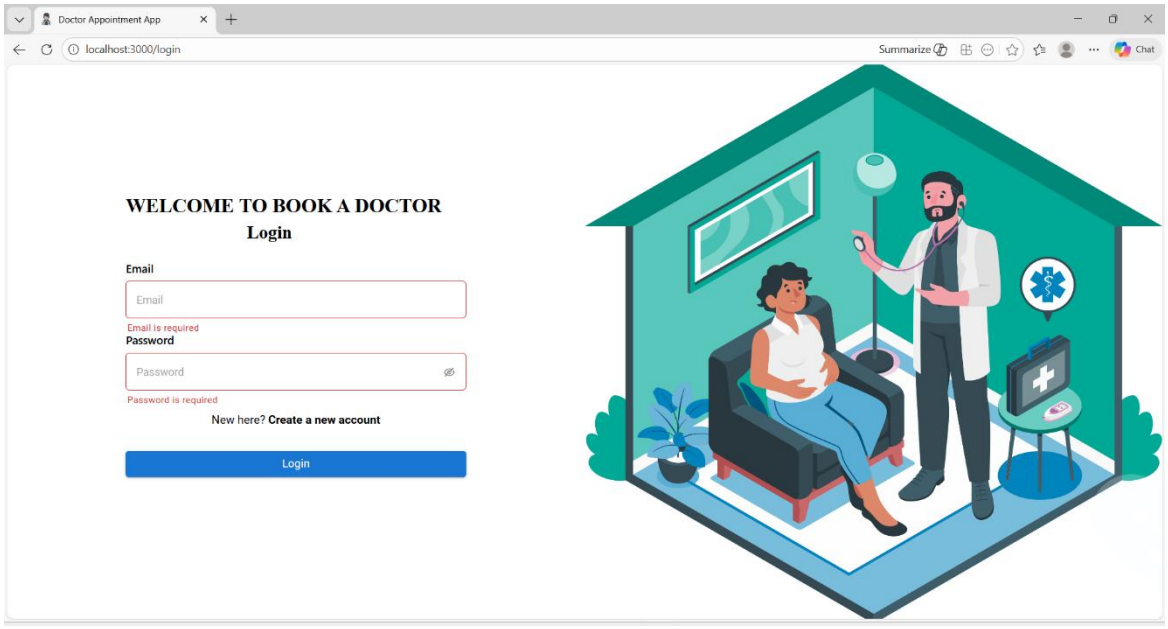
Error handling tested for:

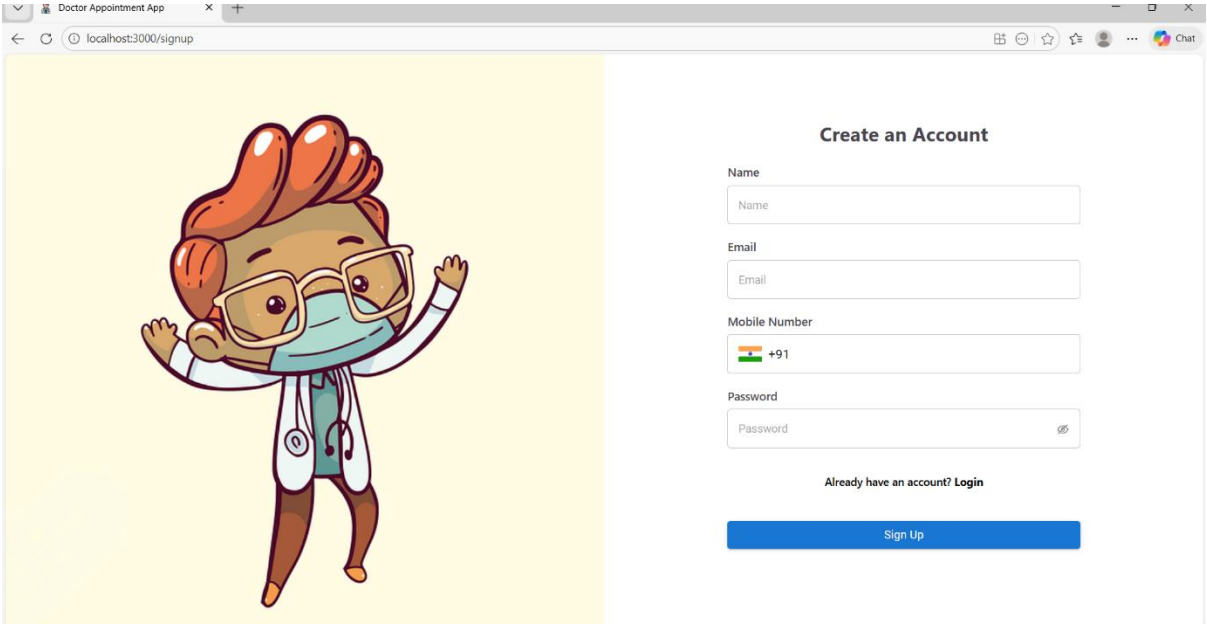
- Invalid credentials
- Unauthorized access
- Invalid form inputs

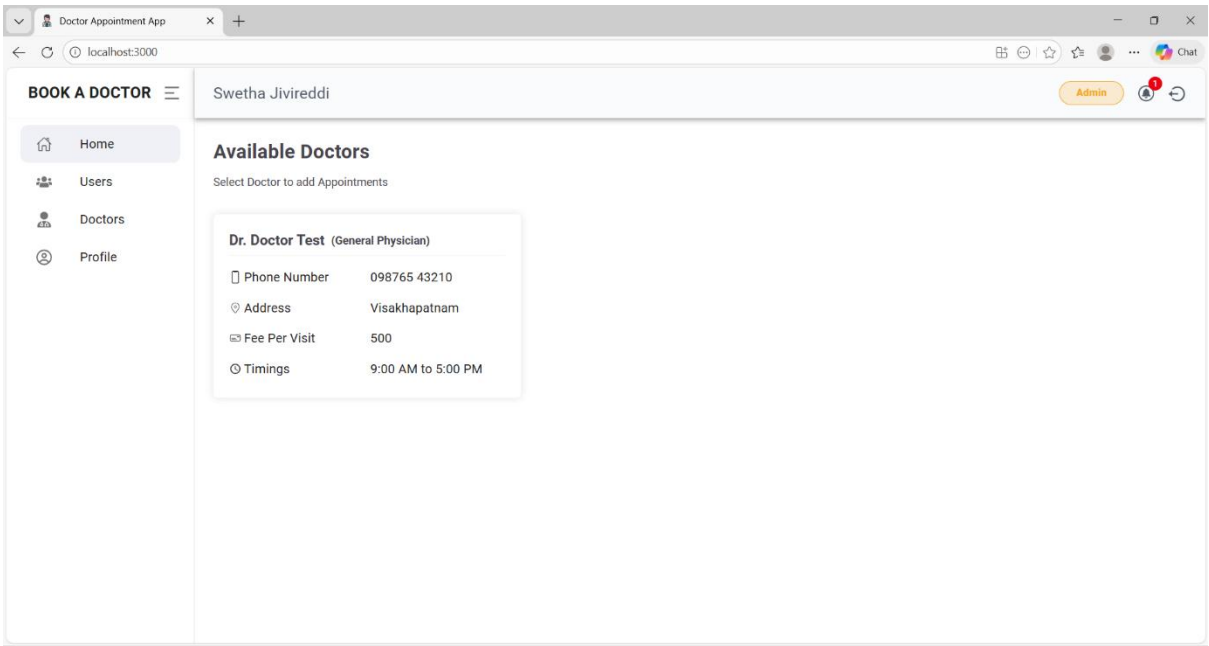
11) Screenshots or Demo

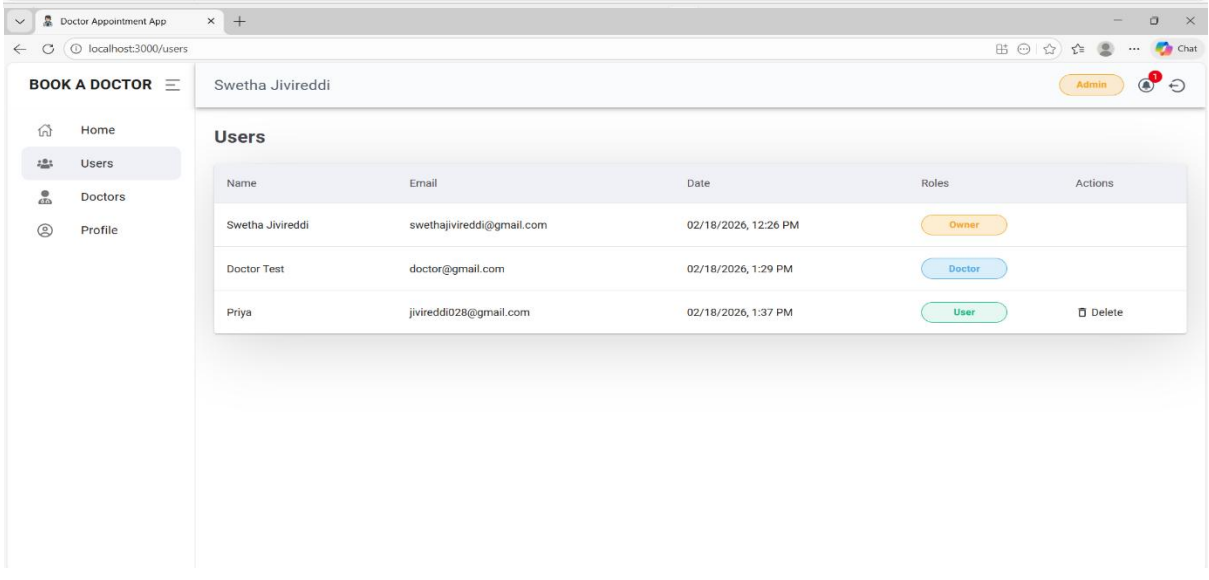
Include:

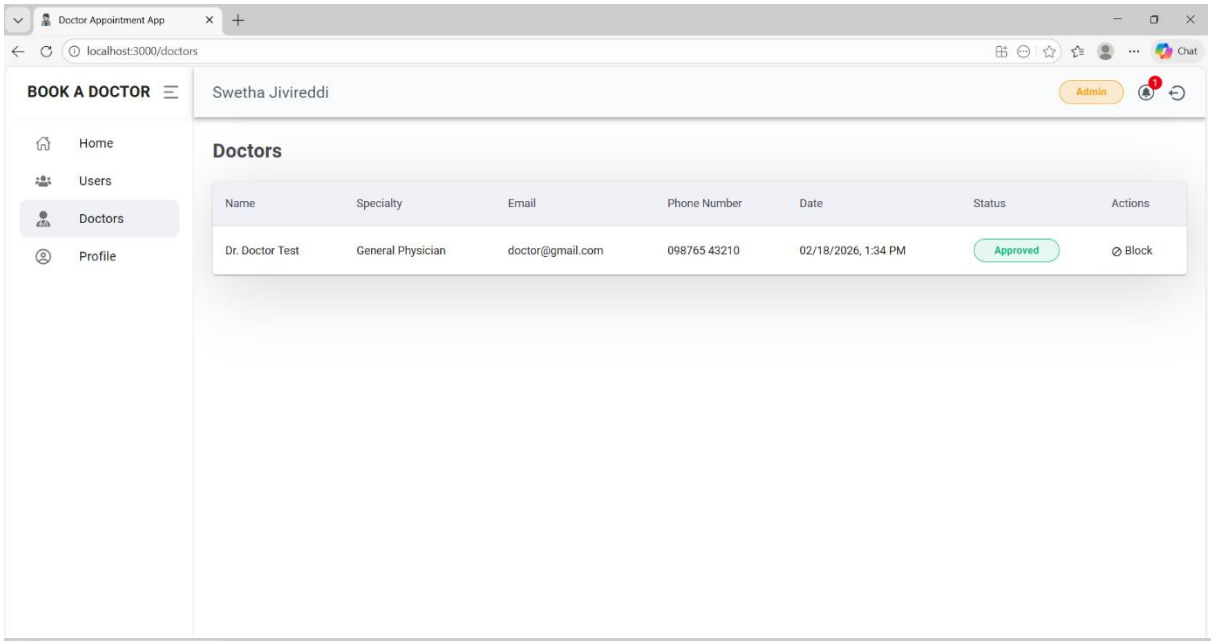
- Login Page Screenshot
- Doctor Registration Screenshot
- Admin Approval Screenshot
- Booking Appointment Screenshot
- MongoDB Collections Screenshot

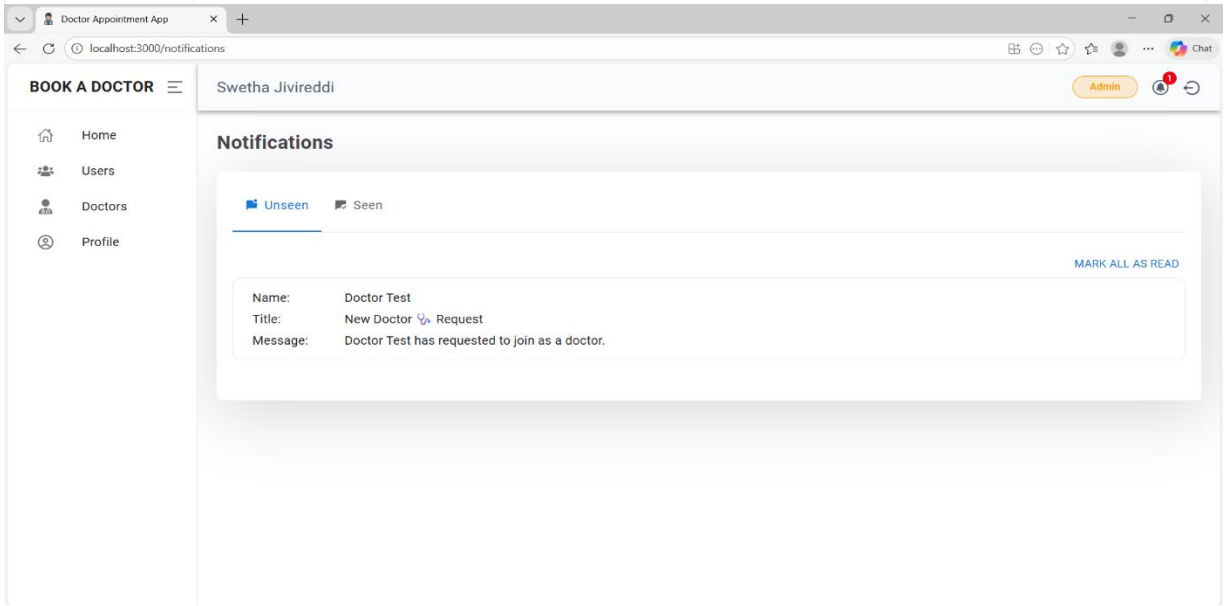


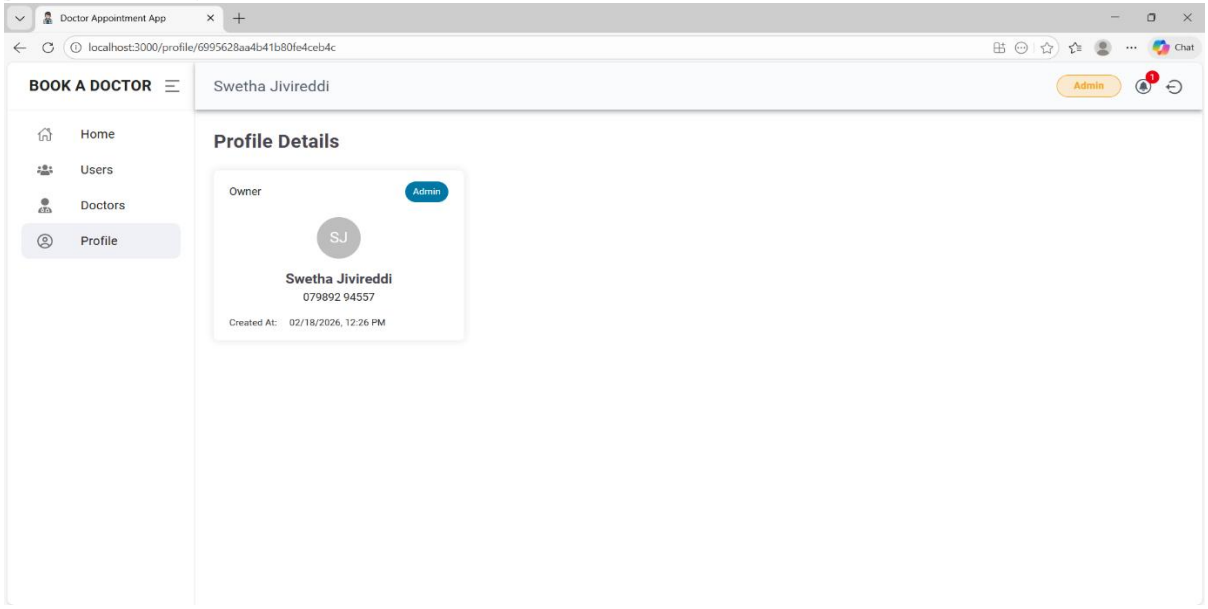

Doctor Appointment App
localhost:3000/signup
Chat
Create an Account
Name
Name
Email
Email
Mobile Number
+91
Password
Password
Already have an account? Login
Sign Up

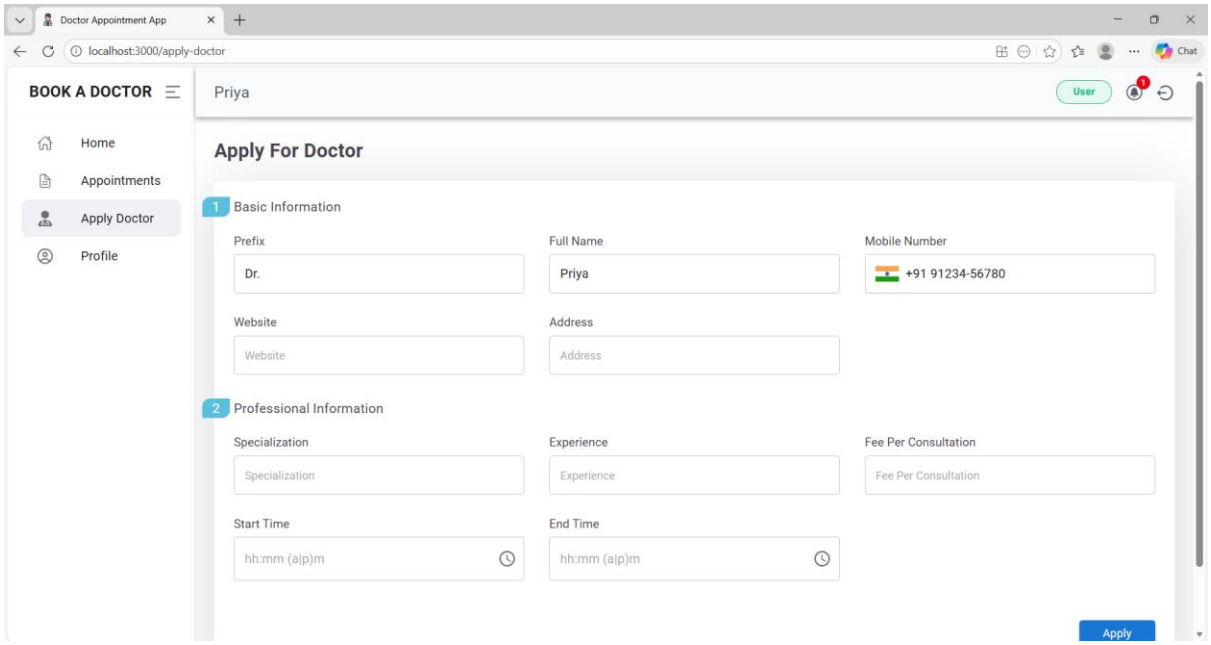

Doctor Appointment App
localhost:3000
Chat
BOOK A DOCTOR
Swetha Jivireddi
Admin
Home
Users
Doctors
Profile
Available Doctors
Select Doctor to add Appointments
Dr. Doctor Test (General Physician)
Phone Number 098765 43210
Address Visakhapatnam
Fee Per Visit 500
Timings 9:00 AM to 5:00 PM

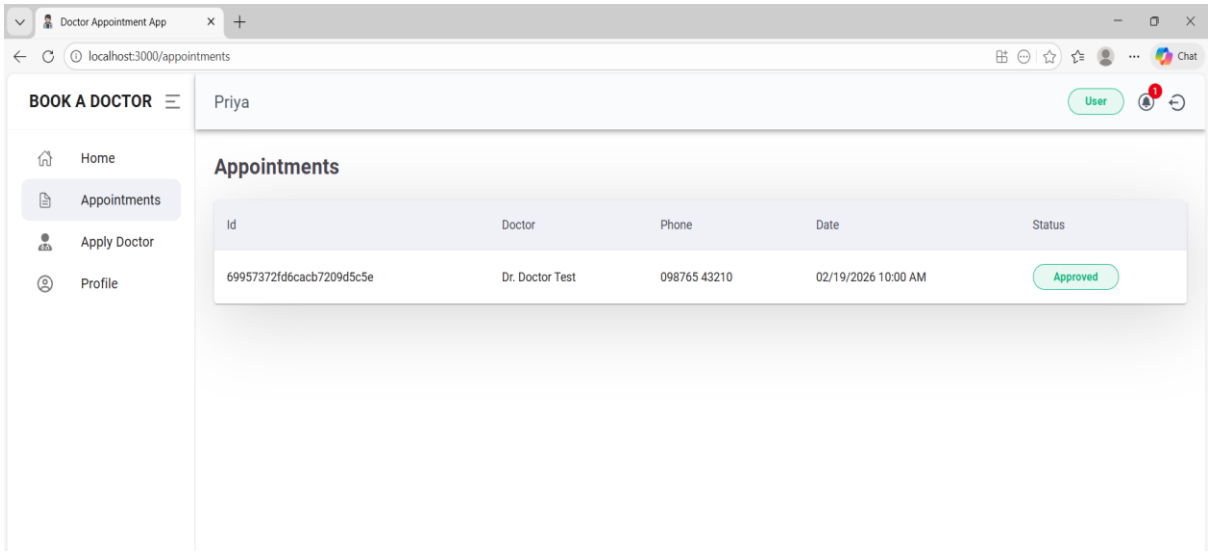

Doctor Appointment App
localhost:3000/users
Chat
BOOK A DOCTOR
Swetha Jivireddi
Admin
Home
Users
Doctors
Profile
Users
Name
Email
Date
Roles
Actions
Swetha Jivireddi
swethajivireddi@gmail.com
02/18/2026, 12:26 PM
Owner
Doctor Test
doctor@gmail.com
02/18/2026, 1:29 PM
Doctor
Priya
jivireddi028@gmail.com
02/18/2026, 1:37 PM
User
Delete

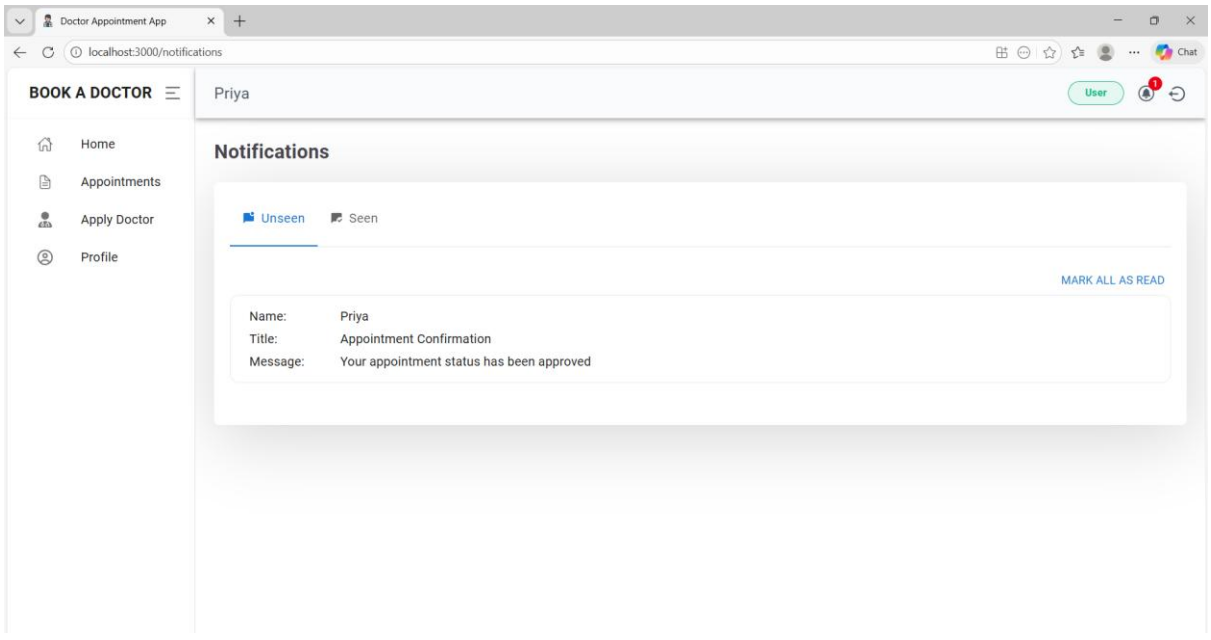

Doctor Appointment App
localhost:3000/doctors
BOOK A DOCTOR
Swetha Jivireddi
Admin
Home
Users
Doctors
Profile
Doctors
Name
Specialty
Email
Phone Number
Date
Status
Actions
Dr. Doctor Test
General Physician
doctor@gmail.com
098765 43210
02/18/2026, 1:34 PM
Approved
Block

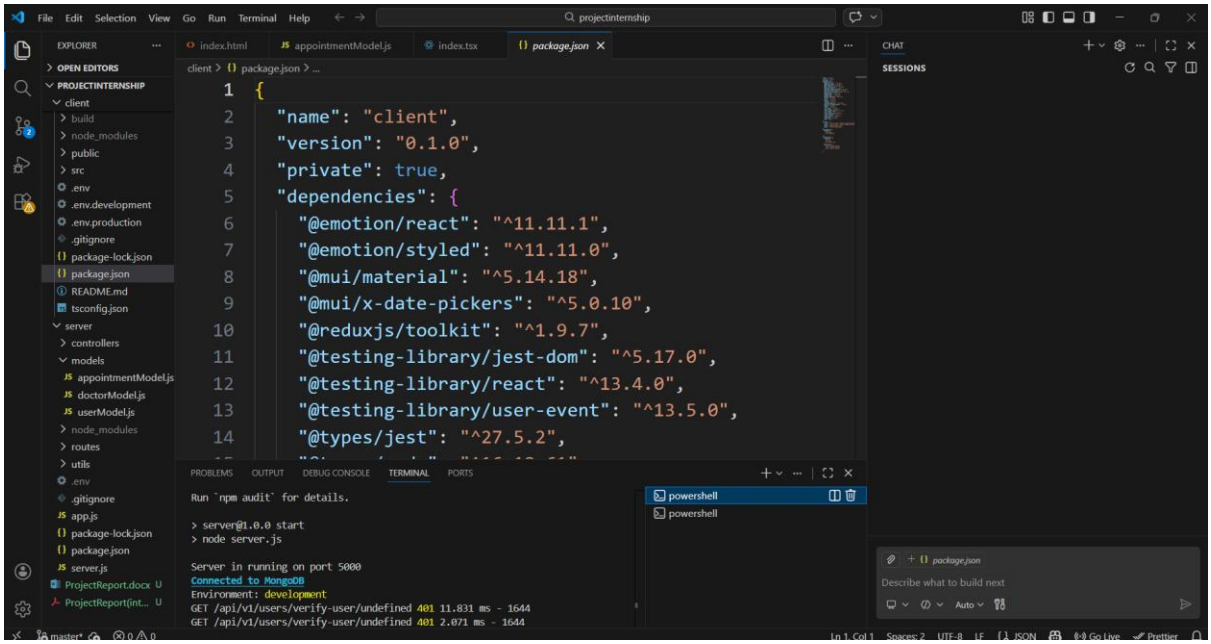

Doctor Appointment App
localhost:3000/notifications
BOOK A DOCTOR
Swetha Jivireddi
Admin
Home
Users
Doctors
Profile
Notifications
Unseen
Seen
MARK ALL AS READ
Name: Doctor Test
Title: New Doctor Request
Message: Doctor Test has requested to join as a doctor.

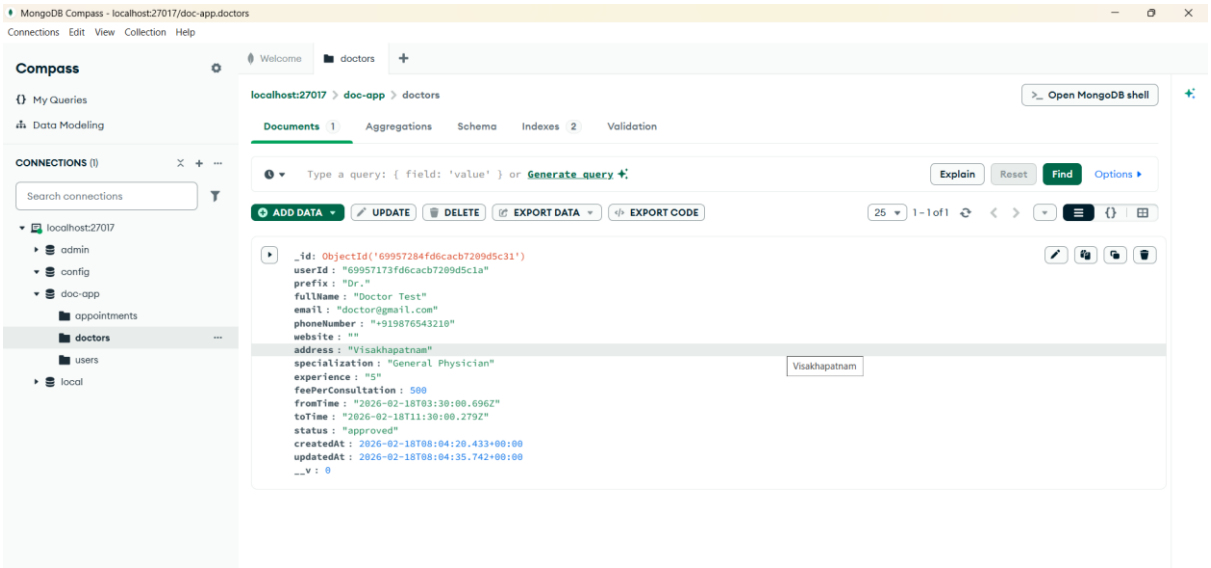

Doctor Appointment App
localhost:3000/profile/6995628aa4b41b80fe4ceb4c
BOOK A DOCTOR
Swetha Jivireddi
Admin
Home
Users
Doctors
Profile
Profile Details
Owner
Admin
SJ
Swetha Jivireddi
079892 94557
Created At: 02/18/2026, 12:26 PM

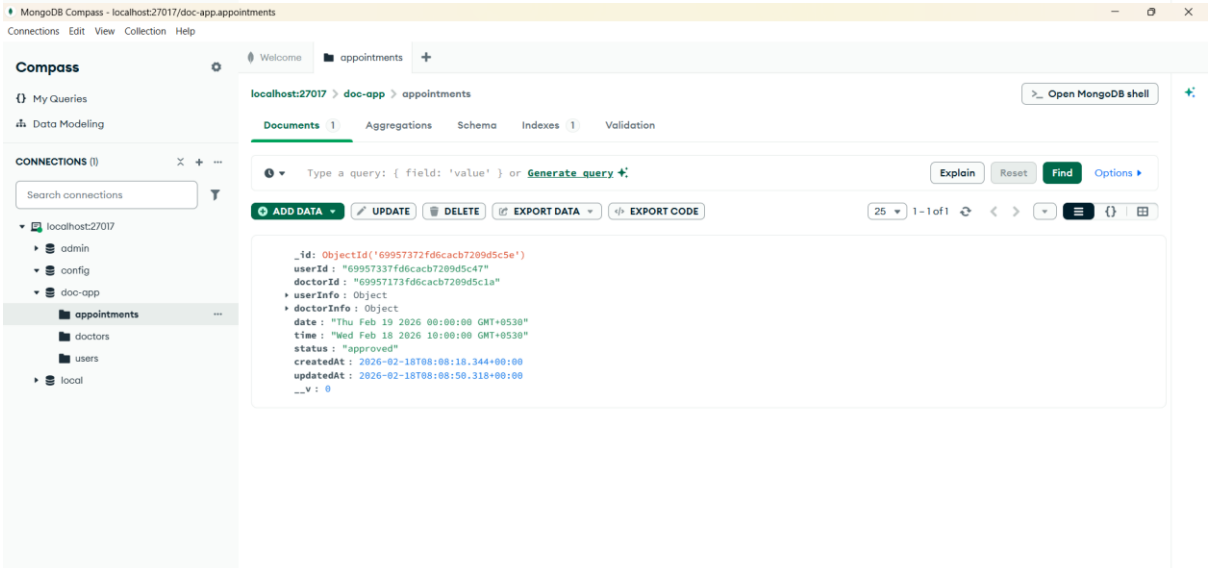

Doctor Appointment App
localhost:3000/apply-doctor
BOOK A DOCTOR
Priya
User
Home
Appointments
Apply Doctor
Profile
Apply For Doctor
1 Basic Information
Prefix
Dr.
Full Name
Priya
Mobile Number
+91 91234-56780
Website
Address
2 Professional Information
Specialization
Experience
Fee Per Consultation
Start Time
End Time
hh:mm (a|p)m
Apply

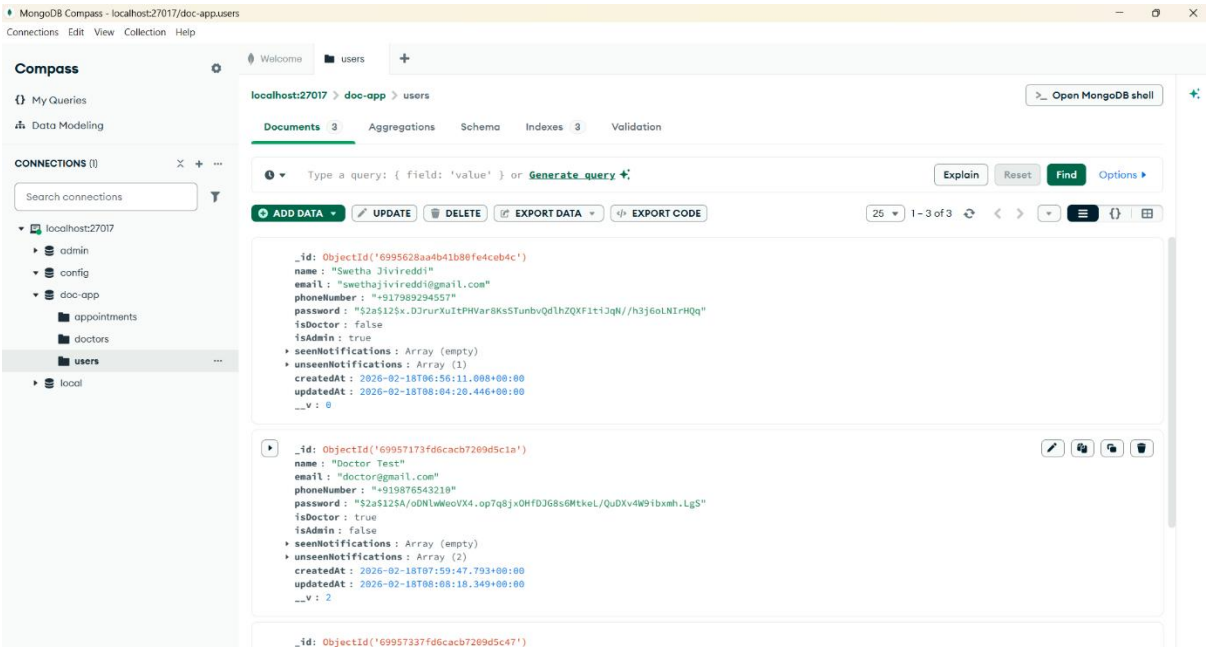

Doctor Appointment App
localhost:3000/appointments
BOOK A DOCTOR
Priya
User
Home
Appointments
Apply Doctor
Profile
Appointments
Id
Doctor
Phone
Date
Status
69957372fd6cacb7209d5c5e
Dr. Doctor Test
098765 43210
02/19/2026 10:00 AM
Approved


Doctor Appointment App
localhost:3000/notifications
BOOK A DOCTOR
Priya
User
Home
Appointments
Apply Doctor
Profile
Notifications
Unseen
Seen
MARK ALL AS READ
Name: Priya
Title: Appointment Confirmation
Message: Your appointment status has been approved


File Edit Selection View Go Run Terminal Help
projectinternship
EXPLORER
OPEN EDITORS
PROJECTINTERNSHIP
client
server
index.html
appointmentModel.js
index.tsx
package.json
CHAT
SESSIONS
{
"name": "client",
"version": "0.1.0",
"private": true,
"dependencies": {
"@emotion/react": "^11.11.1",
"@emotion/styled": "^11.11.0",
"@mui/material": "^5.14.18",
"@mui/x-date-pickers": "^5.0.10",
"@reduxjs/toolkit": "^1.9.7",
"@testing-library/jest-dom": "^5.17.0",
"@testing-library/react": "^13.4.0",
"@testing-library/user-event": "^13.5.0",
"@types/jest": "^27.5.2",
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run `npm audit` for details.
> server@1.0.0 start
> node server.js
Server in 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
powershell
powershell
Describe what to build next


MongoDB Compass - localhost:27017/doc-app.doctors
Connections Edit View Collection Help
Compass
My Queries
Data Modeling
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
doc-app
appointments
doctors
users
local
Welcome
doctors
localhost:27017 > doc-app > doctors
Open MongoDB shell
Documents 1
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
UPDATE
DELETE
EXPORT DATA
EXPORT CODE
1 - 1 of 1
_id: ObjectId('69957284fd6cacb7209d5c31')
userId : "69957173fd6cacb7209d5c1a"
prefix : "Dr."
fullName : "Doctor Test"
email : "doctor@gmail.com"
phoneNumber : "+919876543210"
website : ""
address : "Visakhapatnam"
specialization : "General Physician"
experience : "5"
feePerConsultation : 500
fromTime : "2026-02-18T03:30:00.696Z"
toTime : "2026-02-18T11:30:00.279Z"
status : "approved"
createdAt : 2026-02-18T08:04:20.433+00:00
updatedAt : 2026-02-18T08:04:35.742+00:00
__v : 0
Visakhapatnam


MongoDB Compass - localhost:27017/doc-app.appointments
Connections Edit View Collection Help
Compass
My Queries
Data Modeling
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
doc-app
appointments
doctors
users
local
Welcome
appointments
localhost:27017 > doc-app > appointments
Open MongoDB shell
Documents 1
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
UPDATE
DELETE
EXPORT DATA
EXPORT CODE
1 - 1 of 1
_id: ObjectId('69957372fd6cacb7209d5c5e')
userId : "69957337fd6cacb7209d5c47"
doctorId : "69957173fd6cacb7209d5c1a"
userInfo : Object
doctorInfo : Object
date : "Thu Feb 19 2026 00:00:00 GMT+0530"
time : "Wed Feb 18 2026 10:00:00 GMT+0530"
status : "approved"
createdAt : 2026-02-18T08:08:18.344+00:00
updatedAt : 2026-02-18T08:08:50.318+00:00
__v : 0



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.