Project Documentation – DocSpot

1. Introduction

• Project Title: DocSpot – Seamless Doctor Appointment for Health

• Team Members:
- Vishnupriya Jangati - Full Stack Developer
- Karishma Dudekula - Full Stack Developer

2. Project Overview

• Purpose: DocSpot aims to simplify and digitize the process of booking doctor appointments. The platform enables patients to easily find and book available doctors, while allowing doctors to manage their schedules efficiently.

• Features:
- User & Doctor Registration/Login
- Doctor Profile & Availability
- Appointment Booking & Viewing
- Admin Dashboard
- Notifications via Email

3. Architecture

• Frontend: Built with React.js using React Router for navigation, styled with Material UI and Bootstrap.

• Backend: Node.js + Express.js with modular MVC structure. RESTful APIs built with secure JWT auth.

• Database: MongoDB Atlas with collections for users, doctors, and appointments. Mongoose used for schema validation.

4. Setup Instructions

• Prerequisites:
- Node.js v18+
- MongoDB Atlas cluster
- Git

• Installation:
```bash
git clone https://github.com/Jangativishnupriya/docspot.git
```

```
cd docspot

# Setup Backend
cd server
npm install
cp .env.example .env  # fill in your MongoDB URI, JWT key
npm start

# Setup Frontend
cd ../client
npm install
npm start
```

5. Folder Structure

• Client (React):

```
client/
├── src/
│   ├── components/
│   ├── pages/
│   ├── context/
│   └── App.js
```

• Server (Node.js):

```
server/
├── controllers/
├── routes/
├── models/
├── middleware/
├── utils/
└── server.js
```

6. Running the Application

• Frontend: `npm start` in the `client` directory
• Backend: `npm start` in the `server` directory

7. API Documentation

Example:

```
POST /api/appointments
{
  "doctorId": "64ab12...",
```

```
  "userId": "65cd34...",
  "date": "2025-06-26",
  "time": "10:00 AM"
}
```

Key Endpoints:
- POST /api/auth/register
- POST /api/auth/login
- GET /api/doctors
- POST /api/appointments
- GET /api/admin/users

8. Authentication

• JWT-based authentication and role-based access.
• Tokens are sent via HTTP headers.
• bcrypt.js used to hash passwords securely.

9. User Interface

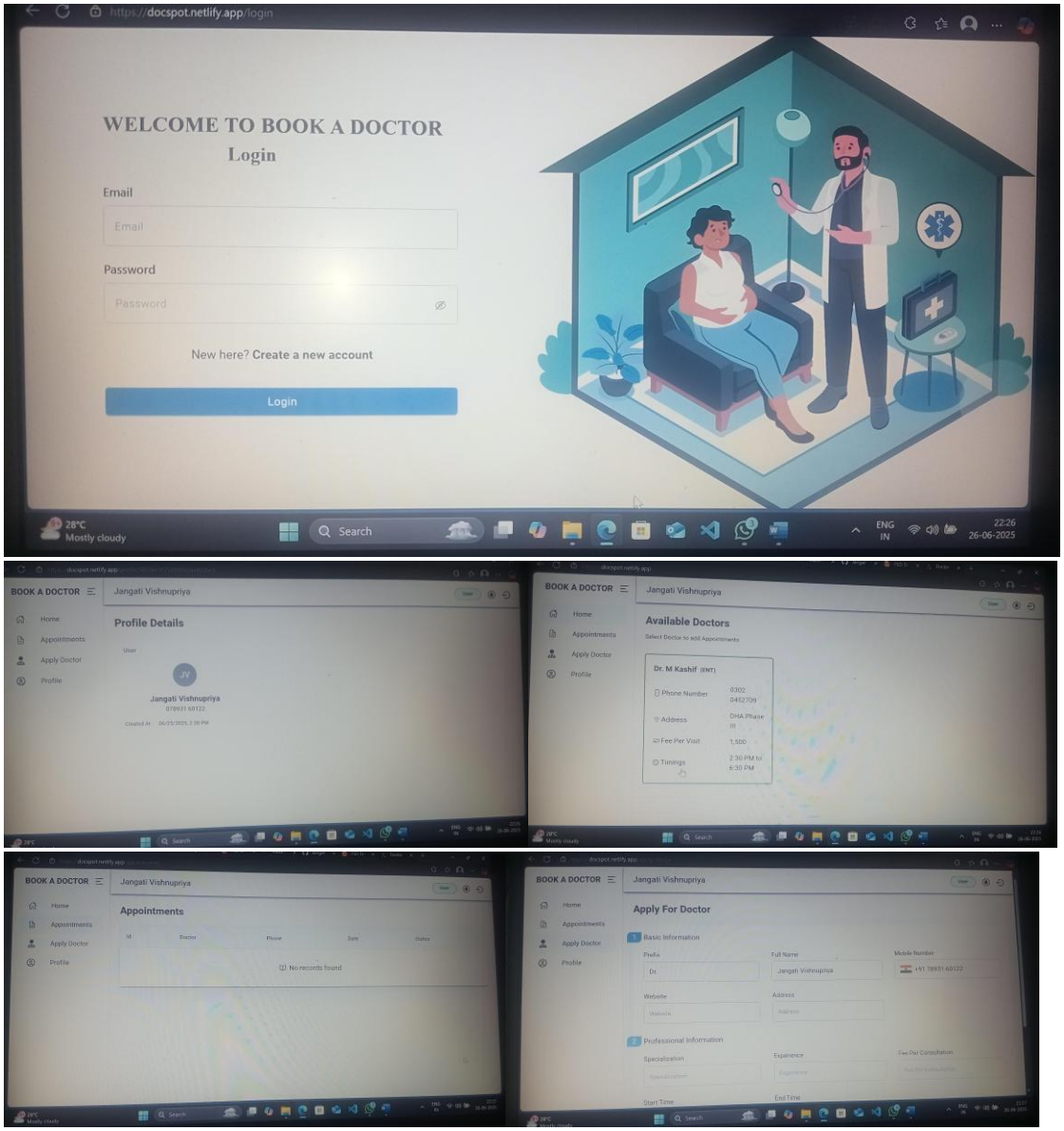
• Clean UI built using Material UI, Bootstrap, and mdb-react-ui-kit.
• Pages: User Dashboard, Doctor Panel, Admin Panel, Appointment Booking.
(Screenshots can be added separately in final PDF format.)

10. Testing

• Manual testing on different browsers.
• Positive and negative test cases written.
• UAT (User Acceptance Testing) completed and signed off.

11. Screenshots or Demo

• Live URL: https://docspot.netlify.app/
• API: https://docspot-tfnm.onrender.com
• **Screenshots:**



12. Known Issues

- Doctor image upload feature is under development.
- Time overlap validation logic can be improved.

13. Future Enhancements

- Mobile App (React Native)
- Google Calendar integration
- Rating system for doctors
- Prescription uploads
- Payment gateway for paid consultations