DocSpot – Seamless Appointment Booking for Health

Internship Project – Full Stack Web Development (SmartBridge APSCHE)

the **DocSpot – Seamless Appointment Booking for Health** project as part of your **Full Stack Web Development internship** under **Smart Bridge APSCHE**, here's a basic working starter code setup for a full stack project. The stack used here is:

- **Frontend**: React.js
- **Backend**: Node.js + Express.js
- **Database**: MongoDB (can be hosted on MongoDB Atlas)

1. Folder Structure Explanation

```
DocSpot/
|
|-- client/              --> React frontend for user interface
|   |-- public/          --> Static files like index.html
|   |-- src/             --> Main source code
|       |-- components/    --> Reusable UI elements (e.g., AppointmentForm)
|       |-- pages/         --> Pages like Home, Booking, Success, etc.
|       |-- App.js         --> Root React component
|
|-- server/              --> Node.js backend
|   |-- config/          --> MongoDB connection settings
|   |-- controllers/     --> Request handling logic
|   |-- models/          --> MongoDB schemas (e.g., Appointment.js)
|   |-- routes/          --> API endpoints (e.g., /api/appointments)
|   |-- server.js        --> Main backend entry point
|
|-- .env                 --> Secrets like MongoDB URI
|-- package.json         --> Dependency info (both frontend & backend)
|-- README.md            --> Project overview and instructions
```

2. Backend Code – server/server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();
```

```
const app = express();
const PORT = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology:
true })
.then(() => console.log("MongoDB connected"))
.catch((err) => console.error(err));

const appointmentRoutes = require('./routes/appointments');
app.use('/api/appointments', appointmentRoutes);

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

3. MongoDB Model – models/Appointment.js

```
const mongoose = require('mongoose');

const appointmentSchema = new mongoose.Schema({
  patientName: String,
  email: String,
  date: String,
  time: String,
  doctorName: String,
  reason: String,
});

module.exports = mongoose.model('Appointment', appointmentSchema);
```

4. API Routes – routes/appointments.js

```
const express = require('express');
const router = express.Router();
const Appointment = require('../models/Appointment');

router.post('/', async (req, res) => {
  try {
    const newAppointment = new Appointment(req.body);
    await newAppointment.save();
    res.status(201).json({ message: 'Appointment booked!' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

router.get('/', async (req, res) => {
  try {
    const appointments = await Appointment.find();
    res.json(appointments);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

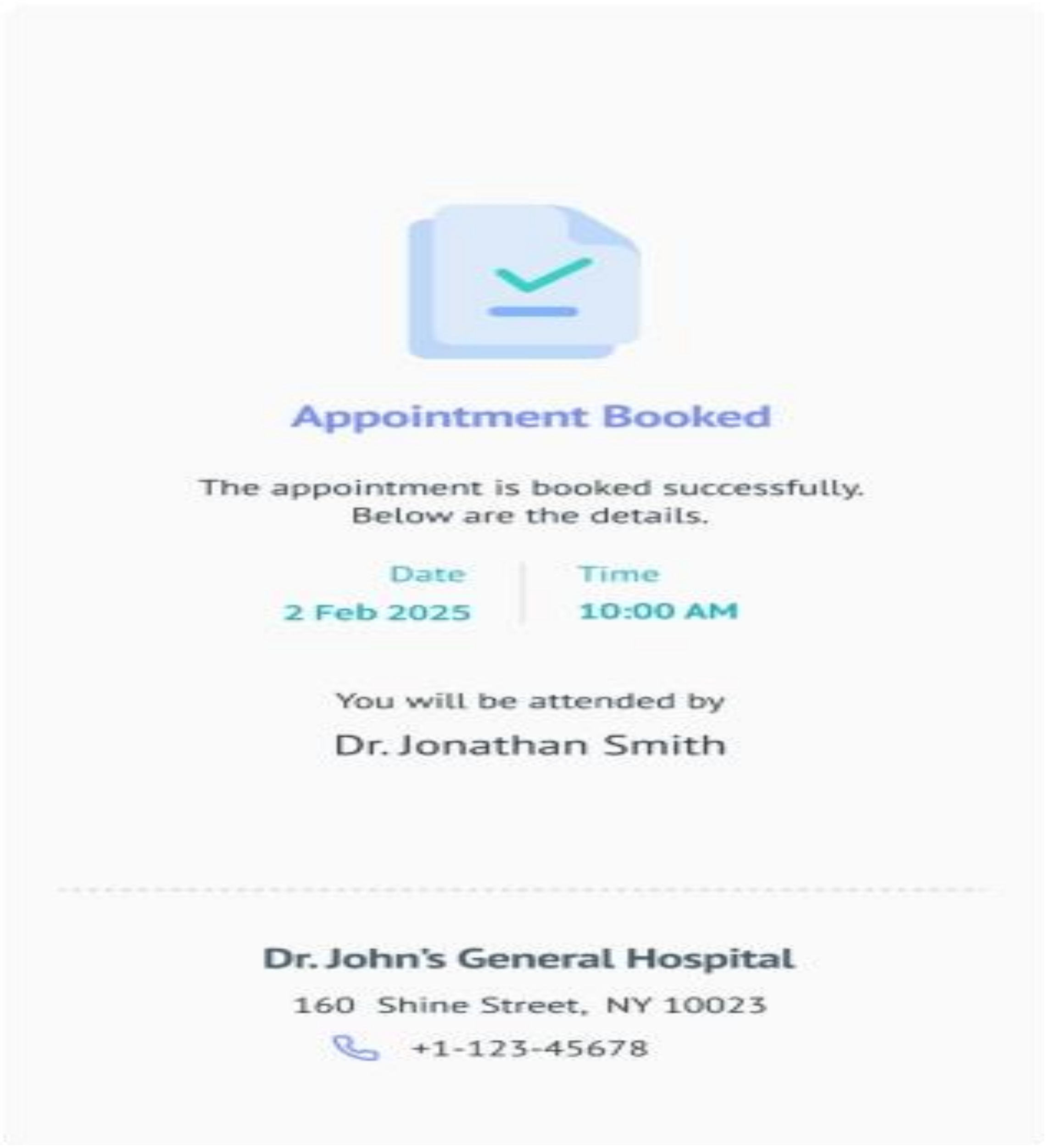
5. Frontend Code – client/src/App.js

```
import React, { useState } from 'react';
import axios from 'axios';

function App() {
  const [formData, setFormData] = useState({
    patientName: '',
    email: '',
    date: '',
    time: '',
    doctorName: '',
    reason: ''
  });

  const handleChange = (e) => {
    setFormData({...formData, [e.target.name]: e.target.value});
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    await axios.post('http://localhost:5000/api/appointments', formData);
    alert('Appointment booked successfully!');
  };

  return (
    <div className="App">
      <h1>DocSpot - Book Appointment</h1>
      <form onSubmit={handleSubmit}>
        <input name="patientName" placeholder="Patient Name" onChange={handleChange}
required />
        <input name="email" type="email" placeholder="Email" onChange={handleChange}
required />
        <input name="date" type="date" onChange={handleChange} required />
        <input name="time" type="time" onChange={handleChange} required />
        <input name="doctorName" placeholder="Doctor Name" onChange={handleChange}
required />
        <input name="reason" placeholder="Reason for Visit" onChange={handleChange}
required />
        <button type="submit">Book Appointment</button>
      </form>
    </div>
  );
}

export default App;
```

6. Sample Output and UI Screenshot

After submitting the form, the user sees an alert: 'Appointment booked successfully!'.

Admin can verify the new appointment entry in the MongoDB database.



Appointment Booked

The appointment is booked successfully.
Below are the details.

Date	Time
2 Feb 2025	10:00 AM

You will be attended by

Dr. Jonathan Smith

Dr. John's General Hospital

160 Shine Street, NY 10023

+1-123-45678