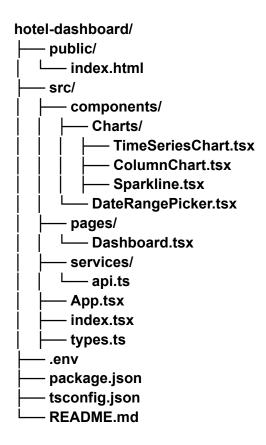
# **HOTEL BOOKING**

## **Project Structure**

#### **CREATED A STRUCTURE STEP BY STEP:**

#### FOLDER STRUCTURE OF REACT AND TYPESCRIPT IN MY PROJECT



#### FOLDER STRUCTURE OF BACKEND IN MY PROJECT

#### **BACKEND**

```
<u>database.js</u>
```

```
const sqlite3 = require('sqlite3').verbose();
const path = require('path');
const db = new sqlite3.Database(path.resolve(__dirname,
'../../data/hotel_bookings_1000.csv'), (err) => {
 if (err) console.error('Error connecting to database:', err);
 console.log('Connected to SQLite database.');
});
module.exports = db;
bookings.js
const express = require('express');
const db = require('../models/database');
const router = express.Router();
// Get all bookings within a date range
router.get('/bookings', (req, res) => {
 const { startDate, endDate } = req.query;
 const query = `
  SELECT arrival_date_year, arrival_date_month, arrival_date_day_of_month,
      adults, children, babies, country
  FROM bookings
  WHERE arrival_date >= ? AND arrival_date <= ?`;
 db.all(query, [startDate, endDate], (err, rows) => {
  if (err) return res.status(500).json({ error: err.message });
  res.json(rows);
```

```
});
});
module.exports = router;
app.js
const express = require('express');
const cors = require('cors');
const dotenv = require('dotenv');
const bookingsRouter = require('./routes/bookings');
dotenv.config();
const app = express();
const PORT = process.env.PORT || 5000;
app.use(cors());
app.use(express.json());
app.use('/api', bookingsRouter);
app.listen(PORT, () => {
 console.log(`Server running at http://localhost:${PORT}`);
});
```

### **FRONTEND**

#### Api.tsx

```
import axios from 'axios';

const API_URL = process.env.REACT_APP_API_URL || 'http://localhost:5000/api';

export const fetchBookings = async (startDate: string, endDate: string) => {
   const response = await axios.get(`${API_URL}/bookings`, {
     params: { startDate, endDate },
   });
   return response.data;
};
```

#### Dashboard.tsx

```
import React, { useState, useEffect } from 'react';
import { fetchBookings } from '../services/api';
import TimeSeriesChart from '../components/Charts/TimeSeriesChart';
import ColumnChart from '../components/Charts/ColumnChart';
import Sparkline from '../components/Charts/Sparkline';
import DateRangePicker from '../components/DateRangePicker';
const Dashboard: React.FC = () => {
 const [data, setData] = useState<any[]>([]);
 const [startDate, setStartDate] = useState(");
 const [endDate, setEndDate] = useState(");
 useEffect(() => {
  if (startDate && endDate) {
   fetchBookings(startDate, endDate).then(setData);
 }, [startDate, endDate]);
 const visitorCount = data.reduce((acc, item) => acc + item.adults + item.children +
item.babies, 0);
 return (
  <div>
   <DateRangePicker setStartDate={setStartDate} setEndDate={setEndDate} />
   <TimeSeriesChart data={data} />
   <ColumnChart data={data} />
   <Sparkline label="Adult Visitors" dataKey="adults" data={data} />
   <Sparkline label="Children Visitors" dataKey="children" data={data} />
   <div>Total Visitors: {visitorCount}</div>
  </div>
);
};
export default Dashboard;
DateRangePicker.tsx
import React from 'react';
```

```
interface Props {
 setStartDate: (date: string) => void;
 setEndDate: (date: string) => void;
}
const DateRangePicker: React.FC<Props> = ({ setStartDate, setEndDate }) => (
 <div>
  <input type="date" onChange={(e) => setStartDate(e.target.value)} />
  <input type="date" onChange={(e) => setEndDate(e.target.value)} />
 </div>
);
export default DateRangePicker;
TimeSeriesChart.tsx
import React from 'react';
import ReactApexChart from 'react-apexcharts';
interface Props {
 data: any[];
}
const TimeSeriesChart: React.FC<Props> = ({ data }) => {
 const series = [
   name: 'Visitors',
   data: data.map((item) => ({
`${item.arrival_date_year}-${item.arrival_date_month}-${item.arrival_date_day_of_month}
    y: item.adults + item.children + item.babies,
   })),
  },
 ];
 const options = {
  chart: { type: 'line', zoom: { enabled: true } },
  xaxis: { type: 'datetime' },
 };
```

```
return <ReactApexChart options={options} series={series} type="line" />;
};
export default TimeSeriesChart;
```