Name: Tejas Redkar

PRN: 1032210937

Roll No.: 44

Panel: C, **Batch:** C2

FSD Lab 07

Aim: Develop a full stack web application using MERN stack to perform CRUD operations.

Objectives:

1. To develop full-stack web projects using the MERN stack.
2. To learn database connectivity using fetch api.
3. To perform insert, update, delete and search operations on database.

Theory:

1. **What is MERN stack?**

   Definition: MERN stack is a popular acronym in web development that stands for MongoDB, Express.js, React.js, and Node.js. It represents a full-stack JavaScript framework used for building dynamic and robust web applications. Each component of the MERN stack plays a specific role in the development process:

   MongoDB: A NoSQL database that stores data in a JSON-like format.

   Express.js: A web application framework for Node.js that simplifies the creation of server-side applications.

   React.js: A JavaScript library for building user interfaces, particularly for single-page applications where data can change without reloading the page.

   Node.js: A JavaScript runtime environment that executes server-side code.

2. **Use of Fetch API:**
   Definition: The Fetch API is a modern web API for making HTTP requests and handling responses. It provides a more flexible and powerful alternative to the older XMLHttpRequest. The Fetch API is native to modern browsers and is used in conjunction with JavaScript Promises.
   Usage: The Fetch API is used to make asynchronous requests to a server. It returns a Promise that resolves to the Response to that request, whether it is successful or not. The basic syntax looks like this:

```
fetch(url)
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

FAQ:

1. **What makes MERN stack the fastest growing tech stack?**

ANS: JavaScript Throughout: One key factor is the use of JavaScript across the entire stack (both client and server). Developers can use the same language for both front-end and back-end development, streamlining the development process and reducing context switching.

Component Reusability: React.js, a part of the MERN stack, promotes the concept of reusable components. This leads to more modular and maintainable code, enabling developers to efficiently build and scale applications.

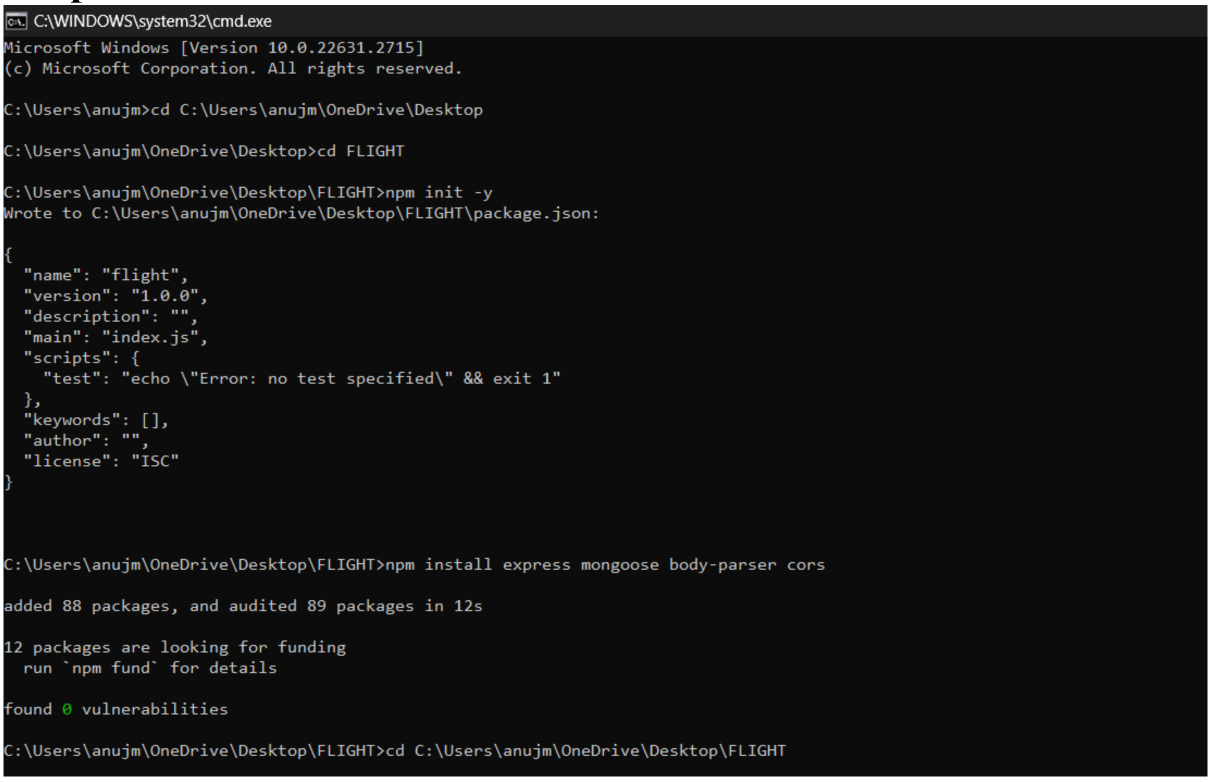
Isomorphic/Universal JavaScript: MERN allows for the use of JavaScript on both the client and server sides. This enables server-side rendering and improves performance, especially in terms of initial page load times.

Active Community and Ecosystem: The MERN stack has a large and active developer community. There are numerous libraries, tools, and resources available, making it easier for developers to find solutions and best practices.

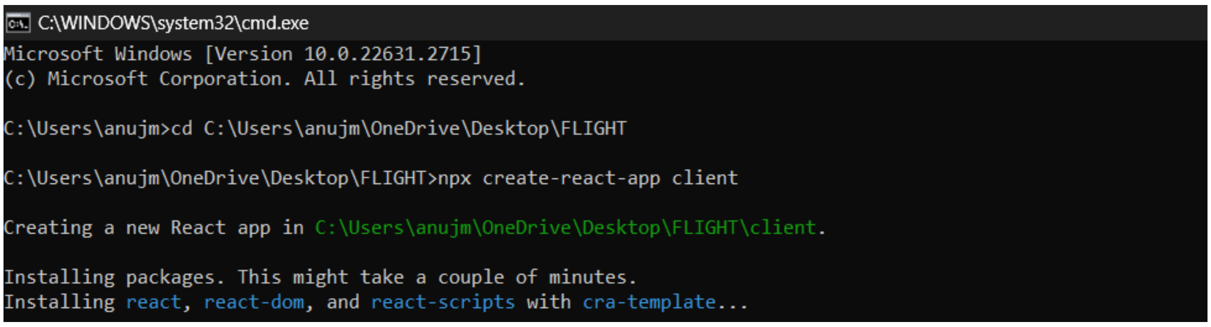
Scalability and Flexibility: MongoDB, a NoSQL database in the stack, provides flexibility in handling different types of data. Node.js, with its non-blocking I/O, contributes to the scalability of applications, making them capable of handling a large number of simultaneous connections.

Rapid Development: The combination of a powerful front-end library (React.js) and a fast, event-driven server (Node.js) allows for rapid development of modern, dynamic web applications.

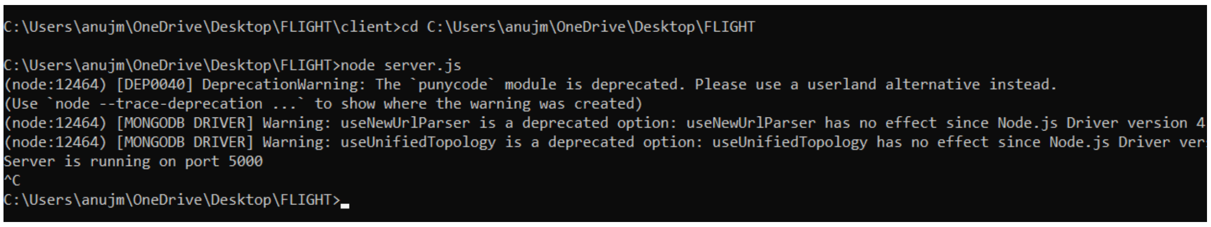
Output Screenshots:



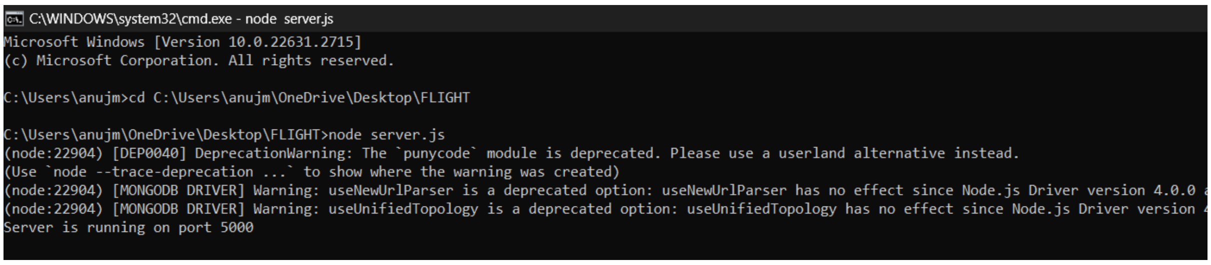
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anujm>cd C:\Users\anujm\OneDrive\Desktop

C:\Users\anujm\OneDrive\Desktop>cd FLIGHT

C:\Users\anujm\OneDrive\Desktop\FLIGHT>npm init -y
Wrote to C:\Users\anujm\OneDrive\Desktop\FLIGHT\package.json:

{
  "name": "flight",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



C:\Users\anujm\OneDrive\Desktop\FLIGHT>npm install express mongoose body-parser cors

added 88 packages, and audited 89 packages in 12s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\anujm\OneDrive\Desktop\FLIGHT>cd C:\Users\anujm\OneDrive\Desktop\FLIGHT
```



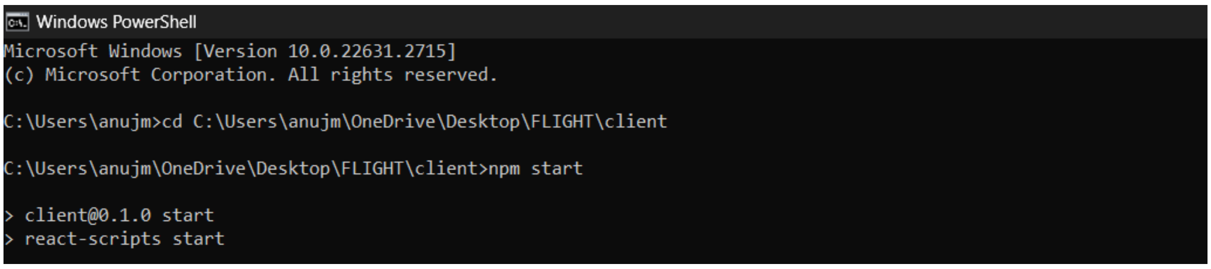
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anujm>cd C:\Users\anujm\OneDrive\Desktop\FLIGHT

C:\Users\anujm\OneDrive\Desktop\FLIGHT>npx create-react-app client

Creating a new React app in C:\Users\anujm\OneDrive\Desktop\FLIGHT\client.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
```



```
C:\Users\anujm\OneDrive\Desktop\FLIGHT\client>cd C:\Users\anujm\OneDrive\Desktop\FLIGHT

C:\Users\anujm\OneDrive\Desktop\FLIGHT>node server.js
(node:12464) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:12464) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4
(node:12464) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver ver
Server is running on port 5000
^C
C:\Users\anujm\OneDrive\Desktop\FLIGHT>
```

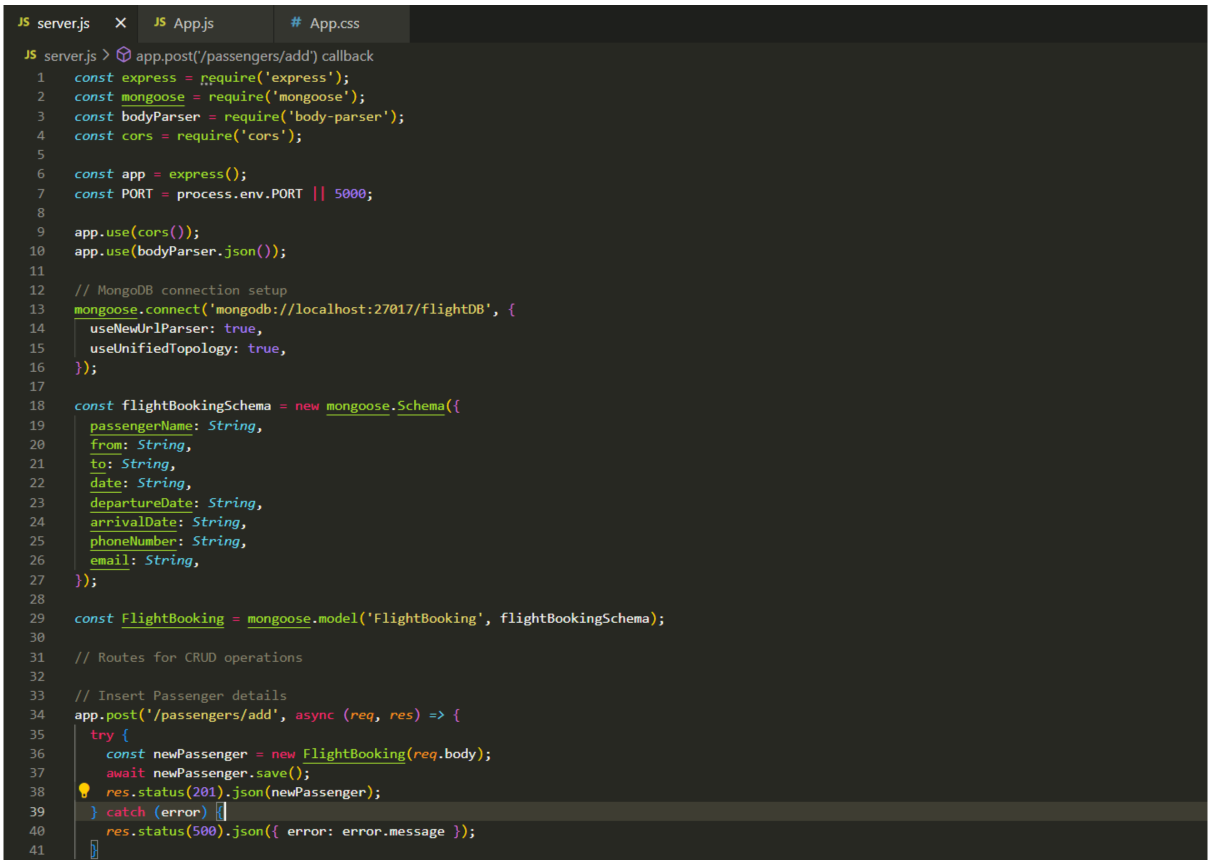


```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anujm>cd C:\Users\anujm\OneDrive\Desktop\FLIGHT

C:\Users\anujm\OneDrive\Desktop\FLIGHT>node server.js
(node:22904) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:22904) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 a
(node:22904) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4
Server is running on port 5000
```

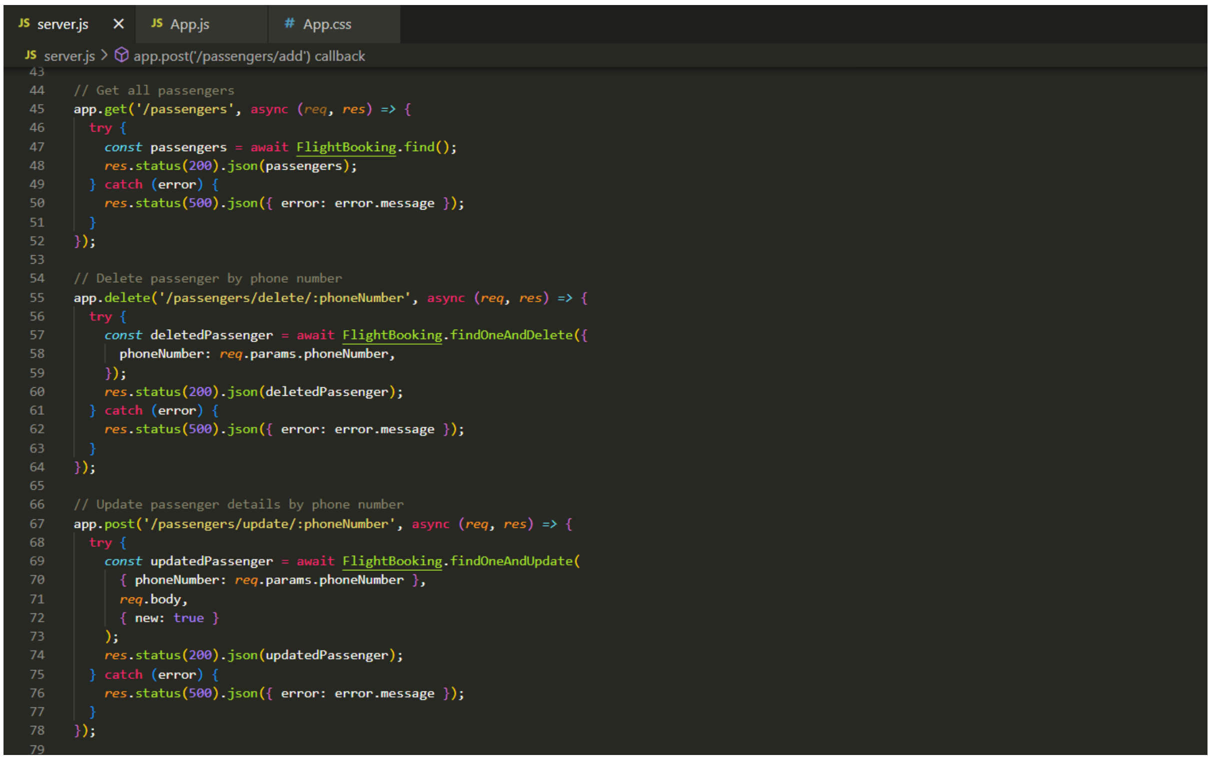


```
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anujm>cd C:\Users\anujm\OneDrive\Desktop\FLIGHT\client

C:\Users\anujm\OneDrive\Desktop\FLIGHT\client>npm start

> client@0.1.0 start
> react-scripts start
```

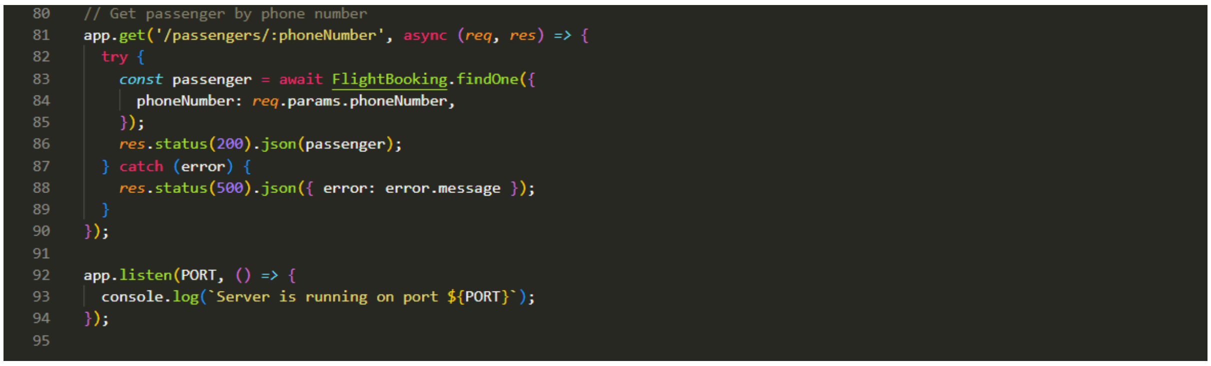
server.js



```javascript
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');

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

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

// MongoDB connection setup
mongoose.connect('mongodb://localhost:27017/flightDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const flightBookingSchema = new mongoose.Schema({
  passengerName: String,
  from: String,
  to: String,
  date: String,
  departureDate: String,
  arrivalDate: String,
  phoneNumber: String,
  email: String,
});

const FlightBooking = mongoose.model('FlightBooking', flightBookingSchema);

// Routes for CRUD operations

// Insert Passenger details
app.post('/passengers/add', async (req, res) => {
  try {
    const newPassenger = new FlightBooking(req.body);
    await newPassenger.save();
    res.status(201).json(newPassenger);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
```

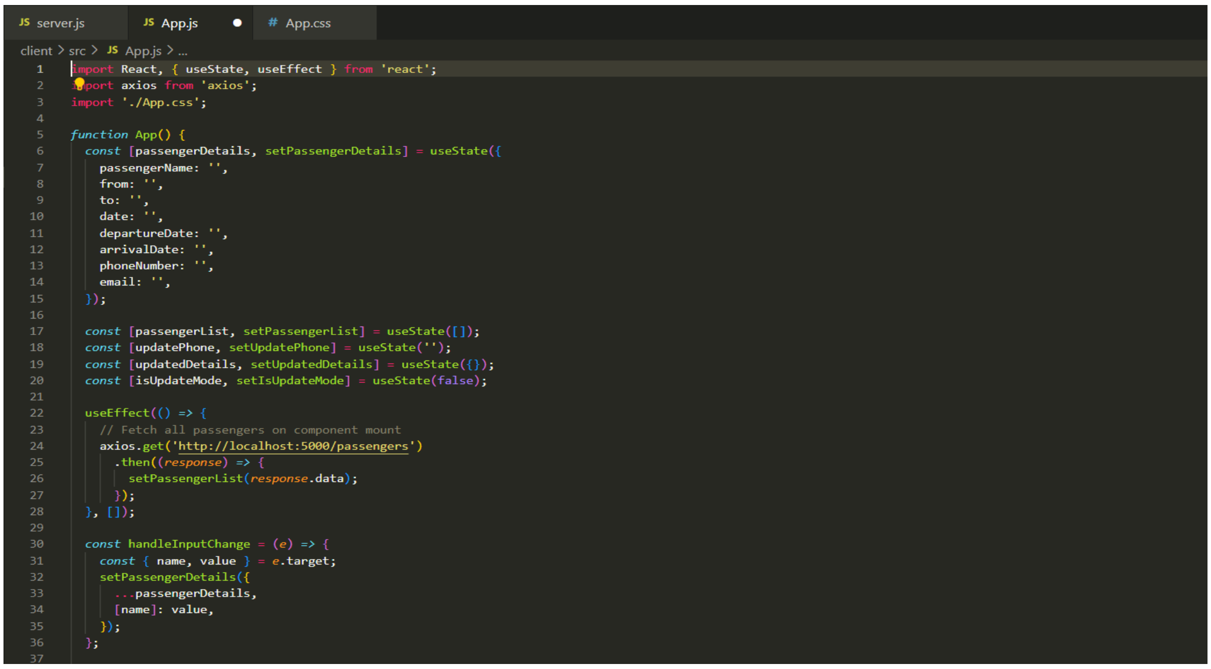


```javascript
// Get all passengers
app.get('/passengers', async (req, res) => {
  try {
    const passengers = await FlightBooking.find();
    res.status(200).json(passengers);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Delete passenger by phone number
app.delete('/passengers/delete/:phoneNumber', async (req, res) => {
  try {
    const deletedPassenger = await FlightBooking.findOneAndDelete({
      phoneNumber: req.params.phoneNumber,
    });
    res.status(200).json(deletedPassenger);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Update passenger details by phone number
app.post('/passengers/update/:phoneNumber', async (req, res) => {
  try {
    const updatedPassenger = await FlightBooking.findOneAndUpdate(
      { phoneNumber: req.params.phoneNumber },
      req.body,
      { new: true }
    );
    res.status(200).json(updatedPassenger);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

```



```javascript
// Get passenger by phone number
app.get('/passengers/:phoneNumber', async (req, res) => {
  try {
    const passenger = await FlightBooking.findOne({
      phoneNumber: req.params.phoneNumber,
    });
    res.status(200).json(passenger);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

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

```

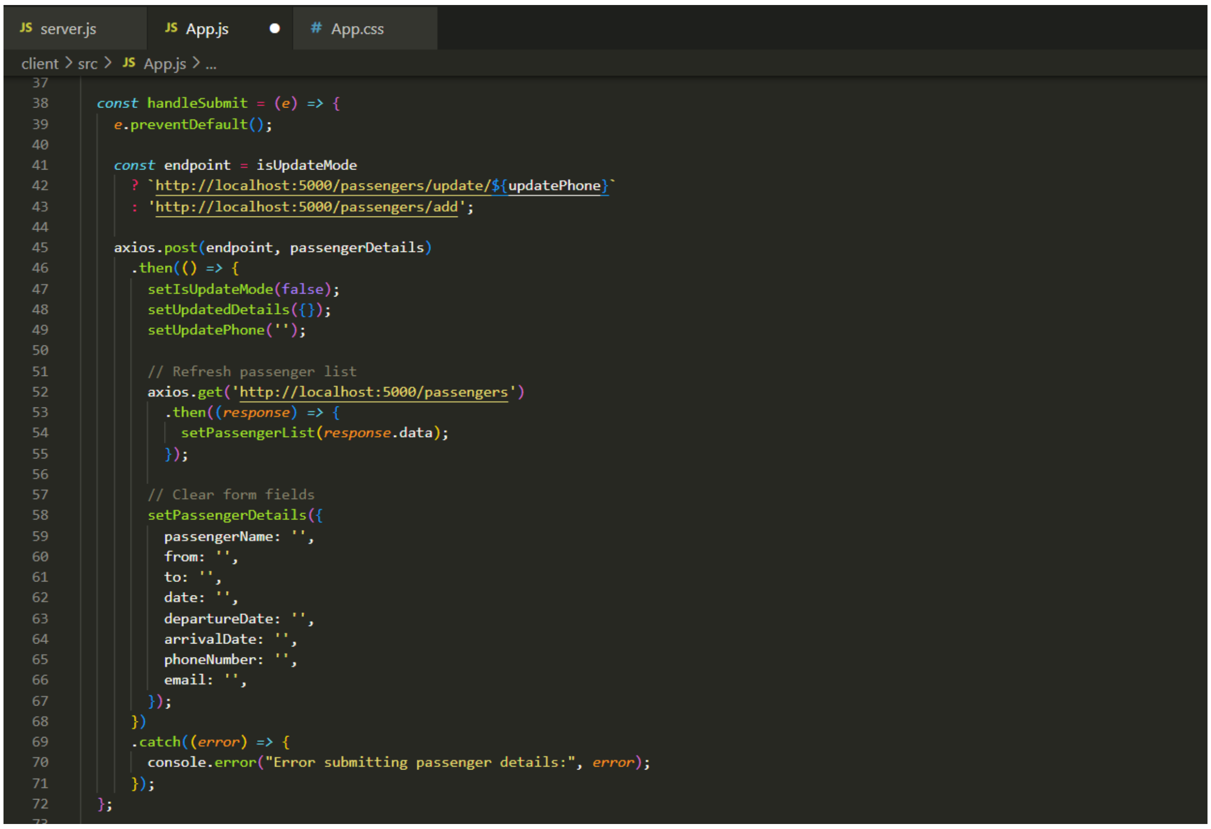
App.js



server.js App.js App.css

client > src > App.js > ...

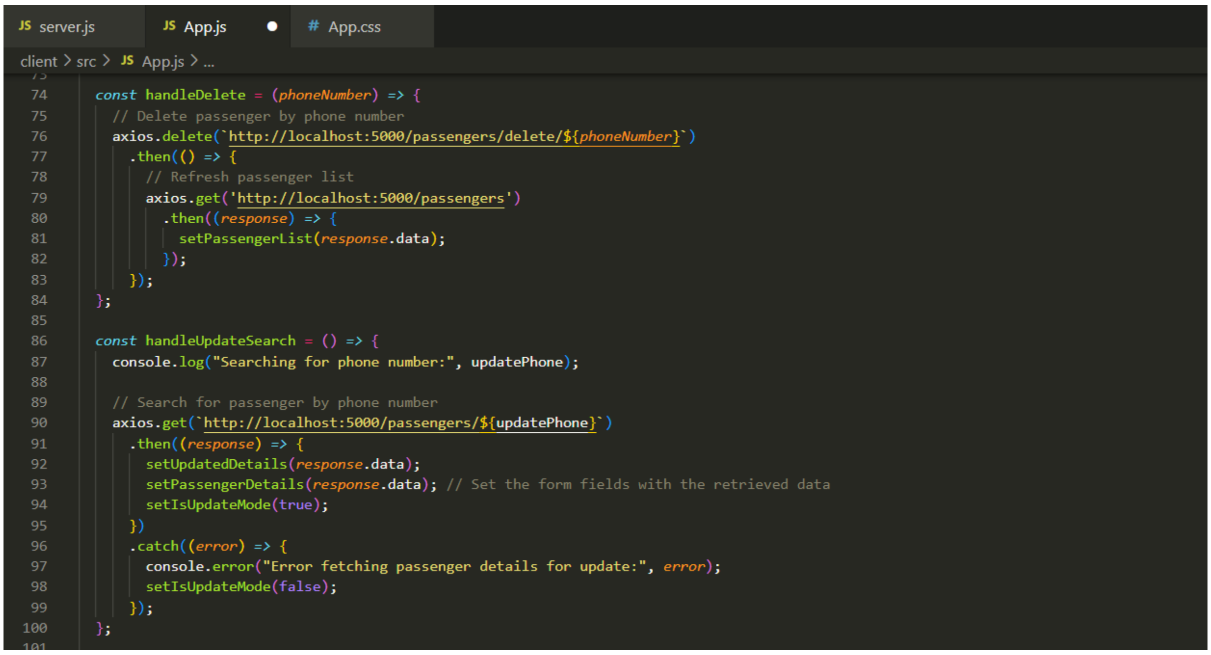
```javascript
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './App.css';

function App() {
  const [passengerDetails, setPassengerDetails] = useState({
    passengerName: '',
    from: '',
    to: '',
    date: '',
    departureDate: '',
    arrivalDate: '',
    phoneNumber: '',
    email: '',
  });

  const [passengerList, setPassengerList] = useState([]);
  const [updatePhone, setUpdatePhone] = useState('');
  const [updatedDetails, setUpdatedDetails] = useState({});
  const [isUpdateMode, setIsUpdateMode] = useState(false);

  useEffect(() => {
    // Fetch all passengers on component mount
    axios.get('http://localhost:5000/passengers')
      .then((response) => {
        setPassengerList(response.data);
      });
  }, []);

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setPassengerDetails({
      ...passengerDetails,
      [name]: value,
    });
  };

```



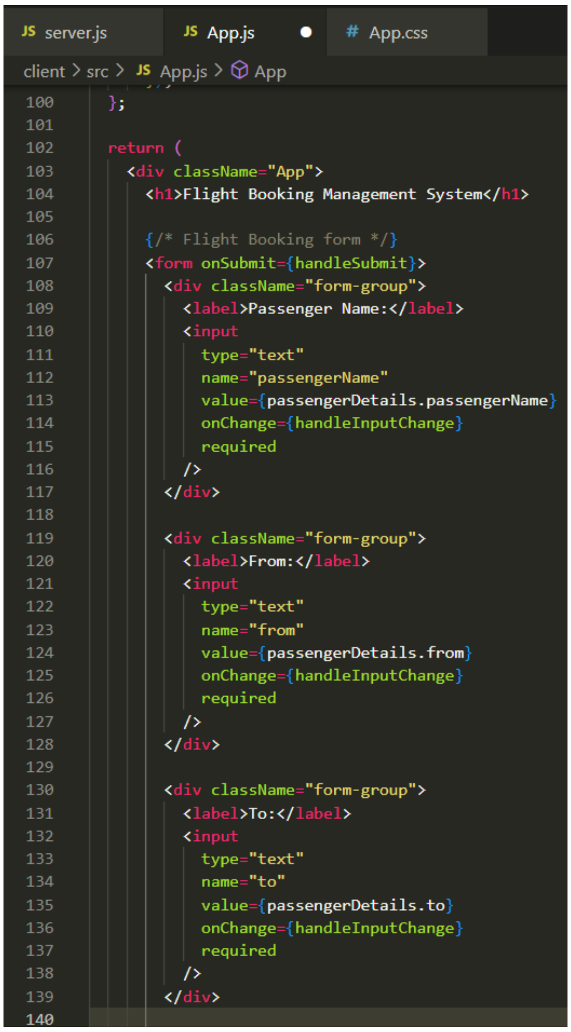
server.js App.js App.css

client > src > App.js > ...

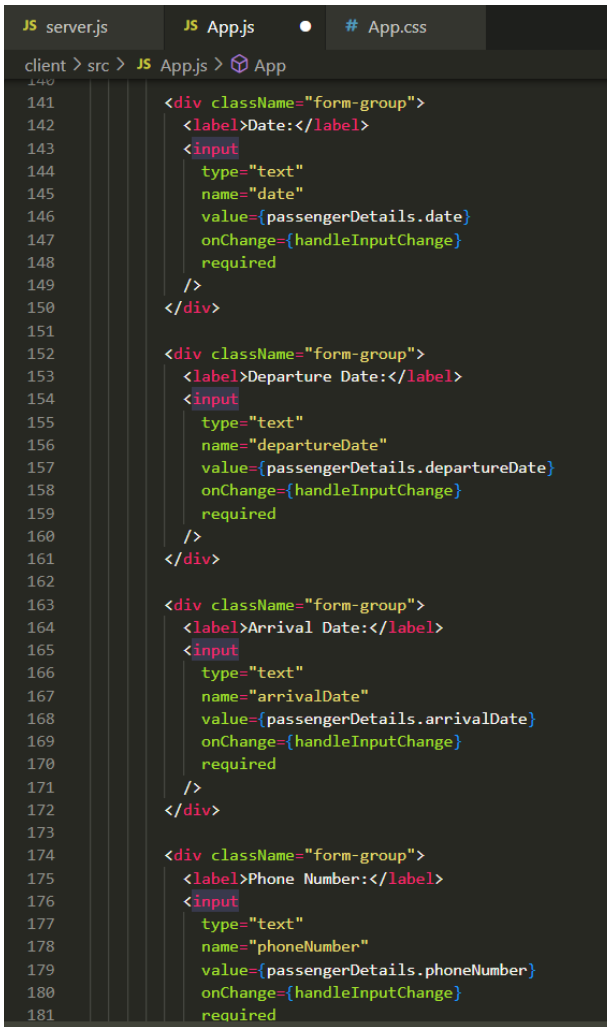
```javascript

  const handleSubmit = (e) => {
    e.preventDefault();

    const endpoint = isUpdateMode
      ? `http://localhost:5000/passengers/update/${updatePhone}`
      : 'http://localhost:5000/passengers/add';

    axios.post(endpoint, passengerDetails)
      .then(() => {
        setIsUpdateMode(false);
        setUpdatedDetails({});
        setUpdatePhone('');

        // Refresh passenger list
        axios.get('http://localhost:5000/passengers')
          .then((response) => {
            setPassengerList(response.data);
          });

        // Clear form fields
        setPassengerDetails({
          passengerName: '',
          from: '',
          to: '',
          date: '',
          departureDate: '',
          arrivalDate: '',
          phoneNumber: '',
          email: '',
        });
      })
      .catch((error) => {
        console.error("Error submitting passenger details:", error);
      });
  };
```



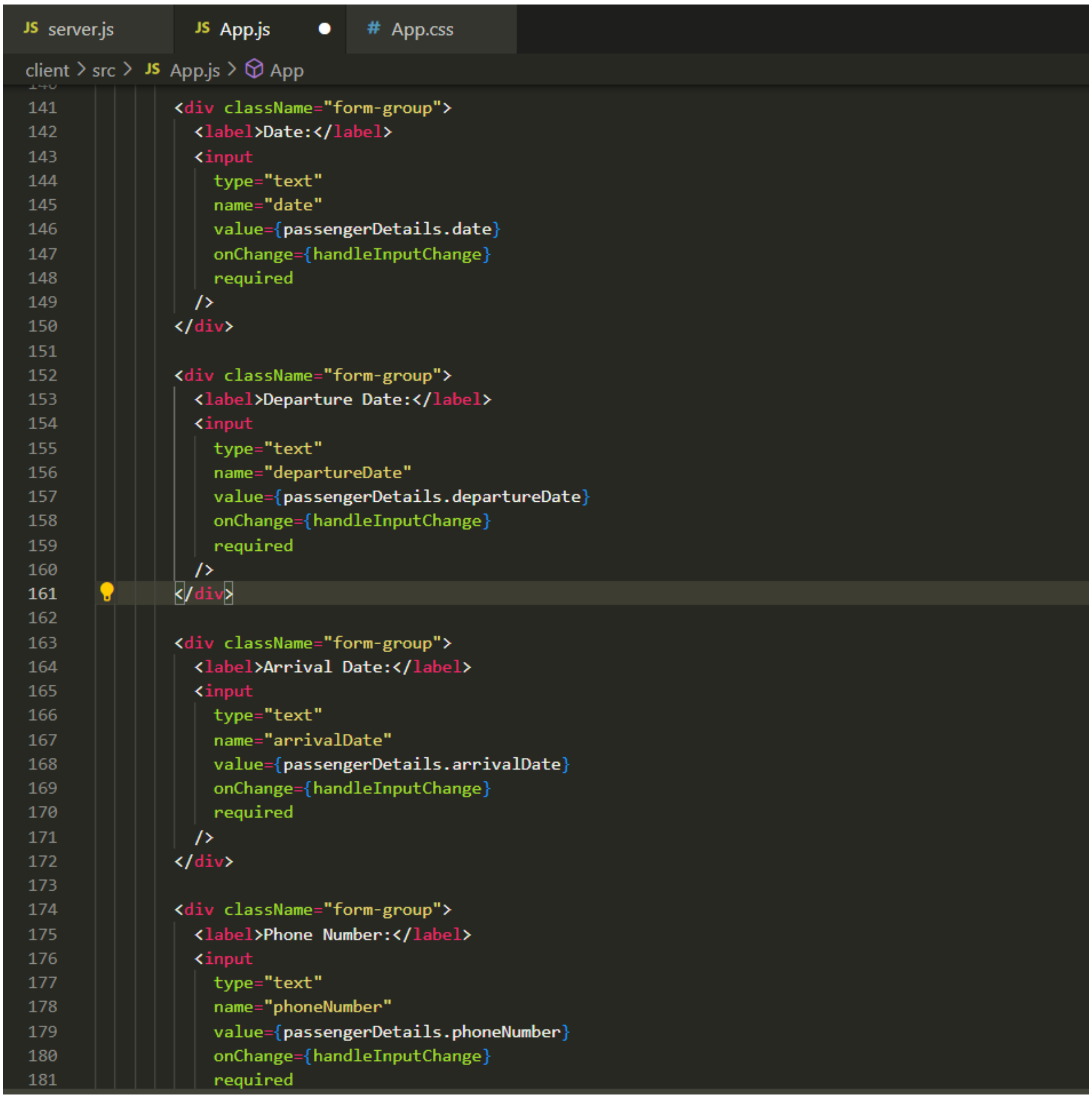
```
  const handleDelete = (phoneNumber) => {
    // Delete passenger by phone number
    axios.delete(`http://localhost:5000/passengers/delete/${phoneNumber}`)
      .then(() => {
        // Refresh passenger list
        axios.get('http://localhost:5000/passengers')
          .then((response) => {
            setPassengerList(response.data);
          });
      });
  };

  const handleUpdateSearch = () => {
    console.log("Searching for phone number:", updatePhone);

    // Search for passenger by phone number
    axios.get(`http://localhost:5000/passengers/${updatePhone}`)
      .then((response) => {
        setUpdatedDetails(response.data);
        setPassengerDetails(response.data); // Set the form fields with the retrieved data
        setIsUpdateMode(true);
      })
      .catch((error) => {
        console.error("Error fetching passenger details for update:", error);
        setIsUpdateMode(false);
      });
  };
```



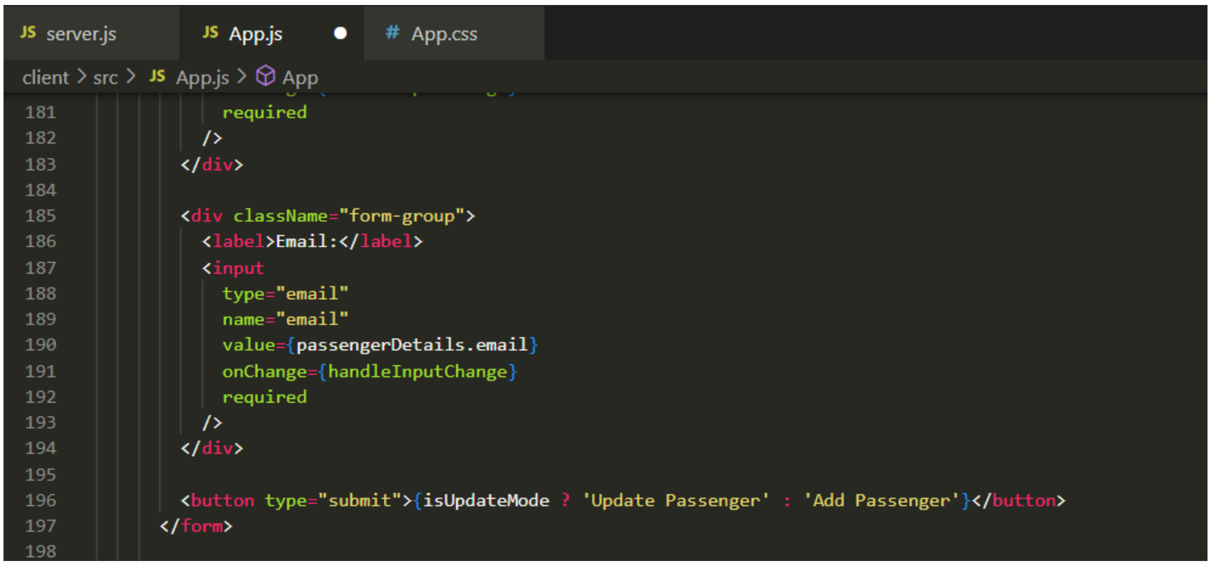
```
  };

  return (
    <div className="App">
      <h1>Flight Booking Management System</h1>

      {/* Flight Booking form */}
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label>Passenger Name:</label>
          <input
            type="text"
            name="passengerName"
            value={passengerDetails.passengerName}
            onChange={handleInputChange}
            required
          />
        </div>

        <div className="form-group">
          <label>From:</label>
          <input
            type="text"
            name="from"
            value={passengerDetails.from}
            onChange={handleInputChange}
            required
          />
        </div>

        <div className="form-group">
          <label>To:</label>
          <input
            type="text"
            name="to"
            value={passengerDetails.to}
            onChange={handleInputChange}
            required
          />
        </div>

```



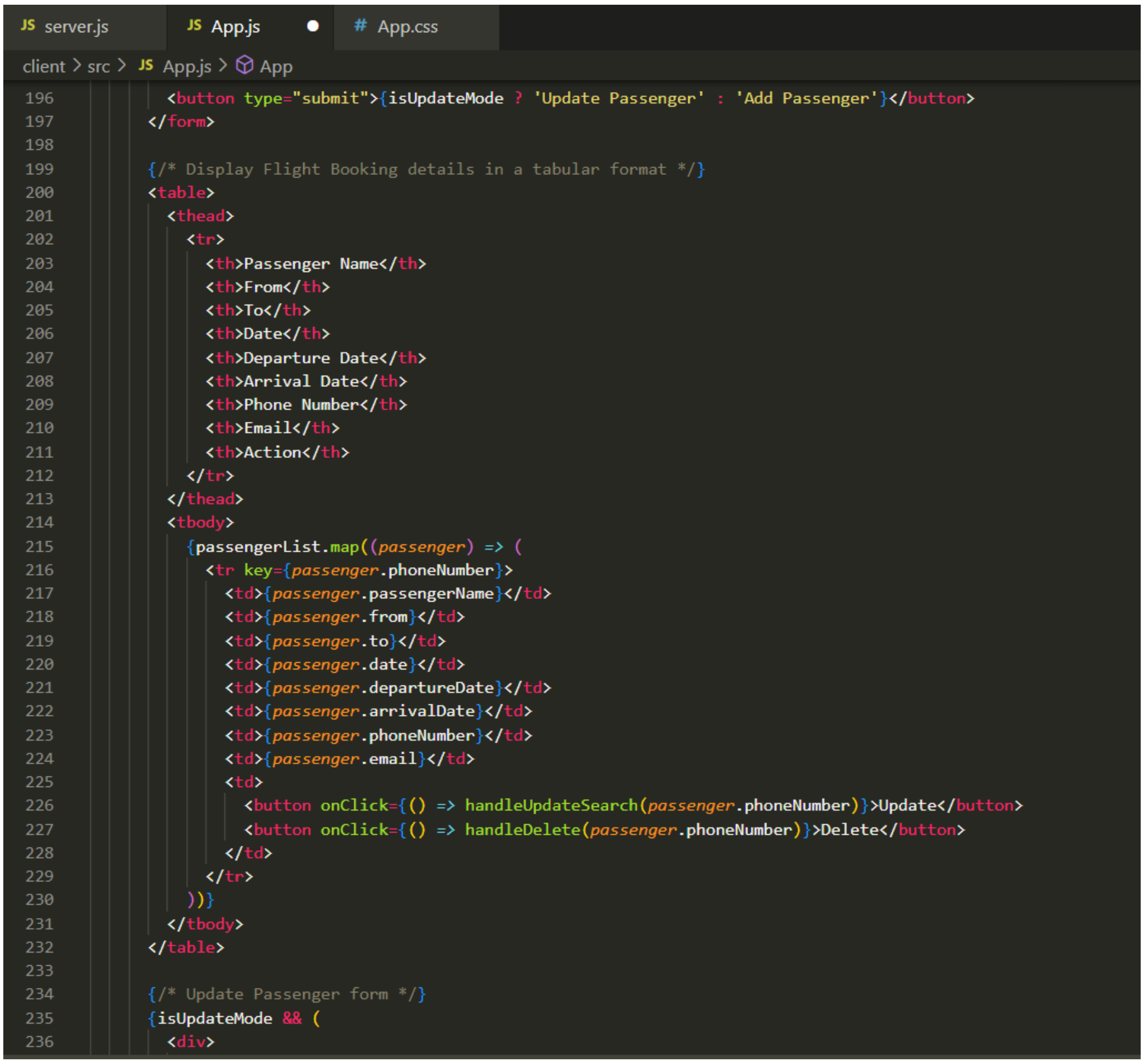
```
        <div className="form-group">
          <label>Date:</label>
          <input
            type="text"
            name="date"
            value={passengerDetails.date}
            onChange={handleInputChange}
            required
          />
        </div>

        <div className="form-group">
          <label>Departure Date:</label>
          <input
            type="text"
            name="departureDate"
            value={passengerDetails.departureDate}
            onChange={handleInputChange}
            required
          />
        </div>

        <div className="form-group">
          <label>Arrival Date:</label>
          <input
            type="text"
            name="arrivalDate"
            value={passengerDetails.arrivalDate}
            onChange={handleInputChange}
            required
          />
        </div>

        <div className="form-group">
          <label>Phone Number:</label>
          <input
            type="text"
            name="phoneNumber"
            value={passengerDetails.phoneNumber}
            onChange={handleInputChange}
            required
```



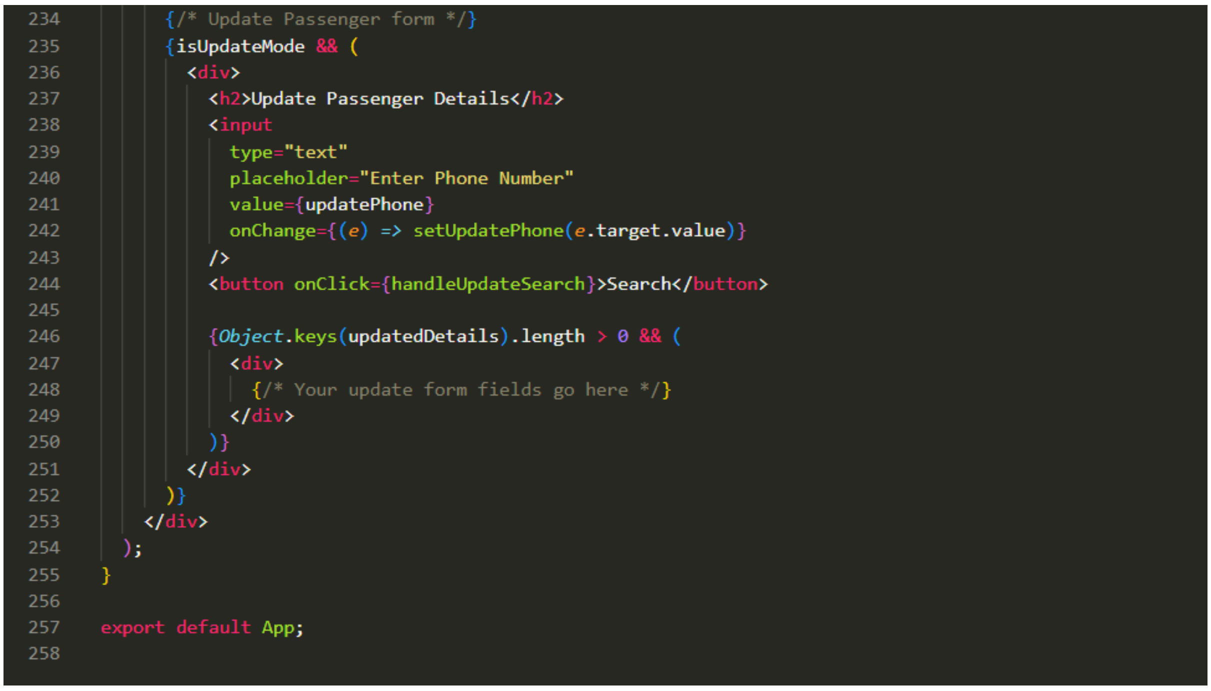
```
          <div className="form-group">
            <label>Date:</label>
            <input
              type="text"
              name="date"
              value={passengerDetails.date}
              onChange={handleInputChange}
              required
            />
          </div>

          <div className="form-group">
            <label>Departure Date:</label>
            <input
              type="text"
              name="departureDate"
              value={passengerDetails.departureDate}
              onChange={handleInputChange}
              required
            />
          </div>

          <div className="form-group">
            <label>Arrival Date:</label>
            <input
              type="text"
              name="arrivalDate"
              value={passengerDetails.arrivalDate}
              onChange={handleInputChange}
              required
            />
          </div>

          <div className="form-group">
            <label>Phone Number:</label>
            <input
              type="text"
              name="phoneNumber"
              value={passengerDetails.phoneNumber}
              onChange={handleInputChange}
              required
```



```
              required
            />
          </div>

          <div className="form-group">
            <label>Email:</label>
            <input
              type="email"
              name="email"
              value={passengerDetails.email}
              onChange={handleInputChange}
              required
            />
          </div>

          <button type="submit">{isUpdateMode ? 'Update Passenger' : 'Add Passenger'}</button>
        </form>
```

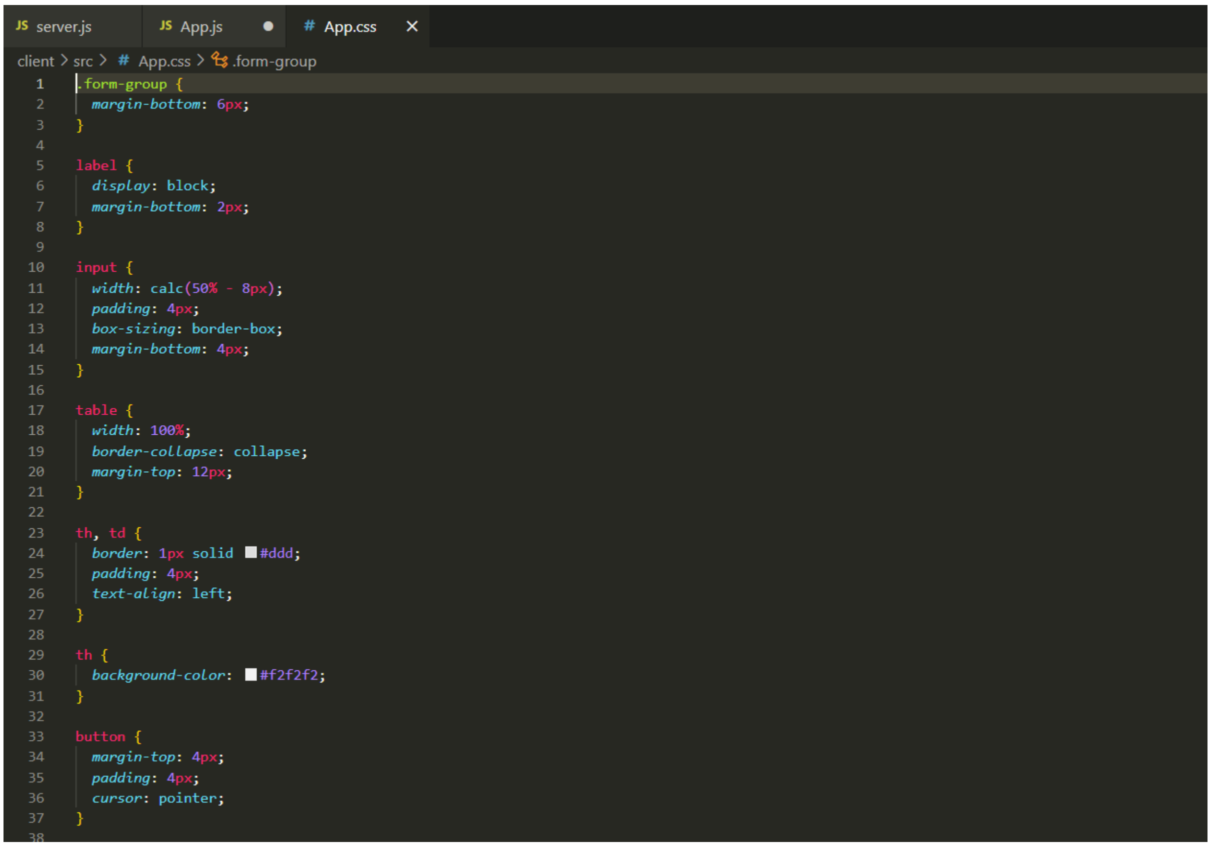


```
          <button type="submit">{isUpdateMode ? 'Update Passenger' : 'Add Passenger'}</button>
        </form>

        {/* Display Flight Booking details in a tabular format */}
        <table>
          <thead>
            <tr>
              <th>Passenger Name</th>
              <th>From</th>
              <th>To</th>
              <th>Date</th>
              <th>Departure Date</th>
              <th>Arrival Date</th>
              <th>Phone Number</th>
              <th>Email</th>
              <th>Action</th>
            </tr>
          </thead>
          <tbody>
            {passengerList.map((passenger) => (
              <tr key={passenger.phoneNumber}>
                <td>{passenger.passengerName}</td>
                <td>{passenger.from}</td>
                <td>{passenger.to}</td>
                <td>{passenger.date}</td>
                <td>{passenger.departureDate}</td>
                <td>{passenger.arrivalDate}</td>
                <td>{passenger.phoneNumber}</td>
                <td>{passenger.email}</td>
                <td>
                  <button onClick={() => handleUpdateSearch(passenger.phoneNumber)}>Update</button>
                  <button onClick={() => handleDelete(passenger.phoneNumber)}>Delete</button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>

        {/* Update Passenger form */}
        {isUpdateMode && (
          <div>
```

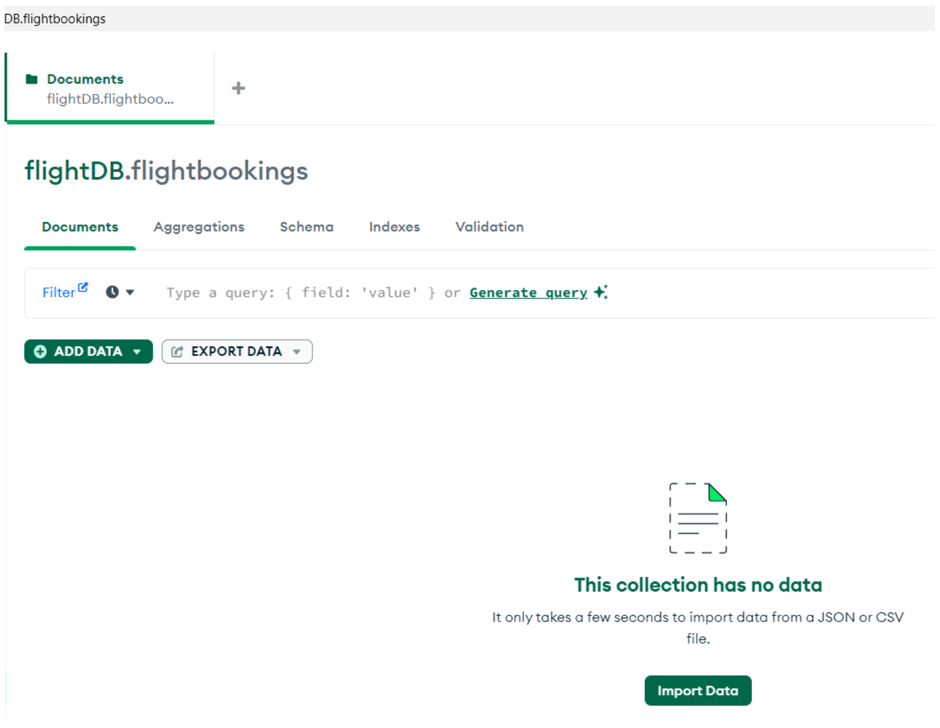


```
      {/* Update Passenger form */}
      {isUpdateMode && (
        <div>
          <h2>Update Passenger Details</h2>
          <input
            type="text"
            placeholder="Enter Phone Number"
            value={updatePhone}
            onChange={(e) => setUpdatePhone(e.target.value)}
          />
          <button onClick={handleUpdateSearch}>Search</button>

          {Object.keys(updatedDetails).length > 0 && (
            <div>
              {/* Your update form fields go here */}
            </div>
          )}
        </div>
      )}
    </div>
  );
}

export default App;
```

App.css



```css
.form-group {
  margin-bottom: 6px;
}

label {
  display: block;
  margin-bottom: 2px;
}

input {
  width: calc(50% - 8px);
  padding: 4px;
  box-sizing: border-box;
  margin-bottom: 4px;
}

table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 12px;
}

th, td {
  border: 1px solid #ddd;
  padding: 4px;
  text-align: left;
}

th {
  background-color: #f2f2f2;
}

button {
  margin-top: 4px;
  padding: 4px;
  cursor: pointer;
}
```


DB.flightbookings

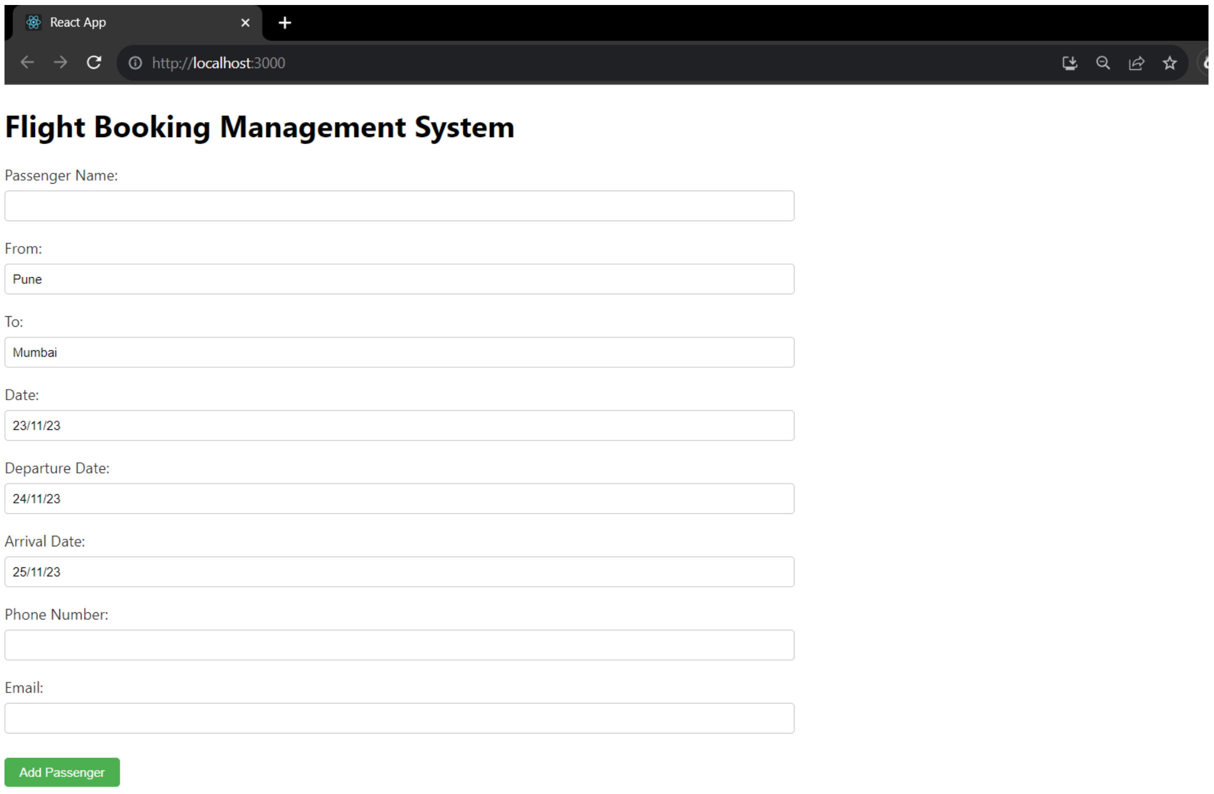
Documents
flightDB.flightboo...

flightDB.flightbookings

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

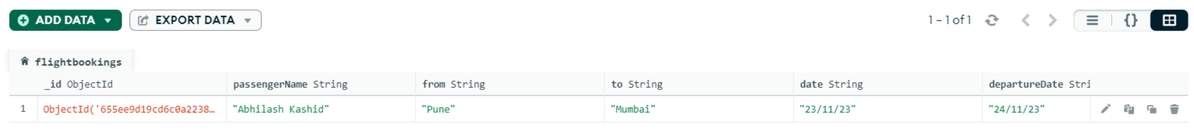
Insert:


React App

http://localhost:3000

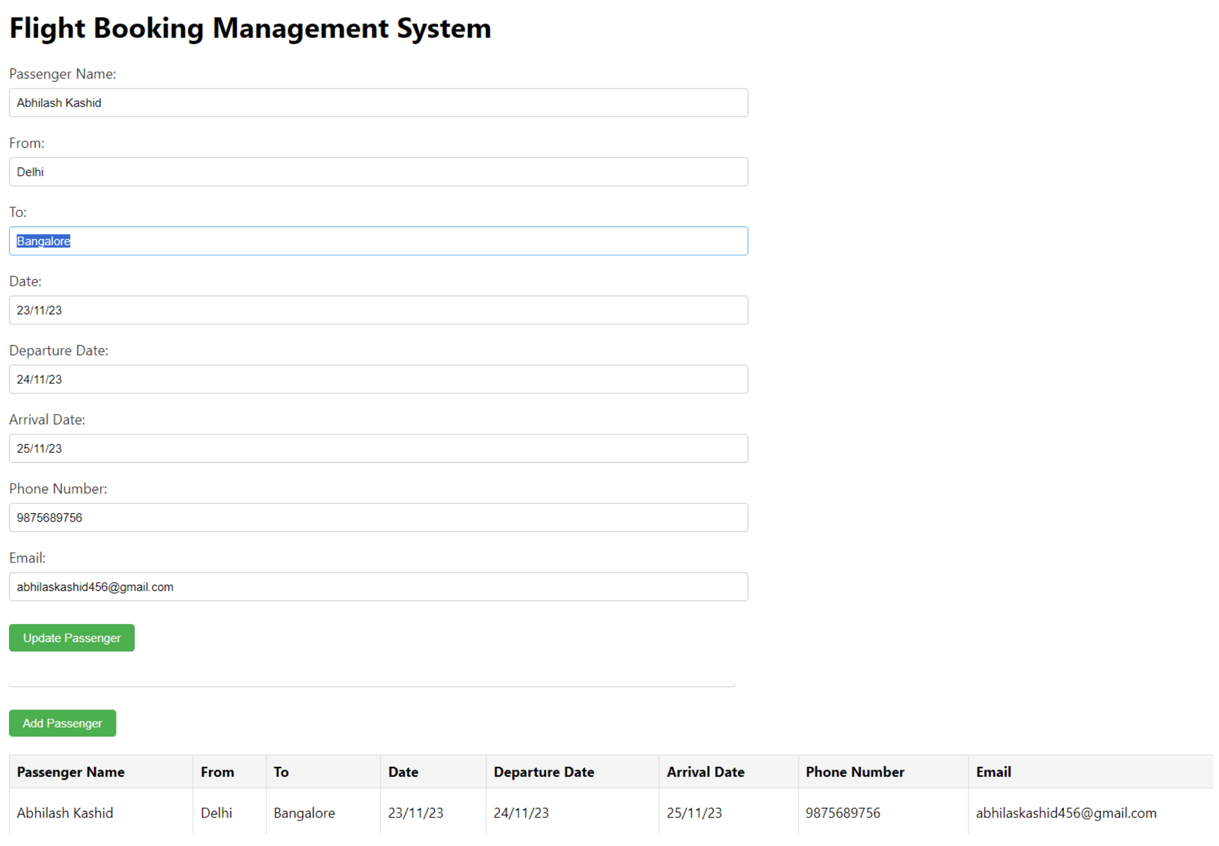
Flight Booking Management System

Passenger Name:	
From:	Pune
To:	Mumbai
Date:	23/11/23
Departure Date:	24/11/23
Arrival Date:	25/11/23
Phone Number:	
Email:	



	_id ObjectId	passengerName String	from String	to String	date String	departureDate Stri
1	ObjectId('655ee9d19cd6c0a2238…	"Abhilash Kashid"	"Pune"	"Mumbai"	"23/11/23"	"24/11/23"

Update:

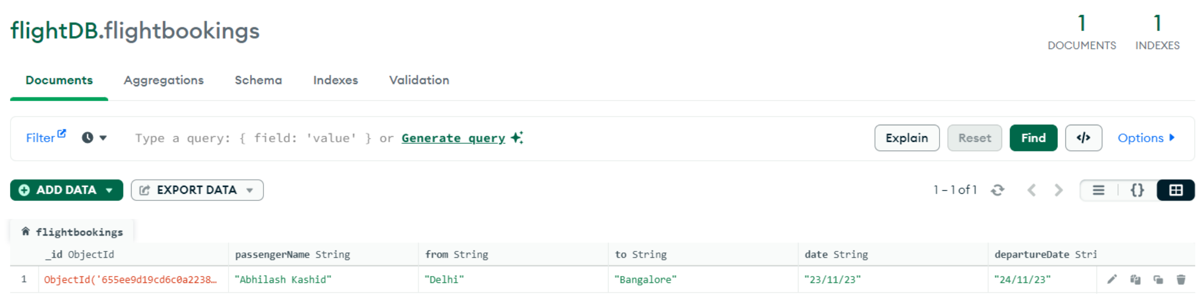


Flight Booking Management System

Passenger Name:	Abhilash Kashid
From:	Delhi
To:	Bangalore
Date:	23/11/23
Departure Date:	24/11/23
Arrival Date:	25/11/23
Phone Number:	9875689756
Email:	abhilaskashid456@gmail.com

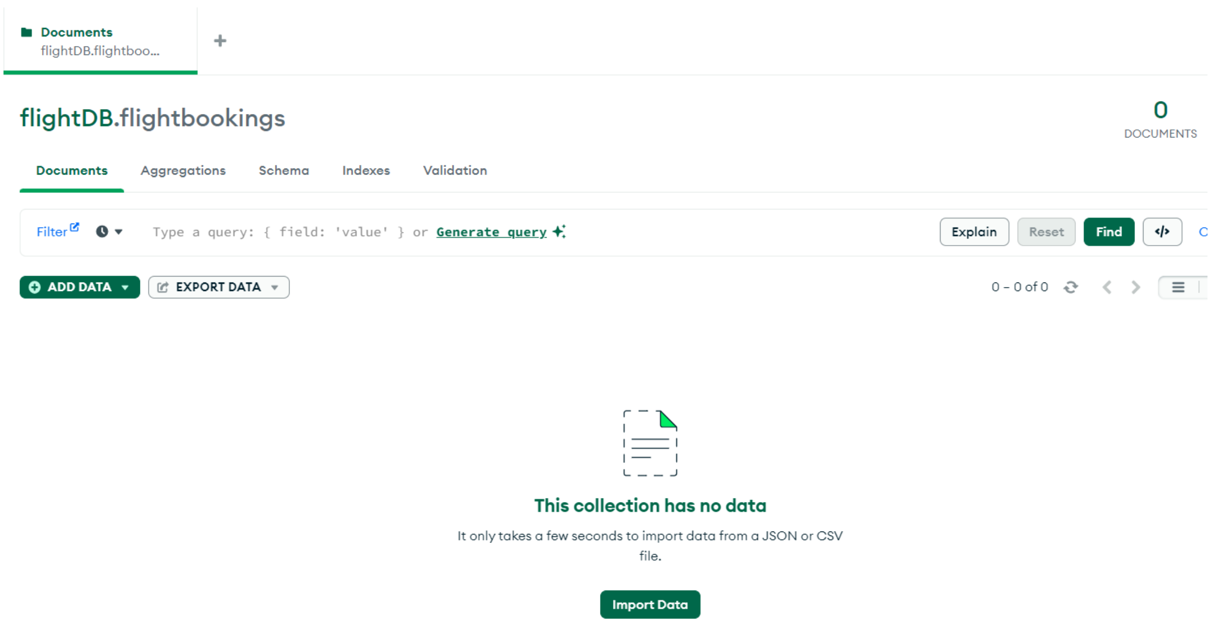
Passenger Name	From	To	Date	Departure Date	Arrival Date	Phone Number	Email
Abhilash Kashid	Delhi	Bangalore	23/11/23	24/11/23	25/11/23	9875689756	abhilaskashid456@gmail.com

Updated:



	_id ObjectId	passengerName String	from String	to String	date String	departureDate Stri
1	ObjectId('655ee9d19cd6c0a2238…	"Abhilash Kashid"	"Delhi"	"Bangalore"	"23/11/23"	"24/11/23"

Deleted:



flightDB.flightbookings

0
DOCUMENTS

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.