Report

Airline Management System

Aman Thakur
dept. of Data Science
University of Massachusetts
Dartmouth, MA

Part 1: Database driven application

I have developed the Airline Management System using the MERN (MongoDB, Express.js, React.js, Node.js) stack. It is a comprehensive solution aimed at automating various aspects of airline operations. By leveraging a database-driven approach, the system achieves enhanced scalability, real-time data processing, and efficient management of airline resources.

The primary use-case of the database-driven application revolves around centralizing airline operations, including flight schedules, crew management, passenger bookings, safety protocols, and performance tracking. This system facilitates easier coordination and communication across different modules, enabling stakeholders to make data-driven decisions for maximizing operational efficiency.

The implementation of database paradigms, particularly MongoDB, allows for flexible data modeling and storage. MongoDB's document-oriented architecture aligns well with the dynamic nature of airline data, enabling easy modification and expansion of schemas as per evolving business requirements. Additionally, MongoDB's support for horizontal scalability ensures the system can handle increasing data loads without compromising performance.

System Architecture:

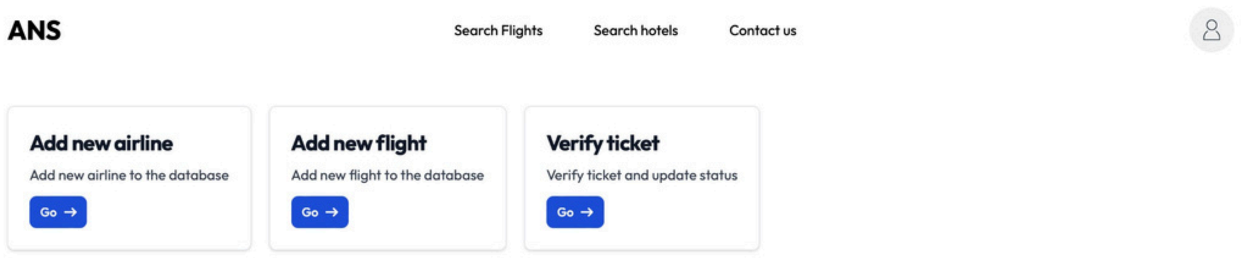
The system follows the MERN stack (MongoDB, Express.js, React, and Node.js) architecture:

- MongoDB(NoSQLDatabase):
    - Storesdataasflexibledocuments(BSONformat).
    - Allowsdynamicschemadesign,accommodatingevolvingrequirements.
    - Collectionsrepresententities(flights,crewmembers,passengers).
- Express.js(Backend):
    - HandlesHTTPrequestsandroutes.
    - CommunicateswithMongoDBfordataretrievalandstorage.
    - Implementsauthenticationandauthorization.
- React(Frontend):
    - Constructsuserinterfaces.
    - ComponentsinteractwiththebackendviaAPIendpoints.
    - Providesaresponsiveandintuitiveuserexperience.
- Node.js(RuntimeEnvironment):
    - Ensuresnon-blockingI/Oforhandlingconcurrentrequests.
    - Scalableandefficientforbackendoperations.

Part 2: Working Incorporation of Data Store (Database Engine)

The Airline Management System efficiently handles the activities of creating, loading, updating, and querying data by utilizing MongoDB as the database engine. Here are the main features of the programme as seen in screenshots:

- DataCreation:Newflightsandtheairlinescanbeaddedbytheadmin,including relevant details such as flight numbers, destinations, departure times, etc.
- Step1:

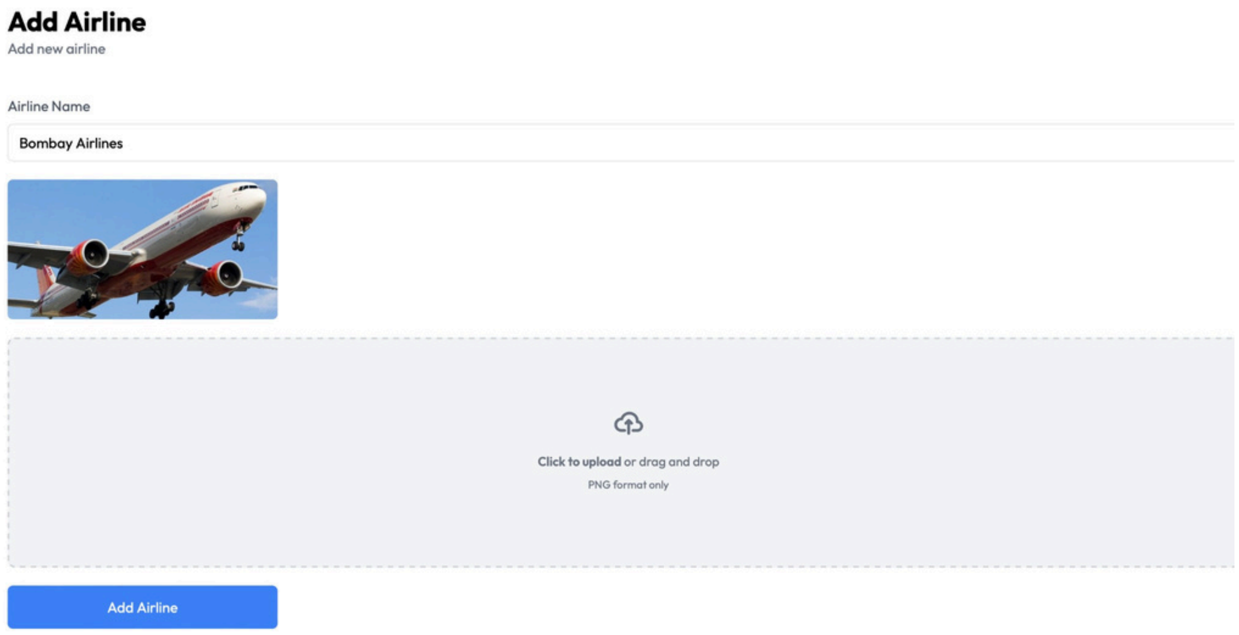


Add new airline
Add new airline to the database

Add new flight
Add new flight to the database

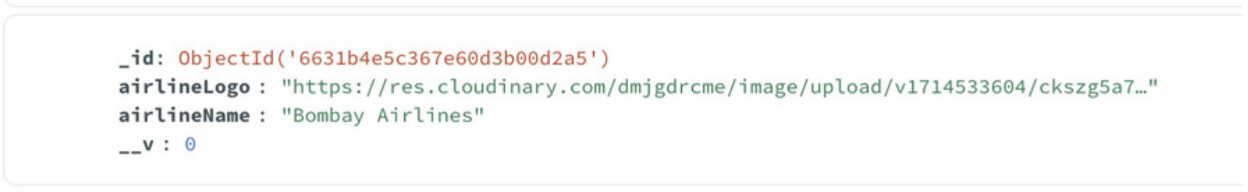
Verify ticket
Verify ticket and update status

- Step 2:



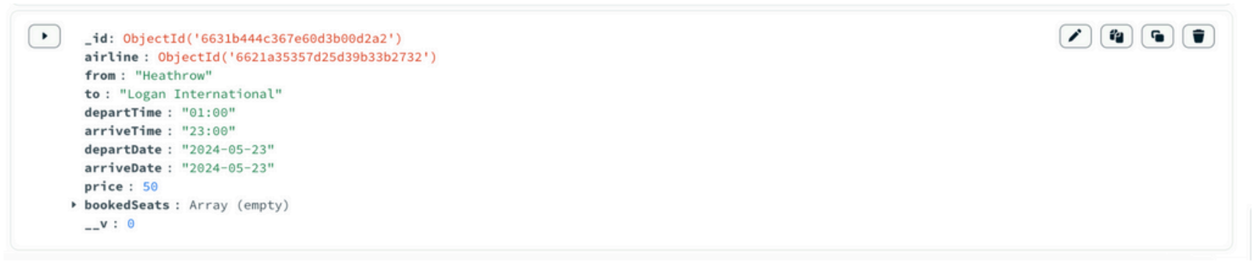
Add Airline
Add new airline

Airline Name
Bombay Airlines



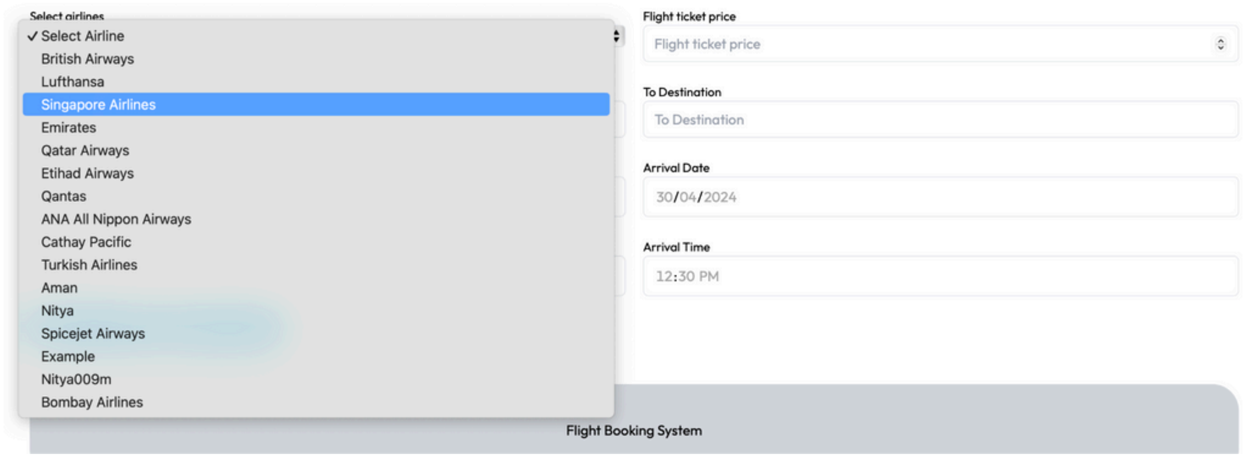
```
_id: ObjectId('6631b4e5c367e60d3b00d2a5')
airlineLogo : "https://res.cloudinary.com/dmjgdrcme/image/upload/v1714533604/ckszg5a7…"
airlineName : "Bombay Airlines"
__v : 0
```

- **Data Loading**: Available flight schedules loaded into the system, displaying available flights for users to book.

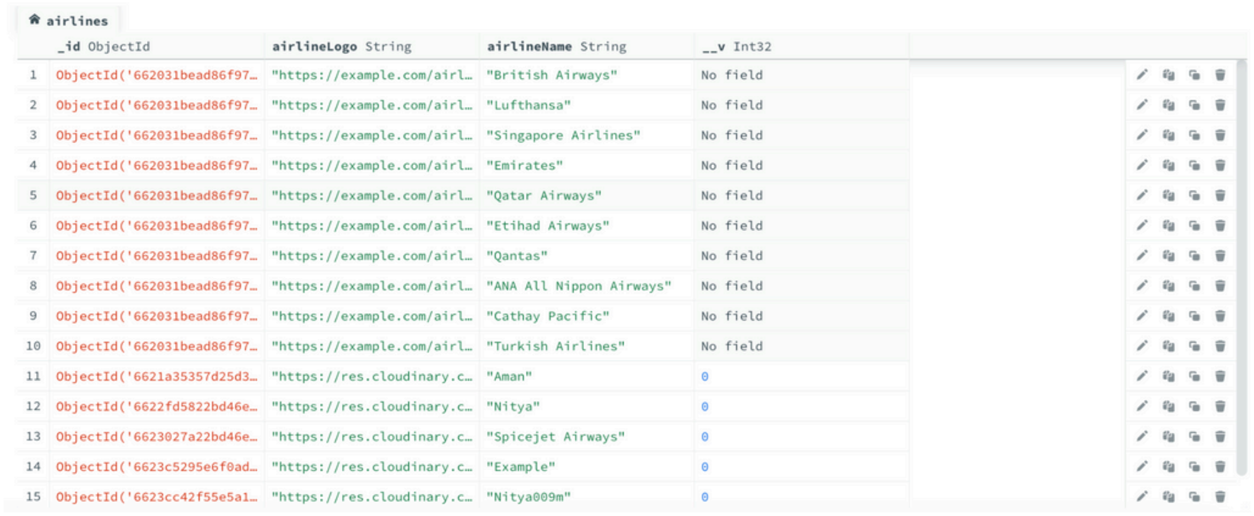


```
_id: ObjectId('6631b444c367e60d3b00d2a2')
airline : ObjectId('6621a35357d25d39b33b2732')
from : "Heathrow"
to : "Logan International"
departTime : "01:00"
arriveTime : "23:00"
departDate : "2024-05-23"
arriveDate : "2024-05-23"
price : 50
bookedSeats : Array (empty)
__v : 0
```

-
- Alltheavailableairlines:UIside



-
- Database:

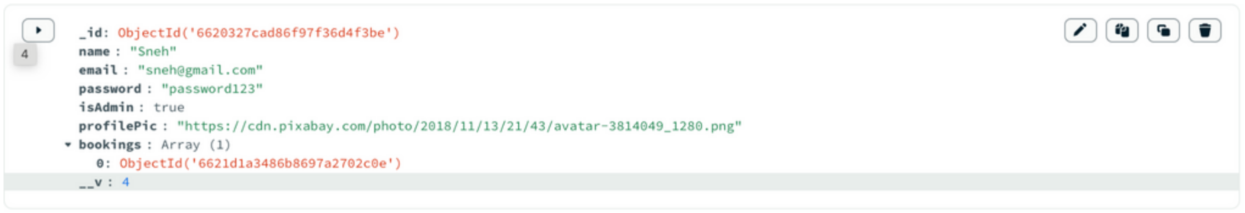


airlines

	_id ObjectId	airlineLogo String	airlineName String	__v Int32
1	ObjectId('662031bead86f97…	"https://example.com/airl…	"British Airways"	No field
2	ObjectId('662031bead86f97…	"https://example.com/airl…	"Lufthansa"	No field
3	ObjectId('662031bead86f97…	"https://example.com/airl…	"Singapore Airlines"	No field
4	ObjectId('662031bead86f97…	"https://example.com/airl…	"Emirates"	No field
5	ObjectId('662031bead86f97…	"https://example.com/airl…	"Qatar Airways"	No field
6	ObjectId('662031bead86f97…	"https://example.com/airl…	"Etihad Airways"	No field
7	ObjectId('662031bead86f97…	"https://example.com/airl…	"Qantas"	No field
8	ObjectId('662031bead86f97…	"https://example.com/airl…	"ANA All Nippon Airways"	No field
9	ObjectId('662031bead86f97…	"https://example.com/airl…	"Cathay Pacific"	No field
10	ObjectId('662031bead86f97…	"https://example.com/airl…	"Turkish Airlines"	No field
11	ObjectId('6621a35357d25d3…	"https://res.cloudinary.c…	"Aman"	0
12	ObjectId('6622fd5822bd46e…	"https://res.cloudinary.c…	"Nitya"	0
13	ObjectId('6623027a22bd46e…	"https://res.cloudinary.c…	"Spicejet Airways"	0
14	ObjectId('6623c5295e6f0ad…	"https://res.cloudinary.c…	"Example"	0
15	ObjectId('6623cc42f55e5a1…	"https://res.cloudinary.c…	"Nitya009m"	0

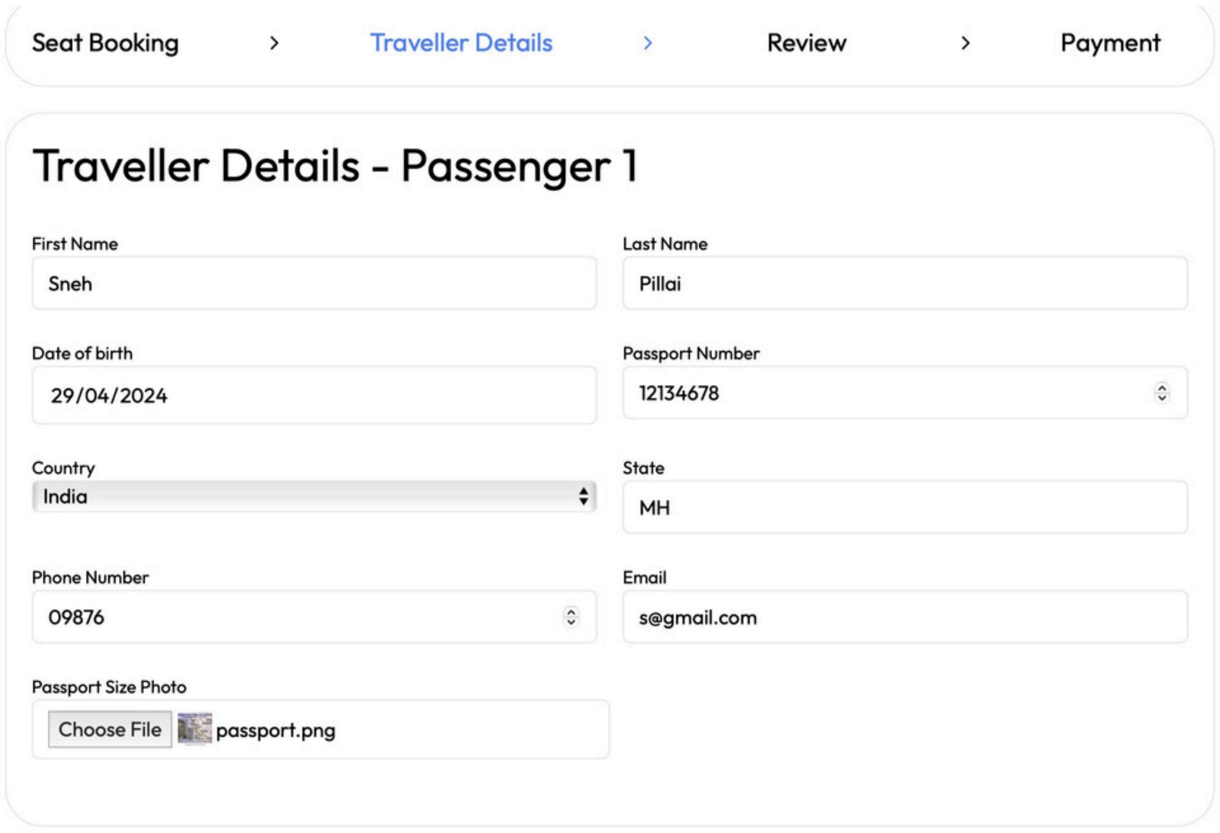
-

- **DataUpdating**: We can update the flight information, such as changing departure times or adding/removing available seats.
- Bookingcount1:



```
_id: ObjectId('6620327cad86f97f36d4f3be')
name : "Sneh"
email : "sneh@gmail.com"
password : "password123"
isAdmin : true
profilePic : "https://cdn.pixabay.com/photo/2018/11/13/21/43/avatar-3814049_1280.png"
bookings : Array (1)
  0: ObjectId('6621d1a3486b8697a2702c0e')
__v : 4
```

-
- Booking another flight for user Sneh:

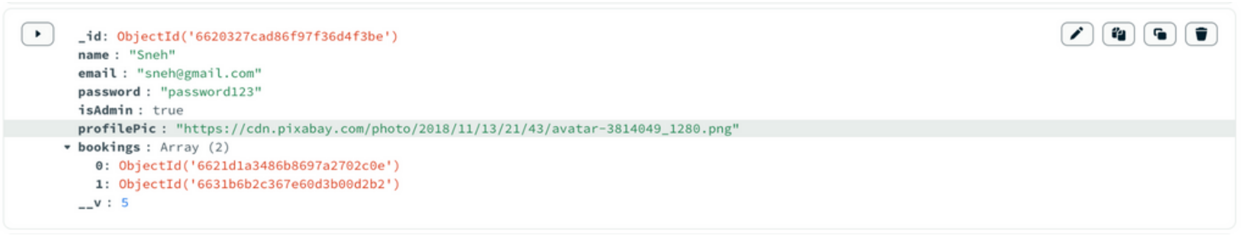


Traveller Details - Passenger 1

First Name: Sneh	Last Name: Pillai
Date of birth: 29/04/2024	Passport Number: 12134678
Country: India	State: MH
Phone Number: 09876	Email: s@gmail.com
Passport Size Photo: passport.png	

-

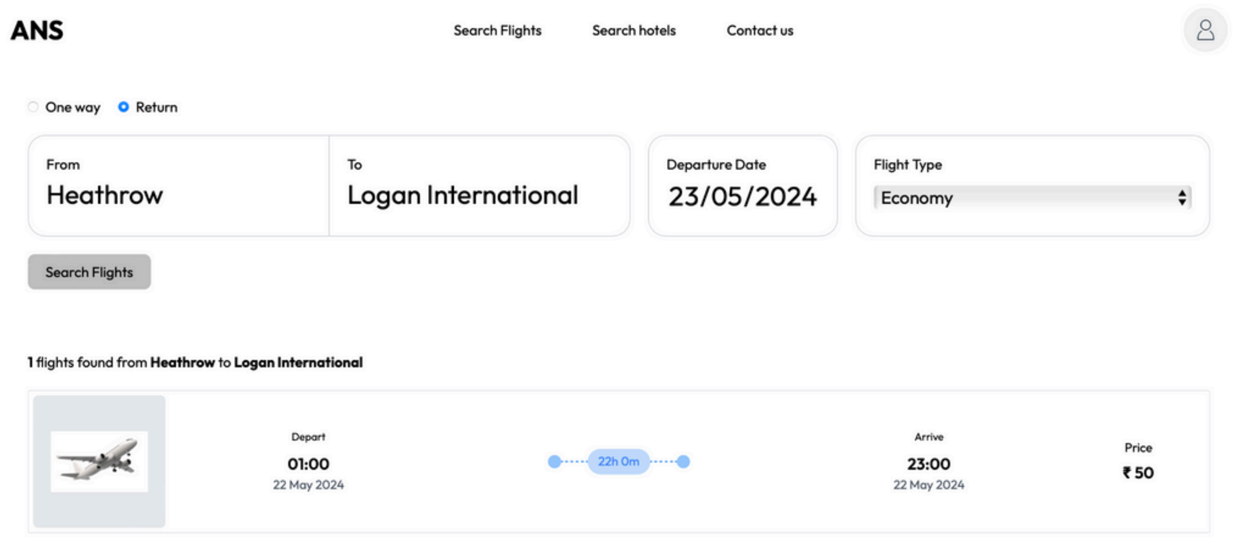
- Bookingcountupdatedto2:



```
_id: ObjectId('6620327cad86f97f36d4f3be')
name : "Sneh"
email : "sneh@gmail.com"
password : "password123"
isAdmin : true
profilePic : "https://cdn.pixabay.com/photo/2018/11/13/21/43/avatar-3814049_1280.png"
bookings : Array (2)
  0: ObjectId('6621d1a3486b8697a2702c0e')
  1: ObjectId('6631b6b2c367e60d3b00d2b2')
__v : 5
```

-

- **Data Querying**: Demonstrating the search functionality, where users can find available flights based on specified criteria like destination, departure time, or airline.



One way [✓ Return]

From: Heathrow
To: Logan International
Departure Date: 23/05/2024
Flight Type: Economy

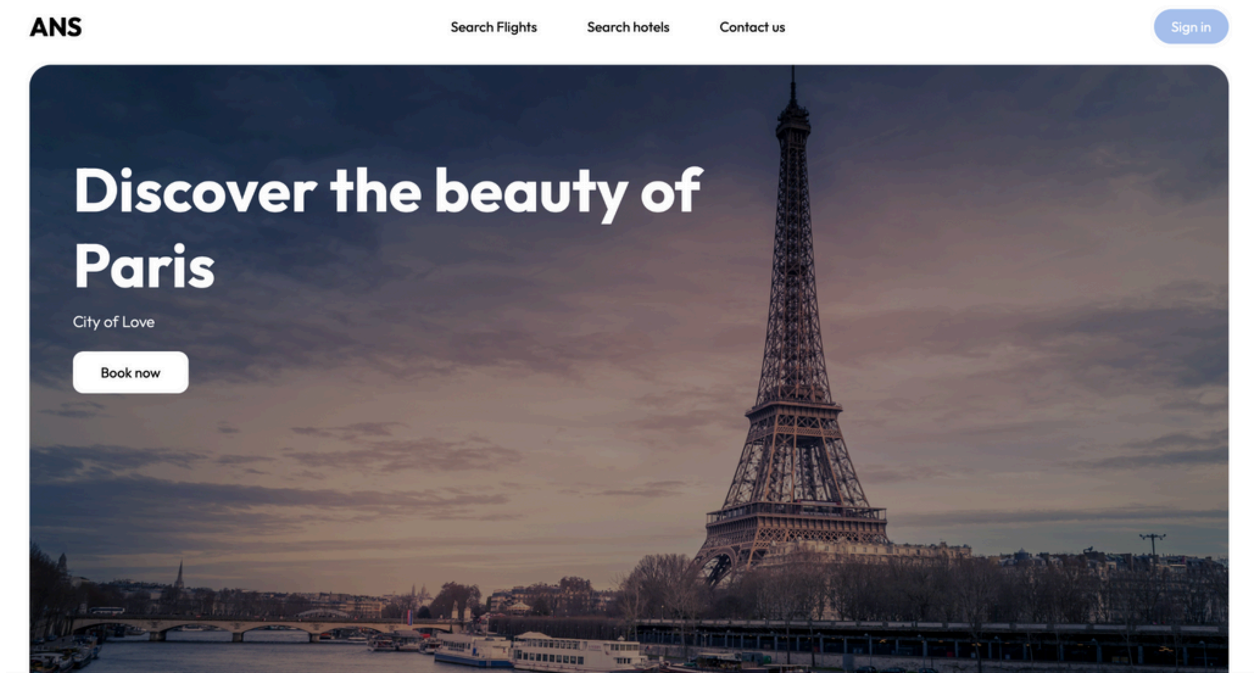
1 flights found from **Heathrow** to **Logan International**

Depart 01:00 22 May 2024 — 22h 0m — Arrive 23:00 22 May 2024 — Price ₹ 50

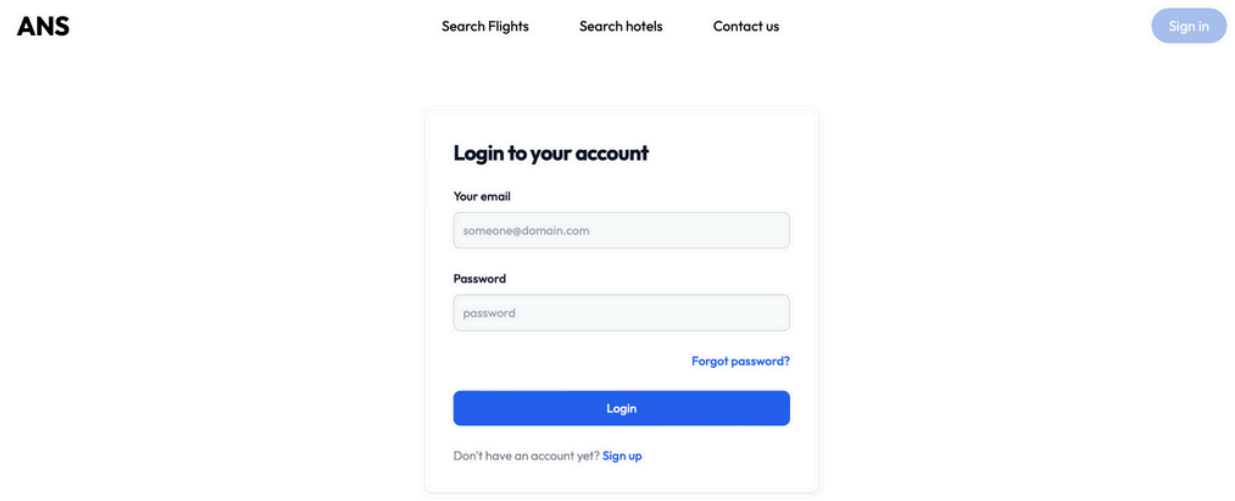
-

Part 3: Front-end Interface

I have created a user-friendly front end user interface platform to interact with the system's functionalities. The interface, developed using React.js, offers easy navigation and responsive design to enhance user experience. These are the screenshots of our front end design.

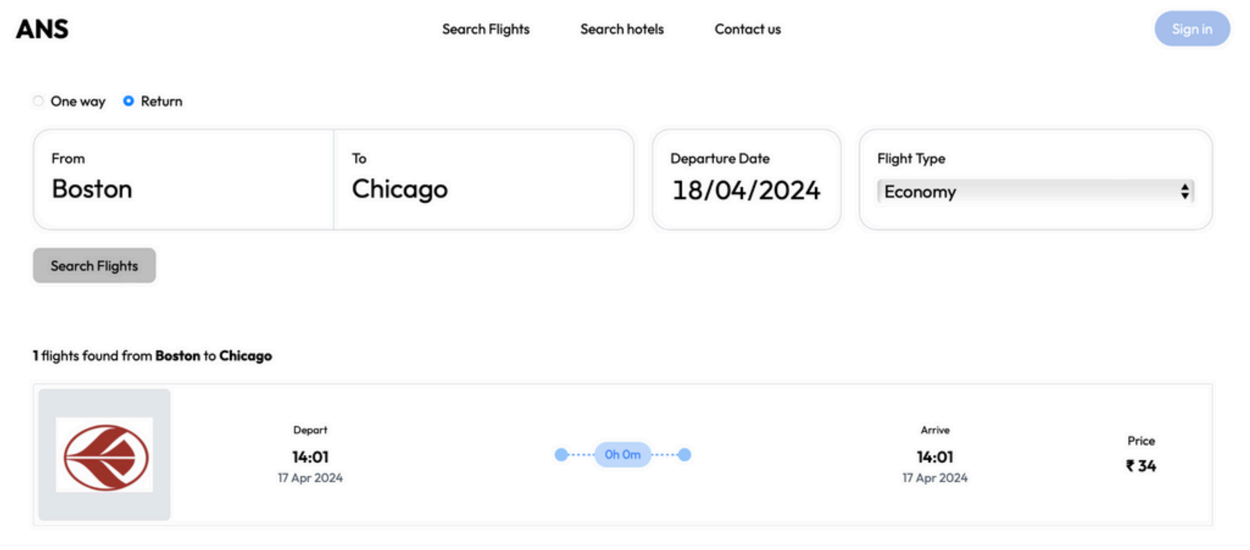


- Login Page: I have created a login page for both admin and users, showcasing the authentication process.



- Adding Flights and Airlines: Adding new flights or airlines can only be seen in the admin page. The screenshots are attached.

- FlightSearch:Examplesoftheflightsearchfeature,enablinguserstofind available flights which were added by the admin based on specified criteria.



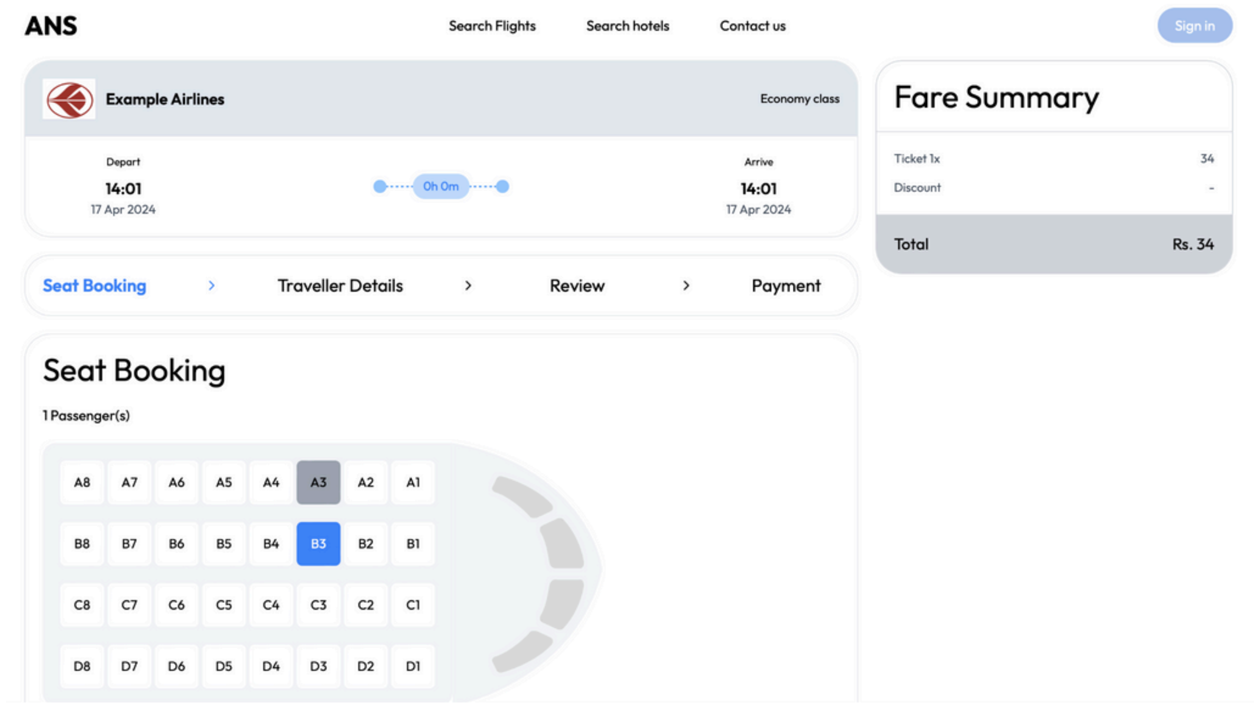
○ One way ◉ Return

From Boston | To Chicago | Departure Date 18/04/2024 | Flight Type Economy

1 flights found from **Boston** to **Chicago**

Depart 14:01 17 Apr 2024 — 0h 0m — Arrive 14:01 17 Apr 2024 — Price ₹ 34

- Flight Booking: Screenshots demonstrating the flight booking process, including flight selection, seat reservation, and payment.



Example Airlines — Economy class

Depart 14:01 17 Apr 2024 — 0h 0m — Arrive 14:01 17 Apr 2024

Fare Summary

Ticket 1x	34
Discount	-
Total	Rs. 34

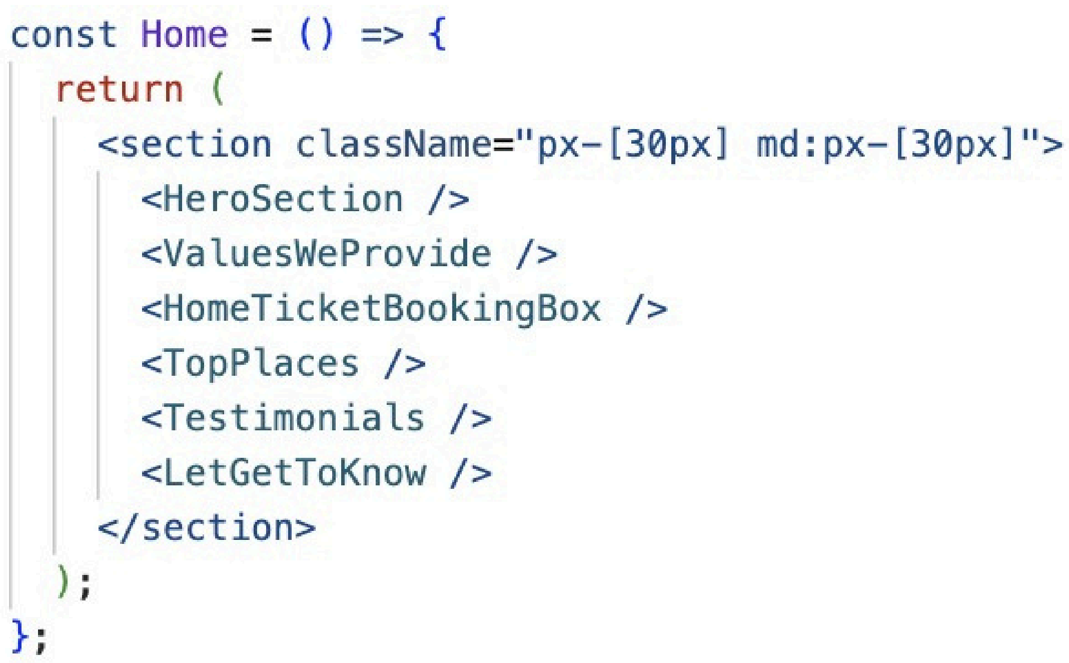
Seat Booking

1 Passenger(s)

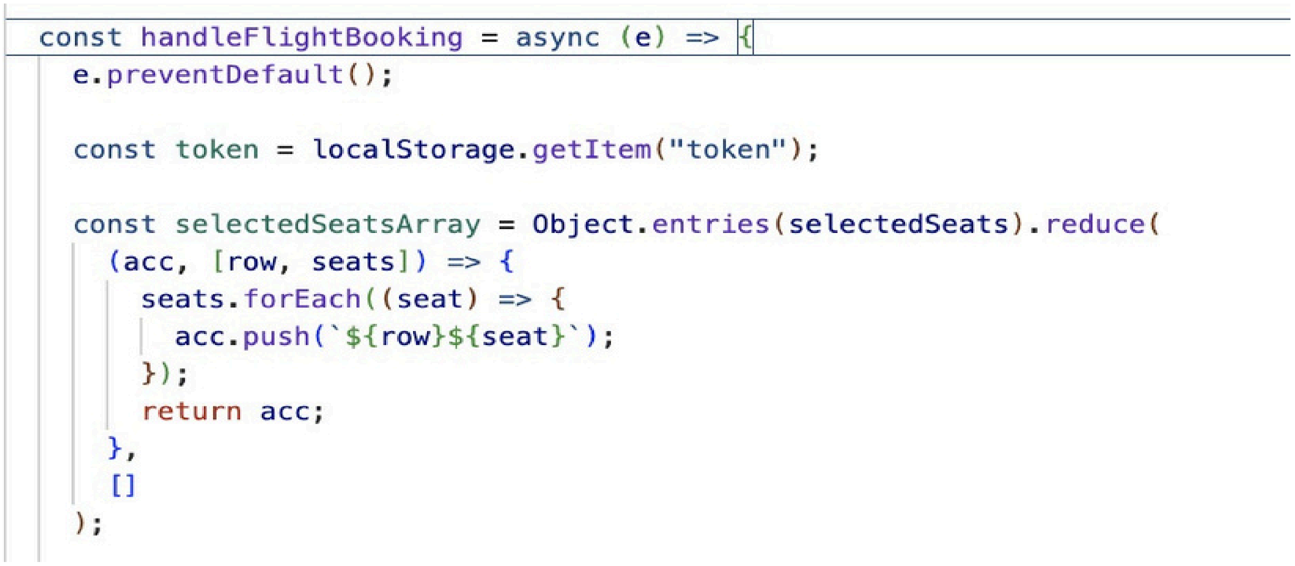
Part 4: Source Code

The source code of the Airline Management System contains various components, including the front-end interface, backend logic, and database communication. Screenshots of relevant code snippets showcase:

- InterfaceCode:ExamplesofReactcomponentsresponsibleforrenderingthe user interface and handling user interactions.

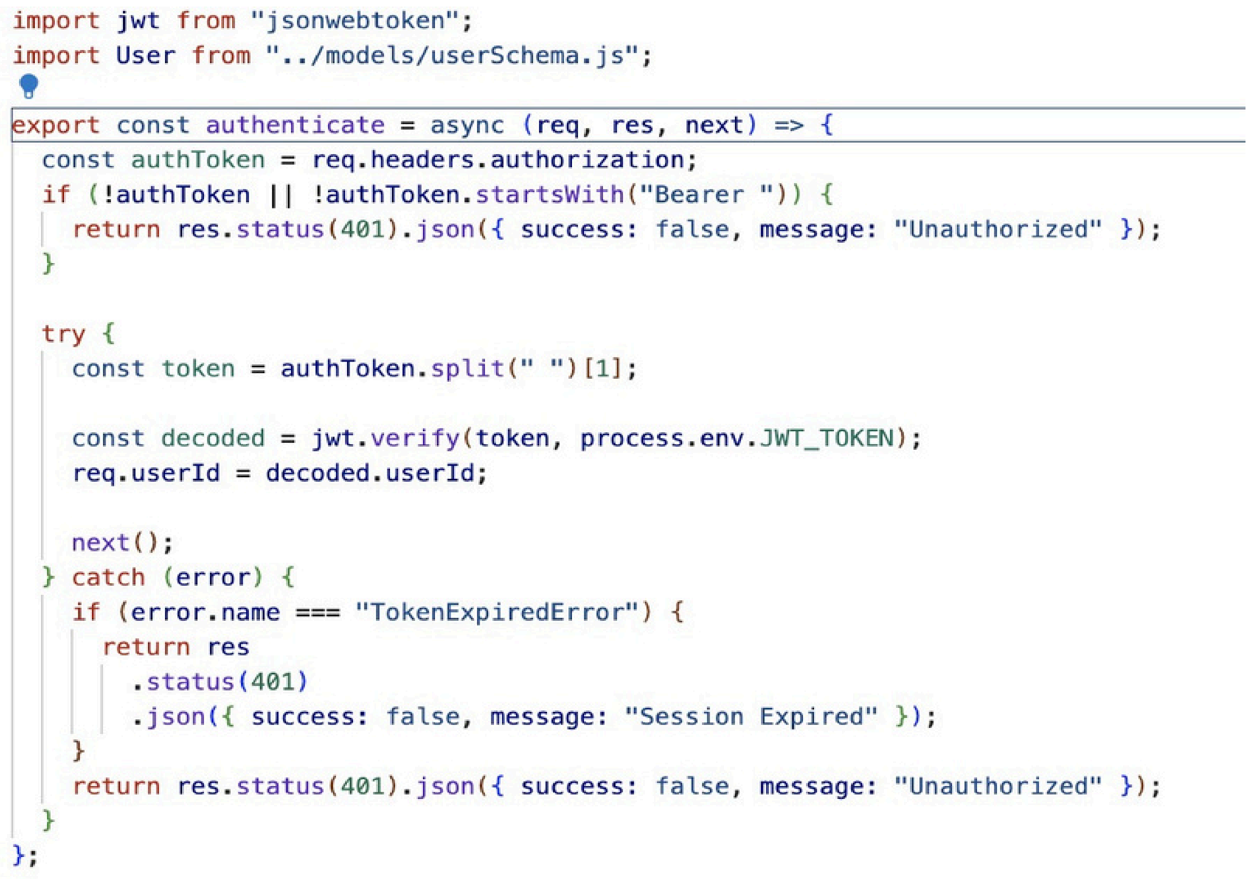


```
const Home = () => {
  return (
    <section className="px-[30px] md:px-[30px]">
      <HeroSection />
      <ValuesWeProvide />
      <HomeTicketBookingBox />
      <TopPlaces />
      <Testimonials />
      <LetGetToKnow />
    </section>
  );
};
```

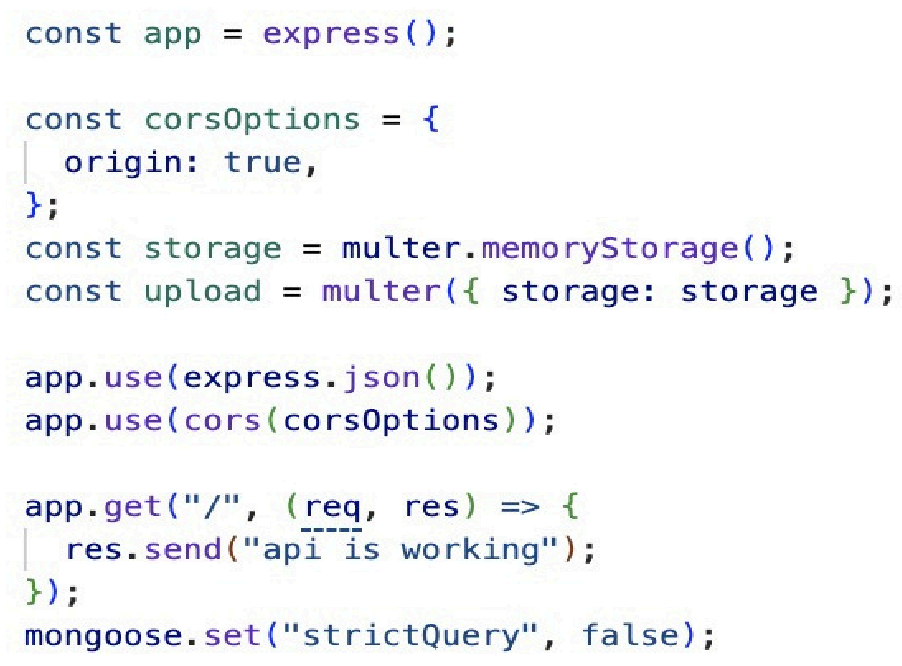


```
const handleFlightBooking = async (e) => {
  e.preventDefault();

  const token = localStorage.getItem("token");

  const selectedSeatsArray = Object.entries(selectedSeats).reduce(
    (acc, [row, seats]) => {
      seats.forEach((seat) => {
        acc.push(`${row}${seat}`);
      });
      return acc;
    },
    []
  );
```

- Backend Logic: Code snippets from the Node.js backend, illustrating server-side operations such as authentication, data validation, and request handling.



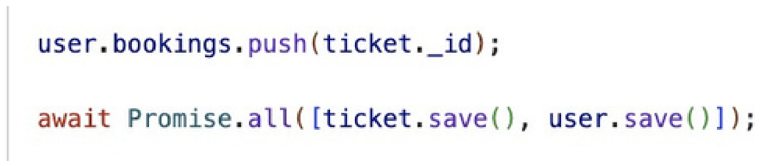
```
import jwt from "jsonwebtoken";
import User from "../models/userSchema.js";

export const authenticate = async (req, res, next) => {
  const authToken = req.headers.authorization;
  if (!authToken || !authToken.startsWith("Bearer ")) {
    return res.status(401).json({ success: false, message: "Unauthorized" });
  }

  try {
    const token = authToken.split(" ")[1];

    const decoded = jwt.verify(token, process.env.JWT_TOKEN);
    req.userId = decoded.userId;

    next();
  } catch (error) {
    if (error.name === "TokenExpiredError") {
      return res
        .status(401)
        .json({ success: false, message: "Session Expired" });
    }
    return res.status(401).json({ success: false, message: "Unauthorized" });
  }
};
```



```
const app = express();

const corsOptions = {
  origin: true,
};
const storage = multer.memoryStorage();
const upload = multer({ storage: storage });

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

app.get("/", (req, res) => {
  res.send("api is working");
});
mongoose.set("strictQuery", false);
```

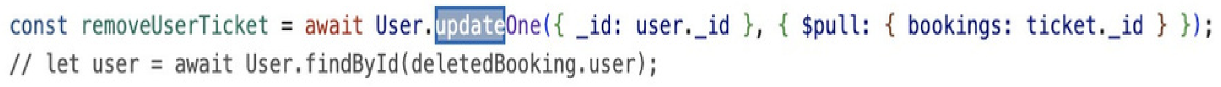
- Database Operations: Screenshots of code segments demonstrating CRUD (Create, Read, Update, Delete) operations with MongoDB, including data retrieval, insertion, updating, and deletion.

Create



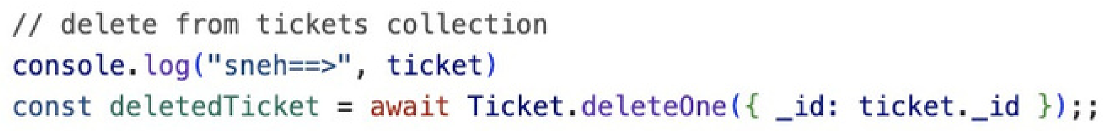
```
user.bookings.push(ticket._id);

await Promise.all([ticket.save(), user.save()]);
```

Update



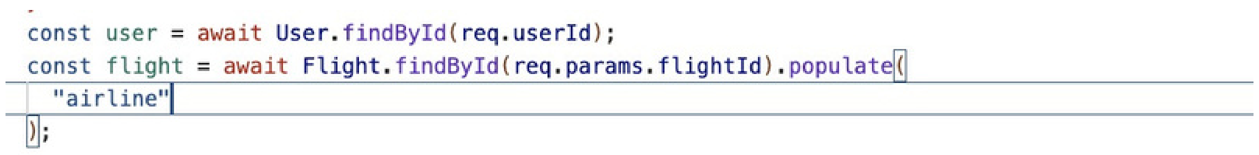
```
const removeUserTicket = await User.updateOne({ _id: user._id }, { $pull: { bookings: ticket._id } });
// let user = await User.findById(deletedBooking.user);
```

Delete

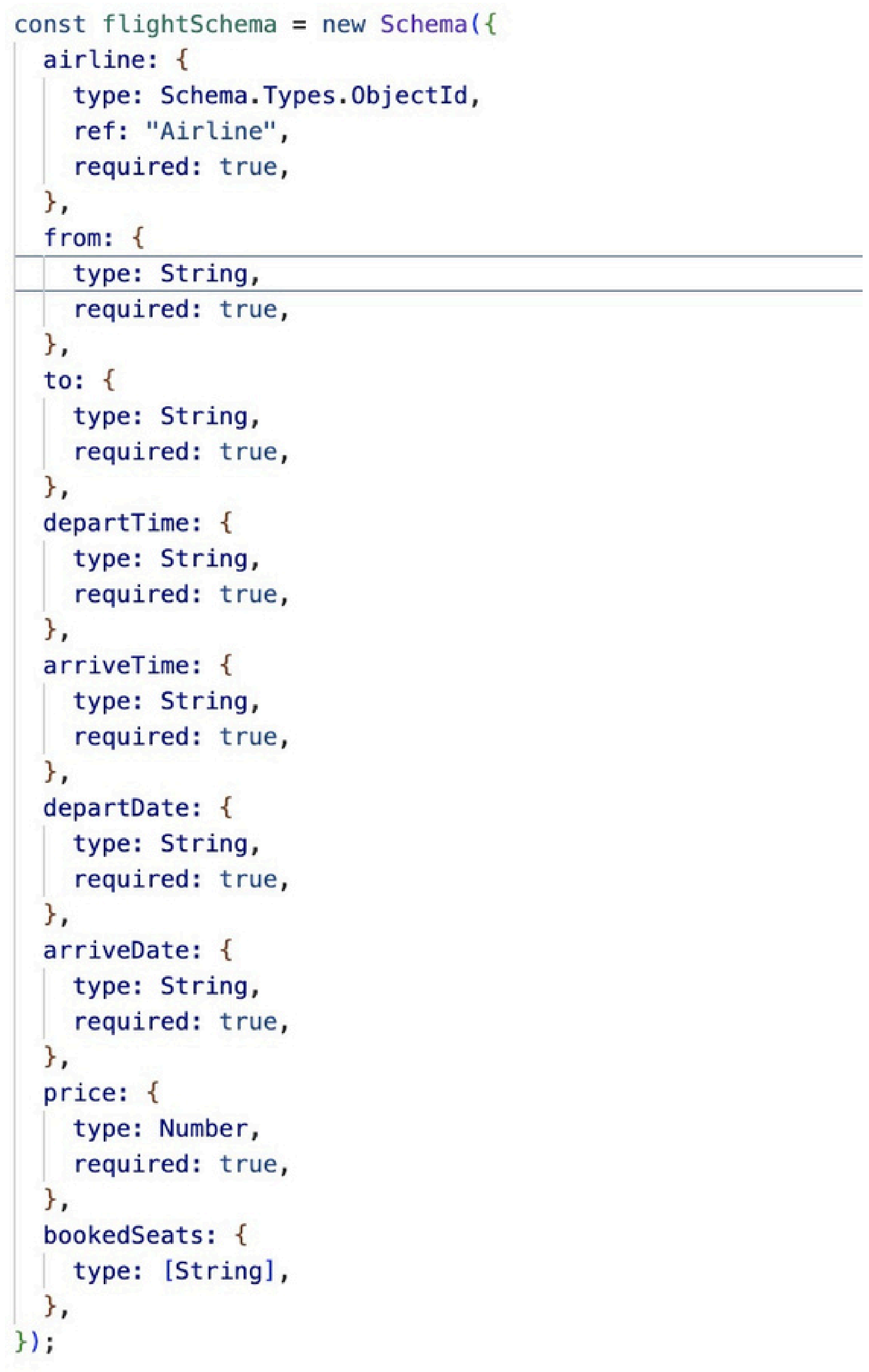


```
// delete from tickets collection
console.log("sneh==>", ticket)
const deletedTicket = await Ticket.deleteOne({ _id: ticket._id });;
```

Read



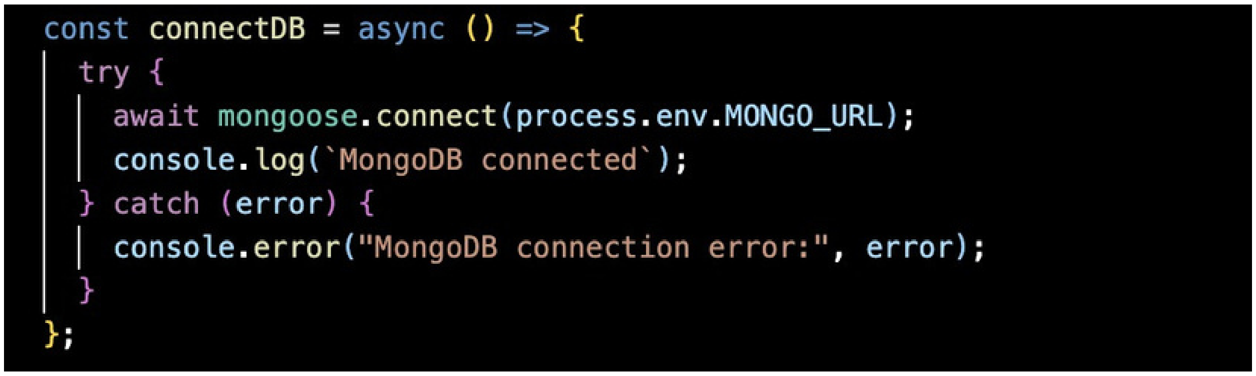
```
const user = await User.findById(req.userId);
const flight = await Flight.findById(req.params.flightId).populate(
  "airline"
);
```



```
const flightSchema = new Schema({
  airline: {
    type: Schema.Types.ObjectId,
    ref: "Airline",
    required: true,
  },
  from: {
    type: String,
    required: true,
  },
  to: {
    type: String,
    required: true,
  },
  departTime: {
    type: String,
    required: true,
  },
  arriveTime: {
    type: String,
    required: true,
  },
  departDate: {
    type: String,
    required: true,
  },
  arriveDate: {
    type: String,
    required: true,
  },
  price: {
    type: Number,
    required: true,
  },
  bookedSeats: {
    type: [String],
  },
});
```

Connection to MongoDB

I kept the mongo url in the environment file for security and seamless configuration.



```
const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URL);
    console.log(`MongoDB connected`);
  } catch (error) {
    console.error("MongoDB connection error:", error);
  }
};
```

Conclusion Remarks

I have developed the Airline Management System using the MERN stack which presents a robust solution for automating various aspects of airline operations. By using MongoDB as the database engine, the system ensures efficient data management, scalability, and real-time processing capabilities.The system addresses core functionalities of airline management, including flight scheduling, crew management, passenger bookings, safety protocols, and performance tracking. Using MongoDB, it enables flexible data modeling, accommodating evolving business requirements. By leveraging MongoDB, Express.js, React, and Node.js, the system ensures efficient handling of HTTP requests, seamless communication with the database, and responsive user interfaces. Node.js ensures non-blocking I/O for scalable backend operations. We have created a user friendly interface, the front-end interface developed using React.js offers good navigation and responsive design, enhancing the user experience. We have integrated the key functionalities such as flight search and booking into the interface. We have maintained the security by storing sensitive information like MongoDB connection URLs in environment files. This ensures seamless configuration while protecting sensitive data.

Therefore, the Airline Management System demonstrates a successful integration of database-driven architecture with the MERN stack, providing stakeholders with a comprehensive and efficient solution for managing airline operations.

References

https://www.mongodb.com/resources/languages/mern-stack-tutorial
https://www.mongodb.com/docs/atlas/getting-started/
https://www.mongodb.com/docs/manual/core/authentication/
https://jwt.io/introduction
https://www.npmjs.com/package/mongoose