TABLE OF CONTENT

Flight finder: Navigating your air travel options.

INTRODUCTION

In today's fast-paced world, travel has become an essential part of our lives, whether it's for business, leisure, or personal reasons. With the advent of technology, booking flights has become more accessible and convenient than ever before, thanks to flight booking apps. A flight booking app is a mobile application that allows users to search, compare, and book flights easily from the comfort of their smartphones or tablets. The primary purpose of a flight booking app is to streamline the process of planning and booking air travel. These apps provide users with a range of features and functionalities that simplify the entire journey, from searching for flights to managing bookings and receiving travel updates.

Description:

This Flight Booking APP is the ultimate digital platform designed to revolutionize the way you book flight tickets. With this app your flight travel experience will be elevated to new heights of convenience and efficiency. Our user-friendly web app empowers travelers to effortlessly discover, explore, and reserve flight tickets based on their unique preferences. Whether you're a frequent commuter or an occasional traveler, finding the perfect flight journey has never been easier.

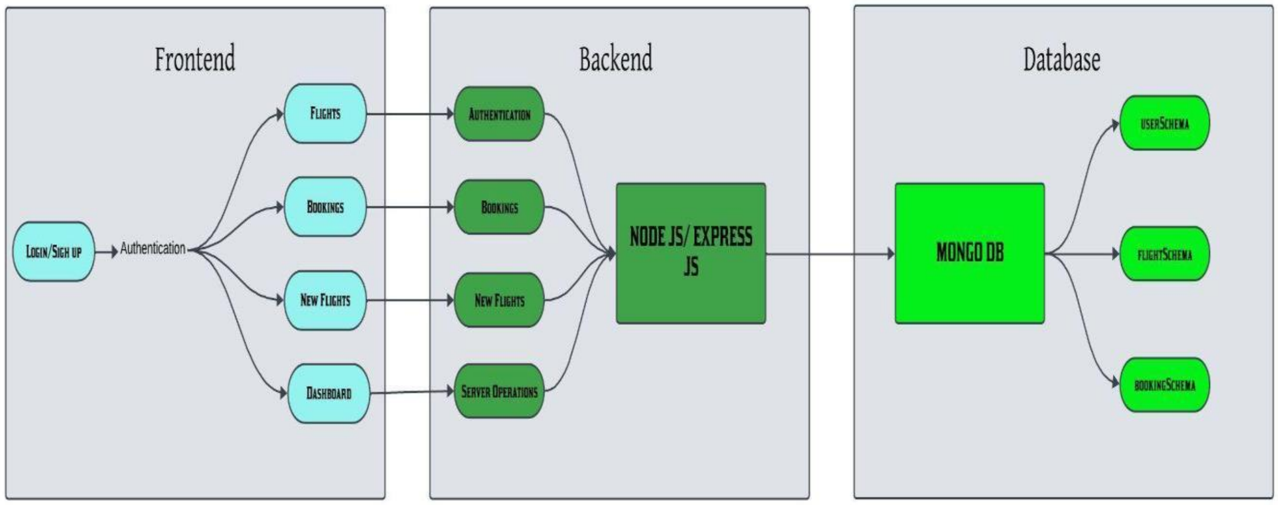
This successful flight booking app combines a user-friendly interface, efficient search and booking capabilities, personalized features, robust security measures, reliable performance, and continuous improvement based on user feedback.

Scenario

- John, a frequent traveler and business professional, needs to book a flight for an upcoming conference in Paris. He prefers using a flight booking app for its convenience and features.
- John opens the flight booking app on his smartphone and enters his travel details for Departure as New York City, Destination as Paris, Date of Departure on April 10th and return on April 15th and Class as Business class, Number of passengers as 1

- The app quickly retrieves available flight options based on John's preferences. He sees a range of choices from different airlines, including direct flights and those with layovers. The results show details such as price, airline, duration, and departure times.
- Using the app's filters, John narrows down the options to show only direct flights with convenient departure times. He also selects his preferred airline based on past experiences and loyalty programs.
- After choosing a flight, John proceeds to select his seat in the business class cabin. The app provides a seat map with available seats highlighted, allowing John to pick a window seat with extra legroom.
- John securely enters his payment information using the app's integrated payment gateway. The app processes the payment and generates a booking confirmation with his e-ticket and itinerary details.
- This scenario demonstrates how a flight booking app streamlines the entire travel process for users like John, offering convenience, customization, and real-time assistance throughout their journey.

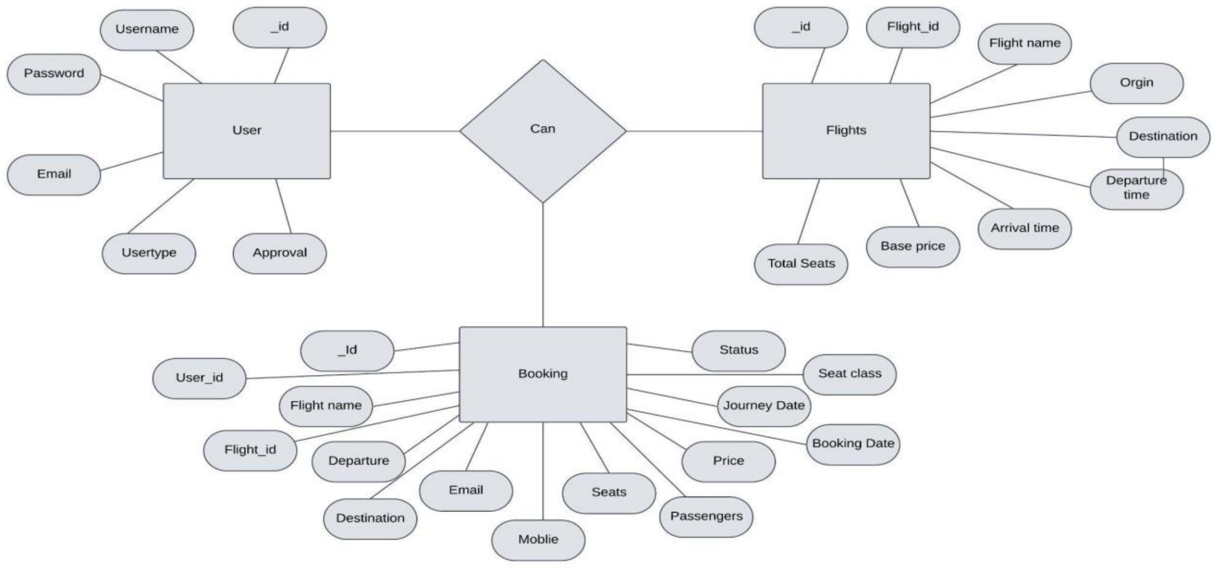
TECHNICAL ARCHITECTURE



In this architecture diagram:

- The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Flight Search, and Booking.
- The backend is represented by the "Backend" section, consisting of API endpoints for Users, Flights, Admin and Bookings. It also includes Admin Authentication and an Admin Dashboard.
- The Database section represents the database that stores collections for Users, Flights, and Flight Bookings.

ER DIAGRAM



The flight booking ER-diagram represents the entities and relationships involved in a flight booking system. It illustrates how users, bookings, flights, passengers, and payments are interconnected. Here is a breakdown of the entities and their relationships:

USER: Represents the individuals or entities who book flights. A customer can place multiple bookings and make multiple payments.

BOOKING: Represents a specific flight booking made by a customer. A booking includes a particular flight details and passenger information. A customer can have multiple bookings.

FLIGHT: Represents a flight that is available for booking. Here, the details of flight will be provided and the users can book them as much as the available seats.

ADMIN: Admin is responsible for all the backend activities. Admin manages all the bookings, adds new flights etc.

PREREQUISITES:

To develop a full-stack flight booking app using React JS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Node.js and npm: Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

- Download: https://nodejs.org/en/download/
- Installation instructions: https://nodejs.org/en/download/package-manager/

MongoDB: Set up a MongoDB database to store hotel and booking information. Install MongoDB locally using a cloud-based MongoDB service.

- Download: https://www.mongodb.com/try/download/community
- Installation instructions: https://docs.mongodb.com/manual/installation/

Express.js: Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

- Installation: Open your command prompt or terminal and run the following command: npm install express

React.js: React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. To install React.js, a JavaScript library for building user interfaces, follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

Front-end Framework: Utilize Angular to build the user-facing part of the application, including product listings, booking forms, and user interfaces for the admin dashboard.

Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

- Git: Download and installation instructions can be found at: https://gitscm.com/downloads

Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

- Visual Studio Code: Download from https://code.visualstudio.com/download
- Sublime Text: Download from https://www.sublimetext.com/download
- WebStorm: Download from https://www.jetbrains.com/webstorm/download

To Connect the Database with Node JS go through the below provided link:

- https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb

To run the existing Flight Booking App project downloaded from GitHub:

Follow below steps:

Clone the repository:

- Open your terminal or command prompt.
- Navigate to the directory where you want to store the e-commerce app.
- Execute the following command to clone the repository:

  Git clone: Flightfinder - Navigating Your Air Travel Options

Install Dependencies:

- Navigate into the cloned repository directory:

  cd Flight-Booking-App-MERN

- Install the required dependencies by running the following command:

  npm install

Start the Development Server:

- To start the development server, execute the following command:

  npm run dev or npm run start

- The e-commerce app will be accessible at http://localhost:3000 by default. You can change the port configuration in the .env file if needed.

Access the App:

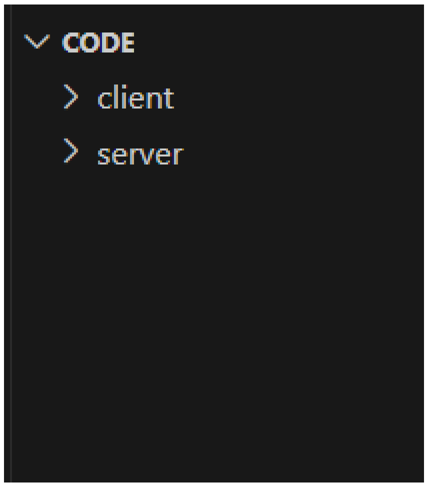
• Open your web browser and navigate to http://localhost:3000

• You should see the flight booking app's homepage, indicating that the installation and the setup was successful.

You have successfully installed and set up the flight booking app on your local machine. You can now proceed with further customization, development, and testing as needed.

PROJECT STRUCTURE:

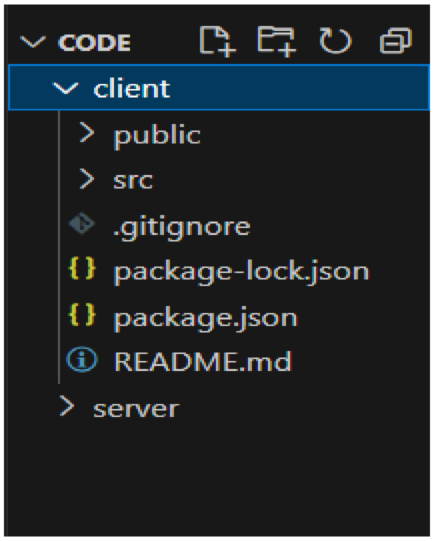
- Inside the Flight Booking app directory, we have the following folders



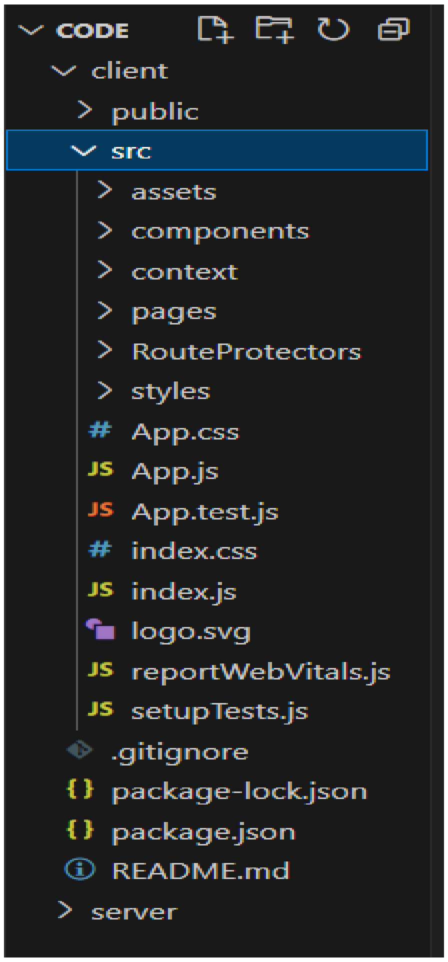
```
CODE
  client
  server
```

- **Client directory:**

  The below directory structure represents the directories and files in the client folder (front end) where, react js is used along with Api's.



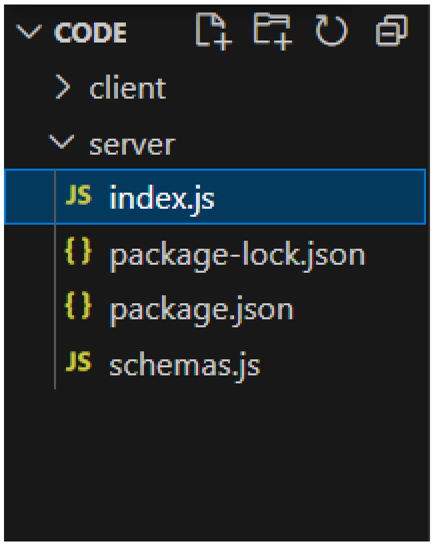
```
CODE
  client
    public
    src
    .gitignore
    package-lock.json
    package.json
    README.md
  server
```



```
CODE
  client
    public
    src
      assets
      components
      context
      pages
      RouteProtectors
      styles
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
  server
```

- **Server directory:**

  The below directory structure represents the directories and files in the server folder (back end) where, node js, express is and MongoDB are used along with Api.



```
CODE
  client
  server
    index.js
    package-lock.json
    package.json
    schemas.js
```

APPLICATION FLOW:

- **USER:**
    - Create their account.
    - Search for his destination.
    - Search for flights as per his time convenience.
    - Book a flight with a particular seat.
    - Make his payment.
    - And also cancel bookings.
- **ADMIN**
    - Manages all bookings.
    - Adds new flights and services.
    - Monitor User activity.

PROJECT FLOW:

Milestone 1: Project setup and configuration.

Folder setup:

To start the project from scratch, firstly create frontend and backend folders to install essential libraries and write code.

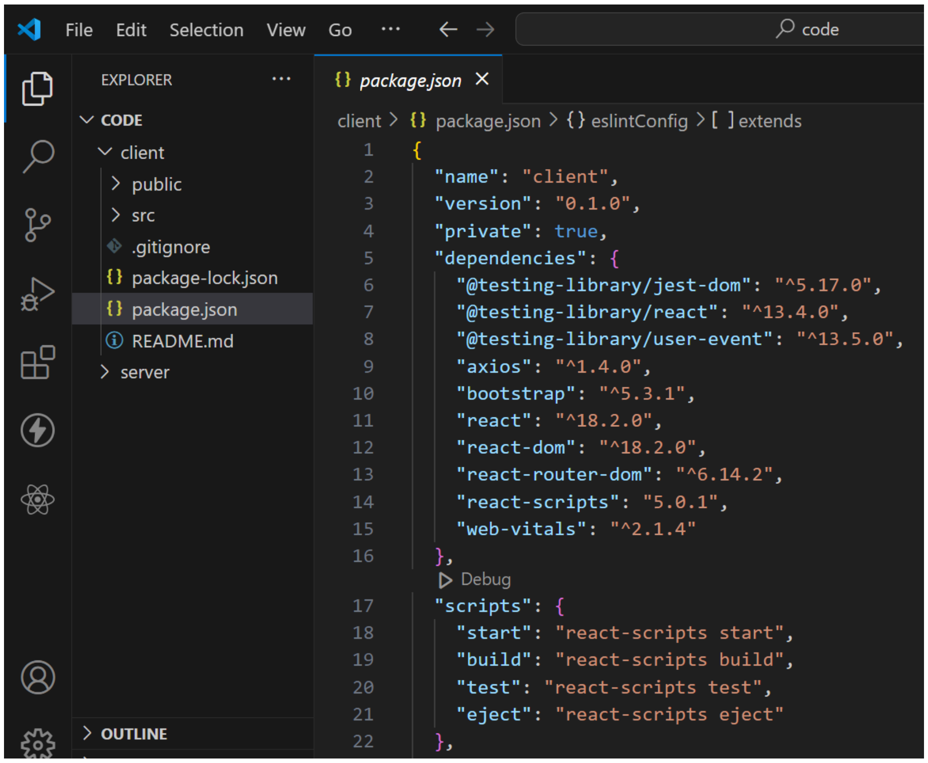
- client
- Server

Installation of required tools:

Now, open the frontend folder to install all the necessary tools we use.
For frontend, we use:

- React Js
- Bootstrap
- Axios

After installing all the required libraries, we'll be seeing the package.json file similar to the one below.



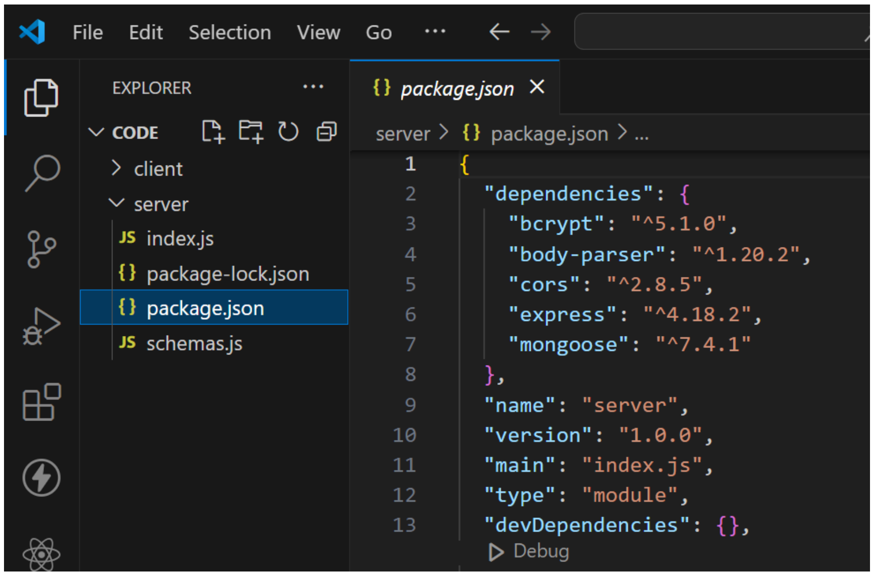
```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.3.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.14.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
```

Now, open the backend folder to install all the necessary tools that we use in the backend.

For backend, we use:

- bcrypt
- body-parser
- cors
- express
- mongoose

After installing all the required libraries, we'll be seeing the package. Json file similar to the one below.



```
{
  "dependencies": {
    "bcrypt": "^5.1.0",
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "mongoose": "^7.4.1"
  },
  "name": "server",
  "version": "1.0.0",
  "main": "index.js",
  "type": "module",
  "devDependencies": {},
```

Milestone 2: Backend Development:

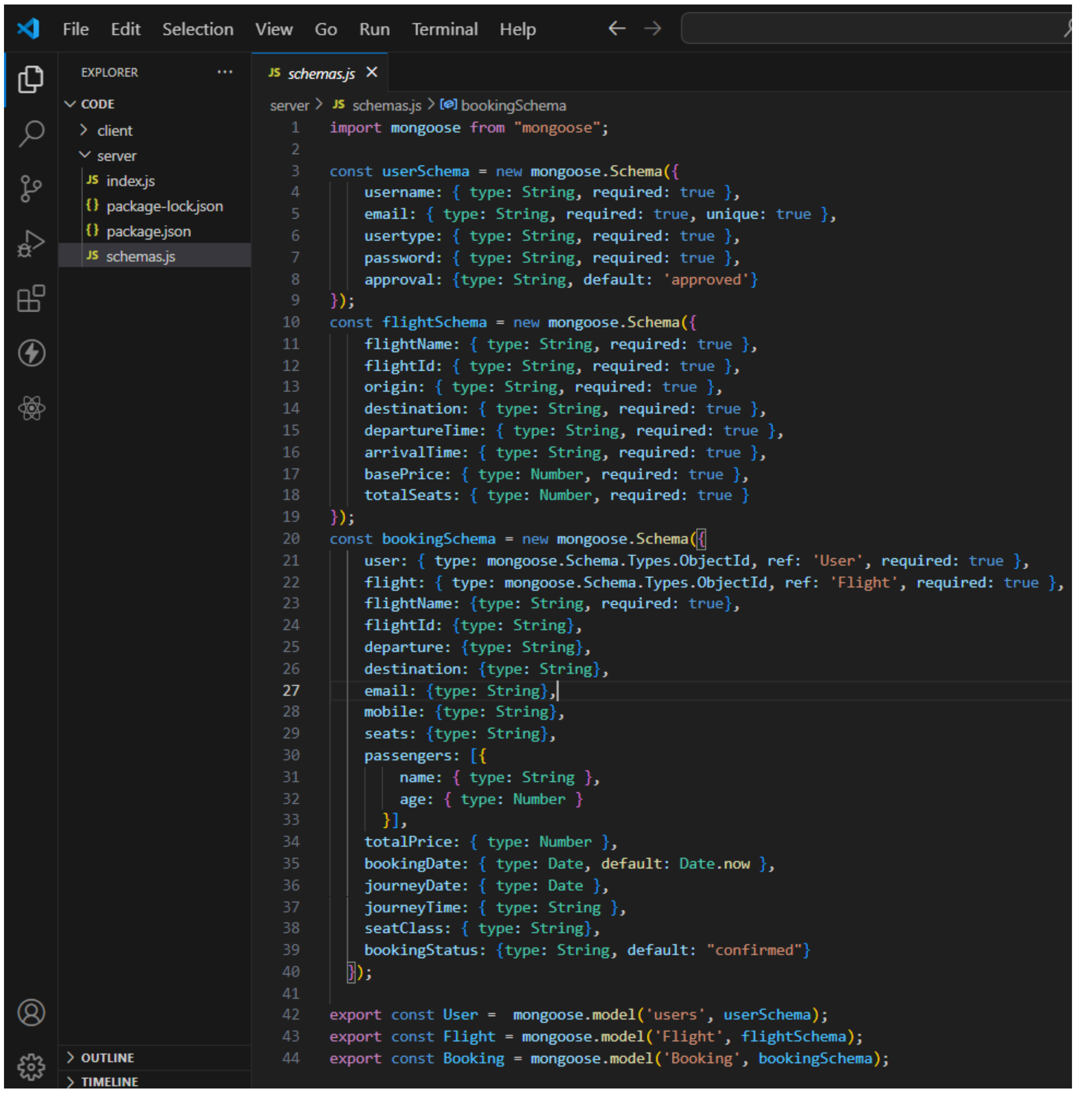
1. **Database Configuration:**
   - Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.
   - Create a database and define the necessary collections for flights, users, bookings, and other relevant data.
2. **Create Express.js Server:**
   - Set up an Express.js server to handle HTTP requests and serve API endpoints.
   - Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
3. **Define API Routes:**
   - Create separate route files for different API functionalities such as flights, users, bookings, and authentication.
   - Define the necessary routes for listing flights, handling user registration and login managing bookings, etc.
   - Implement route handlers using Express.js to handle requests and interact with the database.

4. **Implement Data Models:**
    - Define Mongoose schemas for the different data entities like flights, users, and bookings.
    - Create corresponding Mongoose models to interact with the MongoDB database. Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.
5. **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.
6. **Handle new Flights and Bookings:**
    - Create routes and controllers to handle new flight listings, including fetching flight data from the database and sending it as a response.
    - Implement booking functionality by creating routes and controllers to handle booking requests, including validation and database updates.
7. **Admin Functionality:**
    - Implement routes and controllers specific to admin functionalities such as adding flights, managing user bookings, etc.
    - Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.
8. **Error Handling:**
    - Implement error handling middleware to catch and handle any errors that occur during the API requests.
    - Return appropriate error responses with relevant error messages and HTTP status codes.

Milestone 3: Database development

- **Configure schema**

  Firstly, configure the Schemas for MongoDB database, to store the data in such a pattern. Use the data from the ER diagrams to create the schemas. The Schemas for this application look alike to the one provided below.



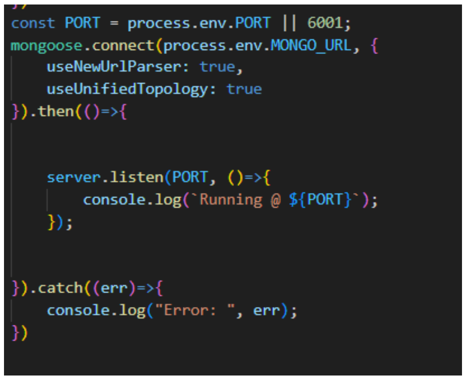
```javascript
import mongoose from "mongoose";

const userSchema = new mongoose.Schema({
    username: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    usertype: { type: String, required: true },
    password: { type: String, required: true },
    approval: {type: String, default: 'approved'}
});
const flightSchema = new mongoose.Schema({
    flightName: { type: String, required: true },
    flightId: { type: String, required: true },
    origin: { type: String, required: true },
    destination: { type: String, required: true },
    departureTime: { type: String, required: true },
    arrivalTime: { type: String, required: true },
    basePrice: { type: Number, required: true },
    totalSeats: { type: Number, required: true }
});
const bookingSchema = new mongoose.Schema({
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    flight: { type: mongoose.Schema.Types.ObjectId, ref: 'Flight', required: true },
    flightName: {type: String, required: true},
    flightId: {type: String},
    departure: {type: String},
    destination: {type: String},
    email: {type: String},
    mobile: {type: String},
    seats: {type: String},
    passengers: [{
        name: { type: String },
        age: { type: Number }
      }],
    totalPrice: { type: Number },
    bookingDate: { type: Date, default: Date.now },
    journeyDate: { type: Date },
    journeyTime: { type: String },
    seatClass: { type: String},
    bookingStatus: {type: String, default: "confirmed"}
});

export const User =  mongoose.model('users', userSchema);
export const Flight = mongoose.model('Flight', flightSchema);
export const Booking = mongoose.model('Booking', bookingSchema);
```

- **Connect database to backend**

  Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.



```javascript
const PORT = process.env.PORT || 6001;
mongoose.connect(process.env.MONGO_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(()=>{


    server.listen(PORT, ()=>{
        console.log(`Running @ ${PORT}`);
    });


}).catch((err)=>{
    console.log("Error: ", err);
})
```

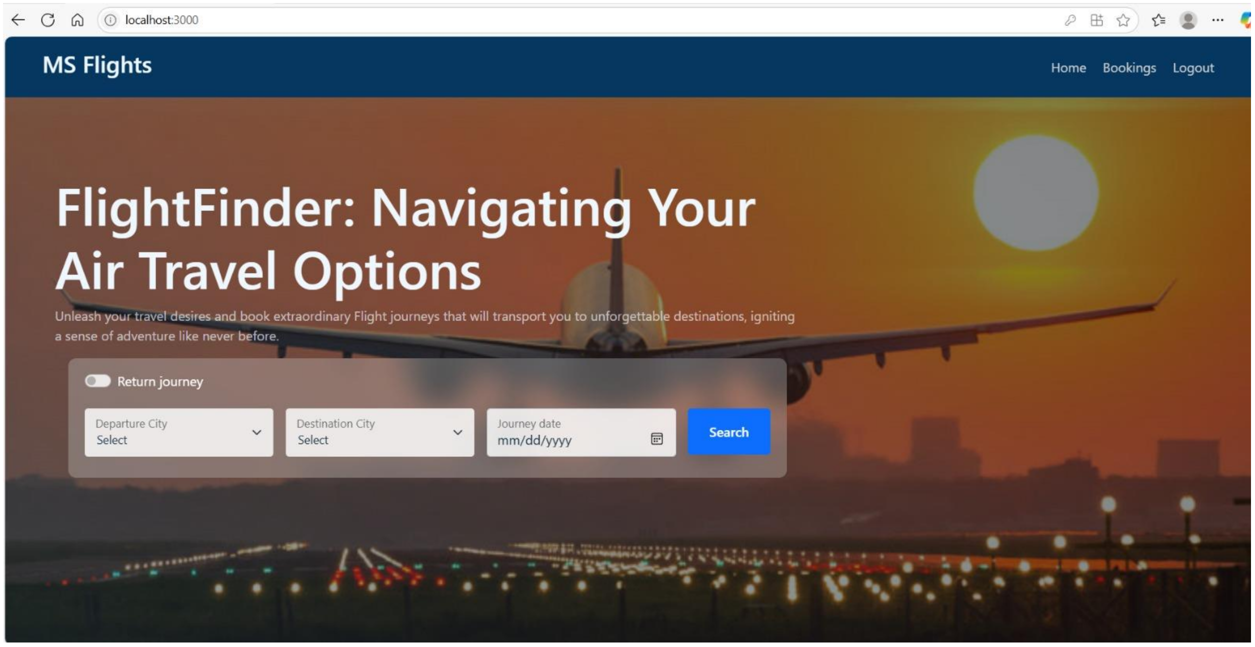
Milestone 4: Frontend development.

1. **Login/Register**
    - Create a Component which contains a form for taking the username and password.
    - If the given inputs match the data of user or admin or flight operator then navigate it to their respective home page
2. **Flight Booking (User):**
    - In the frontend, we implemented all the booking code in a modal. Initially, we need to implement flight searching feature with inputs of Departure city, Destination, etc.,
    - Flight Searching code: With the given inputs, we need to fetch the available flights. With each flight, we add a button to book the flight, which redirects to the flight-Booking page.
3. **Fetching user bookings:**
    - In the bookings page, along with displaying the past bookings, we will also provide an option to cancel that booking.
4. **Add new flight(Admin):**
    - Now, in the admin dashboard, we provide functionality to add new flights.
    - We create a html form with required inputs for the new flight and then send an HTTP request to the server to add it to the database.
5. **Update Flight:**
    - Here, in the admin dashboard, we will update the flight details in case if we want to make any edits to it
    - Along with this, implement additional features to view all flights, bookings, and users in the admin dashboard.

Milestone 5: Project Implementation.

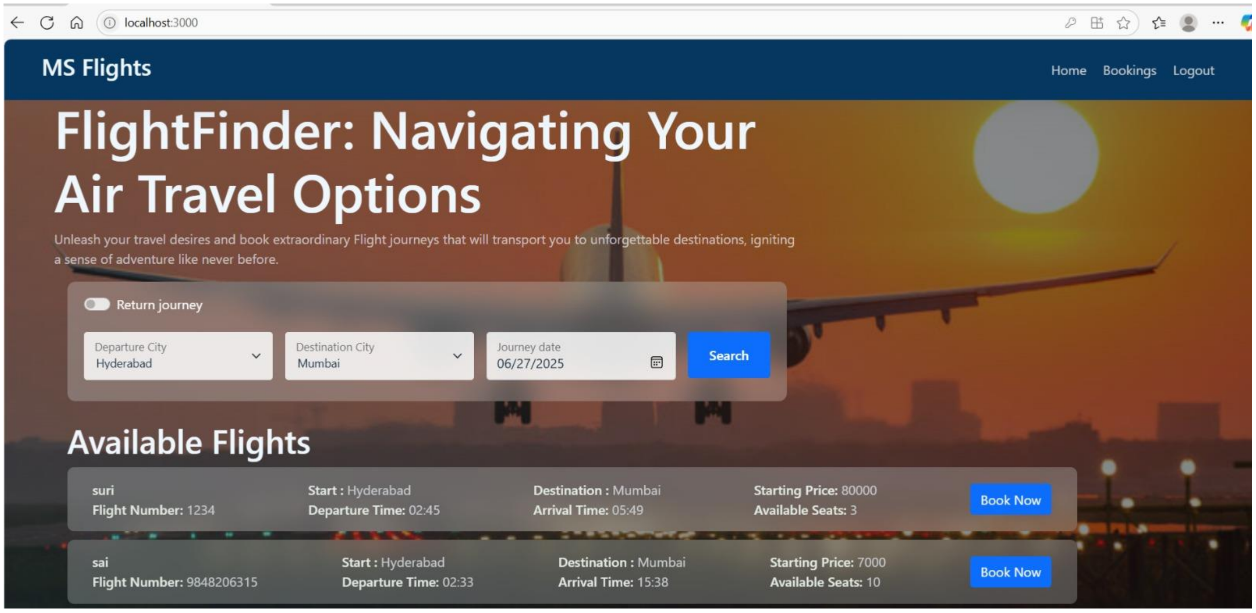
Finally, after finishing coding the projects we run the whole project to test it's working process and look for bugs. Now, let's have a final look at the working of our video conference application

- **Landing page UI**



FlightFinder: Navigating Your Air Travel Options

Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.



FlightFinder: Navigating Your Air Travel Options

Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.

Return journey

Departure City: Hyderabad

Destination City: Mumbai

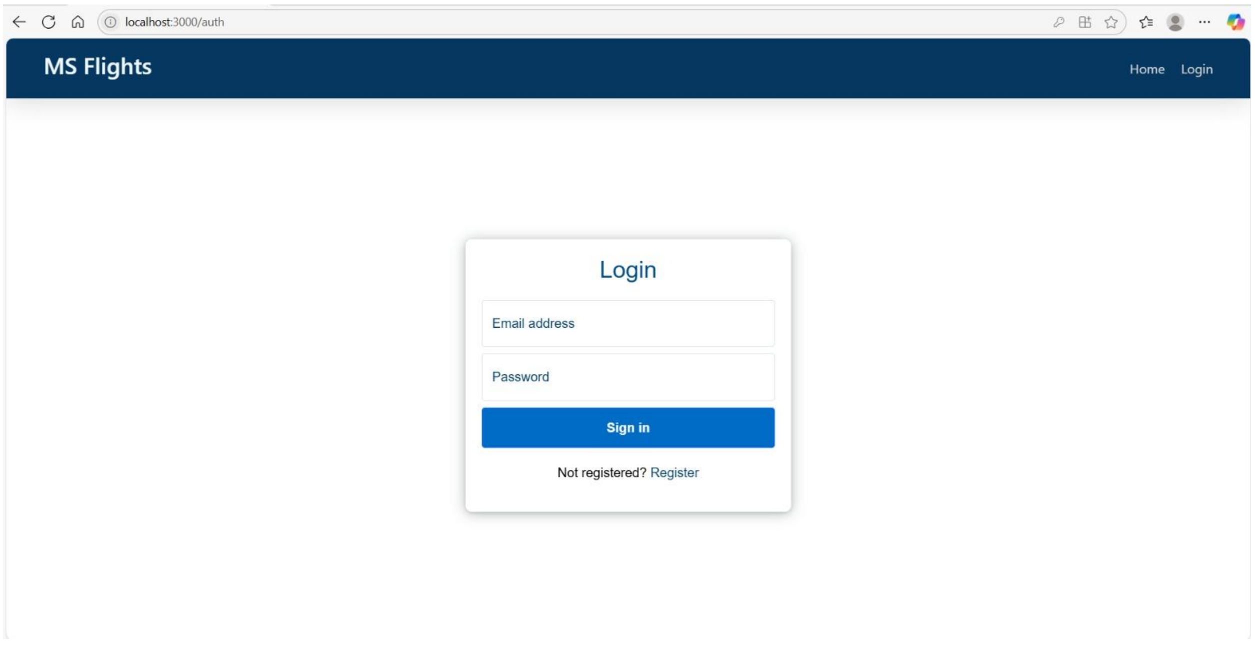
Journey date: 06/27/2025

Available Flights

suri
Flight Number: 1234
Start : Hyderabad
Departure Time: 02:45
Destination : Mumbai
Arrival Time: 05:49
Starting Price: 80000
Available Seats: 3

sai
Flight Number: 9848206315
Start : Hyderabad
Departure Time: 02:33
Destination : Mumbai
Arrival Time: 15:38
Starting Price: 7000
Available Seats: 10

- **Authentication**



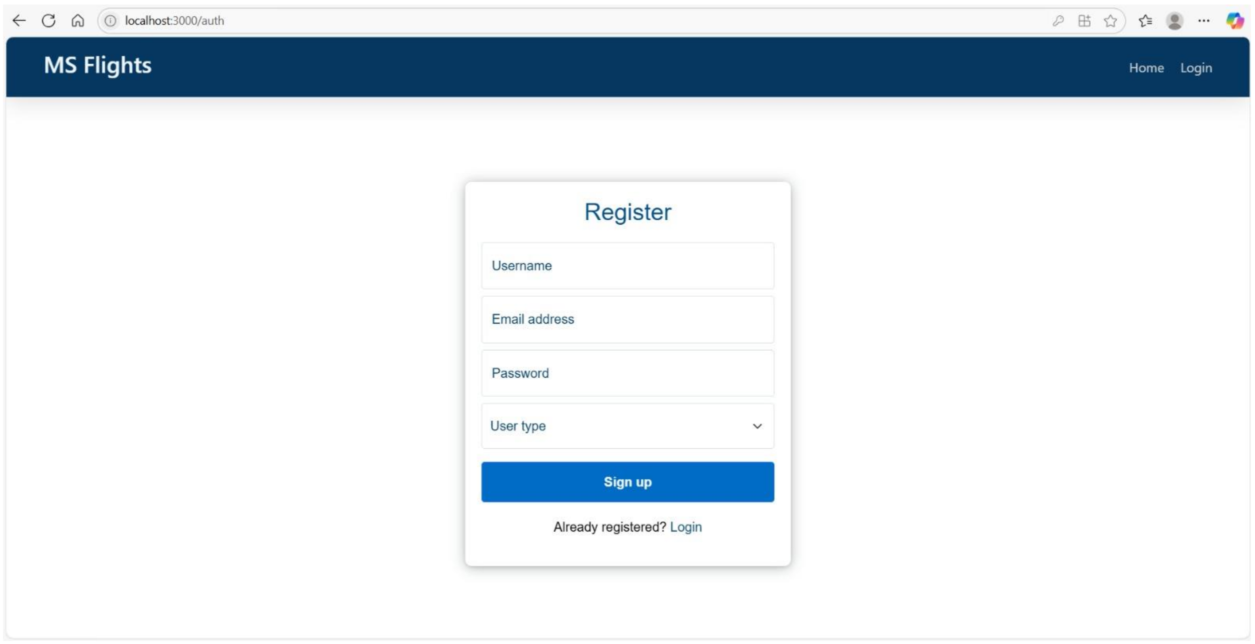
Login

Email address

Password

Sign in

Not registered? Register



Register

Username

Email address

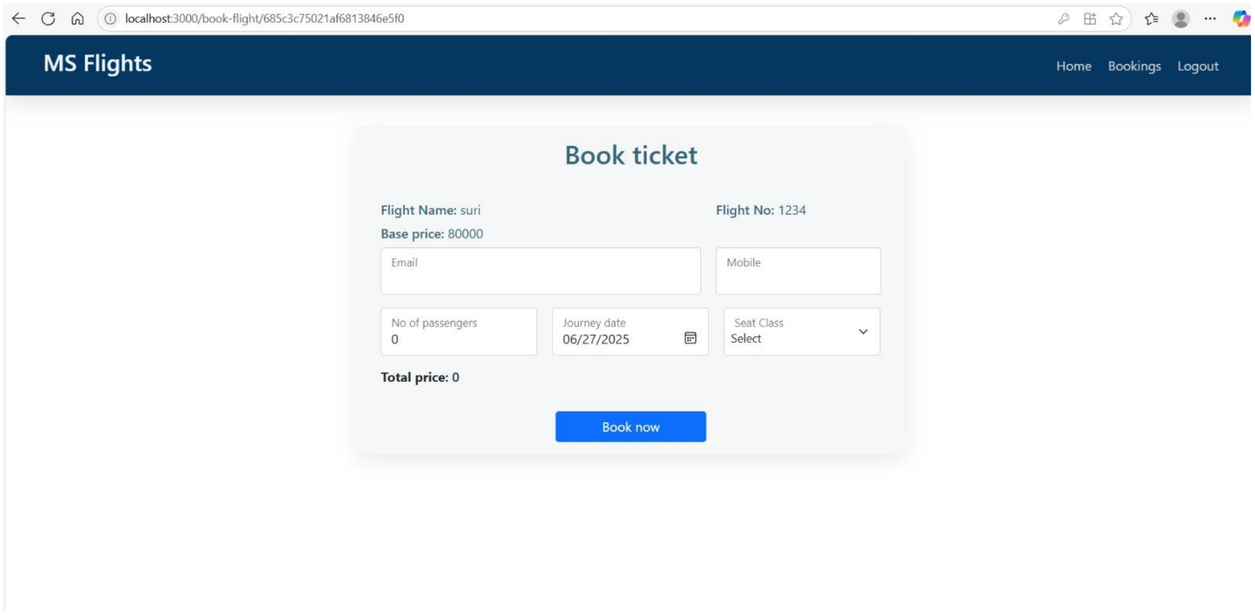
Password

User type

Sign up

Already registered? Login

- **User bookings**


localhost:3000/book-flight/685c3c75021af6813846e5f0

MS Flights

Home Bookings Logout

Book ticket

Flight Name: suri

Flight No: 1234

Base price: 80000

Email

Mobile

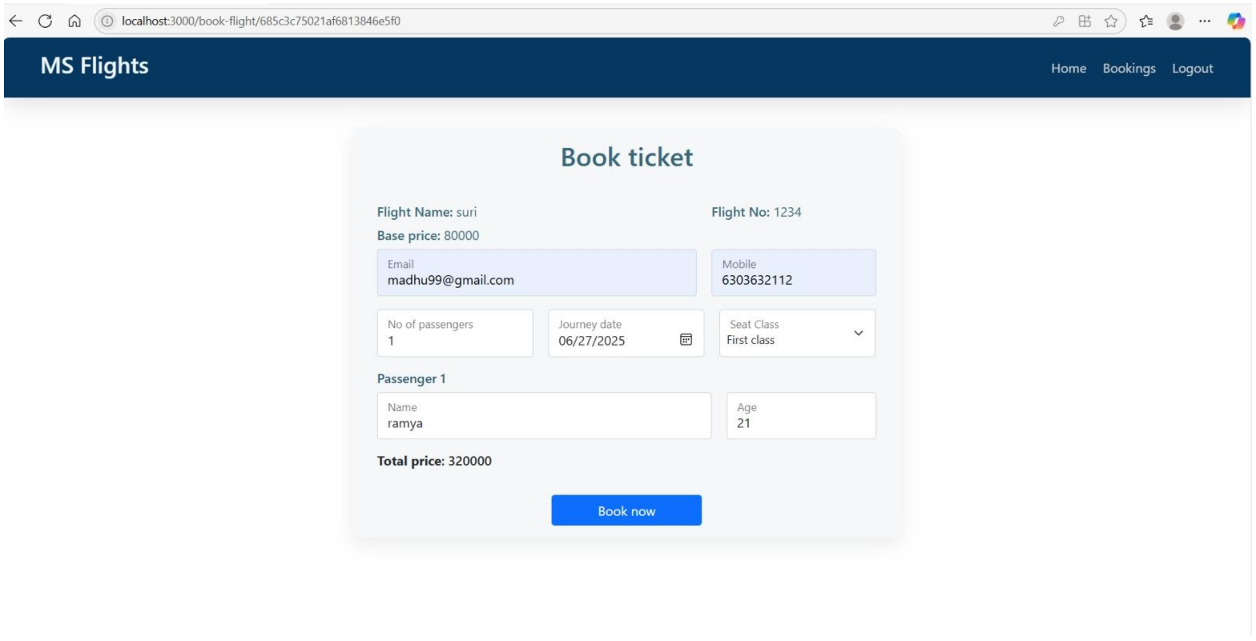
No of passengers: 0

Journey date: 06/27/2025

Seat Class: Select

Total price: 0

Book now


localhost:3000/book-flight/685c3c75021af6813846e5f0

MS Flights

Home Bookings Logout

Book ticket

Flight Name: suri

Flight No: 1234

Base price: 80000

Email: madhu99@gmail.com

Mobile: 6303632112

No of passengers: 1

Journey date: 06/27/2025

Seat Class: First class

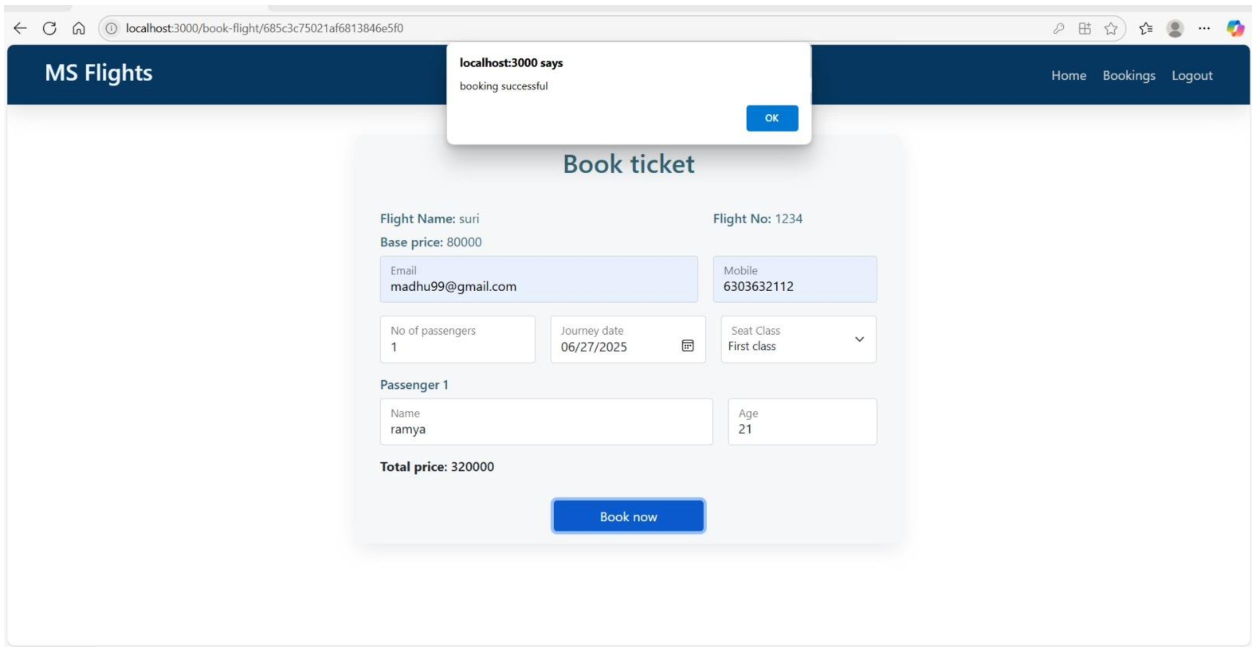
Passenger 1

Name: ramya

Age: 21

Total price: 320000

Book now


localhost:3000/book-flight/685c3c75021af6813846e5f0

MS Flights

Home Bookings Logout

localhost:3000 says

booking successful

OK

Book ticket

Flight Name: suri **Flight No:** 1234

Base price: 80000

Email: madhu99@gmail.com

Mobile: 6303632112

No of passengers: 1

Journey date: 06/27/2025

Seat Class: First class

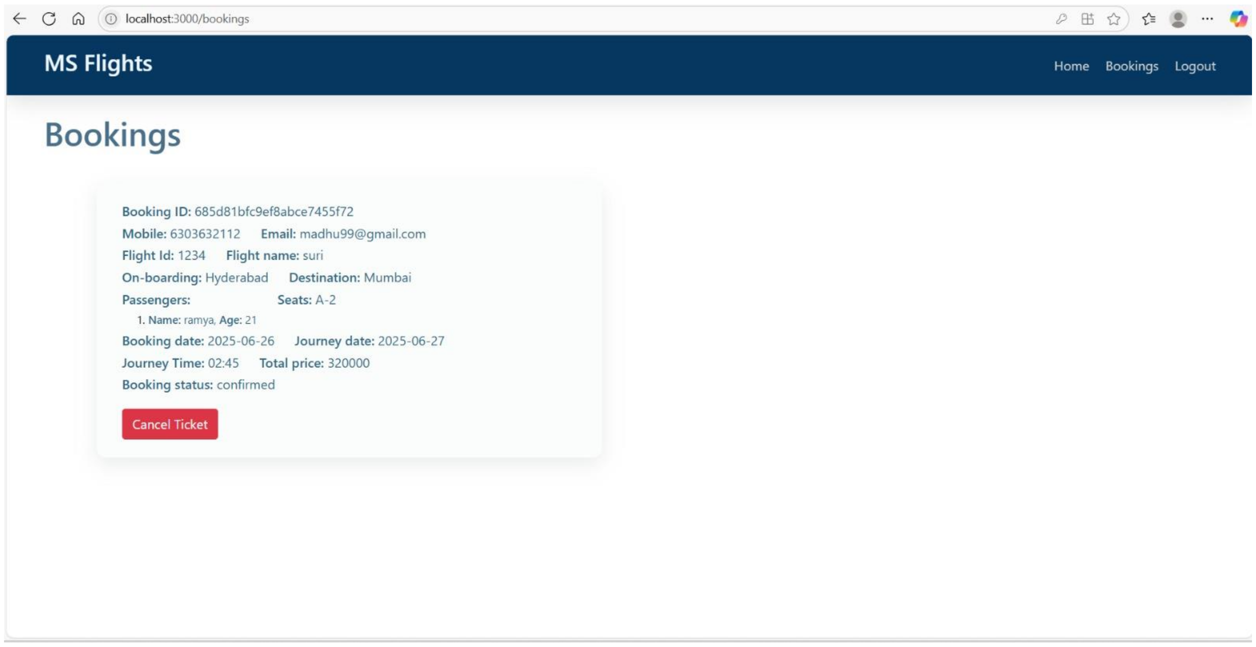
Passenger 1

Name: ramya

Age: 21

Total price: 320000

Book now


localhost:3000/bookings

MS Flights

Home Bookings Logout

Bookings

Booking ID: 685d81bfc9ef8abce7455f72

Mobile: 6303632112 **Email:** madhu99@gmail.com

Flight Id: 1234 **Flight name:** suri

On-boarding: Hyderabad **Destination:** Mumbai

Passengers: **Seats:** A-2

1. **Name:** ramya, **Age:** 21

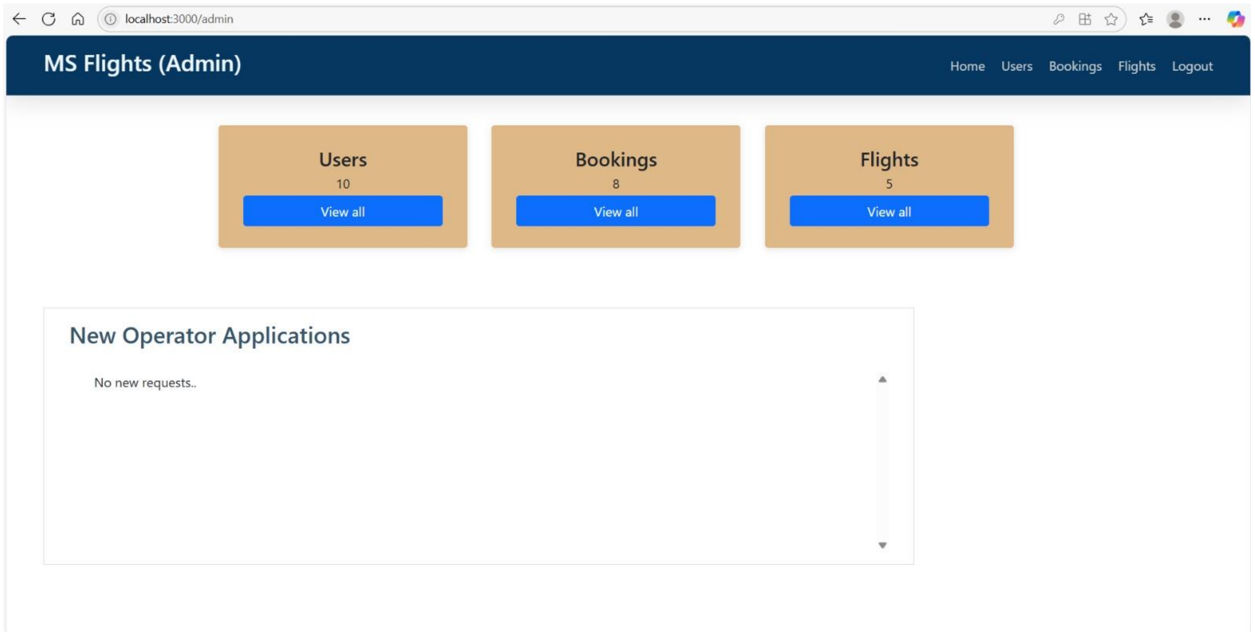
Booking date: 2025-06-26 **Journey date:** 2025-06-27

Journey Time: 02:45 **Total price:** 320000

Booking status: confirmed

Cancel Ticket

- **Admin Dashboard**


localhost:3000/admin

MS Flights (Admin)

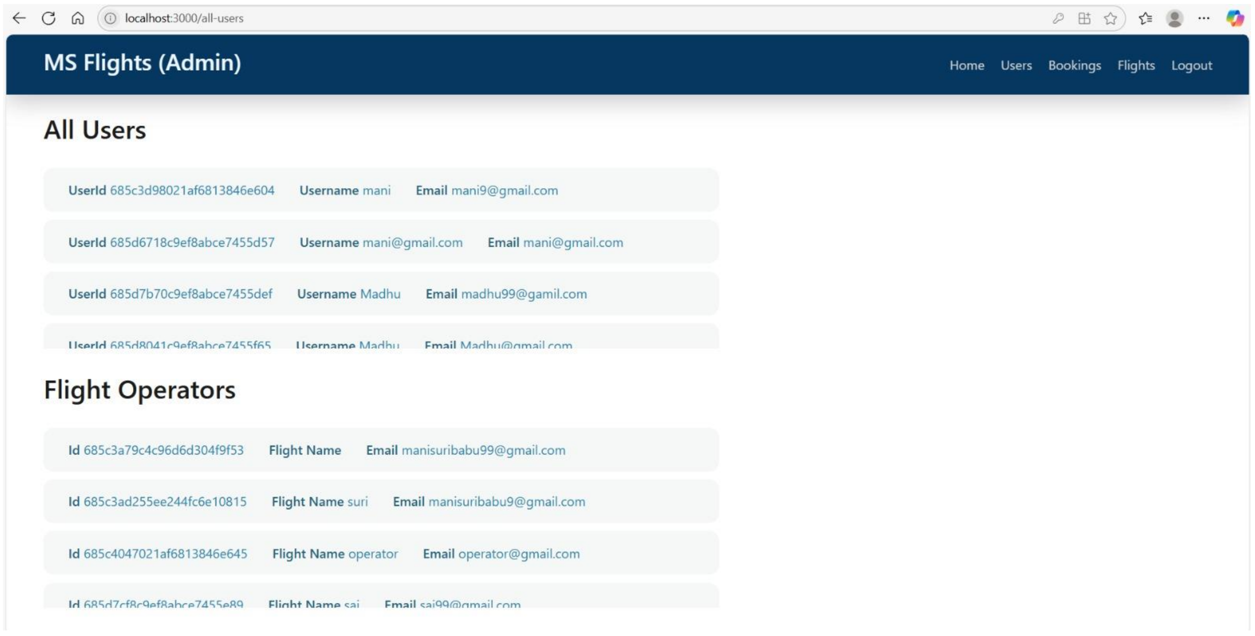
Home Users Bookings Flights Logout

Users	Bookings	Flights
10	8	5
View all	View all	View all

New Operator Applications

No new requests..

- **All users**


localhost:3000/all-users

MS Flights (Admin)

Home Users Bookings Flights Logout

All Users

UserId 685c3d98021af6813846e604 **Username** mani **Email** mani9@gmail.com

UserId 685d6718c9ef8abce7455d57 **Username** mani@gmail.com **Email** mani@gmail.com

UserId 685d7b70c9ef8abce7455def **Username** Madhu **Email** madhu99@gamil.com

UserId 685d8041c9ef8abce7455f65 **Username** Madhu **Email** Madhu@gmail.com

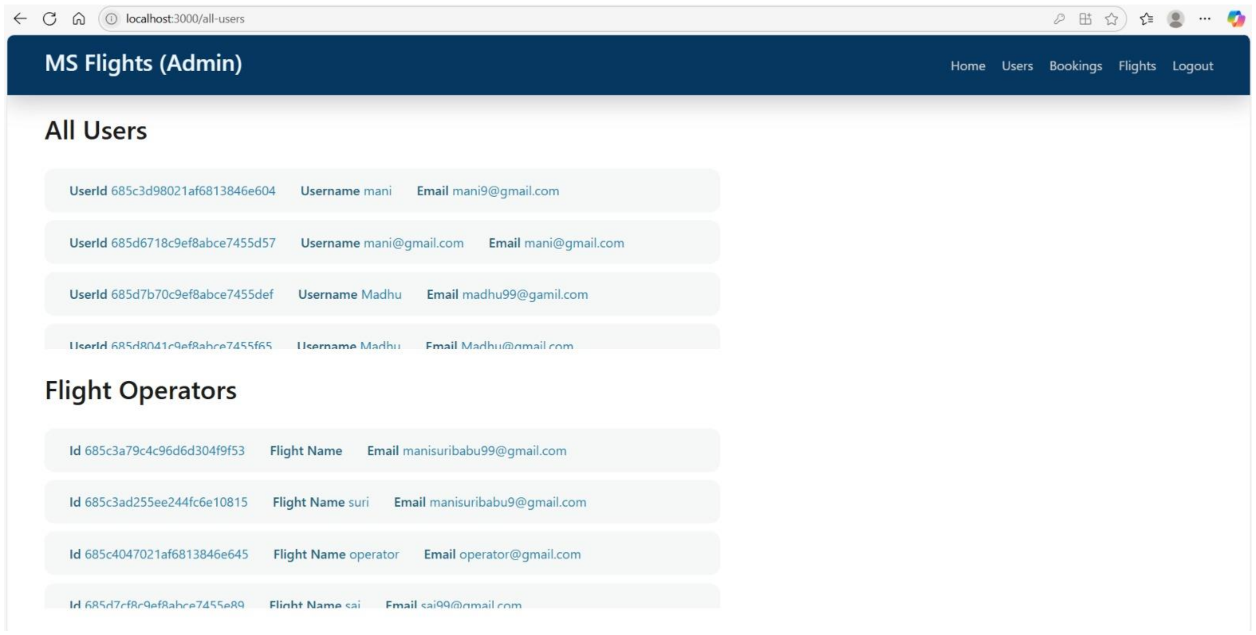
Flight Operators

Id 685c3a79c4c96d6d304f9f53 **Flight Name** **Email** manisuribabu99@gmail.com

Id 685c3ad255ee244fc6e10815 **Flight Name** suri **Email** manisuribabu9@gmail.com

Id 685c4047021af6813846e645 **Flight Name** operator **Email** operator@gmail.com

Id 685d7cf8c9ef8abce7455e89 **Flight Name** sai **Email** sai99@gmail.com


localhost:3000/all-users

MS Flights (Admin) Home Users Bookings Flights Logout

All Users

UserId 685c3d98021af6813846e604 **Username** mani **Email** mani9@gmail.com

UserId 685d6718c9ef8abce7455d57 **Username** mani@gmail.com **Email** mani@gmail.com

UserId 685d7b70c9ef8abce7455def **Username** Madhu **Email** madhu99@gamil.com

UserId 685d8041c9ef8abce7455f65 **Username** Madhu **Email** Madhu@gmail.com

Flight Operators

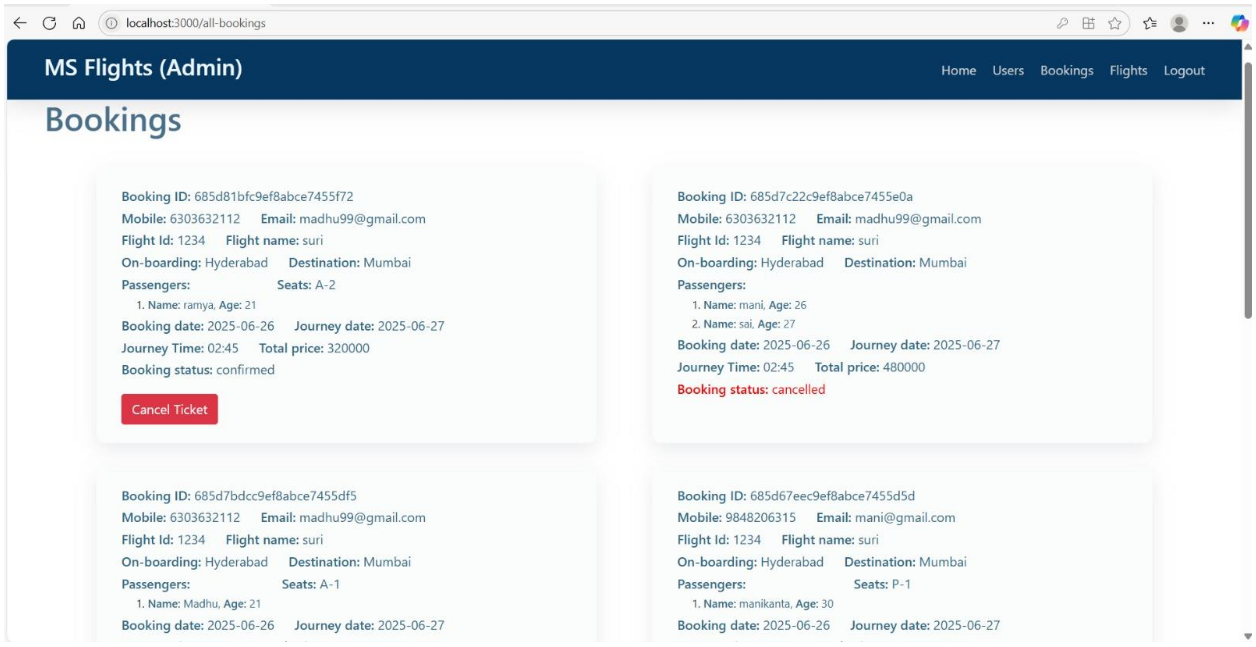
Id 685c3a79c4c96d6d304f9f53 **Flight Name** **Email** manisuribabu99@gmail.com

Id 685c3ad255ee244fc6e10815 **Flight Name** suri **Email** manisuribabu9@gmail.com

Id 685c4047021af6813846e645 **Flight Name** operator **Email** operator@gmail.com

Id 685d7cf8c9ef8abce7455e89 **Flight Name** sai **Email** sai99@gmail.com

- **All Bookings**


localhost:3000/all-bookings

MS Flights (Admin) Home Users Bookings Flights Logout

Bookings

Booking ID: 685d81bfc9ef8abce7455f72
Mobile: 6303632112 **Email:** madhu99@gmail.com
Flight Id: 1234 **Flight name:** suri
On-boarding: Hyderabad **Destination:** Mumbai
Passengers: **Seats:** A-2
1. **Name:** ramya, **Age:** 21

Booking date: 2025-06-26 **Journey date:** 2025-06-27
Journey Time: 02:45 **Total price:** 320000
Booking status: confirmed
Cancel Ticket

Booking ID: 685d7c22c9ef8abce7455e0a
Mobile: 6303632112 **Email:** madhu99@gmail.com
Flight Id: 1234 **Flight name:** suri
On-boarding: Hyderabad **Destination:** Mumbai
Passengers:
1. **Name:** mani, **Age:** 26
2. **Name:** sai, **Age:** 27

Booking date: 2025-06-26 **Journey date:** 2025-06-27
Journey Time: 02:45 **Total price:** 480000
Booking status: cancelled

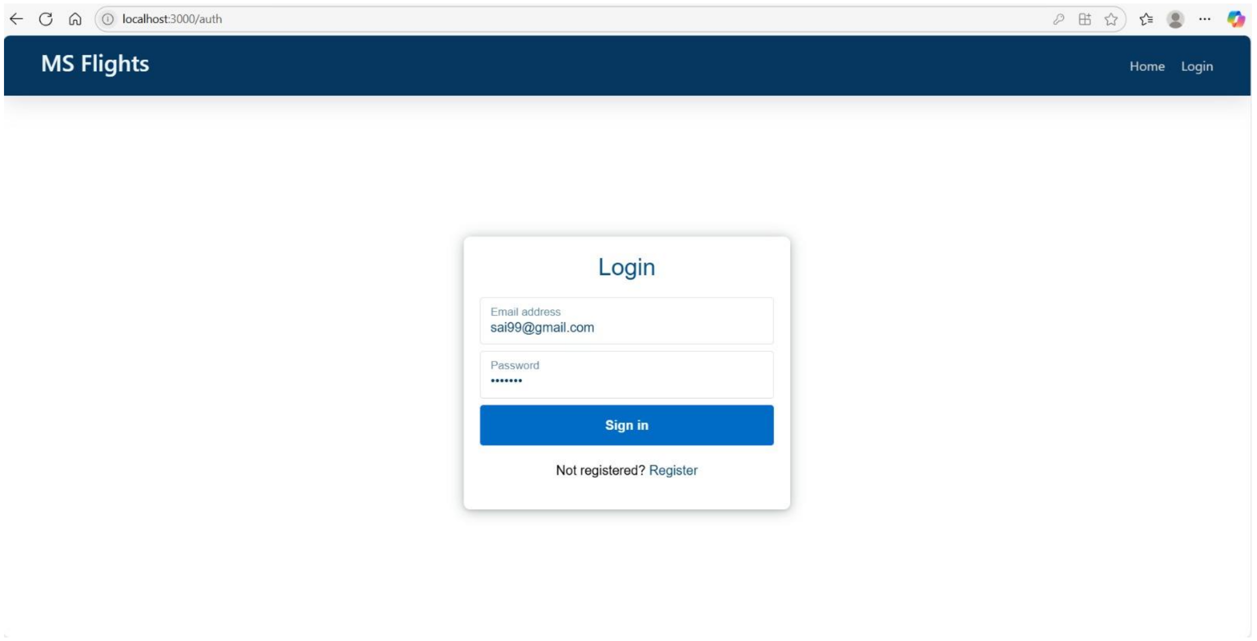
Booking ID: 685d7bdcc9ef8abce7455df5
Mobile: 6303632112 **Email:** madhu99@gmail.com
Flight Id: 1234 **Flight name:** suri
On-boarding: Hyderabad **Destination:** Mumbai
Passengers: **Seats:** A-1
1. **Name:** Madhu, **Age:** 21

Booking date: 2025-06-26 **Journey date:** 2025-06-27

Booking ID: 685d67eec9ef8abce7455d5d
Mobile: 9848206315 **Email:** mani@gmail.com
Flight Id: 1234 **Flight name:** suri
On-boarding: Hyderabad **Destination:** Mumbai
Passengers: **Seats:** P-1
1. **Name:** manikanta, **Age:** 30

Booking date: 2025-06-26 **Journey date:** 2025-06-27

- **Flight Operator**


localhost:3000/auth

MS Flights

Home Login

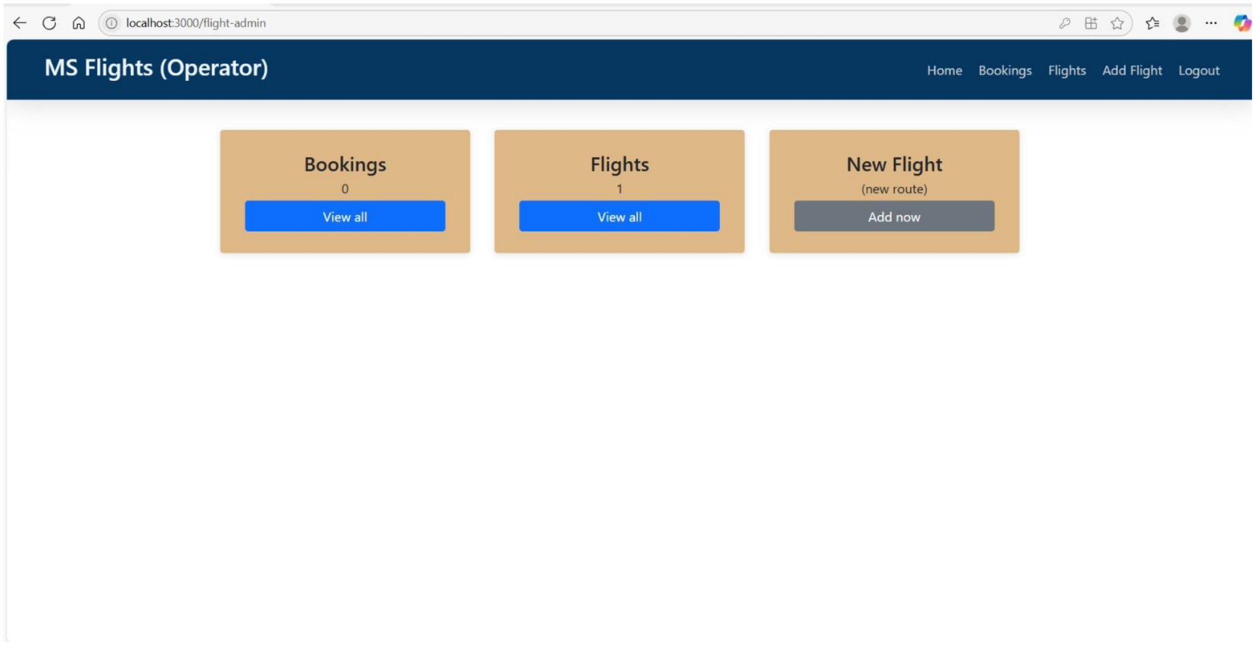
Login

Email address
sai99@gmail.com

Password
•••••••

Sign in

Not registered? Register


localhost:3000/flight-admin

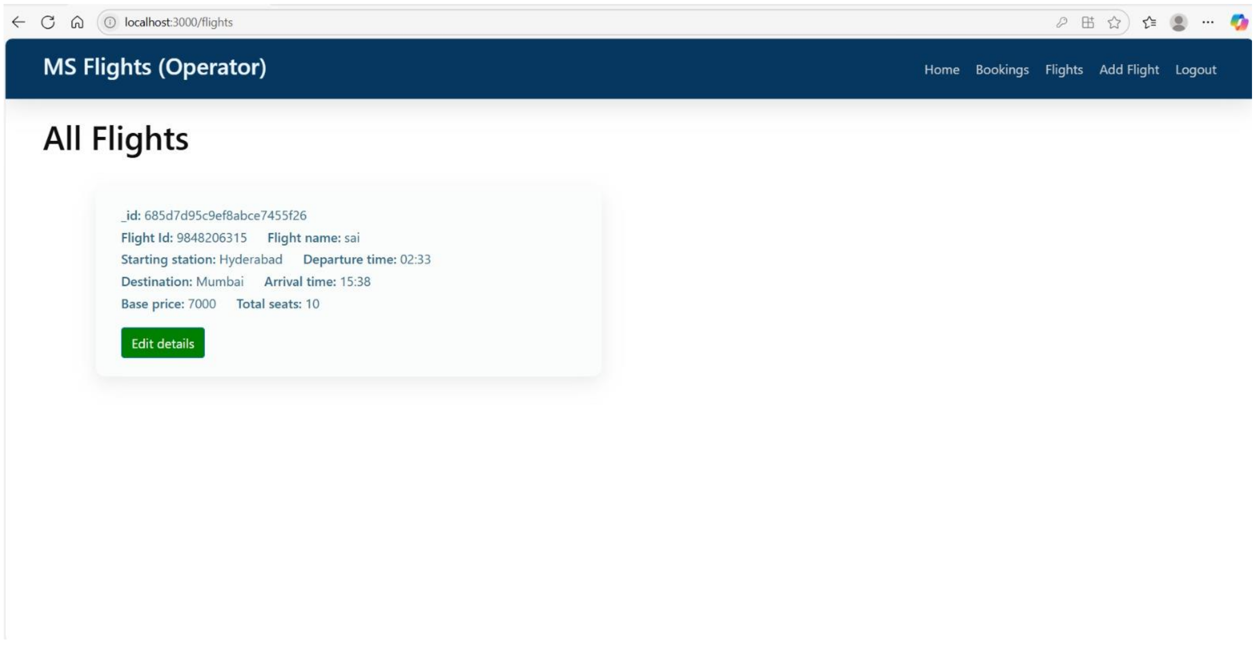
MS Flights (Operator)

Home Bookings Flights Add Flight Logout

Bookings
0
View all

Flights
1
View all

New Flight
(new route)
Add now


localhost:3000/flights

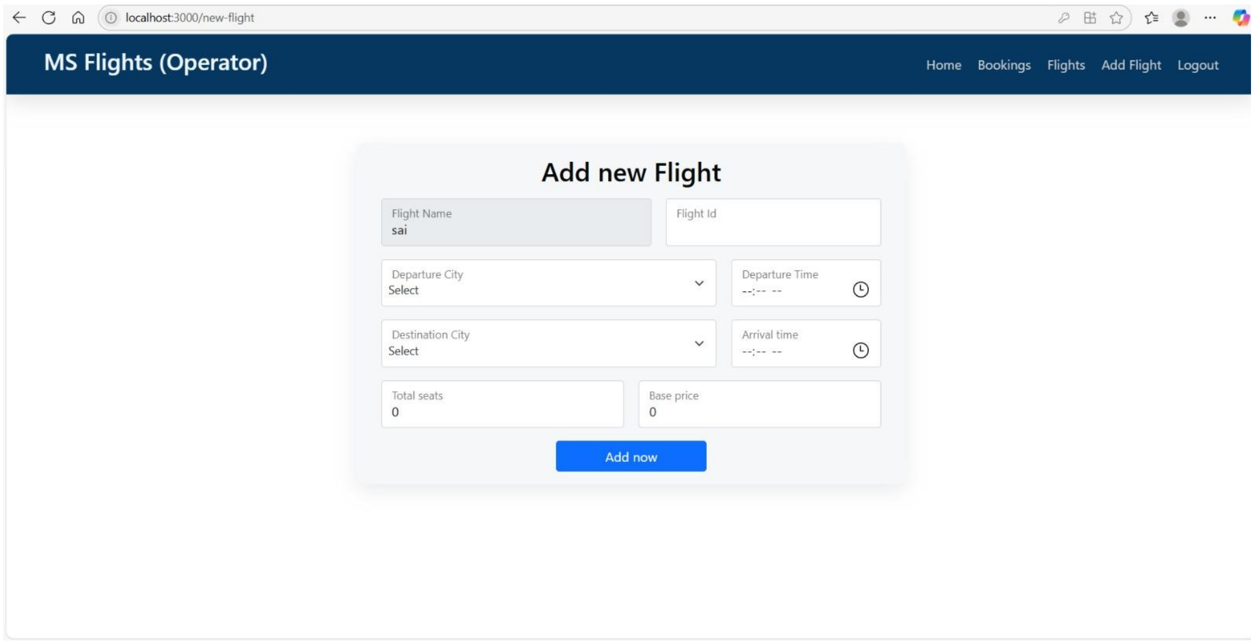
MS Flights (Operator)

Home Bookings Flights Add Flight Logout

All Flights

_id: 685d7d95c9ef8abce7455f26
Flight Id: 9848206315 **Flight name:** sai
Starting station: Hyderabad **Departure time:** 02:33
Destination: Mumbai **Arrival time:** 15:38
Base price: 7000 **Total seats:** 10

- **New Flight**

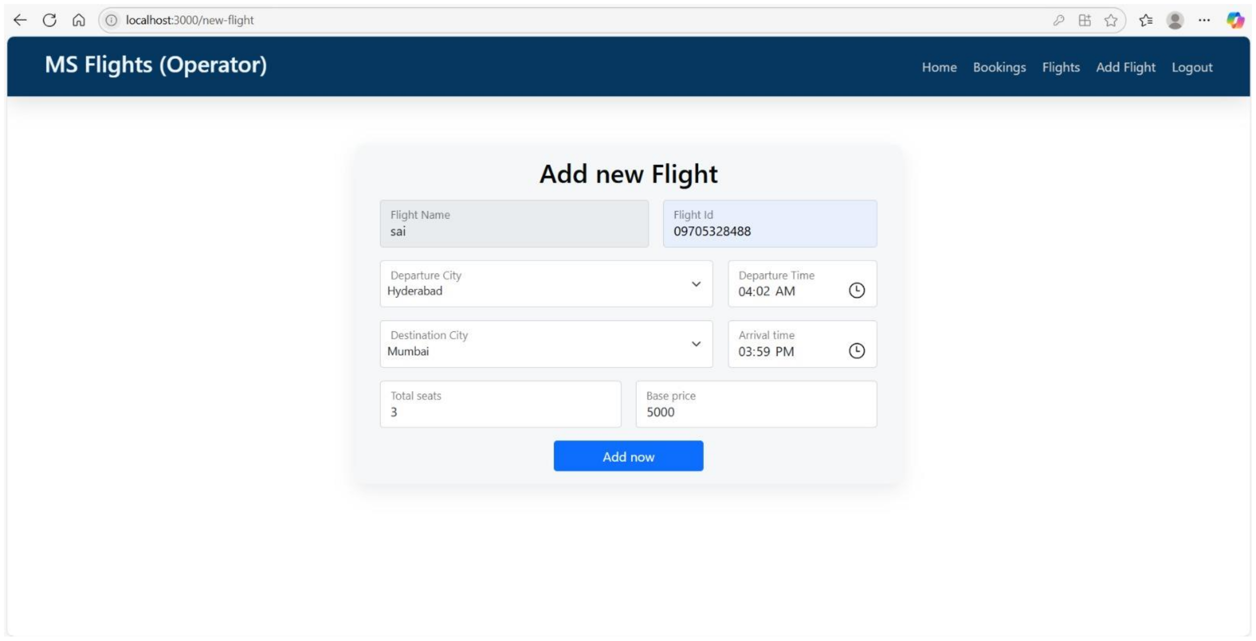

localhost:3000/new-flight

MS Flights (Operator)

Home Bookings Flights Add Flight Logout

Add new Flight

Flight Name: sai
Flight Id:
Departure City: Select
Departure Time: --:-- --
Destination City: Select
Arrival time: --:-- --
Total seats: 0
Base price: 0


localhost:3000/new-flight

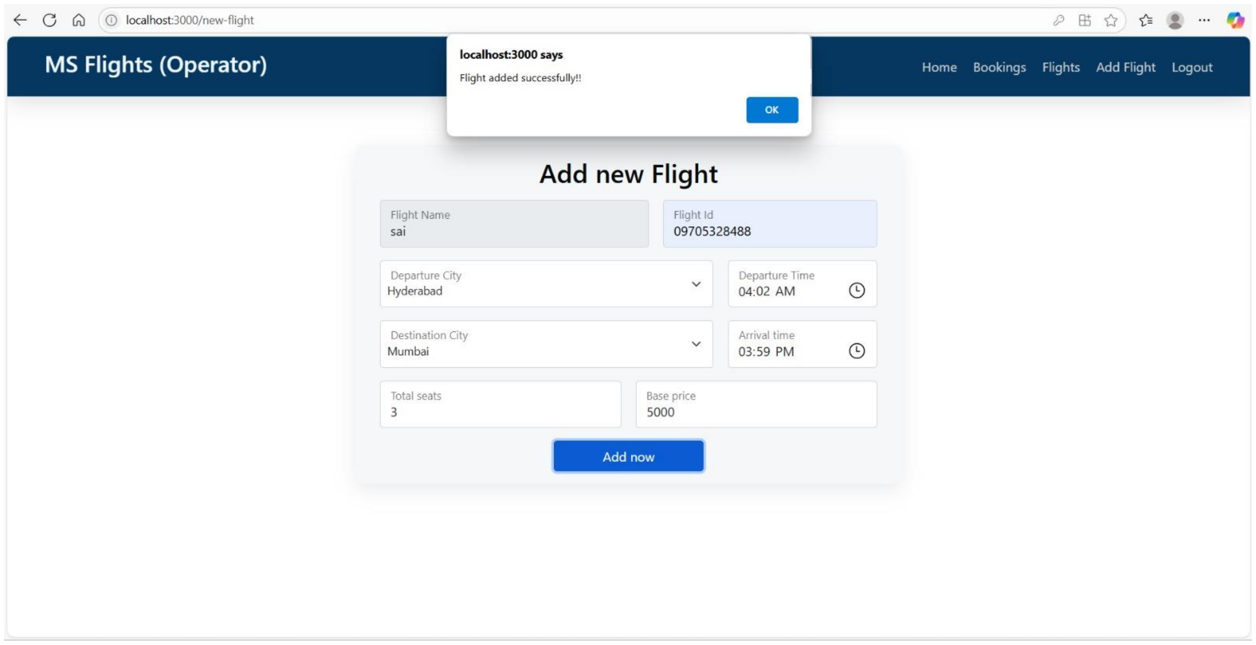
MS Flights (Operator)

Home Bookings Flights Add Flight Logout

Add new Flight

Flight Name: sai	Flight Id: 09705328488
Departure City: Hyderabad	Departure Time: 04:02 AM
Destination City: Mumbai	Arrival time: 03:59 PM
Total seats: 3	Base price: 5000

Add now


localhost:3000/new-flight

MS Flights (Operator)

Home Bookings Flights Add Flight Logout

localhost:3000 says

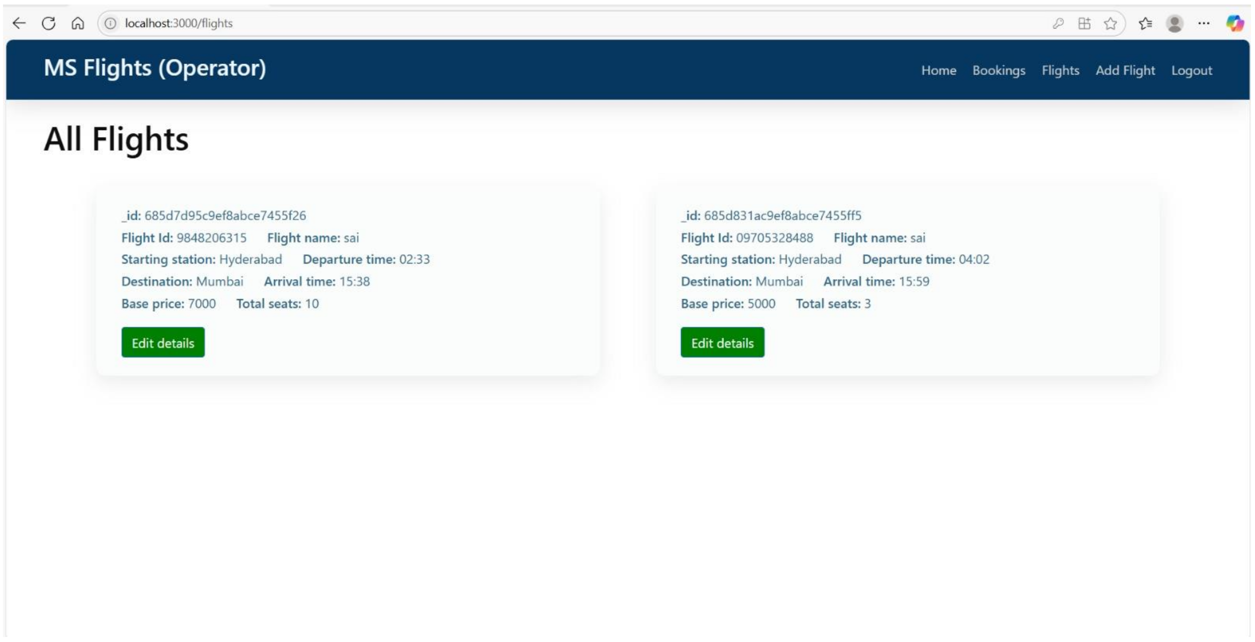
Flight added successfully!!

OK

Add new Flight

Flight Name: sai	Flight Id: 09705328488
Departure City: Hyderabad	Departure Time: 04:02 AM
Destination City: Mumbai	Arrival time: 03:59 PM
Total seats: 3	Base price: 5000

Add now



localhost:3000/flights

MS Flights (Operator)

Home Bookings Flights Add Flight Logout

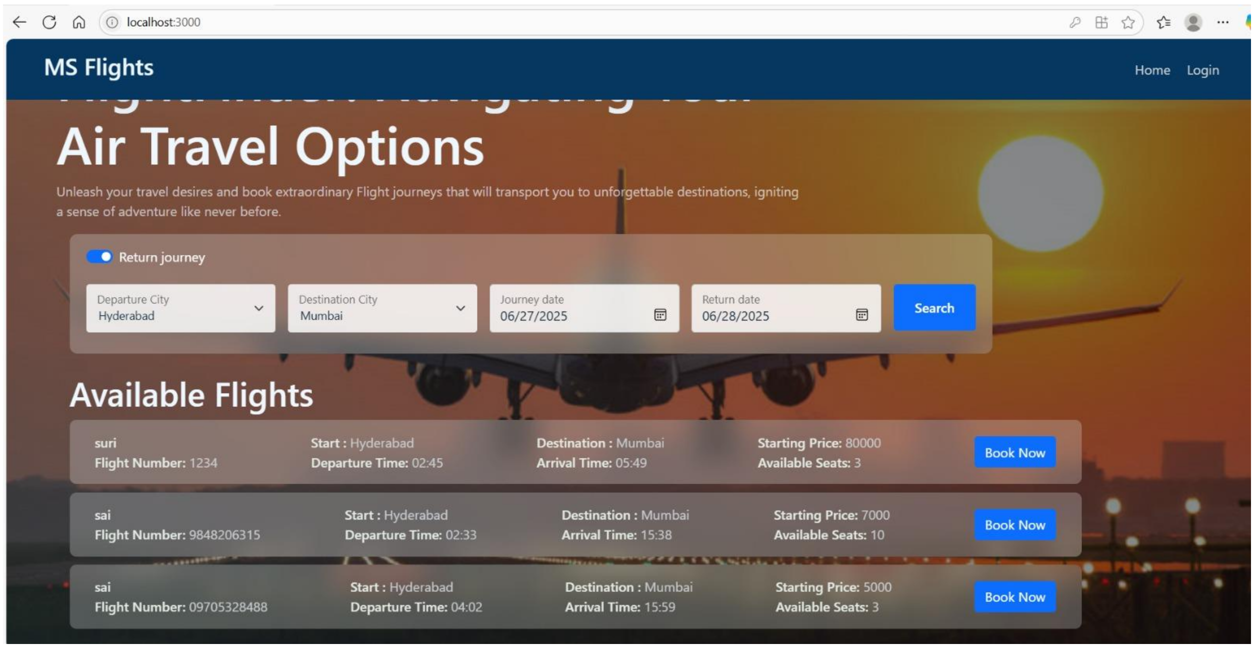
All Flights

_id: 685d7d95c9ef8abce7455f26
Flight Id: 9848206315 **Flight name:** sai
Starting station: Hyderabad **Departure time:** 02:33
Destination: Mumbai **Arrival time:** 15:38
Base price: 7000 **Total seats:** 10

Edit details

_id: 685d831ac9ef8abce7455ff5
Flight Id: 09705328488 **Flight name:** sai
Starting station: Hyderabad **Departure time:** 04:02
Destination: Mumbai **Arrival time:** 15:59
Base price: 5000 **Total seats:** 3

Edit details



localhost:3000

MS Flights

Home Login

Air Travel Options

Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.

Return journey

Departure City: Hyderabad
Destination City: Mumbai
Journey date: 06/27/2025
Return date: 06/28/2025

Search

Available Flights

suri **Flight Number:** 1234	**Start :** Hyderabad **Departure Time:** 02:45	**Destination :** Mumbai **Arrival Time:** 05:49	**Starting Price:** 80000 **Available Seats:** 3	Book Now
sai **Flight Number:** 9848206315	**Start :** Hyderabad **Departure Time:** 02:33	**Destination :** Mumbai **Arrival Time:** 15:38	**Starting Price:** 7000 **Available Seats:** 10	Book Now
sai **Flight Number:** 09705328488	**Start :** Hyderabad **Departure Time:** 04:02	**Destination :** Mumbai **Arrival Time:** 15:59	**Starting Price:** 5000 **Available Seats:** 3	Book Now

Challenge Faced

1. Integration Across the MERN Stack

Coordinating React (frontend), Express/Node, and MongoDB backend often leads to mismatched data formats or API miscommunication. Debugging whether an issue originates in the UI call, Express routing, or DB layer can be time-consuming

2. Authentication & Security

Implementing secure user login, JWT/session handling, role-based access (especially for admin panels), token expiry, and protection against XSS/CSRF requires careful planning. Beginners frequently struggle with securely storing tokens and guarding sensitive routes.

3. API Integration (Flights & Payments)

Connecting flight data APIs or payment gateways introduces challenges around authentication, rate limits, error handling, and transforming inconsistent or complex external data into your app's schema.

4. Performance, Caching & Scalability

As flight records grow, MongoDB performance can degrade without proper indexing, caching, or pagination. Server-side throttling and Node.js clustering may be needed to maintain responsiveness.

5. Error Handling & Monitoring

Centralizing error handling in Express (middleware), using React error-boundary components, and consistent logging (e.g., via Morgan/Winston/Sentry) is essential—but often overlooked in smaller projects ([n-school.com][2]).

6. User Experience & UI Design

Complex UIs—flight search, filters, seat maps, booking forms—are prone to confusion and poor usability. Without iterative design feedback, the interface may feel cluttered or unintuitive.

7. Complex State Management

Managing global state in React (e.g., search filters, flight selections, user profiles) can become messy without Redux, Context API, or custom hooks—leading to bugs and data inconsistencies.

8.Cross-Cutting Concerns Handling

Multilingual support, multiple currencies, time zones, date formatting, and storing user preferences adds complexity in ensuring a smooth, globally functional experience.

9. Testing, Deployment & CI/CD

Setting up automated unit, integration, and end-to-end tests across all stack layers is challenging. Deploying and maintaining runtime consistency (e.g., CORS, environment variables, build pipelines) also adds overhead.

10. Offline & Connectivity Handling

Users may lose connectivity mid-flow. Without robust offline handling, seat map state and booking progress can be lost, degrading trust and usability.

Future Scope of Flight Finder

The current version of Flight Finder provides a solid foundation for flight booking with an intuitive interface, real-time flight listings, and secure booking. To further enhance the user experience and meet evolving traveler expectations, the following future improvements and expansions are proposed:

1. Enhanced Personalization with AI

Smart Recommendations: Use machine learning to suggest flights based on user preferences, travel history, and seasonal trends.

Dynamic Pricing Alerts: Notify users about price drops or better deals using predictive price modeling.
Custom Travel Itineraries: Suggest accommodations, local transportation, and attractions along with flights.

2.Multi-City and Complex Route Support

Allow users to book multi-leg or round-the-world journeys.
Include open-jaw tickets (e.g., flying into one city and returning from another).

3. Real-Time Flight Status and Notifications

Integrate real-time flight tracking APIs to provide updates on delays, gate changes, and cancellations.
Push notifications or SMS/email alerts for changes to bookings or reminders.

4. Loyalty and Reward Integration

Partner with major airlines and loyalty programs.
Enable users to earn and redeem points or miles through the app.

5. In-App Check-In and Boarding Passes

Allow users to check in directly through the app and access digital boarding passes.
Integrate with airline systems for a smoother pre-flight experience.

6. Multilingual and Multi-Currency Support

Support for global users by adding multiple languages and real-time currency conversion during checkout.

7. Chatbot and Virtual Travel Assistant

Implement an AI-powered chatbot for 24/7 support to answer FAQs, guide users, or handle issues like rescheduling.
Provide travel tips, visa information, and local customs for international trips.

8.Eco-Friendly Flight Options

Highlight flights with lower carbon emissions.
Partner with carbon offset programs to give users an option to contribute during checkout.

9. Admin Panel Enhancements

Advanced analytics for bookings, revenue trends, and user engagement.
Real-time management of flight data and promotional campaigns.

10. Mobile App Development

Expand to dedicated Android and iOS apps with offline features such as ticket storage and itinerary access.
Integrate biometric login (FaceID, fingerprint) for faster access and better security.

11. Third-Party Integrations

Integrate with travel insurance providers, ride-hailing apps (e.g., Uber, Lyft), and hotel booking platforms for an all-in-one travel planning ecosystem.

12. Blockchain for Ticket Verification

Explore blockchain technology to prevent fraud and enhance transparency in ticket bookings and transactions.

By implementing these future enhancements, Flight Finder can evolve into a comprehensive travel companion app, capable of addressing the full spectrum of modern traveler needs while maintaining a high standard of convenience, security, and innovation.

CONCLUSION

The **Flight Finder** mini-project—built with the MERN (MongoDB, Express, React, Node.js) stack—successfully delivers a streamlined, full-stack flight booking experience. From secure user authentication and dynamic flight searches to intuitive seat selection and payment handling, the app encapsulates real-world travel app functionality.

Key accomplishments in the project include:

- **End-to-End Workflow**: Users can search, filter, select, and book flights, covering the full booking lifecycle.
- **Componentized Architecture**: React components and modular backend APIs facilitate maintainability and future expansion.
- **Security & Standards**: JWT-based authentication, protected routes, and secure payment-handling illustrate adherence to best practices.
- **Data Handling**: MongoDB collections for users, flights, and bookings ensure efficient management and scalability of core data.

Alongside these strengths, the project helped identify important development lessons: ensuring seamless state management across components, integrating third-party APIs gracefully, and addressing performance tuning and error handling in a real-world context.

Flight Finder establishes a solid foundation for next-level enhancements such as live flight status updates, multi-city itineraries, loyalty programs, multilingual support, and mobile responsiveness. With further iteration, it has the potential to transform into a production-grade travel platform.

In summary, this project not only reinforces modern web development skills but also simulates the demanding architectural and UX considerations faced by real-world travel services—making it a valuable milestone in your development portfolio.

REFERENCES

The following online resources, tools, and platforms were extensively used during the research, development, testing, and documentation of this project:

1. **MDN Web Docs (Mozilla Developer Network)**
   https://developer.mozilla.org
   For foundational understanding of HTML, CSS, JavaScript, and DOM manipulation.
2. **React.js Official Documentation**
   https://react.dev
   For learning about component structure, hooks, state management, and routing.
3. **Node.js & Express.js Documentation**
   https://nodejs.org/en/docs
   https://expressjs.com
   For backend logic, server configuration, and RESTful API development.
4. **MongoDB Documentation**
   https://www.mongodb.com/docs
   For database schema design, CRUD operations, and Mongoose integration.
5. **Bootstrap Documentation**
   https://getbootstrap.com
   Used for building responsive layouts and UI components.
6. **Material UI Documentation**
   https://mui.com
   For implementing polished, accessible, and consistent design elements.
7. **Postman**
   https://www.postman.com
   Used for testing API endpoints and analyzing backend responses.
8. **GitHub**
   https://github.com
   For version control and collaborative development.
9. **YouTube Channels**
   - Traversy Media
   - The Net Ninja
   - Programming with Mosh
     For practical tutorials, full-stack project walkthroughs, and advanced tips.
10. **GeeksforGeeks**
    https://www.geeksforgeeks.org
    Used for reference on algorithms, backend logic, and JavaScript topics.

ChatGPT by OpenAI
https://chat.openai.com