

# Project Report: FlightFinder – Navigating Your Air Travel Options

---

## INTRODUCTION:

### 1. Project Overview

**FlightFinder** is a full-stack web application designed to streamline and enhance the flight booking experience. Built with the MERN stack (MongoDB, Express.js, React.js, and Node.js), it allows users to search, book, and manage flight tickets conveniently, while providing admin functionalities for managing flights and bookings.

Whether you're a frequent flyer or a casual traveller, FlightFinder ensures a seamless, fast, and secure way to plan your air travel.

---

### 2. Project Objective

- Simplify the flight booking process
  - Provide real-time search and filter options
  - Ensure secure and smooth user authentication
  - Enable admin-level control over flights and bookings
  - Create an intuitive and user-friendly UI
- 

### 3. Key Features

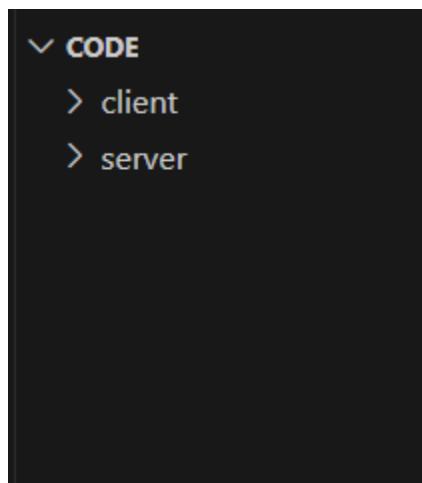
#### For Users

- Account registration and login
- Search flights by destination, date, time, and class
- Filter options: direct flights, preferred airline, etc.
- Book flights with real-time seat selection
- Secure payment simulation (no actual payment gateway)
- View, manage, and cancel bookings

#### For Admin

- Login with dedicated credentials
  - Add, edit, and delete flights
  - Monitor all bookings and users
  - Dashboard for flight and booking analytics
- 

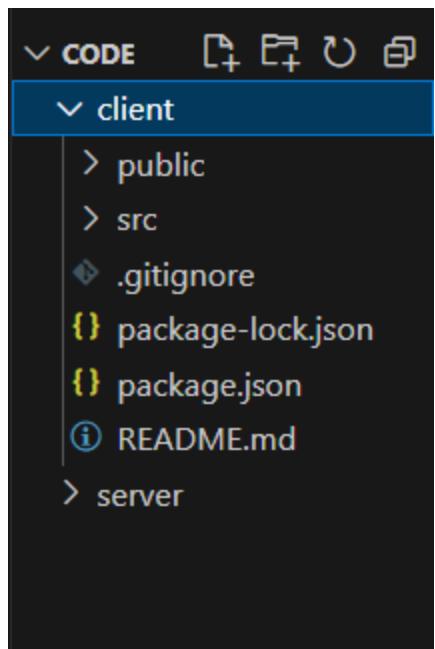
## 4. Technical Architecture

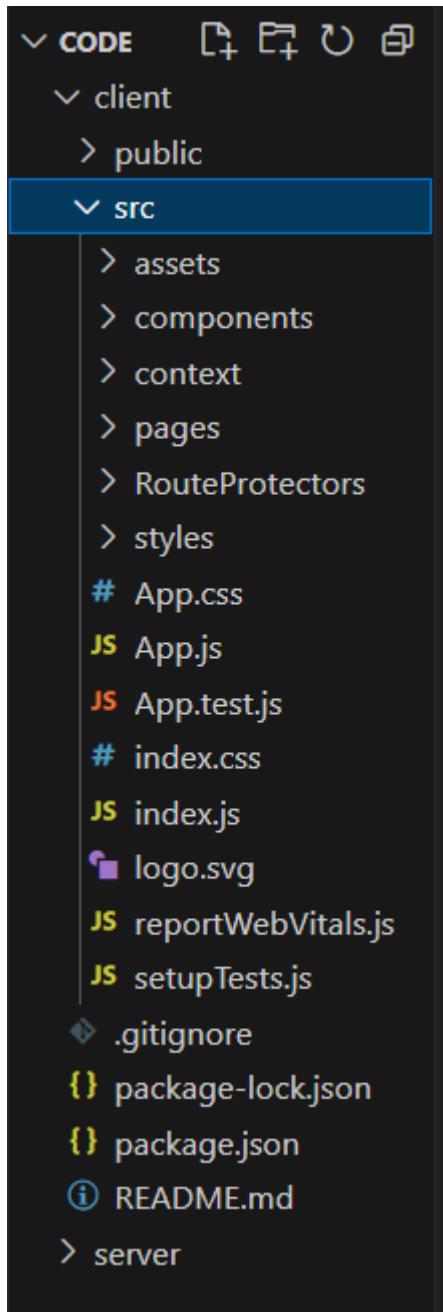


Frontend (React)

|-- User Interface (Login/Register, Search, Booking UI)

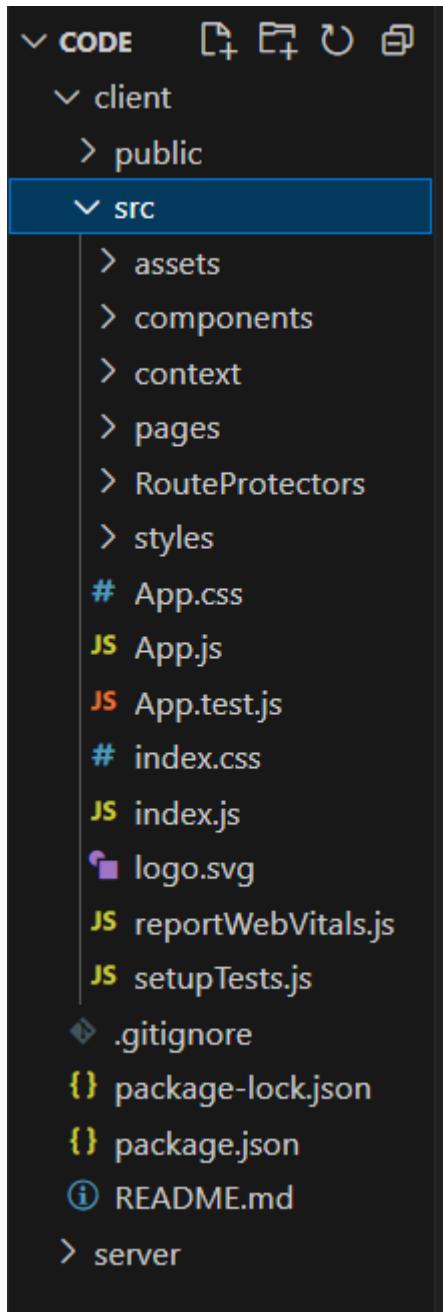
|-- Axios API Calls





## Backend (Node.js + Express)

```
|-- Routes: /users, /flights, /bookings, /admin  
|-- Middleware: Authentication & Authorization  
|-- Controllers: Business Logic for CRUD
```



Database (MongoDB)

|-- Collections: Users, Flights, Bookings

---

## 5. Scenario: Real-World Example

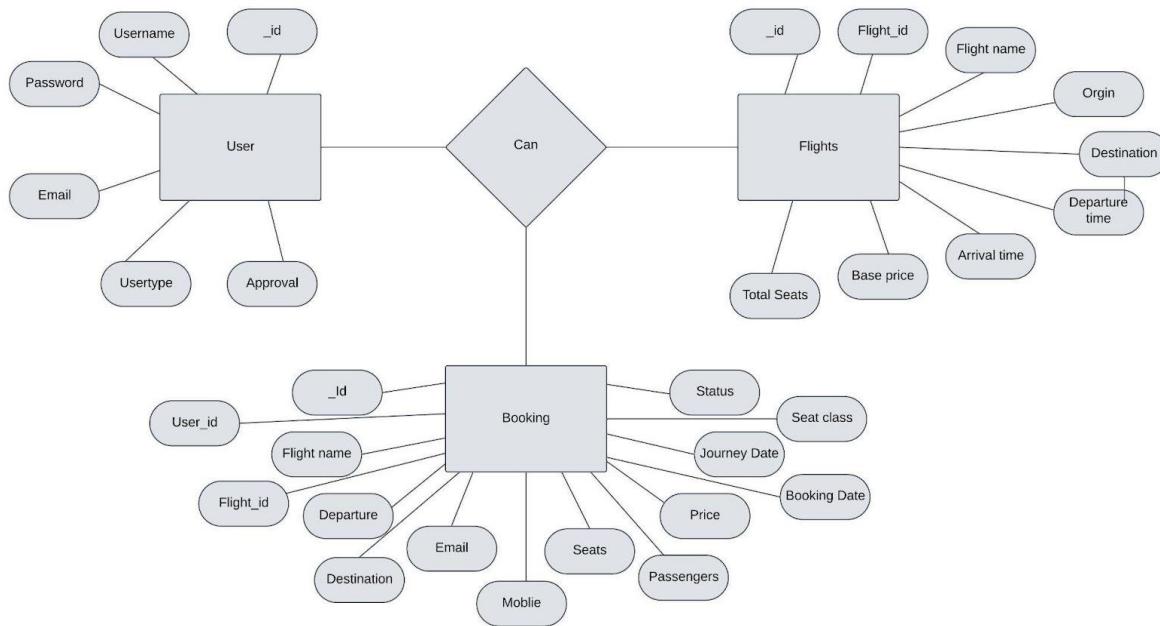
User:

- Books a business class flight from NYC to Paris
- Uses filters for direct flights and preferred airline
- Chooses a window seat with extra legroom
- Pays securely and receives booking confirmation instantly

This showcases the convenience and real-time assistance FlightFinder provides.

---

## 6. ER Diagram Summary



- **User**: Can make multiple bookings and payments
  - **Booking**: Linked to one user and one flight
  - **Flight**: Contains flight details, seat availability
  - **Admin**: Controls backend activities including flight and booking management
- 

## 7. Prerequisites for Development

- Node.js & npm
- MongoDB (local or Atlas)
- React.js

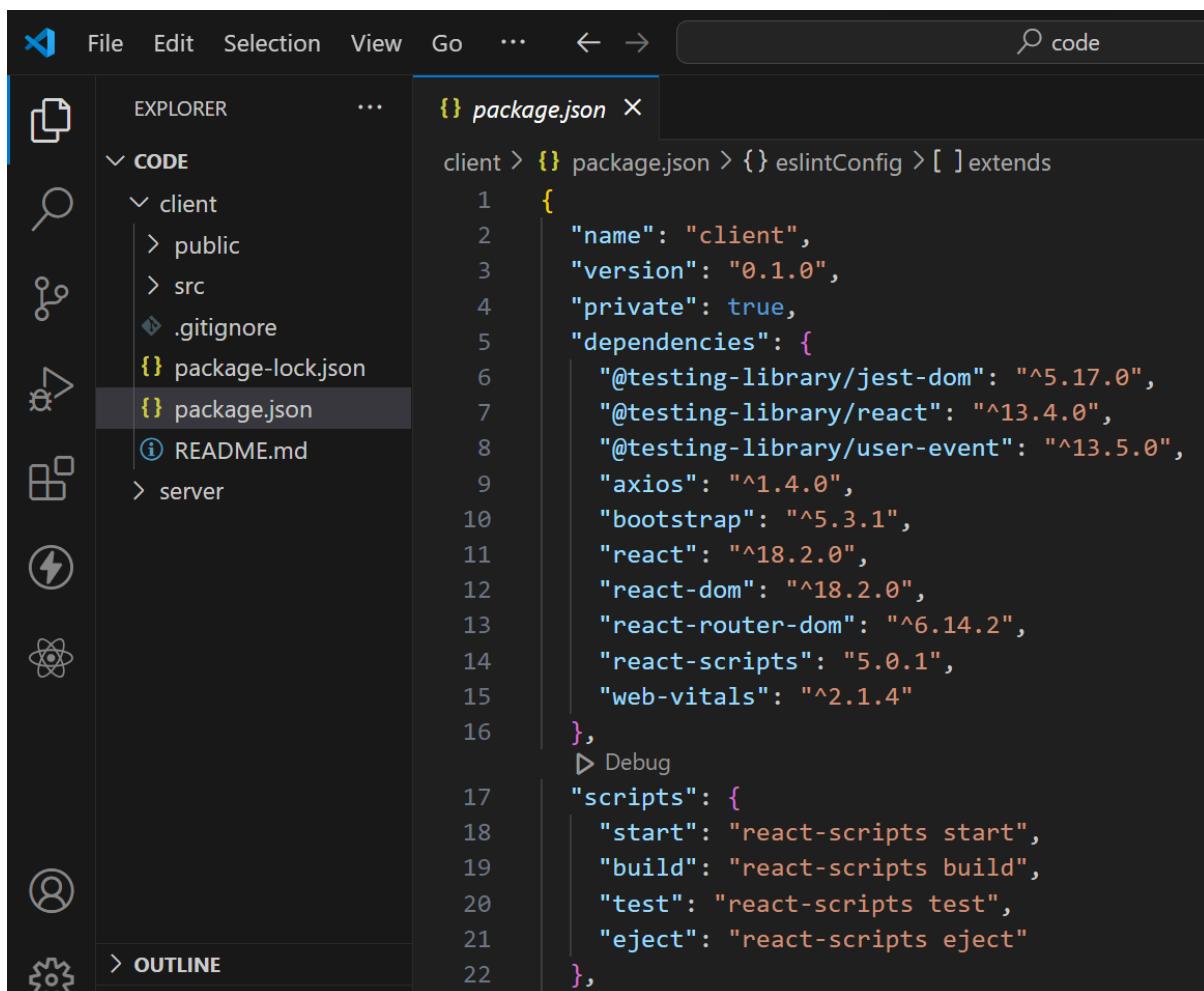
- Express.js
- Git & GitHub
- Visual Studio Code

Useful Links:

- Node.js: [nodejs.org](https://nodejs.org)
- MongoDB Atlas: [mongodb.com](https://mongodb.com)
- React: [reactjs.org](https://reactjs.org)
- Git: [git-scm.com](https://git-scm.com)

---

## 8. Project Setup



The screenshot shows the Visual Studio Code interface with the following details:

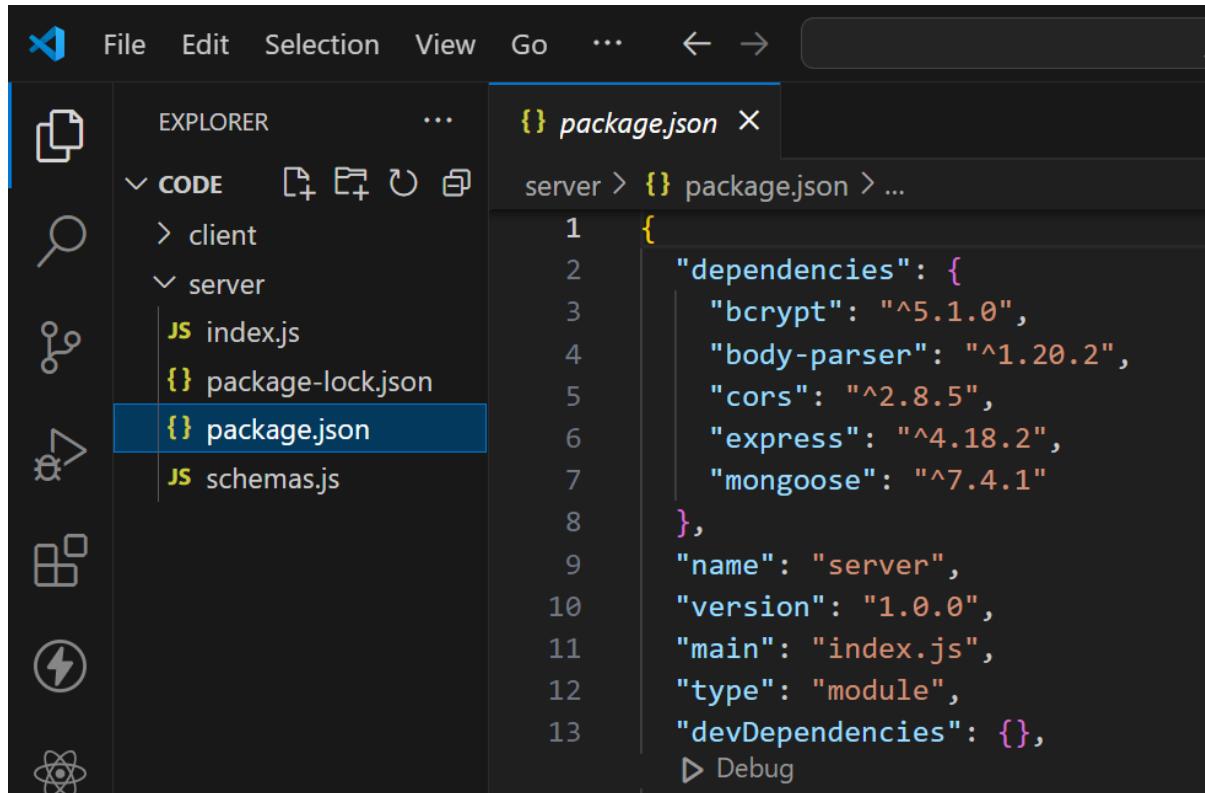
- File Bar:** File, Edit, Selection, View, Go, ..., ⏪ ⏹
- Search Bar:** code
- Left Sidebar (Explorer):** Shows a tree view of the project structure:
  - CODE folder
    - client
      - public
      - src
      - .gitignore
    - package-lock.json
    - package.json
    - README.md
    - server
- Center Panel:** The package.json file is open in the editor.

```
client > {} package.json > {} eslintConfig > [ ] extends
1  {
2    "name": "client",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.17.0",
7      "@testing-library/react": "^13.4.0",
8      "@testing-library/user-event": "^13.5.0",
9      "axios": "^1.4.0",
10     "bootstrap": "^5.3.1",
11     "react": "^18.2.0",
12     "react-dom": "^18.2.0",
13     "react-router-dom": "^6.14.2",
14     "react-scripts": "5.0.1",
15     "web-vitals": "^2.1.4"
16   },
17   ▷ Debug
18   "scripts": {
19     "start": "react-scripts start",
20     "build": "react-scripts build",
21     "test": "react-scripts test",
22     "eject": "react-scripts eject"
23   },
24 }
```
- Bottom Left:** OUTLINE tab

## Clone the Repository

```
git clone https://github.com/harsha-vardhan-reddy-07/Flight-Booking-App-MERN
```

```
cd Flight-Booking-App-MERN
```



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure under the 'CODE' section. It includes a 'client' folder, a 'server' folder which contains 'index.js', 'package-lock.json', and 'schemas.js', and a selected 'package.json' file. The main editor area on the right shows the contents of the 'package.json' file:

```
1  {
2    "dependencies": {
3      "bcrypt": "^5.1.0",
4      "body-parser": "^1.20.2",
5      "cors": "^2.8.5",
6      "express": "^4.18.2",
7      "mongoose": "^7.4.1"
8    },
9    "name": "server",
10   "version": "1.0.0",
11   "main": "index.js",
12   "type": "module",
13   "devDependencies": {}
```

## Install Dependencies

```
npm install
```

## Start Development Server

```
npm run dev
```

## Access

Visit: <http://localhost:3000>

---

# 9. Project Structure

## Client Folder

- Built with React.js and Bootstrap
- Contains components for login, registration, search, booking, dashboard

## Server Folder

- Built with Node.js and Express
  - Contains API routes, models (Mongoose schemas), controllers
- 

## 10. Application Flow

### User Flow

1. Register/Login
2. Search Flights
3. Apply Filters
4. Book Seat
5. Confirm Booking
6. Cancel Booking (if needed)

### Admin Flow

1. Login
  2. Add New Flights
  3. View All Flights & Bookings
  4. Manage User Activity
- 

## 11. Backend Development Highlights

- MongoDB connected using Mongoose
  - API endpoints: /api/flights, /api/bookings, /api/users
  - User roles: User, Admin, Airline Operator
  - Bcrypt used for password encryption
  - Middleware handles route protection
  - Error handling for all major APIs
- 

## 12. Database Design

The screenshot shows a code editor interface with a dark theme. The left sidebar contains icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and various project-related tools like Explorer, Outline, and Timeline. The main area is titled 'schemas.js' and displays the following code:

```
server > JS schemas.js > [e] bookingSchema
1 import mongoose from "mongoose";
2
3 const userSchema = new mongoose.Schema({
4   username: { type: String, required: true },
5   email: { type: String, required: true, unique: true },
6   usertype: { type: String, required: true },
7   password: { type: String, required: true },
8   approval: {type: String, default: 'approved'}
9 });
10 const flightSchema = new mongoose.Schema({
11   flightName: { type: String, required: true },
12   flightId: { type: String, required: true },
13   origin: { type: String, required: true },
14   destination: { type: String, required: true },
15   departureTime: { type: String, required: true },
16   arrivalTime: { type: String, required: true },
17   basePrice: { type: Number, required: true },
18   totalSeats: { type: Number, required: true }
19 });
20 const bookingSchema = new mongoose.Schema([
21   user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
22   flight: { type: mongoose.Schema.Types.ObjectId, ref: 'Flight', required: true },
23   flightName: {type: String, required: true},
24   flightId: {type: String},
25   departure: {type: String},
26   destination: {type: String},
27   email: {type: String},
28   mobile: {type: String},
29   seats: {type: String},
30   passengers: [
31     {
32       name: { type: String },
33       age: { type: Number }
34     },
35   ],
36   totalPrice: { type: Number },
37   bookingDate: { type: Date, default: Date.now },
38   journeyDate: { type: Date },
39   journeyTime: { type: String },
40   seatClass: { type: String },
41   bookingStatus: {type: String, default: "confirmed"}
42 ]);
43
44 export const User = mongoose.model('users', userSchema);
45 export const Flight = mongoose.model('Flight', flightSchema);
46 export const Booking = mongoose.model('Booking', bookingSchema);
```

## Schemas

- User: name, email, password, role
- Flight: source, destination, date, time, class, seat count
- Booking: userID, flightID, seat number, status

## Database Connection

```
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);
})
```

- Connected via MongoDB Atlas or Compass
  - Handles CRUD operations on all collections
- 

## 13. Frontend Development Highlights

- React components for each page (Home, Bookings, Dashboard)
  - Axios for API communication
  - Modals used for booking flow
  - Conditional rendering for different user roles
  - Admin dashboard features full CRUD on flights
- 

## 14. Demo and UI Screens

# Embark on an Extraordinary Flight Booking Adventure!

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  
Select

Destination City  
Select

Journey date  
dd-mm-yyyy

Search

## Login

Email address

Password

Sign in

[Not registered? Register](#)

## Bookings

Booking ID: 64ec8c3c4622709484005484  
Mobile: 7669678988 Email: harsha@gmail.com  
Flight Id: cni2321 Flight name: Indigo  
On-boarding: Chennai Destination: Banglore  
Passengers: Seats: B-1, B-2  
1. Name: Alex, Age: 44  
2. Name: Snyder, Age: 55  
Booking date: 2023-08-28 Journey date: 2023-08-31  
Journey Time: 18:40 Total price: 7200  
Booking status: confirmed

[Cancel Ticket](#)

Booking ID: 64e608bb2c862c07fa865bca  
Mobile: 7869868765 Email: simon@gmail.com  
Flight Id: hyd239 Flight name: Spicejet  
On-boarding: Hyderabad Destination: Banglore  
Passengers:  
1. Name: Jack, Age: 23  
2. Name: Alex, Age: 33  
3. Name: John, Age: 43  
Booking date: 2023-08-23 Journey date: 2023-08-31  
Journey Time: 20:15 Total price: 17100  
Booking status: cancelled

Booking ID: 64e607242c862c07fa865b8d

Booking ID: 64e604fa1b698133e1a38d19

### Users

6

[View all](#)

### Bookings

7

[View all](#)

### Flights

6

[View all](#)

### New Operator Applications

No new requests..

### All Users

UserId 64e5fc... Username hola Email hola@gmail.com

UserId 64e9d2e0f7964122dbe8d098 Username alex Email alex@gmail.com

### Flight Operators

Id 64e8ce302bb50798fe630779 Flight Name spicejet Email spicejet@gmail.com

Id 64e8d11154e48a90d1c0f26b Flight Name Indigo Email indigo@gmail.com

Id 64e9d38e5d17bcb51a27b36a Flight Name Air Vistara Email vistara@gmail.com

#### Bookings

4

[View all](#)

#### Flights

2

[View all](#)

#### New Flight

(new route)

[Add now](#)

**SB Flights (Admin)**

**Bookings**

Booking ID: 64ec8c3c4622709484005484  
 Mobile: 7669678988 Email: harsha@gmail.com  
 Flight Id: cni2321 Flight name: Indigo  
 On-boarding: Chennai Destination: Banglore  
 Passengers: Seats: B-1, B-2  
     1. Name: Alex, Age: 44  
     2. Name: Snyder, Age: 55  
 Booking date: 2023-08-28 Journey date: 2023-08-31  
 Journey Time: 18:40 Total price: 7200  
 Booking status: confirmed

[Cancel Ticket](#)

Booking ID: 64e9d3fe5d17bcb51a27b3ca  
 Mobile: 9993478322 Email: user@gmail.com  
 Flight Id: dl4092 Flight name: Air Vistara  
 On-boarding: Delhi Destination: Kolkata  
 Passengers: Seats: P-1  
     1. Name: Alex, Age: 32  
 Booking date: 2023-08-26 Journey date: 2023-08-29  
 Journey Time: 18:00 Total price: 4200  
 Booking status: confirmed

[Cancel Ticket](#)

---

Booking ID: 64e9d313f7964122dbe8d0ae  
 Mobile: 9993478322 Email: user@gmail.com

Booking ID: 64e608bb2c862c07fa865bca  
 Mobile: 7869868765 Email: simon@gmail.com

**SB Flights (Operator)**

[Home](#) [Bookings](#) [Flights](#) [Add Flight](#) [Logout](#)

**Add new Flight**

Flight Name  
Indigo

Flight Id

▼
 Departure City  
Select

○
 Departure Time  
--:--

▼
 Destination City  
Select

○
 Arrival time  
--:--

Total seats  
0

Base price  
0

[Add now](#)

## 🔗 Demo Link

[FlightFinder App Demo](#)

## Screenshots Covered

- Landing Page
- Authentication
- Flight Search & Booking
- Admin Dashboard
- User & Booking Management

- Add/Edit Flights
  - View All Users & Bookings
- 

## 15. Conclusion

**FlightFinder** successfully demonstrates how a modern web application can simplify complex travel booking processes. It highlights full-stack development best practices, from secure authentication to real-time CRUD operations with a clean UI. Built with scalability in mind, this app can be further extended with payment integrations, live flight APIs, and user notifications.

---