Full Stack Development with MERN

1. Introduction

- **Project Title:** Flight booking
- **Team Members:** (Team id:NM2024TMID00136)

Ezhilarasi D -413021104009 (Team leader)

Vinisha R-413021104046

Kishore -413021104001

Santhosh R-413021104304

2. Project Overview

- **Purpose:** The purpose of the flight booking application is to provide users with a convenient, efficient, and user-friendly platform to search for, compare, and book flights. It aims to simplify the travel planning process by offering real-time flight availability, pricing, and customization options to meet diverse customer needs.
- **Goals:**

1. **Enhanced User Experience**:
   - Deliver a seamless and intuitive interface for searching and booking flights.
   - Minimize the time and effort required for flight selection and payment.
2. **Cost Efficiency and Transparency**:
   - Provide transparent pricing, including all taxes and fees.
   - Offer promotional deals, discounts, and loyalty program integration.
3. **Accessibility and Convenience**:
   - Ensure the platform is accessible via web and mobile devices.
   - Support multiple languages and currencies for a global audience.
4. **Secure Transactions**:
   - Implement secure payment gateways for a variety of payment methods.
   - Protect user data with robust security protocols.
5. **Customer Support**:
   - Provide 24/7 assistance for queries, changes, or cancellations.
   - Offer real-time updates on flight statuses and booking confirmations.

- **Features:**

    1. **Search Flights**: Users can search for one-way, round-trip, or multi-city flights by entering their origin, destination, travel dates, and passenger details.

    2. **Advanced Filters**: Filter results by:

        - ✓ Price range
        - ✓ Airlines
        - ✓ Departure and arrival times
        - ✓ Layovers and direct flights
        - ✓ Class (economy, premium economy, business, first-class)

3. Architecture

A robust and scalable architecture for a flight booking application must efficiently handle complex operations like real-time search, booking, secure transactions, and integrations with external systems. Below is an overview of a layered architecture:

a) Front-End

This layer interacts directly with the users.

- ❖ **Web Application**: Built using frameworks like React, Angular, or Vue.js for a responsive and interactive user interface.

    - ❖ **Features**:

        - o User-friendly interface for flight search, booking, and payment.
        - o Localization for multi-language and multi-currency support.
        - o Notifications for booking updates, flight status, and offers.

b) Back-End

The core business logic and functionality reside in this layer.

I. **Microservices**:
   a. **Flight Search Service**: Manages search requests and retrieves flight data from external APIs or databases.
   b. **Booking Service**: Handles seat selection, reservation, and payment integration.
   c. **User Management Service**: Manages user profiles, preferences, and travel history.
   d. **Notifications Service**: Sends emails, SMS, and push notifications.

II. **Real-Time Services**:
   a. Manages live updates on seat availability, prices, and flight status.
   b. Technologies like WebSockets or Server-Sent Events (SSE).

c) Database :

- **Relational Database (RDBMS)**: For structured data like user profiles, bookings, and payment records. (e.g., MySQL, PostgreSQL)
- **NoSQL Database**: For unstructured data like search logs and real-time analytics. (e.g., MongoDB, DynamoDB)

Diagram Overview:

1. **Clients**:
   - Users interact via web or mobile applications.
2. **API Gateway**:
   - Directs requests to respective microservices.
3. **Databases**:
   - Stores structured and unstructured data.
4. **External Integrations**:
   - Communicates with GDS, payment gateways, and third-party APIs.
5. **Infrastructure**:
   - Deployed on a cloud platform with monitoring and scaling capabilities.

This architecture ensures modularity, scalability, and fault tolerance, making it ideal for a flight booking application.

4. Setup Instructions

a) **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

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

c) **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.

d) **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.

5 Folder Structure :

Static files like `index.html`, favicon, and manifest files.

Src:

The primary source folder containing all application logic and resources.

Assets :

i. **Purpose**: Store static resources.

ii. **Subfolders**:

- `/images`: Icons, logos, or general images.
- `/icons`: SVGs or other icons.
- `/styles`: Global CSS, SCSS, or theme file

Components:

a) **Purpose**: Reusable UI components, further subdivided by type or functionality.
b) **Subfolders**:

- `/common`: Generic components like buttons, modals, dropdowns.
- `/layout`: Navigation bars, footers, and header components.
- `/search`: Components for flight search and filters.
- `/booking`: Components for seat selection, summary, and booking details.
- `/user`: User profile and account-related components.
- `/notifications`: Toasts, alerts, or messages.

Features :

a) **Purpose**: Modular organization of app features.
b) **Subfolders**:

- `/auth`: Login, signup, and authentication logic.
- `/flightSearch`: Components and logic for flight search and filters.
- `/flightDetails`: Handles flight details and itineraries.
- `/payment`: Payment gateway integration and confirmation.
- `/profile`: User profile management.

Pages :

a) **Purpose**: Full-page views rendered via routing.
b) **Subfolders**:

- `/home`: Landing page.
- `/searchResults`: Results displayed after flight search.
- `/booking`: Flight booking page.
- `/profile`: User profile and history.
- `/error`: Custom error pages like 404 or server errors.

6. Running the Application

- Provide commands to start the frontend and backend servers locally.
  - **Frontend:** `npm start` in the client directory.
  - **Backend:** `npm start` in the server directory.

7. API Documentation

This document provides an overview of the endpoints exposed by the backend. Each endpoint includes the HTTP method, parameters, and example responses.

```
{
  "name": "John Doe",
  "email": "johndoe@example.com",
  "password": "securepassword"
}
```

Response:

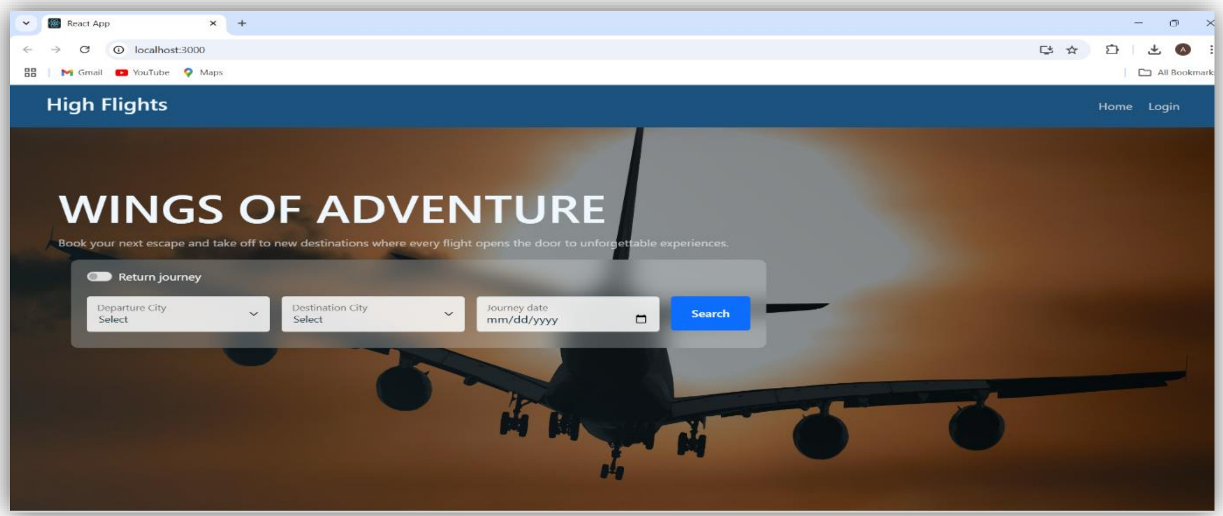
```
{
  "message": "Signup successful",
  "userId": "12345"
}
```

8. Authentication

Authentication verifies the identity of users. In this project, it is implemented using **JSON Web Tokens (JWT)**.

1. **User Signup**:
   - Endpoint: `/api/auth/signup`
   - Users provide their details (e.g., name, email, and password).
   - Passwords are securely hashed using a library like **bcrypt** and stored in the database.
   - After successful registration, the user is notified (but no token is issued yet).
2. **User Login**:
   - Endpoint: `/api/auth/login`
   - Users provide their credentials (email and password).
   - The password is verified against the hashed version stored in the database.
   - Upon successful authentication:
     - A **JWT** is generated and returned to the user.
     - This token contains the user's ID, email, and other claims (e.g., roles) in its payload.
     - The token is signed using a secret key to ensure its integrity.
3. **Token Generation**:
   - The JWT is generated using libraries like **jsonwebtoken**.
   - Claims include:
     - `sub` (subject): User ID.
     - `iat` (issued at): Timestamp of token issuance.
     - `exp` (expiration): Token expiry time .
4. **Storing the Token**:
   - The client stores the JWT in **localStorage** or **cookies** (with `HttpOnly` for security).
   - For mobile apps, secure storage mechanisms like **Keychain** (iOS) or **Keystore** (Android) are used

9. User Interface:


High Flights

Home Login

WINGS OF ADVENTURE

Book your next escape and take off to new destinations where every flight opens the door to unforgettable experiences.

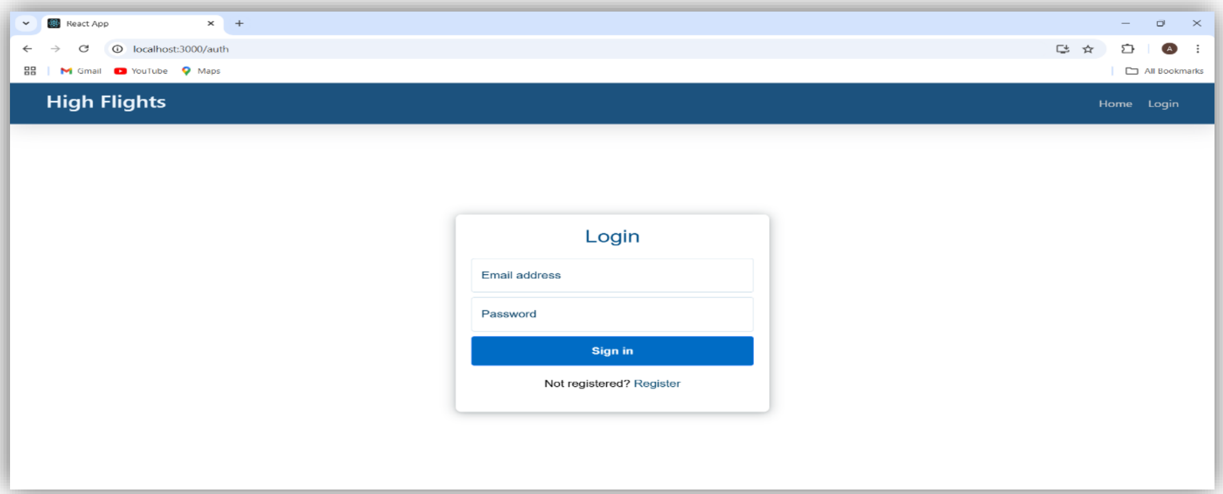
Return journey

Departure City Select

Destination City Select

Journey date mm/dd/yyyy

Search


High Flights

Home Login

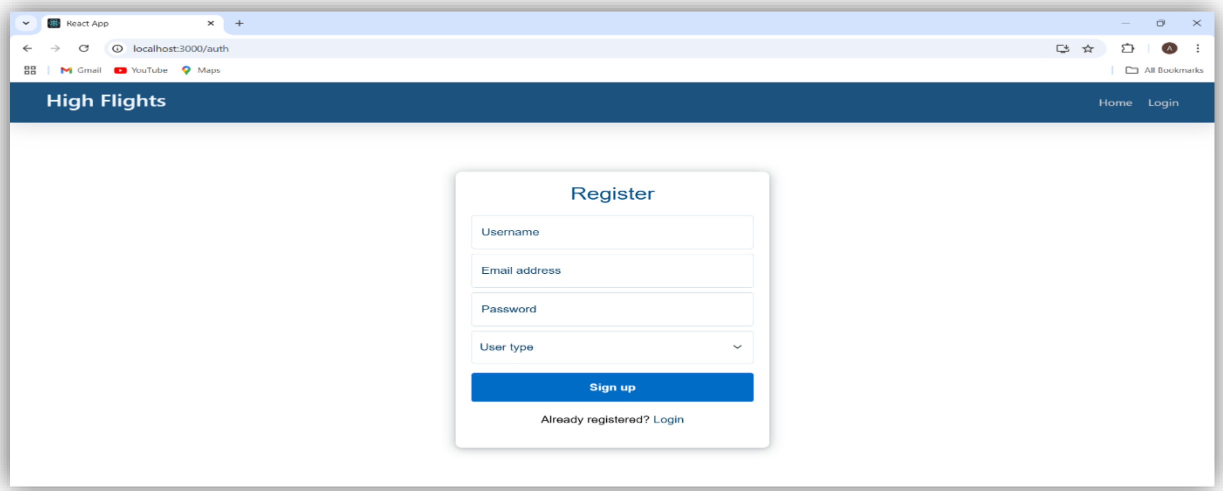
Login

Email address

Password

Sign in

Not registered? Register


High Flights

Home Login

Register

Username

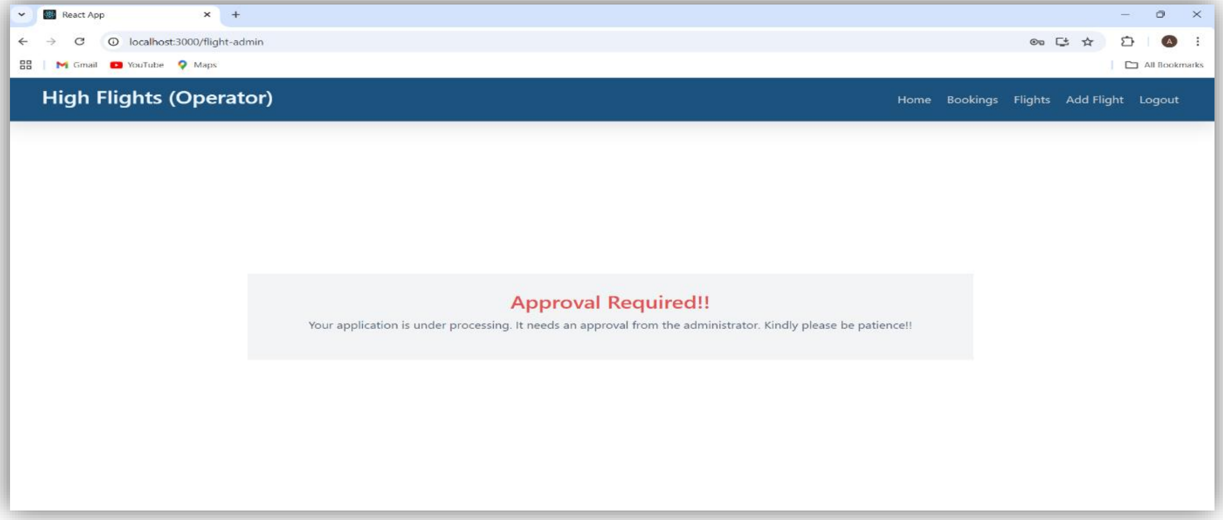
Email address

Password

User type

Sign up

Already registered? Login


React App

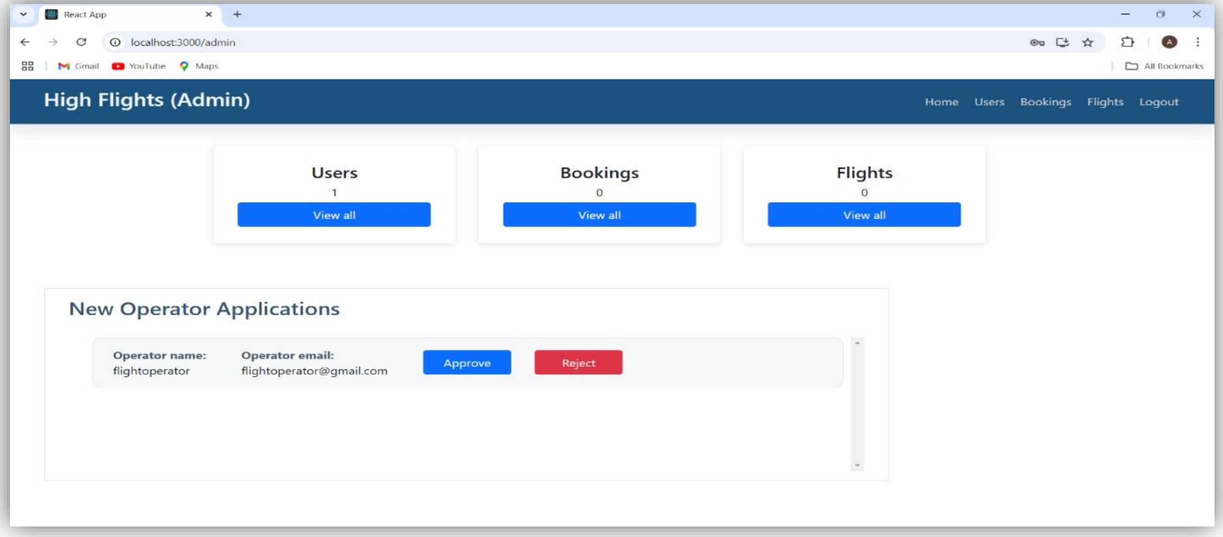
localhost:3000/flight-admin

High Flights (Operator)

Home Bookings Flights Add Flight Logout

Approval Required!!

Your application is under processing. It needs an approval from the administrator. Kindly please be patience!!


React App

localhost:3000/admin

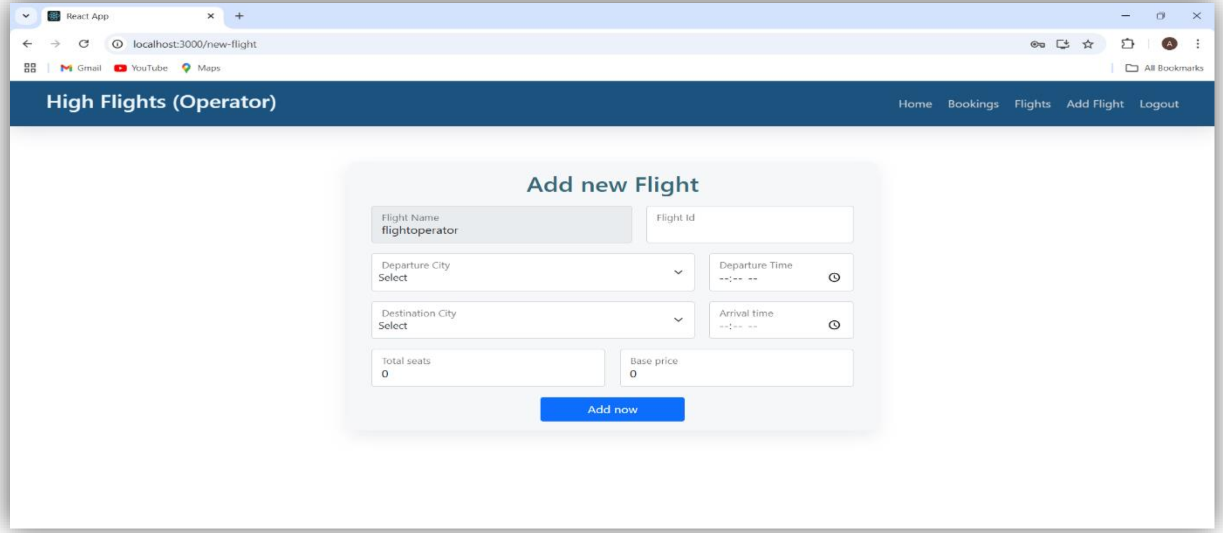
High Flights (Admin)

Home Users Bookings Flights Logout

Users	Bookings	Flights
1	0	0
View all	View all	View all

New Operator Applications

Operator name: flightoperator **Operator email:** flightoperator@gmail.com Approve Reject


React App

localhost:3000/new-flight

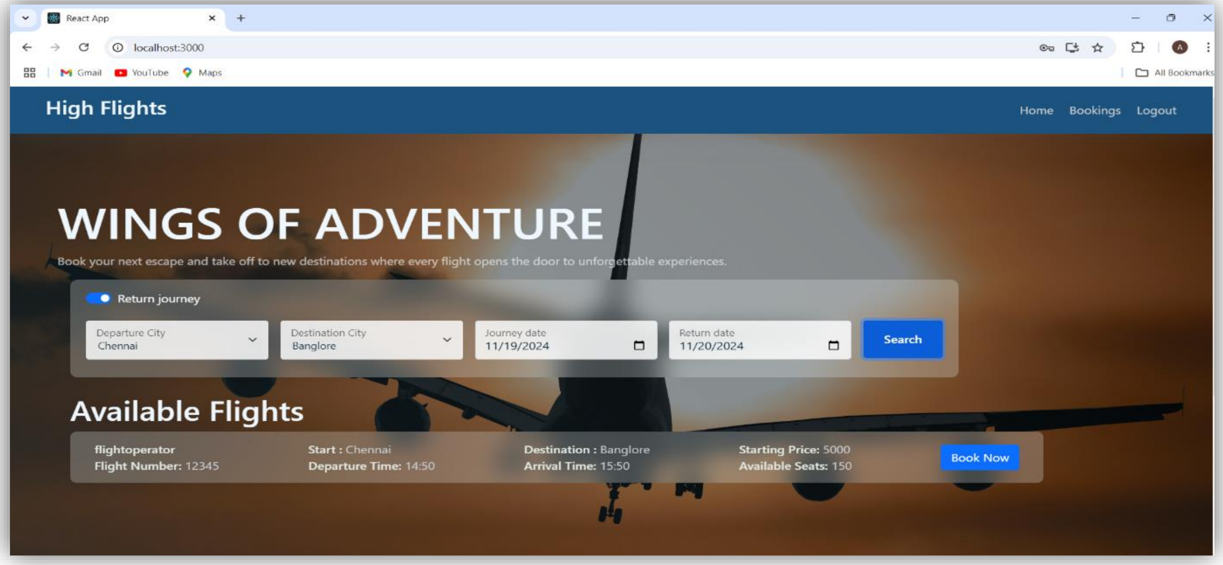
High Flights (Operator)

Home Bookings Flights Add Flight Logout

Add new Flight

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

Add now



High Flights

WINGS OF ADVENTURE

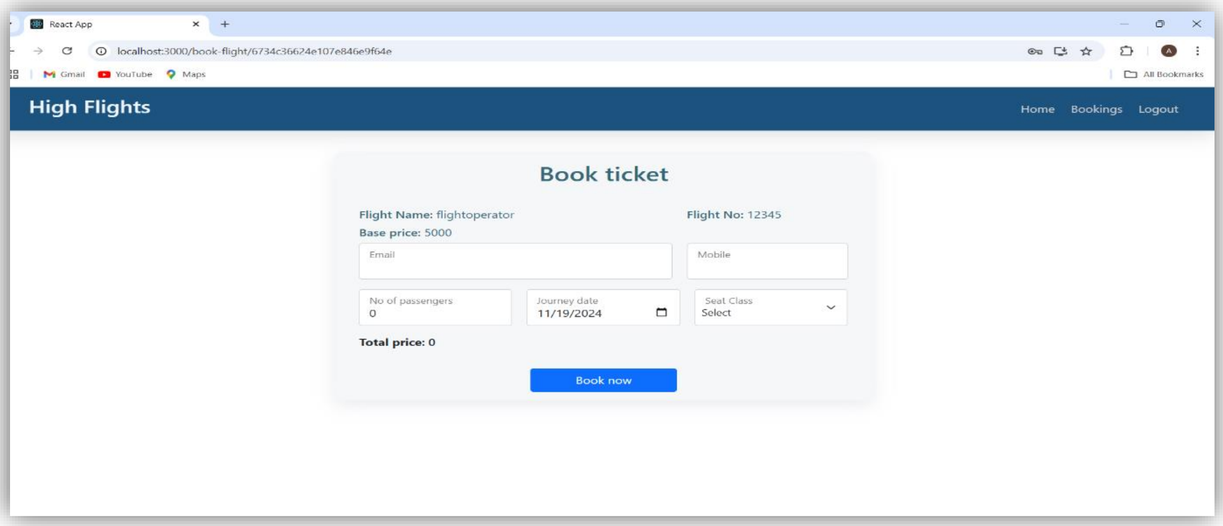
Book your next escape and take off to new destinations where every flight opens the door to unforgettable experiences.

Return journey

Departure City: Chennai
Destination City: Banglore
Journey date: 11/19/2024
Return date: 11/20/2024

Available Flights

flightoperator
Flight Number: 12345
Start : Chennai
Departure Time: 14:50
Destination : Banglore
Arrival Time: 15:50
Starting Price: 5000
Available Seats: 150



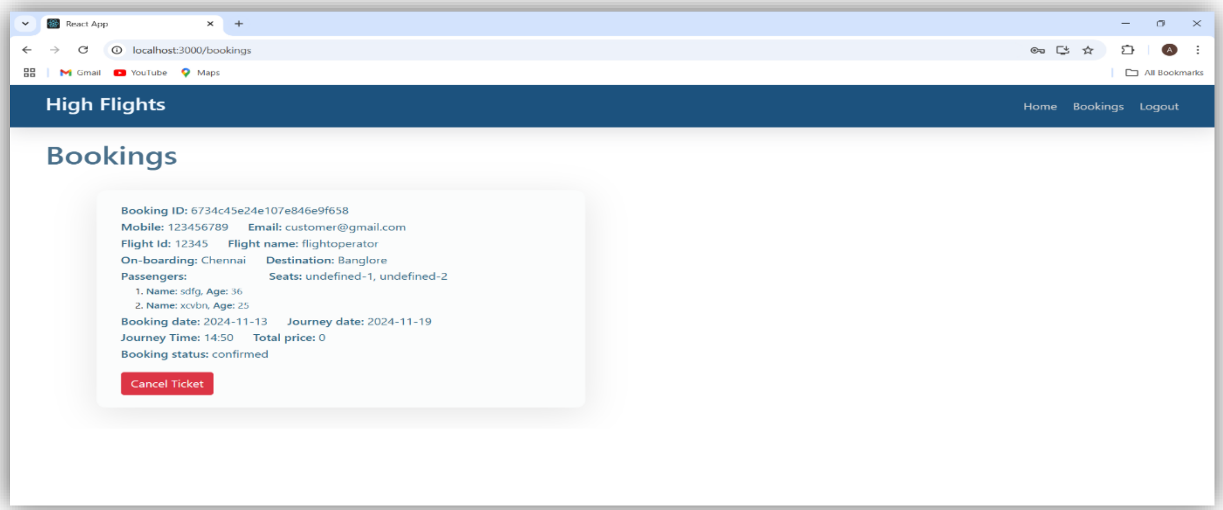
High Flights

Book ticket

Flight Name: flightoperator
Flight No: 12345
Base price: 5000

Email
Mobile
No of passengers: 0
Journey date: 11/19/2024
Seat Class: Select

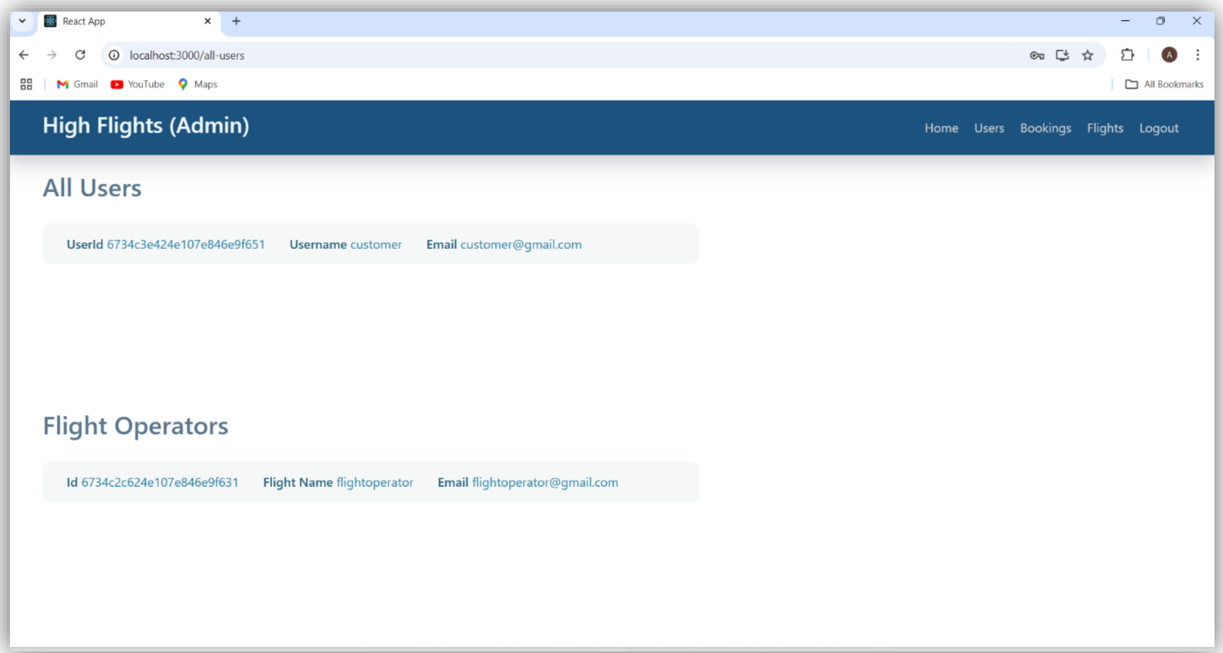
Total price: 0



High Flights

Bookings

Booking ID: 6734c45e24e107e846e9f658
Mobile: 123456789 Email: customer@gmail.com
Flight Id: 12345 Flight name: flightoperator
On-boarding: Chennai Destination: Banglore
Passengers: Seats: undefined-1, undefined-2
1. Name: sdfg, Age: 36
2. Name: xcvbn, Age: 25
Booking date: 2024-11-13 Journey date: 2024-11-19
Journey Time: 14:50 Total price: 0
Booking status: confirmed

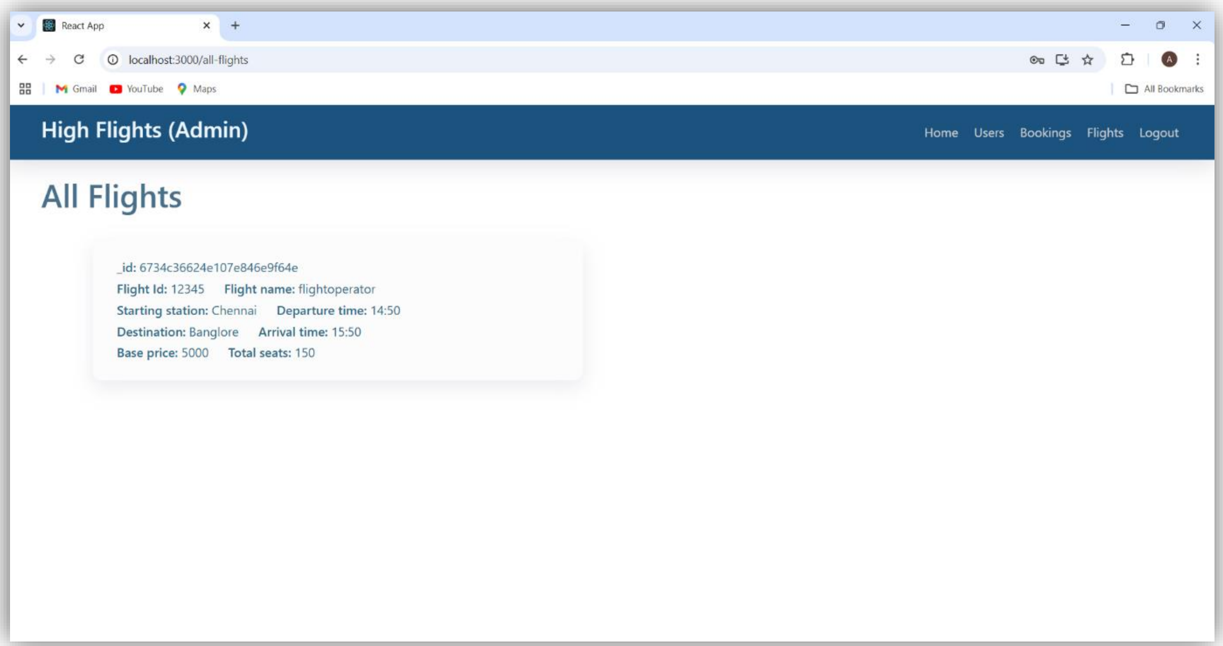


All Users

UserId 6734c3e424e107e846e9f651 **Username** customer **Email** customer@gmail.com

Flight Operators

Id 6734c2c624e107e846e9f631 **Flight Name** flightoperator **Email** flightoperator@gmail.com



All Flights

_id: 6734c36624e107e846e9f64e
Flight Id: 12345 **Flight name:** flightoperator
Starting station: Chennai **Departure time:** 14:50
Destination: Banglore **Arrival time:** 15:50
Base price: 5000 **Total seats:** 150

10. Testing

- Manual Testing

11. Screenshots or Demo

- Screenshot and demo video link: https://github.com/ezhilmahi/NM2024TMID00136_ezhilarasi.git

12. Known Issues

- Initially, if we book a certain number of seats, it will be treated as an unlimited number of seats.
- The page is not fully responsive on some mobile devices, causing some form fields to overlap or appear off-screen.

13. Future Enhancements

- Expand the application to support multiple currencies and languages to accommodate international users.
- Expand the app to allow users to book hotels and car rentals along with their flights in one seamless transaction.