NAAN MUDHALVANPROJECT

MERN stack powered by MongoDB

PROJECT TITLE

Flight Booking Application using MERNStack

Submitted by the team members of Final Year IT‘B’

PRASANTH.J	**311421205063**
PRAVEEN KUMAR.G	**311421205064**
VISHWA.E	**3114212050105**
YUVARAJ.P	**3114212050108**

DEPARTMENT OF INFORMATION TECHNOLOGY

MEENAKSHI COLLEGE OF ENGINEERING

12, VEMBULIAMMAN KOVIL STREET, WESTK.K. NAGAR,

CHENNAI - 600 078, TAMILNADU.

(AFFILATED TO ANNA UNIVERSITY)

Flight booking App

ABSTRACT :

The **Flight Booking Application (FBA**) is a comprehensive travel solution leveraging the MERN stack (MongoDB, Express, React, Node.js) to provide accessible, flexible, and user-friendly booking options for travelers worldwide.

FBA incorporates a suite of features designed to streamline the flight booking experience, including flight search, reservation management, and flexible accessibility across devices.

Through a client-server architecture, FBA enables real-time data exchange and supports robust functionalities such as user authentication, role-based access, and payment processing for secure transactions. By offering a seamless and engaging user experience, this platform empowers users to search for flights, book tickets, manage reservations, and receive travel confirmations.

The platform also supports airline operators in managing flight details and schedules and enables administrators to oversee the system effectively. This documentation details the requirements, architecture, features, and implementation steps necessary to develop the FBA using MERN, serving as a foundational guide for an effective flight booking ecosystem.

Keywords: Flight Booking, Travel Application, MERN Stack, Vite

TABLE OF CONTENTS

1. INTRODUCTION :

In recent years, the demand for convenient and accessible travel solutions has surged, driven by the need for flexible, user-friendly, and efficient booking options that cater to different schedules, destinations, and budget preferences.

A Flight Booking Application (FBA) is a digital system designed to meet this demand, providing a comprehensive travel booking environment that allows users to search for flights, manage reservations, and receive confirmation for booked tickets.

Built using the MERN stack, FBA leverages MongoDB for data storage, Express.js for server-side functionality, React for a dynamic front-end interface, and Node.js for back-end development, ensuring a scalable, efficient, and responsive booking platform. FBA is built with a focus on

accessibility and usability, allowing travelers of all backgrounds and technical proficiency to navigate the platform easily.

It supports features like flight search, seat selection, reservation management, live status updates, and secure payment processing. Moreover, the platform offers flexibility for booking classes across budget and premium options, catering to a broad audience. With roles for travelers, airline operators, and administrators, FBA provides a structured yet adaptable framework for managing bookings and flight details, enhancing the travel experience, and supporting airline operators in schedule management and customer engagement.

1.1. PURPOSE :

The purpose of the Flight Booking Application (FBA) is:

- To provide a centralized online platform that facilitates convenient, flexible flight booking accessible to users worldwide.
- To enable users to search for flights, make reservations, manage bookings, and receive confirmations, supporting seamless travel planning.
- To offer an interactive and user-friendly interface that allows travelers to navigate options easily, select seats, and access real-time flight information.
- To provide airline operators with a streamlined system for managing flight schedules, seat availability, and customer reservations effectively.
- To allow administrators to monitor platform operations, manage user activities, and ensure data security and integrity.

- To create a scalable, efficient, and cost-effective solution for airlines and travel agencies looking to offer online booking services.

1.2. SCOPE :

The scope of the Flight Booking Application (FBA) encompasses the development, deployment, and maintenance of a comprehensive online flight booking system using the MERN stack. Key components and features within this scope include:

- **User Management**: Support user roles for travelers, airline operators, and administrators, including registration, login, and profile management.

- **Flight Management**: Enable airline operators to create, update, organize, and publish flight schedules, pricing, and seat availability.
- **Booking Tools**: Offer features for searching flights, selecting seats, and making reservations to streamline the booking process for travelers.
- **Reservation Tracking**: Implement features for users to monitor their booking status and receive real-time updates on flight schedules.
- **E-Ticketing**: Issue digital tickets upon booking confirmation, providing travelers with necessary flight details and booking credentials.
- **Payment and Pricing Options**: Provide a payment gateway for booking transactions, supporting both one-time payments and promotional discounts.
- **Cross-Device Accessibility**: Ensure compatibility across various devices (PCs, tablets, smartphones) to allow access from any location with an internet connection.
- **Front-end & Back-end Integration**: Leverage the MERN stack for efficient front-end and back-end communication and database management.
- **Admin Panel**: Include an administrative dashboard for monitoring user activity, managing flight listings, and handling platform maintenance tasks.

1.3. PROJECT OBJECTIVE:

The objective of the Flight Booking Application (FBA) project is to develop a versatile and accessible online flight booking system using the MERN stack (MongoDB, Express.js, React, and Node.js). This platform aims to provide a seamless, user-friendly experience for travelers, airline operators, and administrators by offering key features that facilitate efficient flight search, booking, and reservation management.

2. SYSTEM REQUIRMENTS :

2.1. HARDWARE :

- Operating System : Windows 8 or higher
- RAM : 4 GB or more (8 GB recommended for smooth development experience)

2.2. SOFTWARE :

- Node.js : LTS version for back-end and front-end development

- MongoDB : For database management using MongoDB Atlas or a local instance
- React : For front-end framework
- Express.js : For back-end framework
- Git : Version control Code Editor : e.g., Visual Studio Code
- Web Browsers : Two web browsers installed for testing compatibility

  (e.g.,Chrome andFirefox)

2.3. NETWORK :

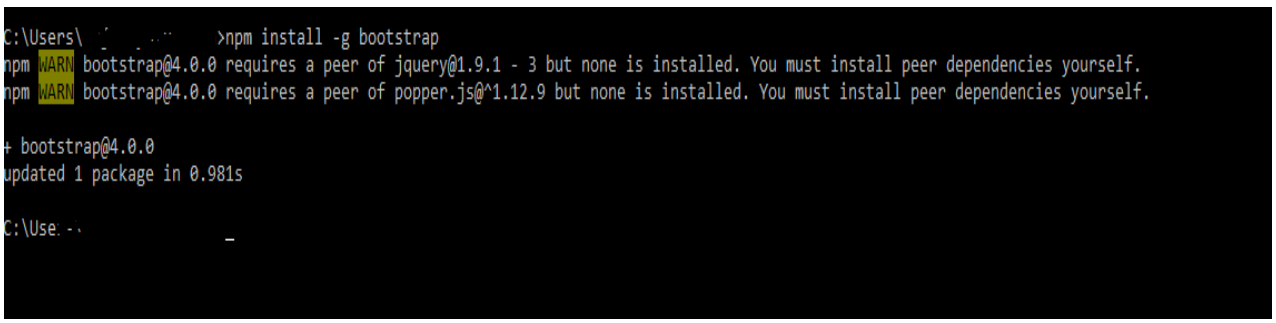
- Bandwidth : 30 Mbps

3. PRE-REQUISITES :

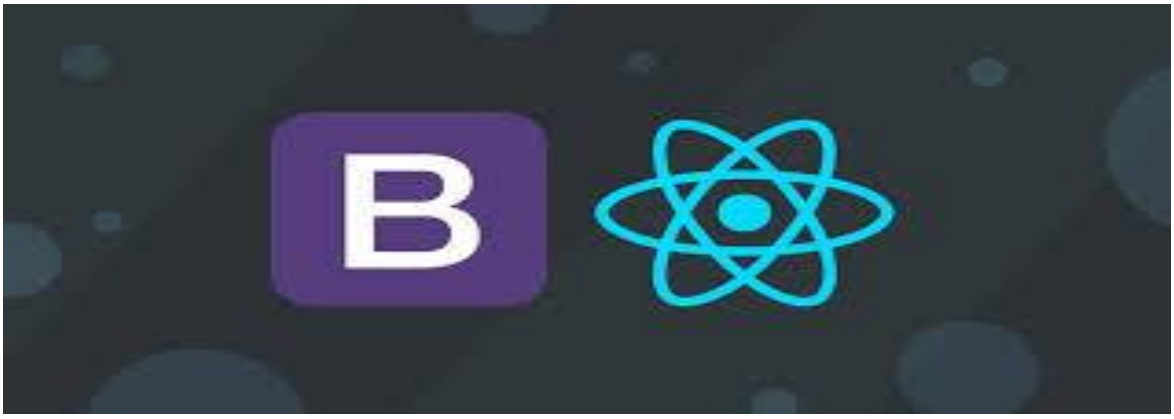
Here are the key prerequisites for developing a full-stack application usingNode.js, Express.js, MongoDB,

✓ **Bootstrap**

Bootstrap is a popular open-source CSS framework focused on simplifying the development of responsive and mobile-first web pages. It includes a collection of pre-designed components, such as navigation bars, forms, buttons, and grids, which are easily customizable to fit various design needs. Bootstrap's grid system and extensive library streamline layout design, ensuring consistency across devices. To install, use: `npm install bootstrap`.
This framework greatly enhances the efficiency and accessibility of front-end development.



```
C:\Users\...>npm install -g bootstrap
npm WARN bootstrap@4.0.0 requires a peer of jquery@1.9.1 - 3 but none is installed. You must install peer dependencies yourself.
npm WARN bootstrap@4.0.0 requires a peer of popper.js@^1.12.9 but none is installed. You must install peer dependencies yourself.

+ bootstrap@4.0.0
updated 1 package in 0.562s

C:\Users\...>npm install -g jquery
+ jquery@3.3.1
updated 1 package in 0.936s

C:\Users\...>npm install -g popper.js
+ popper.js@1.12.9
updated 1 package in 0.89s

C:\Users\...>npm install -g bootstrap
npm WARN bootstrap@4.0.0 requires a peer of jquery@1.9.1 - 3 but none is installed. You must install peer dependencies yourself.
npm WARN bootstrap@4.0.0 requires a peer of popper.js@^1.12.9 but none is installed. You must install peer dependencies yourself.

+ bootstrap@4.0.0
updated 1 package in 1.067s
```



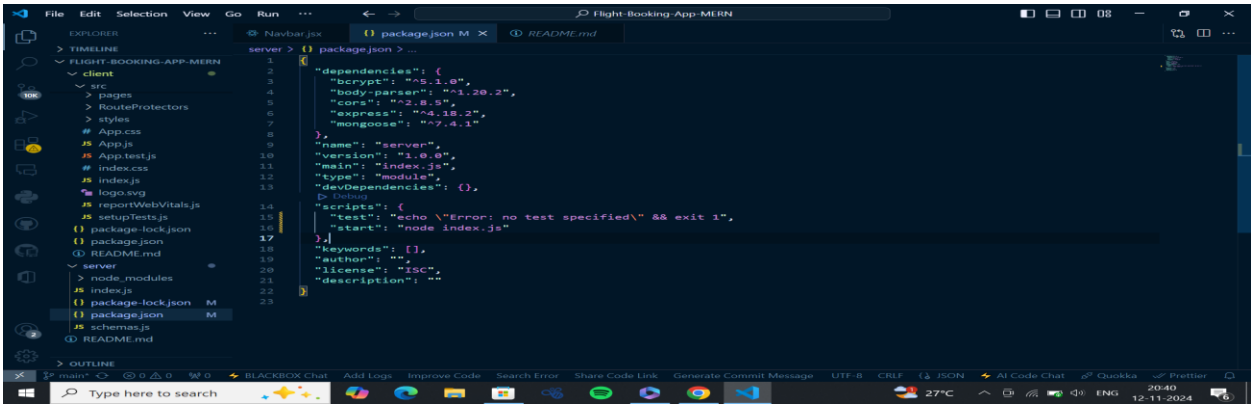
- ✓ **Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScriptcode on the server-side. It provides a scalable and efficient platformfor buildingnetworkapplications. Install Node.js and npm on your development machine, as they are requiredtorunJavaScript on the server-side.

Download: https://nodejs.org/en/download/

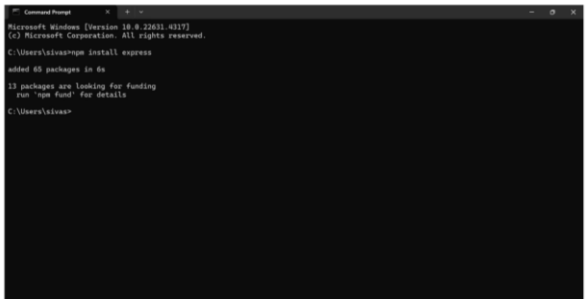
Installation instructions: https://nodejs.org/en/download/package-manager/

Run “npm init” to get default dependencies



✓ **Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifiestheprocess of creating robust APIs and web applications, offering features like routing, middlewaresupport, and modular architecture. Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development. Installation: Open your command prompt or terminal and run the following command: npm install express



✓ **MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-likeformat. It provides high performance, horizontal scalability, and seamless integration withNode.js, making it ideal for handling large amounts of structured and unstructured data. Set up a MongoDB database to store your application's data. Download: https://www.mongodb.com/try/download/community Installation instructions: https://docs.mongodb.com/manual/installation/

Installation instructions: https://docs.mongodb.com/manual/installation/

✓ **React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developersto create interactive and reusable UI components, making it easier to build dynamicandresponsive web applications. 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, andJavaScript for client-side interactivity is essential.

✓ **Database Connectivity:**

Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoosetoconnect your Node.js server with the MongoDB database and performCRUD(Create, Read, Update, Delete) operations.

✓ **Front-end Framework:**

Utilize React.js to build the user-facing part of the application, including enteringbookingroom, status of the booking, and user interfaces for the admin dashboard. For making better UI wehave also used some libraries like material UI and boostrap. Install the required dependencies by running the following commands:

cd frontend || npm install

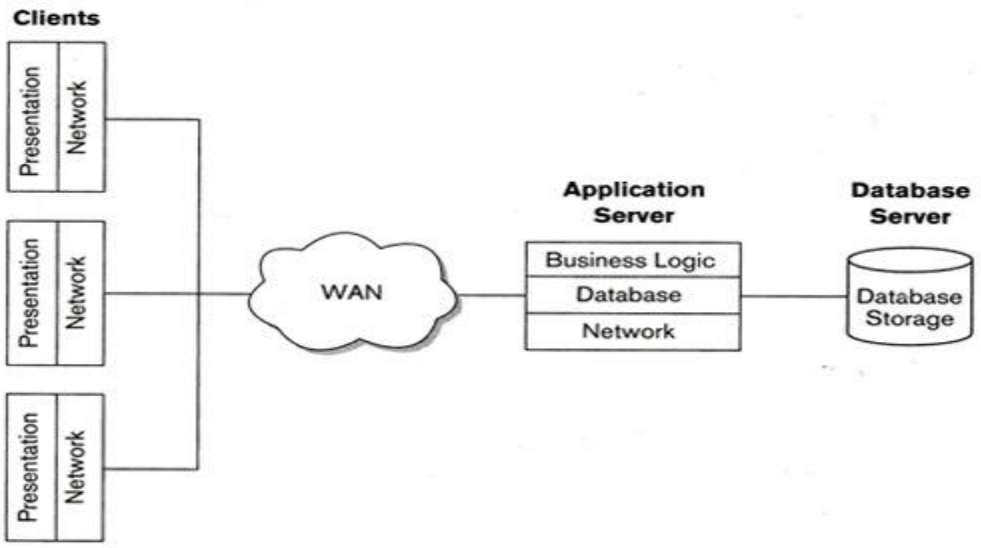
cd ../backend || npm install

Start the Development Server:

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

- The Flight Booking app will be accessible at http://localhost:3000

4..ARCHITECTURE:

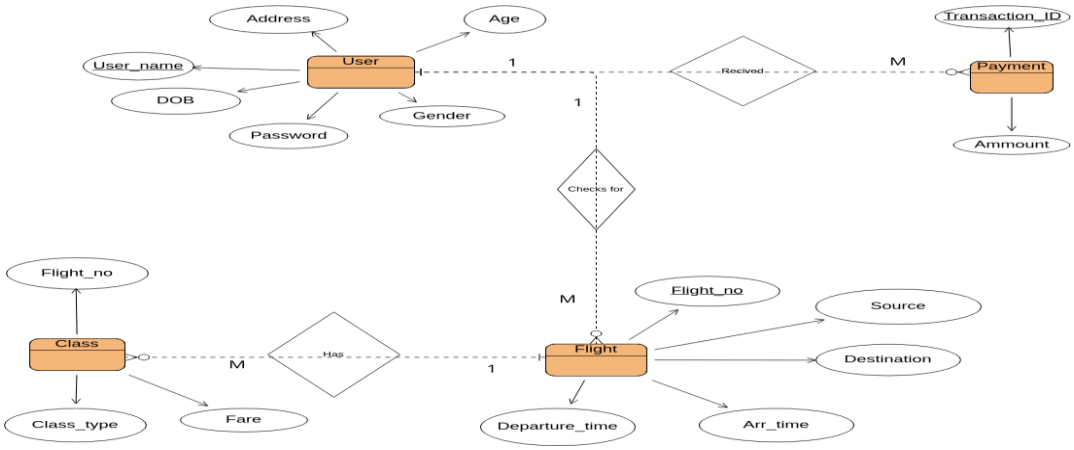


4.1. TECHNICAL ARCHITECTURE:

The technical architecture of the Flight Booking Application (FBA) follows a client-server model, where the front-end acts as the client and the back-end serves as the server. The front-end not only handles the user interface and presentation but also utilizes the axios library to connect seamlessly with the back-end through RESTful APIs. The front-end employs the Bootstrap and Material UI libraries to provide a responsive and visually appealing user experience. On the back-end, Express.js is used to manage server-side logic and communication. MongoDB is leveraged for data storage and retrieval, offering scalable and efficient storage for user data, flight details, and booking information.

For smooth communication between the front-end and back-end, RESTful APIs are implemented, allowing efficient data exchange and modular integration. To secure user access, authentication and authorization protocols, such as JWT (JSON Web Tokens), are used to enforce role-based permissions for travelers, airline operators, and admins. The platform includes real-time updates, such as flight status notifications, to keep travelers informed. For payment processing, third-party payment gateway integration enables secure transactions, while data analytics and visualization tools provide insights into booking trends, helping administrators make data-driven decisions. Finally, systematic updates and security audits are conducted to maintain the platform's integrity and safeguard user data. This robust set of technical components allows the FBA to deliver a secure, scalable, and user-friendly flight booking experience.

5. ER - DIAGRAM :



Here there is 2 collections which have their own fields in :

- USERS
- FLIGHT

a. USERS ENTITY

a. USERS ENTITY

- Represents the users of the platform. **Attributes:**
  - **userID**: Unique identifier for each user (likely primary key).
  - **name**: Name of the user.
  - **email**: Email address associated with the user account.
  - **password**: User's password for secure login.
  - **type**: Indicates the type of user (e.g., traveler, airline operator, admin).

b. FLIGHTS ENTITY

- Represents the various flights available on the platform. **Attributes:**
  - **flightID**: Unique identifier for each flight.
  - **flight_number**: Flight number for identification.
  - **airline**: The airline operating the flight.

- **origin**: Departure location (city or airport).
- **destination**: Arrival location (city or airport).
- **departure_time**: Scheduled departure time.
- **arrival_time**: Scheduled arrival time.
- **price**: Ticket price, which may vary based on seating class or demand.
- **available_seats**: Tracks the number of seats available on the flight.

C. RELATIONSHIP - "CAN"

- This relationship shows the interaction between Users and Flights.
- **Users (travelers) can:**
  - Browse available flights.
  - Book tickets for selected flights.
  - View and manage their bookings, including seat selection and changes.
- **Users (airline operators) can:**
  - Create, edit, and manage flight schedules.
  - Update flight details, such as timings, destinations, and pricing.
  - Track seat availability and bookings.
- **Admin users can:**
  - Oversee all user activities.
  - Manage flight listings, pricing, and seat allocations.
  - Monitor platform transactions and data security.

Summary in Use Case Points:

- **User Registration/Login**: Users can register and log in with a unique userID, email, and password.

- **Flight Browsing and Booking**: Users can browse available flights and book tickets.
- **Flight Management for Operators**: Airline operators can create and manage flights, update details, and oversee seat reservations.
- **Booking Interaction**: Travelers can manage their bookings, including viewing details, selecting seats, and making changes.
- **Payment Handling**: Flight bookings can be paid for securely through an integrated payment gateway.
- **Data Tracking**: The platform tracks user bookings, flight schedules, and seat availability for efficient management and reporting.

6. PROJECT STRUCTURE:

The first image represents the front-end part, showing all the files and folders used in the UI development.

- **Root Folders and Files:**
  - **Front-end**:
    The main project folder containing all files and folders for the front-end part of the application.
  - **node_modules**:
    Contains all the dependencies and modules installed via npm (Node Package Manager). This folder is automatically generated when dependencies are installed.
  - **public**:
    Stores static files that can be served directly, such as `index.html` (the main HTML file for a React app), images, and other static assets. Files in this folder are accessible directly by the browser.

- **src**:
  This is the main source folder for the React components and other application code. All core logic, components, and styles are stored here.
- **Folders and Files Inside src**:
- **assets**:
  This folder typically contains images, icons, or other media assets used throughout the frontend.
- **components**:
  Houses all the components used in the application. Components are organized into subfolders based on functionality or user roles (e.g., admin, common, traveler, airline operator).
- **admin Folder**:
  Contains components specific to the admin functionalities of the application.

- **AdminHome.jsx**: The main dashboard or homepage component for admin users.
- **AllFlights.jsx**: Displays all flights available on the platform, likely for administrative oversight.
- **common Folder**: Contains shared components accessible by all users.
  - **AllFlights.jsx**: Shows a list of all available flights (possibly a common view accessible by all users).
  - **AxiosInstance.jsx**: Configures Axios for API requests, likely setting up base URLs or authorization headers.
  - **Dashboard.jsx**: A general dashboard component that could serve as the main page for logged-in users.
  - **Home.jsx**: The homepage or landing page for the application.
  - **Login.jsx**: The login component for user authentication.
  - **NavBar.jsx**: The navigation bar component, providing links to different parts of the application.
  - **Register.jsx**: The registration component for new users.

    - **UserHome.jsx**: The main page or dashboard for general users.

- **user Folder**: Contains subfolders for different user roles with specific components.
    - **traveler**:
        - **BookingDetails.jsx**: Displays details of a specific booking for travelers.
        - **MyBookings.jsx**: Lists the flights a traveler has booked.
        - **TravelerHome.jsx**: The main page or dashboard for travelers.
    - **airlineOperator**:
        - **AddFlight.jsx**: Component for airline operators to add or create new flights.
        - **OperatorHome.jsx**: The main page or dashboard for airline operators, showing relevant information such as upcoming flights and seat availability.
- **App.css**:
  Contains global styles for the application, defining the look and feel of the entire frontend.
- **App.jsx**:
  The root component of the application. It usually contains routes to various pages and loads other major components.
- **main.jsx**:
  The main entry point for the application, where the React app is rendered into the HTML document (usually `index.html` in the public folder).
- **.eslintrc.js**:
  Configuration file for ESLint, a tool used to enforce coding standards and style in JavaScript code.

This structure is well-organized, with separate folders for different user roles (admin, traveler, airline operator), as well as shared components for common functionality. Each user role has its specific set of components to encapsulate role-based features, making it easier to manage and extend the code. This setup supports scalability and maintainability for a comprehensive flight booking application.

The second image is of Backend part which is showing all the files and folders that havebeen used in backend development

- **Folder and File Structure**
  - **Config**:
    - This folder usually contains configuration files for setting up connections or other environmental variables needed by the application.
    - It likely includes settings for connecting to the database, such as the database URL and credentials, along with any other application-level configurations (e.g., API keys for payment gateways, flight data APIs, etc.).
  - **Controllers**:
    - **adminController.js**: Contains functions to handle admin-specific actions, such as managing flights, users, and payments.
    - **userController.js**: Manages actions related to regular users, such as registration, login, booking flights, managing bookings, or fetching user-specific information.
  - **Middlewares**:
    - **authMiddleware.js**: Contains middleware for handling authentication and authorization. It checks if users are authenticated before allowing access to certain routes, ensuring only logged-in users can access restricted areas of the application, such as booking or managing flights.

- **Routers**:
  - **adminRoutes.js**: Defines API endpoints specific to admin functionalities and maps each endpoint to corresponding controller functions in `adminController.js`. This may include flight management, user management, and payment processing.
  - **userRoutes.js**: Defines API endpoints for user functionalities and maps them to the functions in `userController.js`. This could include endpoints for login, registration, flight search, booking, and managing bookings.

- **Schemas**:
  - This folder includes all the models that define the structure of different collections or tables in the database.
  - **flightModel.js**: Defines the schema for flights, including properties like flight number, origin, destination, departure time, available seats, and other relevant flight details.
  - **paymentModel.js**: Handles the schema for flight payment details, including fields for transaction IDs, payment status, amount, and payment method.
  - **bookingModel.js**: Manages information on users who have booked flights, linking user IDs to specific flight IDs and storing details like booking status, booking date, and seat selection.
  - **userModel.js**: Defines the user schema, including fields like name, email, password, role (e.g., traveler or admin), and booking history.
- **uploads**:
  - This folder is used for storing files uploaded by users, such as profile pictures, booking-related documents (e.g., passports), or any other user-uploaded assets. It ensures that uploaded files are kept in an organized directory.
- **Environment and Configuration Files**:

    - **.env**: Stores environment variables such as database credentials, API keys (e.g., payment gateway credentials), and other sensitive information that should not be hard-coded.
    - **.gitignore**: Specifies files and folders to be ignored by Git, such as `node_modules`, `.env`, and other directories that do not need to be tracked by version control.
- **Entry Points and Package Files**:
    - **index.js**: The main entry point for the backend application. It typically sets up the server, connects to the database, and initializes middleware, routes, and error handling.
    - **package.json**: Lists dependencies, scripts, and metadata for the project. It manages the backend libraries and tools required to run the server.
    - **package-lock.json**: Records the exact versions of dependencies installed in `node_modules`, ensuring consistency across installations.

This backend structure supports a modular approach to handling a flight booking platform, where different sections are divided by purpose. Controllers handle specific actions, routes define accessible API endpoints, and middlewares enforce security and validation. Models structure the database, and configurations ensure easy setup across environments.

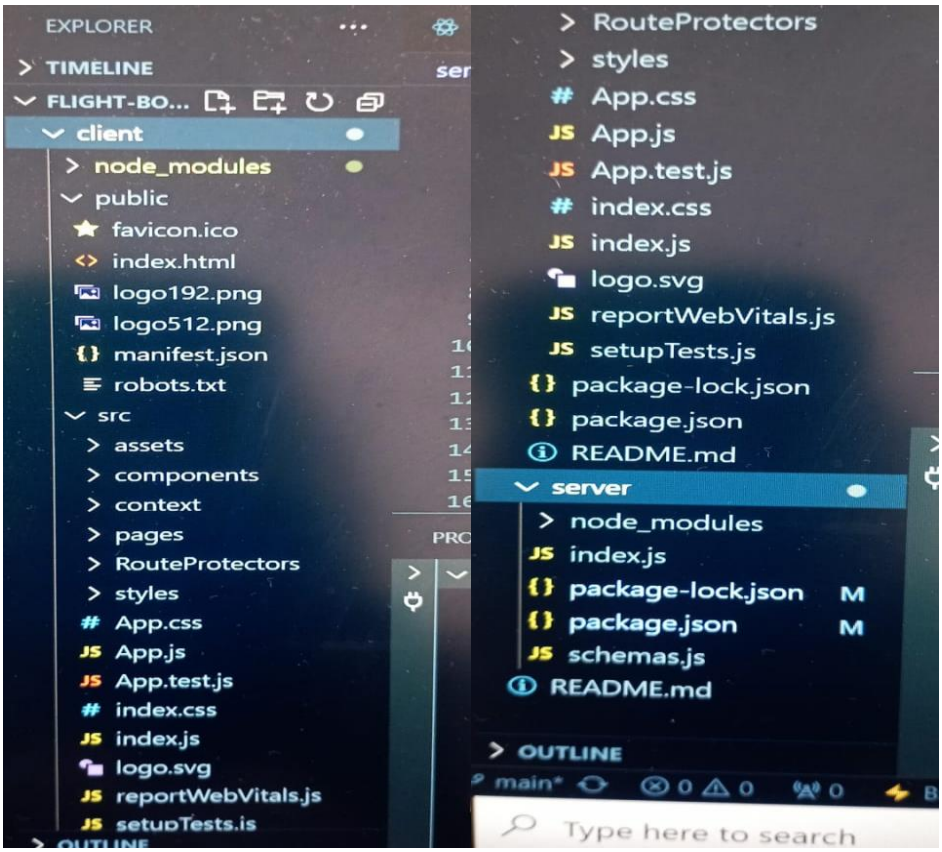
Here's a basic explanation for the third image, which includes:

- **.gitignore**:
    - Specifies files and directories that should be ignored by Git, preventing them from being tracked in version control. Common entries include `node_modules`, `.env`, `uploads` (for user-uploaded files), and any other files that are either sensitive (e.g., API keys, database credentials) or unnecessary for sharing in the repository. This helps keep the repository clean and secure.

Here's a basic explanation for the flight booking app's foundational files:

- **index.html**:
    - The main HTML file for the application. It serves as the entry point for the front-end application, where JavaScript bundles and stylesheets are injected. Typically, this file includes a root `<div>` where the front-end framework (e.g., React) mounts the application, ensuring a smooth user experience.
- **package-lock.json**:
    - Automatically generated by npm to track the exact versions of dependencies and sub-dependencies installed in the project. This file ensures that other developers or environments installing the project will have the same dependency versions, improving consistency and reliability.
- **package.json**:
    - The main configuration file for the Node.js project. It includes metadata about the project (like its name and version), as well as scripts, dependencies, and other configurations. This file is essential for installing and managing packages and for running tasks like `npm start` or `npm build`.
- **README.md**:
    - A markdown file that typically contains documentation for the project. It often includes an introduction, setup instructions, usage information, and any other details that help developers understand and contribute to the project.
- **vite.config.js**:
    - The configuration file for Vite, a fast build tool and development server for modern web projects. This file is used to customize Vite's behavior, such as setting up plugins, defining alias paths, and configuring the development and build environments.

This set of files represents the foundational setup for a modern flight booking app, including configuration, documentation, and project dependencies.



7.APPLICATION FLOW:

Here's an application flow for a flight booking app with roles and responsibilities:**I. Customer (User):**

- Can search for available flights based on departure and destination cities, travel dates, and number of passengers.
- Can view available flights, including flight details such as airline, flight number, departure and arrival times, and prices.
- Can filter flights based on preferences like flight duration, price range, and airline.
- Can book a flight by selecting available flights and entering required passenger details.
- Can make payments for the flight booking via secure payment gateways.

- Can view and manage their booked flights, such as checking flight status, canceling or rescheduling bookings.
- Receives email or SMS confirmations for flight bookings.

II. Airline Staff (Administrator):

- Can manage available flights, including adding, editing, and removing flight details such as departure/arrival times, prices, and seat availability.
- Can manage the schedule of flights, ensuring that all flight details are up-to-date.
- Can monitor and update booking statuses, including handling cancellations and rescheduling.
- Can manage customer bookings, issuing refunds or rebookings when needed.
- Can access user booking data for administrative purposes and customer support.

III. Admin:

- Can oversee all customer and staff activities on the platform.
- Can manage the entire system, including users, flight data, payment transactions, and policies.
- Can generate reports and track system performance and revenue.
- Can handle customer complaints or issues related to bookings, payments, and other services.

This flow ensures a smooth and efficient process for booking flights, managing users, and maintaining flight-related data, creating a streamlined user experience.

8. PROJECT SETUP & CONFIGURATION :

- Folder Setup
  - Create Frontend and Backend folder

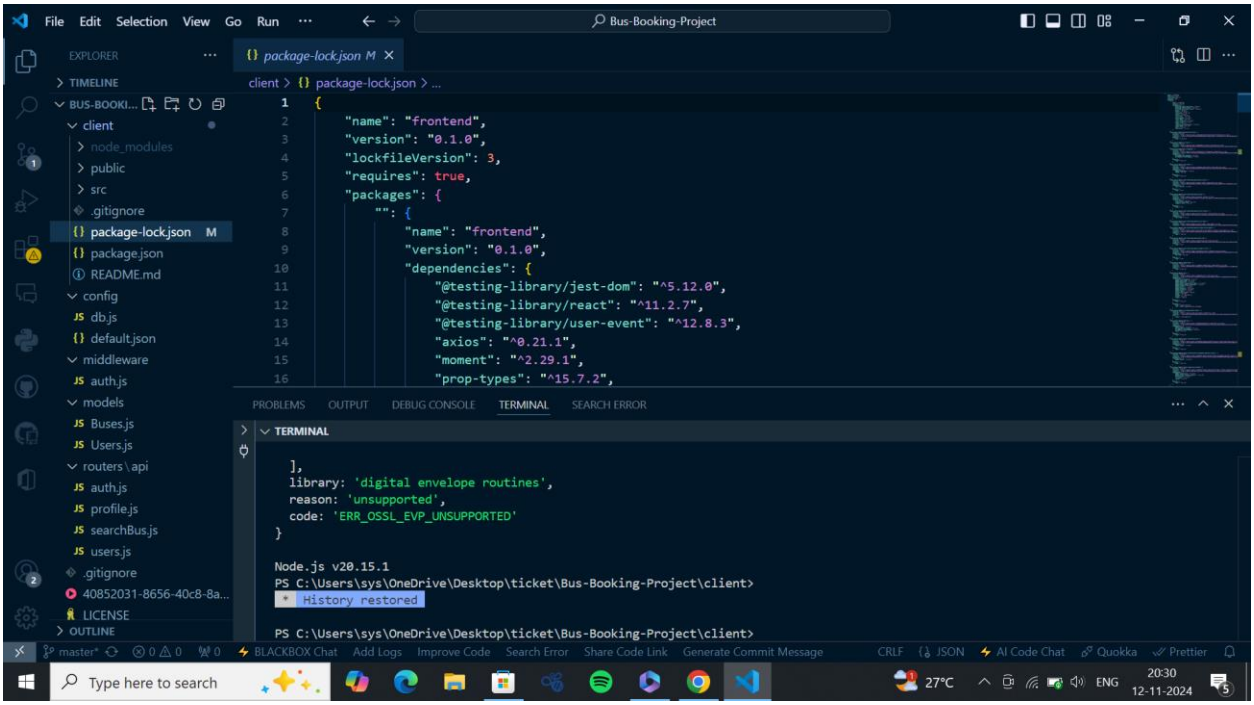
8.1. FRONT-END DEVELOPMENT:

For the front-end of the flight booking app, the following dependencies are used:

- **React**: A JavaScript library for building user interfaces, enabling the creation of reusable components and managing state in a smooth and efficient way.

- **Material UI**: A popular React UI framework that provides a comprehensive set of pre-built, customizable components for faster and consistent design. It helps in creating a visually appealing and responsive interface.
- **Bootstrap**: A front-end framework for designing responsive websites and web applications. It includes a variety of components like buttons, forms, and grids that help in quickly building a structured UI.
- **React-Bootstrap**: The React implementation of Bootstrap, providing Bootstrap components as React components for easier integration with React apps.
- **Axios**: A promise-based HTTP client for making requests to the backend server. It is used to send and receive data from the server (such as flight data, booking information, etc.).
- **Antd (Ant Design)**: A popular React UI library for building enterprise-level applications with ready-to-use components like tables, forms, date pickers, etc., which are highly customizable.
- **Mdb-react-ui-kit**: A lightweight library of responsive components built with React and Material Design principles, offering ready-to-use elements like buttons, cards, modals, etc.

These dependencies help build an interactive, modern, and responsive front-end for the flight booking app, ensuring smooth user experiences across various devices and browser



8.2. BACK-END DEVELOPMENT:

For the back-end of the flight booking app, the following steps and dependencies are used:

Setup Express Server:

- **Create `index.js` File in the Server (Backend Folder)**:
- This file will be the entry point for the server. It will handle setting up the Express server, listening to the appropriate port, and establishing connection to the database.
- **Define Port Number, MongoDB Connection String, and JWT Key in `.env` File**:
    - Port Number: Defines the port where the server will listen (usually `3000` or `5000`).
    - MongoDB Connection String: Stores the connection string for connecting to the MongoDB database securely.
    - JWT Key: Defines the secret key for creating and verifying JSON Web Tokens (JWT), used for secure user authentication.
- **Configure the Server**:

- Add **CORS** (Cross-Origin Resource Sharing): This allows requests from different origins to interact with the server, ensuring that the flight booking app works across various domains.
- Add **Body-Parser**: This middleware parses incoming request bodies in a middleware before the handlers, making it easier to extract data from `POST` requests.

Add Authentication:

- **Create Middleware Folder and `authMiddleware.js` File**:

This middleware will handle the authentication logic, ensuring that only authorized users (based on JWT) can access protected routes, like booking flights or viewing personal details.

The `authMiddleware.js` will:

- Verify the token in the request header.
- Decode and check the JWT token to validate user identity and roles (e.g., admin, user, etc.).

Required Back-End Dependencies:

- **CORS**: This package is used to enable Cross-Origin Resource Sharing, allowing your back-end API to be accessible from various front-end applications running on different domains or ports.
- **Bcryptjs**: A library for hashing passwords before storing them in the database. It ensures that user credentials are stored securely and can be verified during login.
- **Multer**: A middleware for handling `multipart/form-data`, which is used for uploading files. In the flight booking app, it might be used for uploading user profiles or images related to bookings.
- **Dotenv**: A zero-dependency module to load environment variables from a `.env` file into `process.env`. This is essential for storing sensitive information like database credentials, JWT secret, and other configuration settings securely.
- **Express**: The Node.js web application framework used to build the server, handle routes, and manage HTTP requests.
- **Mongoose**: An ODM (Object Data Modeling) library for MongoDB and Node.js. It simplifies database interactions by providing schema-based solutions for MongoDB collections, like users, flights, and bookings.

- **Nodemon**: A development tool that automatically restarts the server when file changes are detected. It makes development more efficient by eliminating the need to manually restart the server each time.
- **Jsonwebtoken (JWT)**: A library used to issue and verify JSON Web Tokens for user authentication. It helps secure routes and ensures that only authenticated users can access protected resources.

Steps for Back-End Setup:

1. **Set up the Express Server**: Create the `index.js` file, configure the middleware, and set up routes for handling flight bookings, user login, registration, and authentication.
2. **Set up Authentication Middleware**: Implement the `authMiddleware.js` to check if requests contain a valid JWT token and ensure authorized access to booking-related routes.

This back-end architecture, built with the mentioned dependencies and setup steps, ensures the flight booking app runs securely, efficiently, and offers smooth integration with the front-end application.

8.3. DATABASE DEVELOPMENT :

- Configure MongoDB
- Import Mongoose
- Add database connection from config.js file present in config folder.
- Create a model folder to store all the DB schemas.

9. PROJECT IMPLEMENTATION & EXECUTION :

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
```

```
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"                                                integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's         mobile         device         or         desktop.         See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js
"                                                               integrity="sha384-
```

```
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
  </body>
</html>
```

Admin.jsx

```
import React, { useEffect, useState } from 'react'
import '../styles/Admin.css'
import { useNavigate } from 'react-router-dom'
import axios from 'axios'

const Admin = () => {

  const navigate = useNavigate();
  const [users, setUsers] = useState([]);
  const [userCount, setUserCount] = useState(0);
  const [bookingCount, setbookingCount] = useState(0);
  const [flightsCount, setFlightsCount] = useState(0);



  useEffect(()=>{

    fetchData();
  }, [])

  const fetchData = async () =>{
    await axios.get('http://localhost:6001/fetch-users').then(
      (response)=>{
```

```
      setUserCount(response.data.length -1);
      setUsers(response.data.filter(user => user.approval === 'not-approved'));
    }
  );
  await axios.get('http://localhost:6001/fetch-bookings').then(
    (response)=>{
      setbookingCount(response.data.length);
    }
  );
  await axios.get('http://localhost:6001/fetch-flights').then(
    (response)=>{
      setFlightsCount(response.data.length);
    }
  );
}



const approveRequest = async (id) =>{
    try{

        await axios.post('http://localhost:6001/approve-operator', {id}).then(
          (response)=>{
            alert("Operator approved!!");
            fetchData();
          }
        )

    }catch(err){

    }
}

const rejectRequest = async (id) =>{
  try{

    await axios.post('http://localhost:6001/reject-operator', {id}).then(
      (response)=>{
        alert("Operator rejected!!");
        fetchData();
      }
```

```
      )

    }catch(err){

    }
  }

  return (
    <>

      <div className="admin-page">

        <div className="admin-page-cards">

            <div className="card admin-card users-card">
                <h4>Users</h4>
                <p> {userCount} </p>
                <button  className="btn btn-primary" onClick={()=>navigate('/all-
users')}>View all</button>
            </div>

            <div className="card admin-card transactions-card">
                <h4>Bookings</h4>
                <p> {bookingCount} </p>
                <button  className="btn btn-primary" onClick={()=>navigate('/all-
bookings')}>View all</button>
            </div>

            <div className="card admin-card deposits-card">
                <h4>Flights</h4>
                <p> {flightsCount} </p>
                <button  className="btn btn-primary" onClick={()=>navigate('/all-
flights')}>View all</button>
            </div>



        </div>

        <div className="admin-requests-container">

            <h3>New Operator Applications</h3>
```

```
            <div className="admin-requests">

              {
                users.length === 0 ?
                  <p>No new requests..</p>
                :
                  <>
                  {users.map((user)=>{
                    return(
                      <div className="admin-request" key={user._id}>
                        <span><b>Operator name: </b> {user.username}</span>
                        <span><b>Operator email: </b> {user.email}</span>
                        <div className="admin-request-actions">
                          <button   className='btn   btn-primary'   onClick={()=>
approveRequest(user._id)}>Approve</button>
                          <button   className='btn   btn-danger'   onClick={()=>
rejectRequest(user._id)}>Reject</button>
                        </div>
                      </div>
                    )
                  })}
                  </>

              }


            </div>

        </div>

    </div>

    </>
  )
}

export default Admin
```

AllBooking.jsx

```
import axios from 'axios';
import React, { useEffect, useState } from 'react'
```

```
const AllBookings = () => {

  const [bookings, setBookings] = useState([]);

  const userId = localStorage.getItem('userId');

  useEffect(()=>{
    fetchBookings();
  }, [])

  const fetchBookings = async () =>{
    await axios.get('http://localhost:6001/fetch-bookings').then(
      (response)=>{
        setBookings(response.data.reverse());
      }
    )
  }

  const cancelTicket = async (id) =>{
    await axios.put(`http://localhost:6001/cancel-ticket/${id}`).then(
      (response)=>{
        alert("Ticket cancelled!!");
        fetchBookings();
      }
    )
  }

  return (
    <div className="user-bookingsPage">
      <h1>Bookings</h1>

      <div className="user-bookings">

        {bookings.map((booking)=>{
          return(
            <div className="user-booking" key={booking._id}>
            <p><b>Booking ID:</b> {booking._id}</p>
            <span>
              <p><b>Mobile:</b> {booking.mobile}</p>
              <p><b>Email:</b> {booking.email}</p>
            </span>
```

```
            <span>
              <p><b>Flight Id:</b> {booking.flightId}</p>
              <p><b>Flight name:</b> {booking.flightName}</p>
            </span>
            <span>
              <p><b>On-boarding:</b> {booking.departure}</p>
              <p><b>Destination:</b> {booking.destination}</p>
            </span>
            <span>
              <div>
                <p><b>Passengers:</b></p>
                <ol>
                  {booking.passengers.map((passenger, i)=>{
                    return(
                      <li key={i}><p><b>Name:</b> {passenger.name},  <b>Age:</b>
{passenger.age}</p></li>
                    )
                  })}
                </ol>
              </div>
              {booking.bookingStatus   ===   'confirmed'   ?   <p><b>Seats:</b>
{booking.seats}</p> : ""}
            </span>
            <span>
              <p><b>Booking date:</b> {booking.bookingDate.slice(0,10)}</p>
              <p><b>Journey date:</b> {booking.journeyDate.slice(0,10)}</p>
            </span>
            <span>
              <p><b>Journey Time:</b> {booking.journeyTime}</p>
              <p><b>Total price:</b> {booking.totalPrice}</p>
            </span>
              {booking.bookingStatus === 'cancelled' ?
                <p      style={{color:      "red"}}><b>Booking      status:</b>
{booking.bookingStatus}</p>
                :
                <p><b>Booking status:</b> {booking.bookingStatus}</p>
              }
            {booking.bookingStatus === 'confirmed' ?
              <div>
                <button      className="btn      btn-danger"      onClick={()=>
cancelTicket(booking._id)}>Cancel Ticket</button>
              </div>

            :
```

```
            <></>}
          </div>
          )
        })}


      </div>
    </div>
  )
}

export default AllBookings
```

AllFlights.jsx

```
import axios from 'axios';
import React, { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom';
import '../styles/AllFlights.css';

const AllFlights = () => {
    const [flights, setFlights] = useState([]);
    const navigate = useNavigate();



    const fetchFlights = async () =>{
      await axios.get('http://localhost:6001/fetch-flights').then(
        (response)=>{
          setFlights(response.data);
          console.log(response.data)
        }
        )
      }

      useEffect(()=>{
        fetchFlights();
      }, [])

    return (
      <div className="allFlightsPage">
        <h1>All Flights</h1>
```

```
        <div className="allFlights">

          {flights.map((Flight)=>{
            return(

                <div className="allFlights-Flight" key={Flight._id}>
                  <p><b>_id:</b> {Flight._id}</p>
                  <span>
                    <p><b>Flight Id:</b> {Flight.flightId}</p>
                    <p><b>Flight name:</b> {Flight.flightName}</p>
                  </span>
                  <span>
                    <p><b>Starting station:</b> {Flight.origin}</p>
                    <p><b>Departure time:</b> {Flight.departureTime}</p>
                  </span>
                  <span>
                    <p><b>Destination:</b> {Flight.destination}</p>
                    <p><b>Arrival time:</b> {Flight.arrivalTime}</p>
                  </span>
                  <span>
                    <p><b>Base price:</b> {Flight.basePrice}</p>
                    <p><b>Total seats:</b> {Flight.totalSeats}</p>
                  </span>
                </div>
            )
          })}









        </div>
      </div>
    )
  }

export default AllFlights
```

AllUsers.jsx

```
import React, { useEffect, useState } from 'react'
```

```jsx
import Navbar from '../components/Navbar'
import '../styles/allUsers.css'
import axios from 'axios';

const AllUsers = () => {

  const [users, setUsers] = useState([]);

  useEffect(()=>{
    fetchUsers();
  },[]);

  const fetchUsers = async () =>{
    await axios.get('http://localhost:6001/fetch-users').then(
      (response) =>{
        setUsers(response.data);
      }
    )
  }

  return (
    <>
      <Navbar />

      <div class="all-users-page">
        <h2>All Users</h2>
        <div class="all-users">

        {users.filter(user=> user.usertype === 'customer').map((user)=>{
            return(

              <div class="user" key={user._id}>
                  <p><b>UserId </b>{user._id}</p>
                  <p><b>Username </b>{user.username}</p>
                  <p><b>Email </b>{user.email}</p>
              </div>
            )
          })}

        </div>
```

```
        <h2>Flight Operators</h2>
        <div class="all-users">

        {users.filter(user=> user.usertype === 'flight-operator').map((user)=>{
            return(

              <div class="user" key={user._id}>
                  <p><b>Id </b>{user._id}</p>
                  <p><b>Flight Name </b>{user.username}</p>
                  <p><b>Email </b>{user.email}</p>
              </div>
            )
          })}

        </div>

    </div>
    </>
  )
}

export default AllUsers
```

Authenticate.jsx

```
import React, { useState } from 'react';
import '../styles/Authenticate.css'
import Login from '../components/Login';
import Register from '../components/Register';

const Authenticate = () => {

  const [isLogin, setIsLogin] = useState(true);

  return (
    <div className="AuthenticatePage">

      {isLogin ?

      <Login  setIsLogin = {setIsLogin} />
```

```
      :

      <Register setIsLogin = {setIsLogin} />
      }

    </div>
  )
}

export default Authenticate
```

BookFlight.jsx

```
import React, { useContext, useEffect, useState } from 'react'
import '../styles/BookFlight.css'
import { GeneralContext } from '../context/GeneralContext';
import axios from 'axios';
import { useParams, useNavigate } from 'react-router-dom';

const BookFlight = () => {
    const {id} = useParams();

    const [flightName, setFlightName] = useState('');
    const [flightId, setFlightId] = useState('');
    const [basePrice, setBasePrice] = useState(0);
    const [StartCity, setStartCity] = useState('');
    const [destinationCity, setDestinationCity] = useState('');
    const [startTime, setStartTime] = useState();



    useEffect(()=>{
      fetchFlightData();
    }, [])

    const fetchFlightData = async () =>{
      await axios.get(`http://localhost:6001/fetch-flight/${id}`).then(
        (response) =>{
          setFlightName(response.data.flightName);
          setFlightId(response.data.flightId);
          setBasePrice(response.data.basePrice);
```

```
          setStartCity(response.data.origin);
          setDestinationCity(response.data.destination);
          setStartTime(response.data.departureTime);
        }
      )
    }



    const [email, setEmail] = useState('');
    const [mobile, setMobile] = useState('');
    const [coachType, setCoachType] = useState('');
    const {ticketBookingDate} = useContext(GeneralContext);
    const [journeyDate, setJourneyDate] = useState(ticketBookingDate);

    const [numberOfPassengers, setNumberOfPassengers] = useState(0);
    const [passengerDetails, setPassengerDetails] = useState([]);

    const [totalPrice, setTotalPrice] = useState(0);
    const price = {'economy': 1, 'premium-economy': 2, 'business': 3, 'first-
class': 4}



    const handlePassengerChange = (event) => {
      const value = parseInt(event.target.value);
      setNumberOfPassengers(value);
      };

    const handlePassengerDetailsChange = (index, key, value) => {
      setPassengerDetails((prevDetails) => {
        const updatedDetails = [...prevDetails];
        updatedDetails[index] = { ...updatedDetails[index], [key]: value };
        return updatedDetails;
      });

    };

    useEffect(()=>{
      if(price[coachType] * basePrice * numberOfPassengers){
        setTotalPrice(price[coachType] * basePrice * numberOfPassengers);
      }
    },[numberOfPassengers, coachType])
```

```
    const navigate = useNavigate();


    const bookFlight = async ()=>{

      const inputs = {user: localStorage.getItem('userId'), flight: id, flightName,
                                                  flightId, departure: StartCity,
journeyTime: startTime, destination: destinationCity,
                                                  email,    mobile,    passengers:
passengerDetails, totalPrice,
                                                  journeyDate,         seatClass:
coachType}

      await axios.post('http://localhost:6001/book-ticket', inputs).then(
        (response)=>{
          alert("booking successful");
          navigate('/bookings');
        }
      ).catch((err)=>{
        alert("Booking failed!!")
      })
    }







    return (
      <div className='BookFlightPage'>

        <div className="BookingFlightPageContainer">
          <h2>Book ticket</h2>
        <span>
          <p><b>Flight Name: </b> {flightName}</p>
          <p><b>Flight No: </b> {flightId}</p>
        </span>
        <span>
          <p><b>Base price: </b> {basePrice}</p>
        </span>

        <span>
```

```
          <div className="form-floating mb-3">
                  <input          type="email"          className="form-control"
id="floatingInputemail" value={email} onChange={(e)=> setEmail(e.target.value)} />
                  <label htmlFor="floatingInputemail">Email</label>
            </div>
            <div className="form-floating mb-3">
                  <input          type="text"          className="form-control"
id="floatingInputmobile"             value={mobile}             onChange={(e)=>
setMobile(e.target.value)} />
                  <label htmlFor="floatingInputmobile">Mobile</label>
            </div>
        </span>
        <span className='span3'>
          <div className="no-of-passengers">
            <div className="form-floating mb-3">
                  <input         type="number"          className="form-control"
id="floatingInputreturnDate"                          value={numberOfPassengers}
onChange={handlePassengerChange} />
                  <label         htmlFor="floatingInputreturnDate">No         of
passengers</label>
            </div>
          </div>
          <div className="form-floating mb-3">
                  <input          type="date"          className="form-control"
id="floatingInputreturnDate"                               value={journeyDate}
onChange={(e)=>setJourneyDate(e.target.value)} />
                  <label htmlFor="floatingInputreturnDate">Journey date</label>
          </div>
          <div className="form-floating">
                        <select   className="form-select   form-select-sm   mb-3"
defaultValue=""    aria-label=".form-select-sm    example"    value={coachType}
onChange={(e) => setCoachType(e.target.value) }>
                          <option value="" disabled>Select</option>
                          <option value="economy">Economy class</option>
                          <option                value="premium-economy">Premium
Economy</option>
                          <option value="business">Business class</option>
                          <option value="first-class">First class</option>
                        </select>
                        <label htmlFor="floatingSelect">Seat Class</label>
                      </div>


        </span>
```

```
        <div className="new-passengers">
            {Array.from({ length: numberOfPassengers }).map((_, index) => (
              <div className='new-passenger' key={index}>
                <h4>Passenger {index + 1}</h4>
                <div className="new-passenger-inputs">

                    <div className="form-floating mb-3">
                      <input         type="text"         className="form-control"
id="floatingInputpassengerName"    value={passengerDetails[index]?.name    ||    ''}
onChange={(event)       =>       handlePassengerDetailsChange(index,       'name',
event.target.value) } />
                      <label htmlFor="floatingInputpassengerName">Name</label>
                    </div>
                    <div className="form-floating mb-3">
                          <input      type="number"      className="form-control"
id="floatingInputpassengerAge"    value={passengerDetails[index]?.age    ||    ''}
onChange={(event)       =>       handlePassengerDetailsChange(index,       'age',
event.target.value) } />
                          <label htmlFor="floatingInputpassengerAge">Age</label>
                    </div>



                </div>
              </div>
            ))}

        </div>

        <h6><b>Total price</b>: {totalPrice}</h6>
        <button className='btn btn-primary' onClick={bookFlight}>Book now</button>
      </div>
      </div>
    )
  }
export default BookFlight
```

Bookings.jsx

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

```
const Bookings = () => {



  const [bookings, setBookings] = useState([]);

  const userId = localStorage.getItem('userId');

  useEffect(()=>{
    fetchBookings();
  }, [])

  const fetchBookings = async () =>{
    await axios.get('http://localhost:6001/fetch-bookings').then(
      (response)=>{
        setBookings(response.data.reverse());
      }
    )
  }
  const cancelTicket = async (id) =>{
    await axios.put(`http://localhost:6001/cancel-ticket/${id}`).then(
      (response)=>{
        alert("Ticket cancelled!!");
        fetchBookings();
      }
    )
  }

  return (
    <div className="user-bookingsPage">
      <h1>Bookings</h1>

      <div className="user-bookings">

        {bookings.filter(booking=> booking.user === userId).map((booking)=>{
          return(
            <div className="user-booking" key={booking._id}>
            <p><b>Booking ID:</b> {booking._id}</p>
            <span>
              <p><b>Mobile:</b> {booking.mobile}</p>
              <p><b>Email:</b> {booking.email}</p>
            </span>
            <span>
              <p><b>Flight Id:</b> {booking.flightId}</p>
```

```
            <p><b>Flight name:</b> {booking.flightName}</p>
          </span>
          <span>
            <p><b>On-boarding:</b> {booking.departure}</p>
            <p><b>Destination:</b> {booking.destination}</p>
          </span>
          <span>

            <div>
              <p><b>Passengers:</b></p>
              <ol>
                {booking.passengers.map((passenger, i)=>{
                  return(
                    <li key={i}><p><b>Name:</b> {passenger.name},  <b>Age:</b>
{passenger.age}</p></li>
                  )
                })}
              </ol>
            </div>
            {booking.bookingStatus   ===   'confirmed'   ?   <p><b>Seats:</b>
{booking.seats}</p> : ""}

          </span>
          <span>
            <p><b>Booking date:</b> {booking.bookingDate.slice(0,10)}</p>
            <p><b>Journey date:</b> {booking.journeyDate.slice(0,10)}</p>
          </span>
          <span>
            <p><b>Journey Time:</b> {booking.journeyTime}</p>
            <p><b>Total price:</b> {booking.totalPrice}</p>
          </span>
            {booking.bookingStatus === 'cancelled' ?
              <p      style={{color:      "red"}}><b>Booking      status:</b>
{booking.bookingStatus}</p>
              :
              <p><b>Booking status:</b> {booking.bookingStatus}</p>
            }
          {booking.bookingStatus === 'confirmed' ?
            <div>
              <button      className="btn      btn-danger"      onClick={()=>
cancelTicket(booking._id)}>Cancel Ticket</button>
            </div>
```

```
            :
            <></>}
          </div>
          )
        })}


      </div>
    </div>
  )
}

export default Bookings
```

EditFlight.jsx

```
import React, { useEffect, useState } from 'react'
import '../styles/NewFlight.css'
import axios from 'axios';
import { useParams } from 'react-router-dom';

const EditFlight = () => {
    const [flightName, setFlightName] = useState('');
    const [flightId, setFlightId] = useState('');
    const [origin, setOrigin] = useState('');
    const [destination, setDestination] = useState('');
    const [startTime, setStartTime] = useState();
    const [arrivalTime, setArrivalTime] = useState();
    const [totalSeats, setTotalSeats] = useState(0);
    const [basePrice, setBasePrice] = useState(0);




    const {id} = useParams();

    useEffect(()=>{
      console.log(startTime);
    }, [startTime])

    useEffect(()=>{
```

```
    fetchFlightData();
  }, [])

  const fetchFlightData = async () =>{
    await axios.get(`http://localhost:6001/fetch-flight/${id}`).then(
      (response) =>{
        console.log(response.data);
        setFlightName(response.data.flightName);
        setFlightId(response.data.flightId);
        setOrigin(response.data.origin);
        setDestination(response.data.destination);
        setTotalSeats(response.data.totalSeats);
        setBasePrice(response.data.basePrice);

        const timeParts1 = response.data.departureTime.split(":");
        const startT = new Date();
        startT.setHours(parseInt(timeParts1[0], 10));
        startT.setMinutes(parseInt(timeParts1[1], 10));
        const hours1 = String(startT.getHours()).padStart(2, '0');
        const minutes1 = String(startT.getMinutes()).padStart(2, '0');

        setStartTime(`${hours1}:${minutes1}`);

        const timeParts2 = response.data.arrivalTime.split(":");
        const startD = new Date();
        startD.setHours(parseInt(timeParts2[0], 10));
        startD.setMinutes(parseInt(timeParts2[1], 10));
        const hours2 = String(startD.getHours()).padStart(2, '0');
        const minutes2 = String(startD.getMinutes()).padStart(2, '0');

        setArrivalTime(`${hours2}:${minutes2}`);

      }
    )
  }

  const handleSubmit = async () =>{

    const inputs = {_id: id,flightName, flightId, origin, destination,
      departureTime: startTime, arrivalTime, basePrice, totalSeats};

    await axios.put('http://localhost:6001/update-flight', inputs).then(
```

```
        async (response)=>{
          alert('Flight updated successfully!!');
          setFlightName('');
          setFlightId('');
          setOrigin('');
          setStartTime('');
          setArrivalTime('');
          setDestination('');
          setBasePrice(0);
          setTotalSeats(0);
        }
      )

    }

    return (
      <div className='NewFlightPage'>

        <div className="NewFlightPageContainer">

          <h2>Edit Flight</h2>

        <span className='newFlightSpan1'>
          <div className="form-floating mb-3">
                  <input           type="text"           className="form-control"
id="floatingInputemail"            value={flightName}            onChange={(e)=>
setFlightName(e.target.value)} disabled />
                  <label htmlFor="floatingInputemail">Flight Name</label>
            </div>
            <div className="form-floating mb-3">
                  <input           type="text"           className="form-control"
id="floatingInputmobile"             value={flightId}            onChange={(e)=>
setFlightId(e.target.value)} />
                  <label htmlFor="floatingInputmobile">Flight Id</label>
            </div>
        </span>
        <span>
        <div className="form-floating">
            <select className="form-select form-select-sm mb-3" aria-label=".form-
select-sm example" value={origin} onChange={(e)=> setOrigin(e.target.value)} >
              <option value="" selected disabled>Select</option>
              <option value="Chennai">Chennai</option>
              <option value="Banglore">Banglore</option>
```

```
              <option value="Hyderabad">Hyderabad</option>
              <option value="Mumbai">Mumbai</option>
              <option value="Indore">Indore</option>
              <option value="Delhi">Delhi</option>
              <option value="Pune">Pune</option>
              <option value="Trivendrum">Trivendrum</option>
              <option value="Bhopal">Bhopal</option>
              <option value="Kolkata">Kolkata</option>
              <option value="varanasi">varanasi</option>
              <option value="Jaipur">Jaipur</option>
            </select>
            <label htmlFor="floatingSelect">Departure City</label>
          </div>
            <div className="form-floating mb-3">
                  <input           type="time"           className="form-control"
id="floatingInputmobile"            value={startTime}            onChange={(e)=>
setStartTime(e.target.value)} />
                  <label htmlFor="floatingInputmobile">Departure Time</label>
            </div>
        </span>
        <span>
            <div className="form-floating">
              <select    className="form-select    form-select-sm    mb-3"    aria-
label=".form-select-sm      example"      value={destination}      onChange={(e)=>
setDestination(e.target.value)} >
                <option value="" selected disabled>Select</option>
                <option value="Chennai">Chennai</option>
                <option value="Banglore">Banglore</option>
                <option value="Hyderabad">Hyderabad</option>
                <option value="Mumbai">Mumbai</option>
                <option value="Indore">Indore</option>
                <option value="Delhi">Delhi</option>
                <option value="Pune">Pune</option>
                <option value="Trivendrum">Trivendrum</option>
                <option value="Bhopal">Bhopal</option>
                <option value="Kolkata">Kolkata</option>
                <option value="varanasi">varanasi</option>
                <option value="Jaipur">Jaipur</option>
              </select>
              <label htmlFor="floatingSelect">Destination City</label>
            </div>
            <div className="form-floating mb-3">
```

```
                <input          type="time"          className="form-control"
id="floatingInputArrivalTime"         value={arrivalTime}         onChange={(e)=>
setArrivalTime(e.target.value)} />
                <label htmlFor="floatingInputArrivalTime">Arrival time</label>
            </div>
        </span>
        <span className='newFlightSpan2'>
          <div className="form-floating mb-3">
                <input          type="number"          className="form-control"
id="floatingInpuSeats"             value={totalSeats}             onChange={(e)=>
setTotalSeats(e.target.value)} />
                <label htmlhtmlFor="floatingInpuSeats">Total seats</label>
            </div>
            <div className="form-floating mb-3">
                <input          type="number"          className="form-control"
id="floatingInputBasePrice"           value={basePrice}           onChange={(e)=>
setBasePrice(e.target.value)} />
                <label htmlhtmlFor="floatingInputBasePrice">Base price</label>
            </div>
        </span>

        <button                    className='btn                    btn-primary'
onClick={handleSubmit}>Update</button>
      </div>
      </div>
    )
  }

export default EditFlight
```

FlightAdmin.jsx

```
import React, { useEffect, useState } from 'react'
import axios from 'axios'
import '../styles/FlightAdmin.css'
import { useNavigate } from 'react-router-dom';

const FlightAdmin = () => {

  const navigate = useNavigate();
```

```
  const [userDetails, setUserDetails] = useState();
  const [bookingCount, setbookingCount] = useState(0);
  const [flightsCount, setFlightsCount] = useState(0);

  useEffect(()=>{
    fetchUserData();
  }, [])

  const fetchUserData = async () =>{
    try{
      const id = localStorage.getItem('userId');
      await axios.get(`http://localhost:6001/fetch-user/${id}`).then(
        (response)=>{
          setUserDetails(response.data);
          console.log(response.data);
        }
      )

    }catch(err){

    }
  }



  useEffect(()=>{

    fetchData();
  }, [])

  const fetchData = async () =>{
    await axios.get('http://localhost:6001/fetch-bookings').then(
      (response)=>{
        setbookingCount(response.data.filter(booking  =>  booking.flightName  ===
localStorage.getItem('username')).length);
      }
    );
    await axios.get('http://localhost:6001/fetch-flights').then(
      (response)=>{
        setFlightsCount(response.data.filter(booking  =>  booking.flightName  ===
localStorage.getItem('username')).length);
      }
```

```
    );
  }

  return (
    <div className="flightAdmin-page">

      {userDetails ?
        <>
          {userDetails.approval === 'not-approved' ?
            <div className="notApproved-box">
              <h3>Approval Required!!</h3>
              <p>Your application is under processing. It needs an approval from
the administrator. Kindly please be patience!!</p>
            </div>



          : userDetails.approval === 'rejected' ?
            <div className="notApproved-box">
              <h3>Application Rejected!!</h3>
              <p>We are sorry to inform you that your application has been
rejected!!</p>
          </div>
          : userDetails.approval === 'approved' ?



          <div className="admin-page-cards">

          <div className="card admin-card transactions-card">
              <h4>Bookings</h4>
              <p> {bookingCount} </p>
              <button className="btn btn-primary" onClick={()=>navigate('/flight-
bookings')}>View all</button>
          </div>

          <div className="card admin-card deposits-card">
              <h4>Flights</h4>
              <p> {flightsCount} </p>
              <button                 className="btn                 btn-primary"
onClick={()=>navigate('/flights')}>View all</button>
          </div>

          <div className="card admin-card loans-card">
```

```
              <h4>New Flight</h4>
              <p> (new route) </p>
              <button  className="btn  btn-primary"  onClick={()=>navigate('/new-
flight')}>Add now</button>
          </div>

      </div>

          :
            ""
          }
        </>
      :
       ""
      }

    </div>
  )
}

export default FlightAdmin
```

FlightBookings.jsx

```
import axios from 'axios';
import React, { useEffect, useState } from 'react'

const FlightBookings = () => {
  const [userDetails, setUserDetails] = useState();

  useEffect(()=>{
    fetchUserData();
  }, [])

  const fetchUserData = async () =>{
    try{
      const id = localStorage.getItem('userId');
      await axios.get(`http://localhost:6001/fetch-user/${id}`).then(
        (response)=>{
          setUserDetails(response.data);
          console.log(response.data);
        }
```

```
      )

    }catch(err){

    }
  }


  const [bookings, setBookings] = useState([]);


  useEffect(()=>{
    fetchBookings();
  }, [])

  const fetchBookings = async () =>{
    await axios.get('http://localhost:6001/fetch-bookings').then(
      (response)=>{
        setBookings(response.data.reverse());
      }
    )
  }

  const cancelTicket = async (id) =>{
    await axios.put(`http://localhost:6001/cancel-ticket/${id}`).then(
      (response)=>{
        alert("Ticket cancelled!!");
        fetchBookings();
      }
    )
  }

  return (
    <div className="user-bookingsPage">

      {userDetails ?
        <>
          {userDetails.approval === 'not-approved' ?
            <div className="notApproved-box">
              <h3>Approval Required!!</h3>
              <p>Your application is under processing. It needs an approval from
the administrator. Kindly please be patience!!</p>
```

```
            </div>

          : userDetails.approval === 'approved' ?
            <>
              <h1>Bookings</h1>

              <div className="user-bookings">

                {bookings.filter(booking=>         booking.flightName         ===
localStorage.getItem('username')).map((booking)=>{
                  return(
                    <div className="user-booking" key={booking._id}>
                    <p><b>Booking ID:</b> {booking._id}</p>
                    <span>
                      <p><b>Mobile:</b> {booking.mobile}</p>
                      <p><b>Email:</b> {booking.email}</p>
                    </span>
                    <span>
                      <p><b>Flight Id:</b> {booking.flightId}</p>
                      <p><b>Flight name:</b> {booking.flightName}</p>
                    </span>
                    <span>
                      <p><b>On-boarding:</b> {booking.departure}</p>
                      <p><b>Destination:</b> {booking.destination}</p>
                    </span>
                    <span>
                      <div>
                        <p><b>Passengers:</b></p>
                        <ol>
                          {booking.passengers.map((passenger, i)=>{
                            return(
                              <li    key={i}><p><b>Name:</b>    {passenger.name},
<b>Age:</b> {passenger.age}</p></li>
                            )
                          })}
                        </ol>
                      </div>
                      {booking.bookingStatus  ===  'confirmed'  ?  <p><b>Seats:</b>
{booking.seats}</p> : ""}
                    </span>
                    <span>
                      <p><b>Booking                                        date:</b>
{booking.bookingDate.slice(0,10)}</p>
```

```
                      <p><b>Journey                                    date:</b>
{booking.journeyDate.slice(0,10)}</p>
                    </span>
                    <span>
                      <p><b>Journey Time:</b> {booking.journeyTime}</p>
                      <p><b>Total price:</b> {booking.totalPrice}</p>
                    </span>
                      {booking.bookingStatus === 'cancelled' ?
                        <p    style={{color:    "red"}}><b>Booking    status:</b>
{booking.bookingStatus}</p>
                        :
                        <p><b>Booking status:</b> {booking.bookingStatus}</p>
                      }
                    {booking.bookingStatus === 'confirmed' ?
                      <div>
                        <button    className="btn    btn-danger"    onClick={()=>
cancelTicket(booking._id)}>Cancel Ticket</button>
                      </div>

                    :
                    <></>}
                  </div>
                  )
                })}

              </div>
            </>
          :
            ""
          }
        </>
      :
       ""
      }

    </div>
  )

}

export default FlightBookings
```

FlightRequest.jsx

```jsx
import React from 'react'

const FlightRequests = () => {
  return (
    <div>FlightRequests</div>
  )
}

export default FlightRequests
```

Flights.jsx

```jsx
import axios from 'axios';
import React, { useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom';

const Flights = () => {
  const [userDetails, setUserDetails] = useState();

  useEffect(()=>{
    fetchUserData();
  }, [])

  const fetchUserData = async () =>{
    try{
      const id = localStorage.getItem('userId');
      await axios.get(`http://localhost:6001/fetch-user/${id}`).then(
        (response)=>{
          setUserDetails(response.data);
          console.log(response.data);
        }
      )
```

```
    }catch(err){

    }
  }

  const [flights, setFlights] = useState([]);
  const navigate = useNavigate();


  const fetchFlights = async () =>{
    await axios.get('http://localhost:6001/fetch-flights').then(
      (response)=>{
        setFlights(response.data);
        console.log(response.data)
      }
      )
    }

    useEffect(()=>{
      fetchFlights();
    }, [])

  return (
    <div className="allFlightsPage">

      {userDetails ?
        <>
          {userDetails.approval === 'not-approved' ?
            <div className="notApproved-box">
              <h3>Approval Required!!</h3>
              <p>Your application is under processing. It needs an approval from
the administrator. Kindly please be patience!!</p>
            </div>



          : userDetails.approval === 'approved' ?
            <>
              <h1>All Flights</h1>

              <div className="allFlights">
```

```
                {flights.filter(flight=>          flight.flightName          ===
localStorage.getItem('username')).map((Flight)=>{
                  return(

                      <div className="allFlights-Flight" key={Flight._id}>
                        <p><b>_id:</b> {Flight._id}</p>
                        <span>
                          <p><b>Flight Id:</b> {Flight.flightId}</p>
                          <p><b>Flight name:</b> {Flight.flightName}</p>
                        </span>
                        <span>
                          <p><b>Starting station:</b> {Flight.origin}</p>
                          <p><b>Departure time:</b> {Flight.departureTime}</p>
                        </span>
                        <span>
                          <p><b>Destination:</b> {Flight.destination}</p>
                          <p><b>Arrival time:</b> {Flight.arrivalTime}</p>
                        </span>
                        <span>
                          <p><b>Base price:</b> {Flight.basePrice}</p>
                          <p><b>Total seats:</b> {Flight.totalSeats}</p>
                        </span>
                        <div>
                          <button   className="btn   btn-primary"   onClick={()=>
navigate(`/edit-flight/${Flight._id}`)}>Edit details</button>
                        </div>
                      </div>
                  )
                })}
              </div>
            </>
          :
            ""
          }
        </>
      :
       ""
      }

    </div>
  )
}
```

```
export default Flights
```

LandingPage.jsx

```
import React, { useContext, useEffect, useState } from 'react'
import '../styles/LandingPage.css'
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import { GeneralContext } from '../context/GeneralContext';

const LandingPage = () => {

  const [error, setError] = useState('');
  const [checkBox, setCheckBox] = useState(false);



  const [departure, setDeparture] = useState('');
  const [destination, setDestination] = useState('');
  const [departureDate, setDepartureDate] = useState();
  const [returnDate, setReturnDate] = useState();




  const navigate = useNavigate();
  useEffect(()=>{

    if(localStorage.getItem('userType') === 'admin'){
      navigate('/admin');
    } else if(localStorage.getItem('userType') === 'flight-operator'){
      navigate('/flight-admin');
    }
  }, []);

  const [Flights, setFlights] = useState([]);

  const fetchFlights = async () =>{

    if(checkBox){
      if(departure !== "" && destination !== "" && departureDate && returnDate){
```

```
      const date = new Date();
      const date1 = new Date(departureDate);
      const date2 = new Date(returnDate);
      if(date1 > date && date2 > date1){
        setError("");
        await axios.get('http://localhost:6001/fetch-flights').then(
            (response)=>{
              setFlights(response.data);
              console.log(response.data)
            }
         )
      } else{ setError("Please check the dates"); }
    } else{ setError("Please fill all the inputs"); }
  }else{
    if(departure !== "" && destination !== "" && departureDate){
      const date = new Date();
      const date1 = new Date(departureDate);
      if(date1 >= date){
        setError("");
        await axios.get('http://localhost:6001/fetch-flights').then(
            (response)=>{
              setFlights(response.data);
              console.log(response.data)
            }
         )
      } else{ setError("Please check the dates"); }
    } else{ setError("Please fill all the inputs"); }
  }
  }
  const {setTicketBookingDate} = useContext(GeneralContext);
  const userId = localStorage.getItem('userId');


  const handleTicketBooking = async (id, origin, destination) =>{
    if(userId){

      if(origin === departure){
        setTicketBookingDate(departureDate);
        navigate(`/book-flight/${id}`);
      } else if(destination === departure){
        setTicketBookingDate(returnDate);
        navigate(`/book-flight/${id}`);
      }
```

```
      }else{
        navigate('/auth');
      }
    }



  return (
    <div className="landingPage">
        <div className="landingHero">


          <div className="landingHero-title">
            <h1 className="banner-h1">Embark on an Extraordinary Flight Booking Adventure!</h1>
            <p  className="banner-p">Unleash  your  travel  desires  and  book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.</p>
          </div>



          <div className="Flight-search-container input-container mb-4">

                {/* <h3>Journey details</h3> */}
                <div className="form-check form-switch">
                  <input      className="form-check-input"      type="checkbox" id="flexSwitchCheckDefault"                                      value="" onChange={(e)=>setCheckBox(e.target.checked)} />
                  <label                          className="form-check-label" htmlFor="flexSwitchCheckDefault">Return journey</label>
                </div>
                <div className='Flight-search-container-body'>

                  <div className="form-floating">
                    <select className="form-select form-select-sm mb-3"  aria-label=".form-select-sm                example"                value={departure} onChange={(e)=>setDeparture(e.target.value)}>
                      <option value="" selected disabled>Select</option>
                      <option value="Chennai">Chennai</option>
                      <option value="Banglore">Banglore</option>
                      <option value="Hyderabad">Hyderabad</option>
```

```
                        <option value="Mumbai">Mumbai</option>
                        <option value="Indore">Indore</option>
                        <option value="Delhi">Delhi</option>
                        <option value="Pune">Pune</option>
                        <option value="Trivendrum">Trivendrum</option>
                        <option value="Bhopal">Bhopal</option>
                        <option value="Kolkata">Kolkata</option>
                        <option value="varanasi">varanasi</option>
                        <option value="Jaipur">Jaipur</option>
                      </select>
                      <label htmlFor="floatingSelect">Departure City</label>
                    </div>
                    <div className="form-floating">
                      <select className="form-select form-select-sm mb-3"  aria-
label=".form-select-sm               example"               value={destination}
onChange={(e)=>setDestination(e.target.value)}>
                        <option value="" selected disabled>Select</option>
                        <option value="Chennai">Chennai</option>
                        <option value="Banglore">Banglore</option>
                        <option value="Hyderabad">Hyderabad</option>
                        <option value="Mumbai">Mumbai</option>
                        <option value="Indore">Indore</option>
                        <option value="Delhi">Delhi</option>
                        <option value="Pune">Pune</option>
                        <option value="Trivendrum">Trivendrum</option>
                        <option value="Bhopal">Bhopal</option>
                        <option value="Kolkata">Kolkata</option>
                        <option value="varanasi">varanasi</option>
                        <option value="Jaipur">Jaipur</option>
                      </select>
                      <label htmlFor="floatingSelect">Destination City</label>
                    </div>
                    <div className="form-floating mb-3">
                      <input         type="date"         className="form-control"
id="floatingInputstartDate"                                   value={departureDate}
onChange={(e)=>setDepartureDate(e.target.value)}/>
                      <label             htmlFor="floatingInputstartDate">Journey
date</label>
                    </div>
                    {checkBox ?

                      <div className="form-floating mb-3">
```

```
                        <input        type="date"        className="form-control"
id="floatingInputreturnDate"                                    value={returnDate}
onChange={(e)=>setReturnDate(e.target.value)}/>
                        <label           htmlFor="floatingInputreturnDate">Return
date</label>
                      </div>

                    :

                    ""}
                    <div>
                      <button            className="btn            btn-primary"
onClick={fetchFlights}>Search</button>
                    </div>

                  </div>
                  <p>{error}</p>
              </div>

                {Flights.length > 0
                ?
                <>
                {
                  Flights.filter(Flight   =>   Flight.origin   ===   departure   &&
Flight.destination === destination).length > 0 ?
                  <>
                  <div className="availableFlightsContainer">
                    <h1>Available Flights</h1>

                    <div className="Flights">

                      {checkBox ?

                      <>
                        {Flights.filter(Flight => (Flight.origin === departure &&
Flight.destination  ===  destination  )  ||  (Flight.origin  ===  destination  &&
Flight.destination === departure)).map((Flight)=>{
                        return(

                        <div className="Flight" key={Flight._id}>
                            <div>
                                <p> <b>{Flight.flightName}</b></p>
                                <p ><b>Flight Number:</b> {Flight.flightId}</p>
```

```
                            </div>
                            <div>
                                <p ><b>Start :</b> {Flight.origin}</p>
                                <p            ><b>Departure            Time:</b>
{Flight.departureTime}</p>
                            </div>
                            <div>
                                <p ><b>Destination :</b> {Flight.destination}</p>
                                <p ><b>Arrival Time:</b> {Flight.arrivalTime}</p>
                            </div>
                            <div>
                                <p ><b>Starting Price:</b> {Flight.basePrice}</p>
                                <p            ><b>Available            Seats:</b>
{Flight.totalSeats}</p>
                            </div>
                            <button     className="button     btn     btn-primary"
onClick={()=>handleTicketBooking(Flight._id,                    Flight.origin,
Flight.destination)}>Book Now</button>
                        </div>
                        )
                    })}
                    </>
                    :
                    <>
                    {Flights.filter(Flight  =>  Flight.origin  ===  departure  &&
Flight.destination === destination).map((Flight)=>{
                        return(

                        <div className="Flight">
                            <div>
                                <p> <b>{Flight.flightName}</b></p>
                                <p ><b>Flight Number:</b> {Flight.flightId}</p>
                            </div>
                            <div>
                                <p ><b>Start :</b> {Flight.origin}</p>
                                <p            ><b>Departure            Time:</b>
{Flight.departureTime}</p>
                            </div>
                            <div>
                                <p ><b>Destination :</b> {Flight.destination}</p>
                                <p ><b>Arrival Time:</b> {Flight.arrivalTime}</p>
                            </div>
                            <div>
                                <p ><b>Starting Price:</b> {Flight.basePrice}</p>
```

```
                                <p            ><b>Available            Seats:</b>
{Flight.totalSeats}</p>
                            </div>
                            <button     className="button     btn     btn-primary"
onClick={()=>handleTicketBooking(Flight._id,                      Flight.origin,
Flight.destination)}>Book Now</button>
                        </div>
                        )
                    })}
                    </>}




                  </div>
                </div>
                </>
                :
                <>
                 <div className="availableFlightsContainer">
                  <h1> No Flights</h1>
                  </div>
                </>
              }
              </>
              :
              <></>
              }









        </div>
        <section id="about" className="section-about  p-4">
        <div className="container">
            <h2 className="section-title">About Us</h2>
            <p className="section-description">
```

```
                       Welcome to our Flight ticket booking app, where we are dedicated to providing you with an exceptional travel experience from start to finish. Whether you're embarking on a daily commute, planning an exciting cross-country adventure, or seeking a leisurely scenic route, our app offers an extensive selection of Flight options to cater to your unique travel preferences.
            </p>
            <p className="section-description">
                       We understand the importance of convenience and efficiency in your travel plans. Our user-friendly interface allows you to effortlessly browse through a wide range of Flight schedules, compare fares, and choose the most suitable seating options. With just a few taps, you can secure your Flight tickets and be one step closer to your desired destination. Our intuitive booking process enables you to customize your travel preferences, such as selecting specific departure times, opting for a window seat, or accommodating any special requirements.
            </p>
            <p className="section-description">
                       With our Flight ticket booking app, you can embrace the joy of exploring new destinations, immerse yourself in breathtaking scenery, and create cherished memories along the way. Start your journey today and let us be your trusted companion in making your Flight travel dreams a reality. Experience the convenience, reliability, and comfort that our app offers, and embark on unforgettable Flight adventures with confidence.
            </p>

            <span><h5>2023 SB FlightConnect - &copy; All rights reserved</h5></span>

        </div>
    </section>
    </div>
  )
}

export default LandingPage
```

NewFlights.jsx

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

```
const NewFlight = () => {



    const [userDetails, setUserDetails] = useState();

    useEffect(()=>{
      fetchUserData();
    }, [])

    const fetchUserData = async () =>{
      try{
        const id = localStorage.getItem('userId');
        await axios.get(`http://localhost:6001/fetch-user/${id}`).then(
          (response)=>{
            setUserDetails(response.data);
            console.log(response.data);
          }
        )

      }catch(err){

      }
    }
```

AuthProtector.jsx

```
import { useEffect } from 'react';

const AuthProtector =  ({ children }) => {

  useEffect(() => {

    if (!localStorage.getItem('userType')) {
      window.location.href = '/';
    }
  }, [localStorage]);
```

```
  return children;
};

export default AuthProtector;
```

LoginProtector.jsx

```
import React from 'react'
import { Navigate } from 'react-router-dom';

const LoginProtector = ({children}) => {

    if (localStorage.getItem('userType')){
        if (localStorage.getItem('userType') === 'customer'){
            return <Navigate to='/' replace />
        }else if (localStorage.getItem('userType') === 'admin'){
            return <Navigate to='/admin' replace />
        }
    }

    return children;
}

export default LoginProtector;
```

Admin.css

```
.admin-page{

    margin-top: 15vh;
    height: fit-content;
}

.admin-page-cards{

    display: grid;
    grid-template-columns: auto auto auto auto;
    gap: 30px;
    justify-content: center;
    align-items: center;
```

```
}

.admin-card{

    padding: 3vh 2vw;
    text-align: center;
    width: 20vw;
    height: 150px;
    box-shadow: rgba(99, 99, 99, 0.2) 0px 2px 8px 0px;
    border: none;
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
}

.admin-card:hover{
    box-shadow: rgba(50, 50, 93, 0.25) 0px 13px 27px -5px, rgba(0, 0, 0, 0.3) 0px
8px 16px -8px;
}

.admin-card h4{
    margin: 0;

}

.admin-card p{

    margin: 0;
}



.admin-requests-container{

    margin: 10vh 0 3vh 3vw;
    border: 1px solid #1918182b;
    width: 70%;
    padding: 2vh 2vw;
}

.admin-requests-container h3{

    color: rgb(55, 84, 108);
```

```
}

.admin-requests{

    display: flex;
    flex-direction: column;
    gap: 10px;
    margin-top: 4vh;
    height: 30vh;
    overflow-y: scroll;
    margin-left: 2vw;
}

.admin-request{
    display: flex;
    align-items: center;
    gap: 45px;
    background-color: rgba(241, 242, 244, 0.566);
    border: 1px solid #e7e3e3d6;
    padding: 15px 25px;
    margin-right: 10px;
    border-radius: 0.8rem;
}

.admin-request span{

    display: flex;
    flex-direction: column;
}

.admin-request span b{
    font-weight: 700;
    color: rgb(67, 89, 107);
}




.admin-request-actions{

    display: flex;
    gap: 30px;
```

```
}
.admin-request-actions button{
    width: 7rem;
}
```

AllFlights.css

```
.allFlightsPage{
    margin: 13vh 0 0 0;
    height: fit-content;
    width: 100%;
}

.allFlightsPage h1{
    color: rgb(73, 111, 138);
    margin-left: 3vw;
}

.allFlights{

    margin-top: 5vh;
    width: 90%;
    margin-left: 5%;
    height: fit-content;
    padding-bottom: 5vh;
    display: grid;
    grid-template-columns: 50% 50%;
    row-gap: 30px;
}

.allFlights .allFlights-Flight{

    background-color: rgba(237, 239, 241, 0.212);
    width: 90%;
    margin-left: 5%;
    padding: 3vh 2vw;
    border-radius: 0.7rem;
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;

}
.allFlights .allFlights-Flight:hover{
    box-shadow: rgba(0, 0, 0, 0.1) 0px 10px 50px;
```

```css
}
.allFlights .allFlights-Flight span{
    display: flex;
    gap: 25px;

}

.allFlights .allFlights-Flight p{
    margin: 0;
    margin-bottom: 3px;
    color: rgb(46, 104, 135);
}
.allFlights .allFlights-Flight p b{

    font-weight: 600;
}
.allFlights .allFlights-Flight ol{
    margin: 0;
    font-size: 0.85rem;
}
.allFlights .allFlights-Flight ol b{
   font-weight: 500;
}
.allFlights .allFlights-Flight button{

    margin-top: 15px;
}
```

AllUsers.css

```css
.all-users-page{
    width: 100%;
    height: 100vh;
    padding-top: 13vh;
    padding-left: 3vw
}
.all-users-page h2{
    margin-bottom: 4vh;
    color: rgba(77, 109, 134, 0.934);
}
```

```css
.all-users{
    width: 80%;
    height: 30vh;
    margin-bottom: 4vh;
    display: flex;
    flex-direction: column;
    gap: 10px;
    overflow-y: scroll;
    -ms-overflow-style: none;
    scrollbar-width: none;
}
.all-users::-webkit-scrollbar {
    display: none;
}

.all-users .user{
    width: 70%;
    background-color: rgba(231, 239, 236, 0.416);
    border-radius: 0.8rem;
    display: flex;
    padding: 2vh 2vw;
    gap: 2vw;
}
.all-users .user p{

    margin: 0;
    color: rgb(36, 132, 173);
}
.all-users .user p b{
    font-weight: 600;
    color: rgb(32, 97, 124);
}
```

Authenticate.css

```css
.AuthenticatePage{
    width: 100%;
    height: 100vh;
    padding-top: 12vh;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

```css
.authForm{
    /* margin-top: 5%; */
    /* border: 1px solid #cb7c7c; */
    padding: 20px;
    width: 400px;
    display: flex;
    flex-direction: column;
    /* align-items: center; */
    justify-content: center;
    text-align: center;
    height: max-content;
    font-family: 'Work Sans', sans-serif;

    background-color: rgba(255, 255, 255, 0.263);
    backdrop-filter: blur(10px);
    box-shadow: rgba(14, 30, 37, 0.12) 0px 2px 4px 0px, rgba(14, 30, 37, 0.32) 0px
2px 16px 0px;
    border-radius: 0.5rem;

}
.authForm h2{
    margin-bottom: 20px;
    font-family: 'Poppins', sans-serif;
    color: rgb(0, 79, 135);
    font-size: 1.8rem;
}
.authFormInputs {
    width: 100%;
    height: 50px;
}

.authFormInputs label{
    color: rgb(0, 61, 101);
    /* padding: 0; */
}
.authFormInputs input{
    color: rgb(0, 61, 101);
    border: none;
    outline: none;
    background-color: rgba(231, 235, 235, 0);
    border: 1px solid #9dc1d975;
    /* padding: 0; */
```

```
}
.authForm select {
    color: rgb(0, 61, 101);
    border: none;
    outline: none;
    background-color: rgba(231, 235, 235, 0);
    border: 1px solid #9dc1d975;
    font-size: medium;
    padding: 15px 10px;
}
.authFormInputs input #floatingInput .form-control{
    padding: 0 !important;
}

.authForm button{
    background-color: rgb(0, 108, 197);
    font-weight: bold;
    color: rgb(246, 251, 255);
    height: 50px;
}
.authForm p{
    margin-top: 5%;
    color: black;
}
.authForm p span{
    margin-top: 5%;
    color: rgb(0, 68, 107);
    cursor: pointer;
}
```

BookFlights.css

```
.BookFlightPage{
    width: 100%;
    height: fit-content;
    padding-top: 15vh;
    padding-bottom: 5vh;
    display: flex;
    justify-content: center;
}

.BookingFlightPageContainer{
```

```
    width: 44%;
    background-color: rgba(242, 245, 247, 0.753);
    padding: 2vh 2%;
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;
    border-radius: 0.7rem;
}
.BookingFlightPageContainer h2{
    margin: 0;
    margin-bottom: 5vh;
    text-align: center;
    color: rgb(54, 104, 120);
}
.BookingFlightPageContainer p{
    margin: 0;
    margin-bottom: 5px;
    color: rgb(54, 104, 120);
}
.BookingFlightPageContainer p b{
    font-weight: 600;
}
.BookingFlightPageContainer span{
    display: grid;
    grid-template-columns: 64% 33%;
    gap: 3%;
}
.BookingFlightPageContainer .span3{
    display: grid;
    grid-template-columns: 31.3% 31.3% 31.3%;
    gap: 3%;
}
.BookingFlightPageContainer button{
    width: 30%;
    margin-left: 35%;
    margin-top: 25px;
}
.BookingFlightPageContainer li{
    font-size: 0.8rem;
}

.new-passenger h4{
    font-size: 1rem;
    font-weight: 600;
    color: rgb(48, 91, 114);
}
```

```css
.new-passenger-inputs{
    display: grid;
    grid-template-columns: 67% 30%;
    gap: 3%;
}
.new-passenger-inputs select{
    padding-left: 12px;
}
```

Bookings.css

```css
.user-bookingsPage{
    margin: 13vh 0 0 0;
    height: fit-content;
    width: 100%;
}

.user-bookingsPage h1{
    color: rgb(73, 111, 138);
    margin-left: 3vw;
}

.user-bookings{

    margin-top: 5vh;
    width: 90%;
    margin-left: 5%;
    height: fit-content;
    padding-bottom: 5vh;
    display: grid;
    grid-template-columns: 50% 50%;
    row-gap: 30px;
}

.user-bookings .user-booking{

    background-color: rgba(237, 239, 241, 0.212);
    width: 90%;
    margin-left: 5%;
    padding: 3vh 2vw;
    border-radius: 0.7rem;
```

```
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;

}
.user-bookings .user-booking:hover{
    box-shadow: rgba(0, 0, 0, 0.1) 0px 10px 50px;
}
.user-bookings .user-booking span{
    display: flex;
    gap: 25px;

}

.user-bookings .user-booking p{
    margin: 0;
    margin-bottom: 3px;
    color: rgb(46, 104, 135);
}
.user-bookings .user-booking p b{

    font-weight: 600;
}
.user-bookings .user-booking ol{
    margin: 0;
    font-size: 0.85rem;
}
.user-bookings .user-booking ol b{
   font-weight: 500;
}
.user-bookings .user-booking button{

    margin-top: 15px;
}
```

FlightAdmin.css

```
.flightAdmin-page{

    padding-top: 15vh;
    height: 100vh;
}

.notApproved-box{
```

```
    width: 60%;
    text-align: center;
    margin: 30vh 20%;
    background-color: rgba(237, 239, 241, 0.692);
    padding: 4vh 4vw;
}

.notApproved-box h3{

    color: #e35252;
}

.notApproved-box p{

    color: rgb(82, 102, 120);
}
```

LandingPage.css

```
.landingPage{
    width: 100%;
    height: fit-content;
}

.landingHero{
    width: 100%;
    height: 100vh;
    padding-top: 10vh;
    background-image: linear-gradient(45deg, rgba(36, 59, 69, 0.489),rgba(0, 0, 0,
0.493)), url('../assets/HomeBG1.png');
    background-position: center;
    background-size: cover;
    background-repeat: no-repeat;
    overflow-y: scroll;
    -ms-overflow-style: none;
    scrollbar-width: none;
}
.landingHero::-webkit-scrollbar {
    display: none;
  }
```

```css
.landingHero-title{
    width: 60%;
    margin: 6% 0 0 4%;
}

.landingHero-title h1{
    color: aliceblue;
    font-size: 4rem;
}

.landingHero-title p{
    color: rgba(240, 248, 255, 0.68);
}

.Flight-search-container{
    margin-left: 5vw;
    width: fit-content;
    height: fit-content;
    background-color: rgba(227, 237, 237, 0.267);
    backdrop-filter: blur(15px);

    padding: 15px 20px 0px 20px;
    border-radius: 0.6rem;
}
.Flight-search-container h3{
    margin-bottom: 15px;
    font-size: 1.5rem;
    color: rgb(232, 244, 255);
}
.Flight-search-container p{
    padding-bottom: 10px;
    text-align: center;
    color: rgb(183, 40, 40);
    font-weight: 500;
}
.form-check-label{
    color: rgb(255, 255, 255);
}
.Flight-search-container-body{
    display: flex;
    /* height: 100%; */
```

```
    margin-top: 3vh;
    gap: 15px;
}
.Flight-search-container select{
    padding-left: 15px;
    width: 15vw;
    background-color: rgba(255, 255, 255, 0.831);
    border: none;
    color: rgb(26, 52, 69);
}
.Flight-search-container input{
    padding-left: 15px;
    width: 15vw;
    color: rgb(26, 52, 69);
    background-color: rgba(255, 255, 255, 0.831);
}
.Flight-search-container button{
    /* margin-bottom: 15px; */
    padding: 15px 25px;
    font-size: 1rem;
    font-weight: 500;
}

.availableFlightsContainer{
    margin: 3vh 5vw;
}
.availableFlightsContainer h1{

    color: aliceblue;
}

.availableFlightsContainer .Flights{
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.availableFlightsContainer .Flights .Flight{
    width: 90%;
    display: flex;
    align-items: center;
    justify-content: space-between;
    background-color: rgba(227, 237, 237, 0.267);
```

```
    backdrop-filter: blur(15px);
    padding: 2vh 2vw;
    border-radius: 0.5rem;
}
.availableFlightsContainer .Flights .Flight p{
    margin: 0;
    color: rgb(201, 205, 209);
}
.availableFlightsContainer .Flights .Flight p b{
    font-weight: 600;
    color: rgb(227, 235, 227);
}



.section-about {
    min-height: 50vh;
    /* background-color: rgb(207, 221, 233); */
    background: rgb(75,143,209);
    background: linear-gradient(145deg, rgb(205, 227, 249) 0%, rgb(232, 239, 243)
55%, rgb(213, 189, 222) 100%);
  }
  .section-about h2{

    font-weight: 600;
    color: #055791;

}

.section-about p{
  text-align: justify;
  color: #055791;
}
.section-about span{
  text-align: center;
  color: #6badeb;
}
.section-about span h5{
  margin-top: 8vh;
  font-size: 1rem;
}
```

Navbar.css

```css
.navbar{

    height: 10vh;
    width: 100%;
    position: fixed;
    top: 0vh;
    z-index: 10;
    background-color: rgb(28, 82, 126) !important;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 0 3%;
    box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;
}

.navbar h3{
    color: aliceblue;
}

.nav-options{
    display: flex;
    gap: 20px;
}

.nav-options p{
    margin: 0;
    color: rgb(209, 210, 211);
    text-decoration: none;
    cursor: pointer;
}

.nav-options p:hover{
    color: aliceblue;
}
```

NewFlight.css

```css
.NewFlightPage{
    width: 100%;
    height: fit-content;
    padding-top: 18vh;
    padding-bottom: 5vh;
```

```
    display: flex;
    justify-content: center;
}

.NewFlightPageContainer{
    width: 44%;
    background-color: rgba(242, 245, 247, 0.753);
    padding: 2vh 2%;
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;
    border-radius: 0.7rem;
}
.NewFlightPageContainer button{
    width: 30%;
    margin-left: 35%;
}
.NewFlightPageContainer h2{
    margin: 0;
    margin-bottom: 15px;
    color: rgb(54, 104, 120);
    text-align: center;
}
.NewFlightPageContainer p b{
    font-weight: 600;
}
.NewFlightPageContainer span{
    display: grid;
    grid-template-columns: 67% 30%;
    gap: 3%;
}
.NewFlightPageContainer .newFlightSpan1{
    display: grid;
    grid-template-columns: 54% 43%;
    gap: 3%;
}
.NewFlightPageContainer .newFlightSpan2{
    display: grid;
    grid-template-columns: 48.5% 48.5%;
    gap: 3%;
}

.new-passenger h4{
    font-size: 1rem;
    font-weight: 600;
    color: rgb(48, 91, 114);
```

```
}

.new-passenger-inputs{
    display: grid;
    grid-template-columns: 49% 15% 30%;
    gap: 3%;
}
.new-passenger-inputs select{
    padding-left: 12px;
}
```

App.css

```
.App{

  overflow-x: hidden;
  height: fit-content;
}
```

App.js

```
import logo from './logo.svg';
import './App.css';
import Navbar from './components/Navbar';
import LandingPage from './pages/LandingPage';
import Authenticate from './pages/Authenticate';
import Bookings from './pages/Bookings';
import Admin from './pages/Admin';
import AllUsers from './pages/AllUsers';
import AllBookings from './pages/AllBookings';
import AllFlights from './pages/AllFlights';
import NewFlight from './pages/NewFlight';
import {Routes, Route} from 'react-router-dom'
import LoginProtector from './RouteProtectors/LoginProtector';
import AuthProtector from './RouteProtectors/AuthProtector';
import BookFlight from './pages/BookFlight';
import EditFlight from './pages/EditFlight';
import FlightAdmin from './pages/FlightAdmin';
import FlightBookings from './pages/FlightBookings.jsx';
import Flights from './pages/Flights.jsx';

function App() {
```

```
  return (
    <div className="App">
      <Navbar />

      <Routes>
        <Route exact path = '' element={<LandingPage />} />
        <Route    path='/auth'    element={<LoginProtector>    <Authenticate    />
</LoginProtector>} />
        <Route  path='/book-Flight/:id'  element={<AuthProtector>  <BookFlight  />
</AuthProtector>} />
        <Route    path='/bookings'    element={<AuthProtector>    <Bookings    />
</AuthProtector>} />

        <Route path='/admin' element={<AuthProtector><Admin /> </AuthProtector>}
/>
        <Route     path='/all-users'     element={<AuthProtector><AllUsers     />
</AuthProtector>} />
        <Route   path='/all-bookings'   element={<AuthProtector><AllBookings   />
</AuthProtector>} />
        <Route    path='/all-flights'    element={<AuthProtector><AllFlights    />
</AuthProtector>} />

        <Route   path='/flight-admin'   element={<AuthProtector><FlightAdmin   />
</AuthProtector>} />
        <Route path='/flight-bookings' element={<AuthProtector><FlightBookings />
</AuthProtector>} />
        <Route      path='/flights'      element={<AuthProtector><Flights      />
</AuthProtector>} />
        <Route    path='/new-flight'    element={<AuthProtector><NewFlight    />
</AuthProtector>} />
        <Route   path='/edit-flight/:id'   element={<AuthProtector><EditFlight   />
</AuthProtector>} />
      </Routes>

    </div>
  );
}

export default App;
```

App.test.js

```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

Index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
import GeneralContextProvider from './context/GeneralContext';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
```

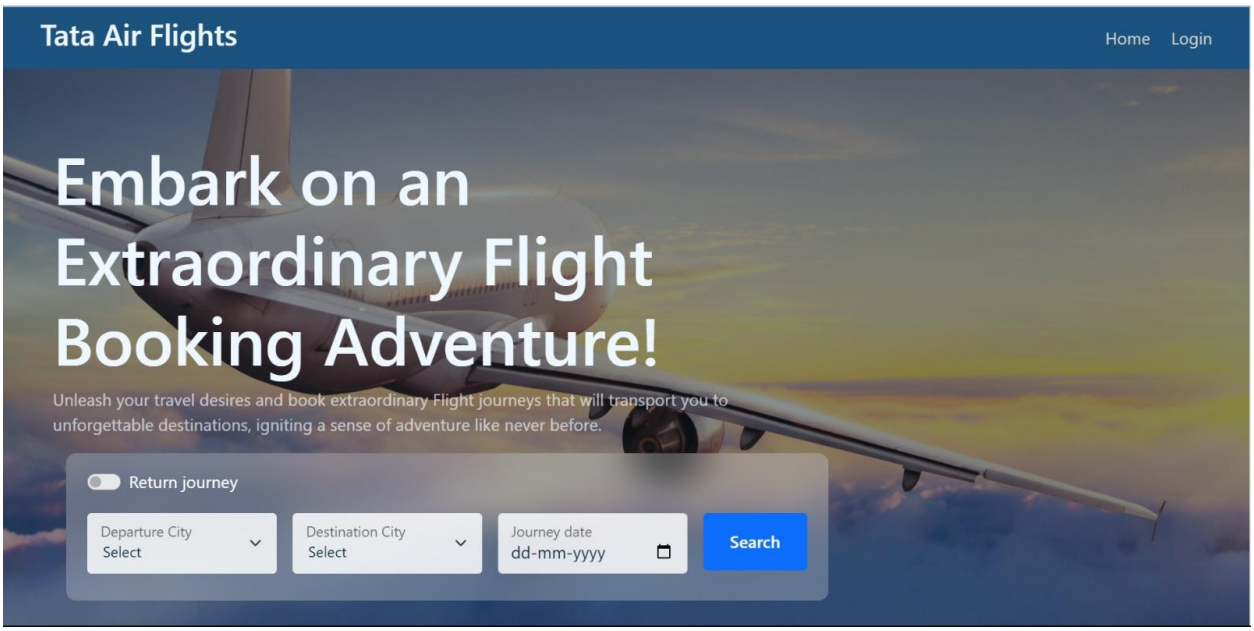
```
          <GeneralContextProvider>
              <App />
          </GeneralContextProvider>
    </BrowserRouter>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

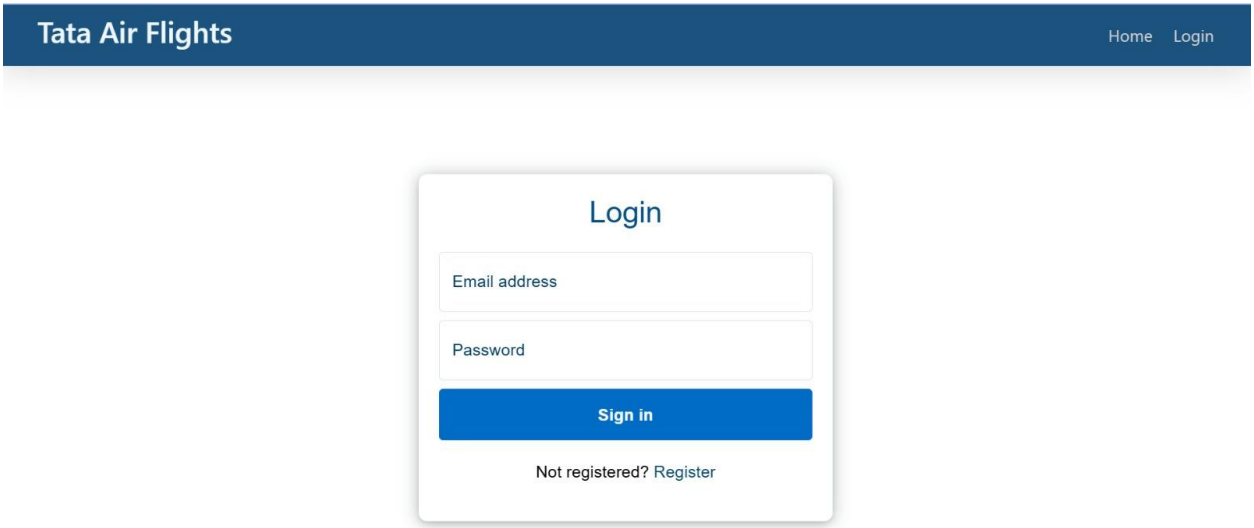
ReportWebVitals.js

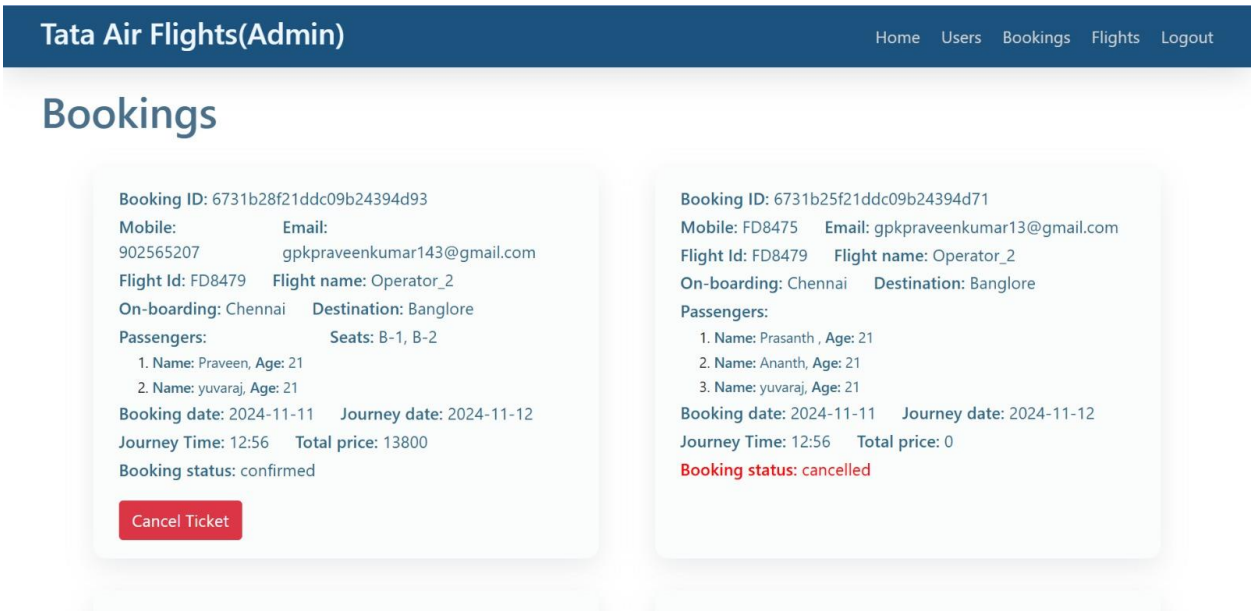
```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

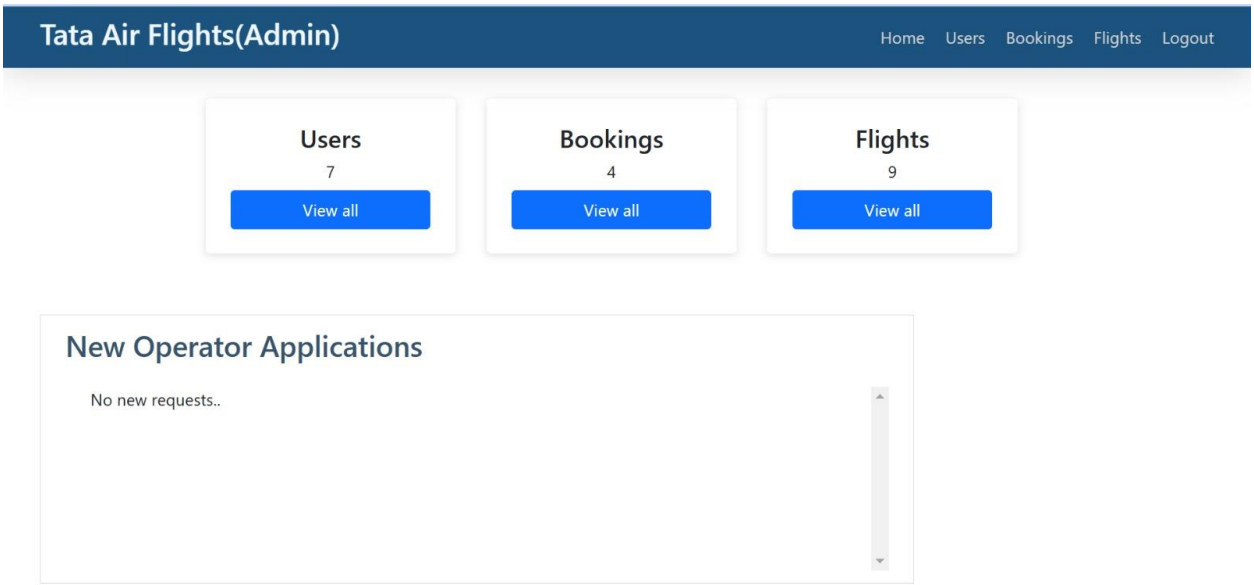
9.3. SCREENSHOTS :

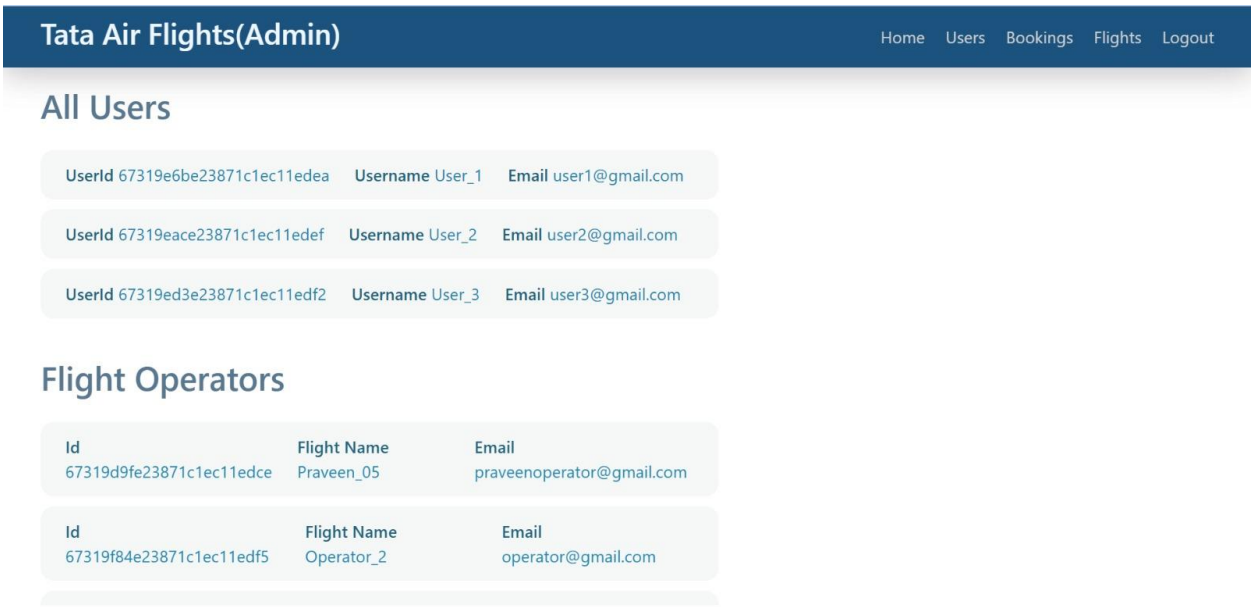
HOME PAGE:

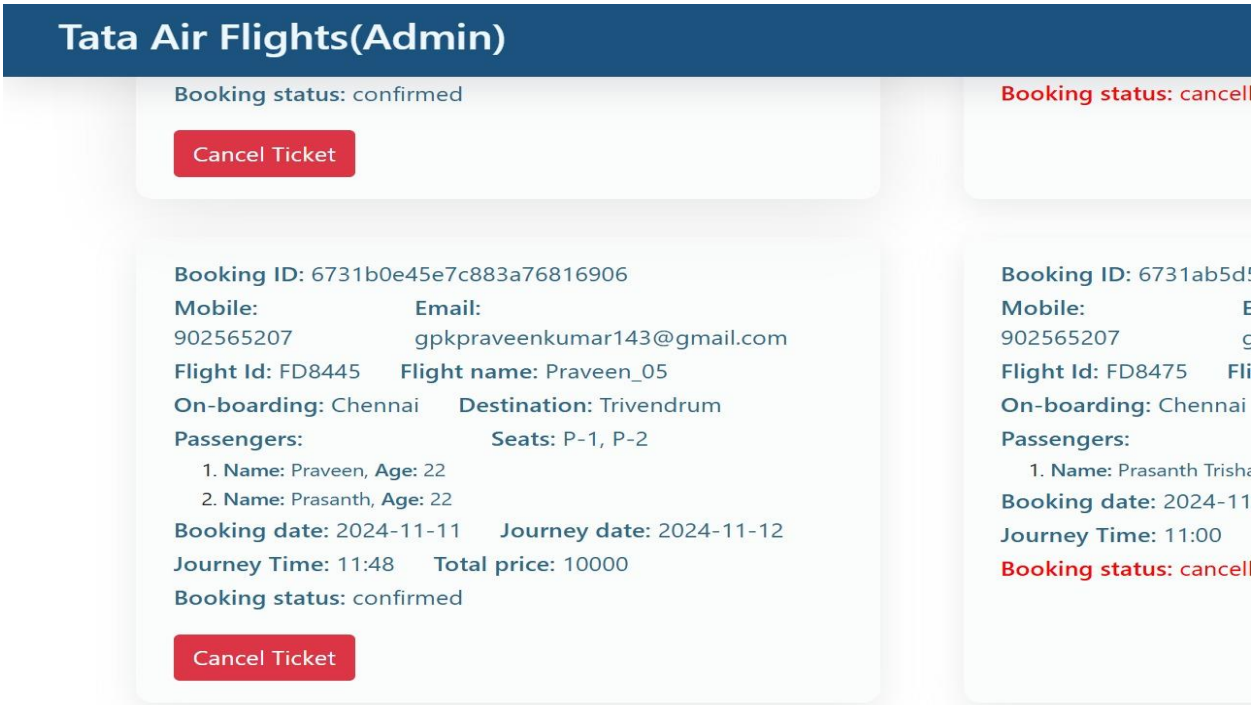

Tata Air Flights
Home
Login
Embark on an Extraordinary Flight Booking Adventure!
Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.
Return journey
Departure City
Select
Destination City
Select
Journey date
dd-mm-yyyy
Search

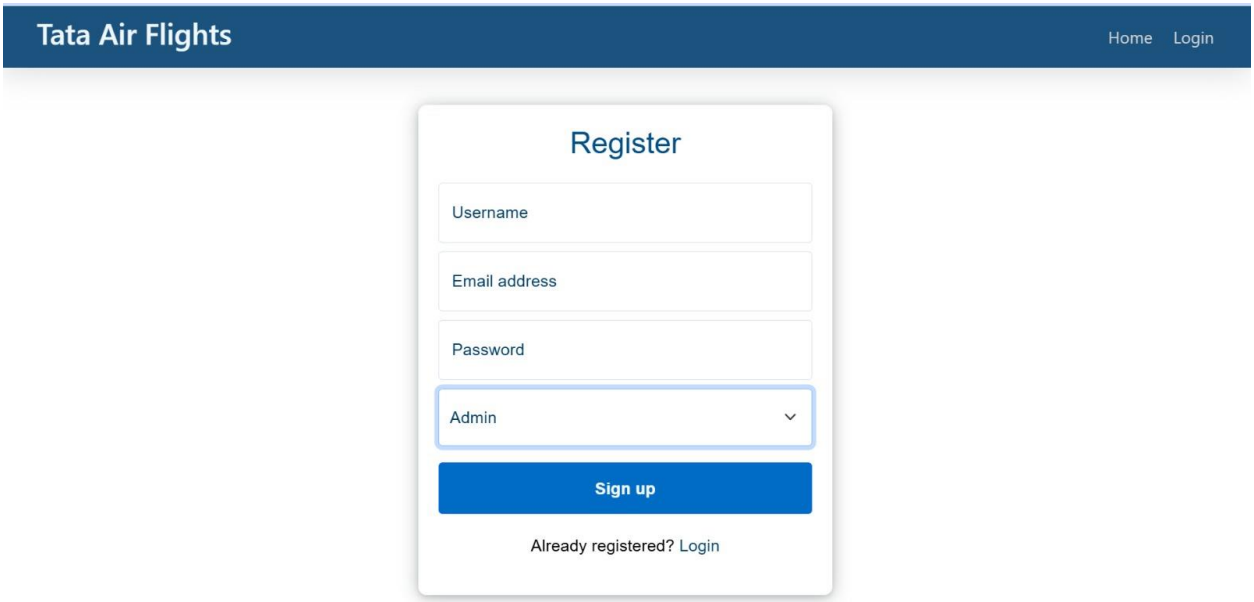

Tata Air Flights
Home
Login
Login
Email address
Password
Sign in
Not registered? Register

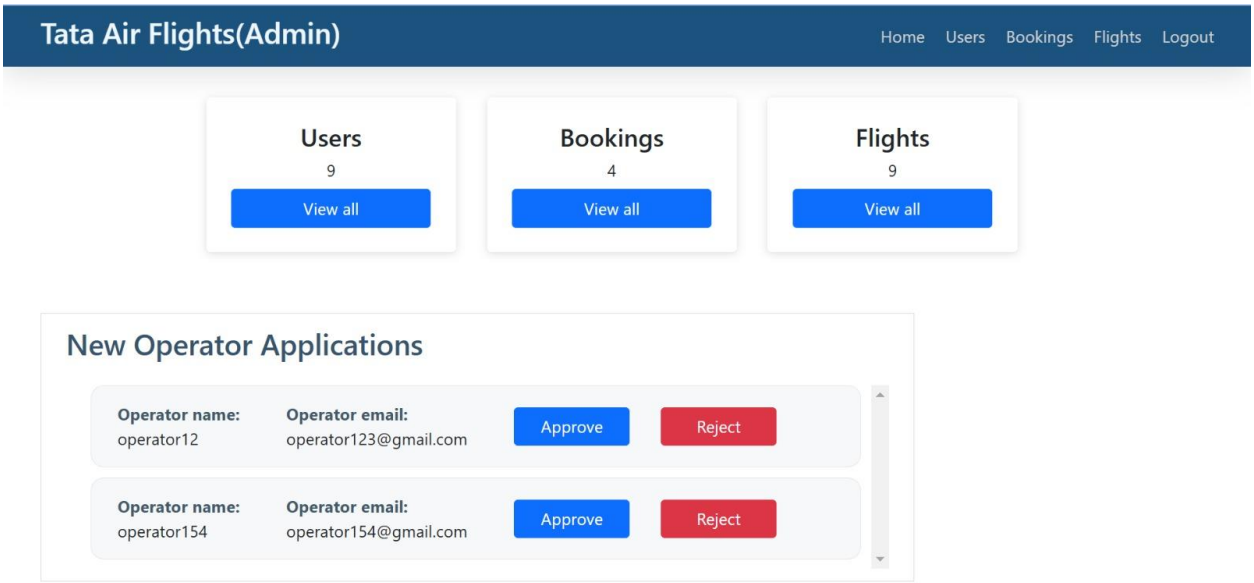

Tata Air Flights(Admin)
Home Users Bookings Flights Logout
Bookings
Booking ID: 6731b28f21ddc09b24394d93
Mobile: 902565207 Email: gpkpraveenkumar143@gmail.com
Flight Id: FD8479 Flight name: Operator_2
On-boarding: Chennai Destination: Banglore
Passengers: Seats: B-1, B-2
1. Name: Praveen, Age: 21
2. Name: yuvaraj, Age: 21
Booking date: 2024-11-11 Journey date: 2024-11-12
Journey Time: 12:56 Total price: 13800
Booking status: confirmed
Cancel Ticket
Booking ID: 6731b25f21ddc09b24394d71
Mobile: FD8475 Email: gpkpraveenkumar13@gmail.com
Flight Id: FD8479 Flight name: Operator_2
On-boarding: Chennai Destination: Banglore
Passengers:
1. Name: Prasanth , Age: 21
2. Name: Ananth, Age: 21
3. Name: yuvaraj, Age: 21
Booking date: 2024-11-11 Journey date: 2024-11-12
Journey Time: 12:56 Total price: 0
Booking status: cancelled

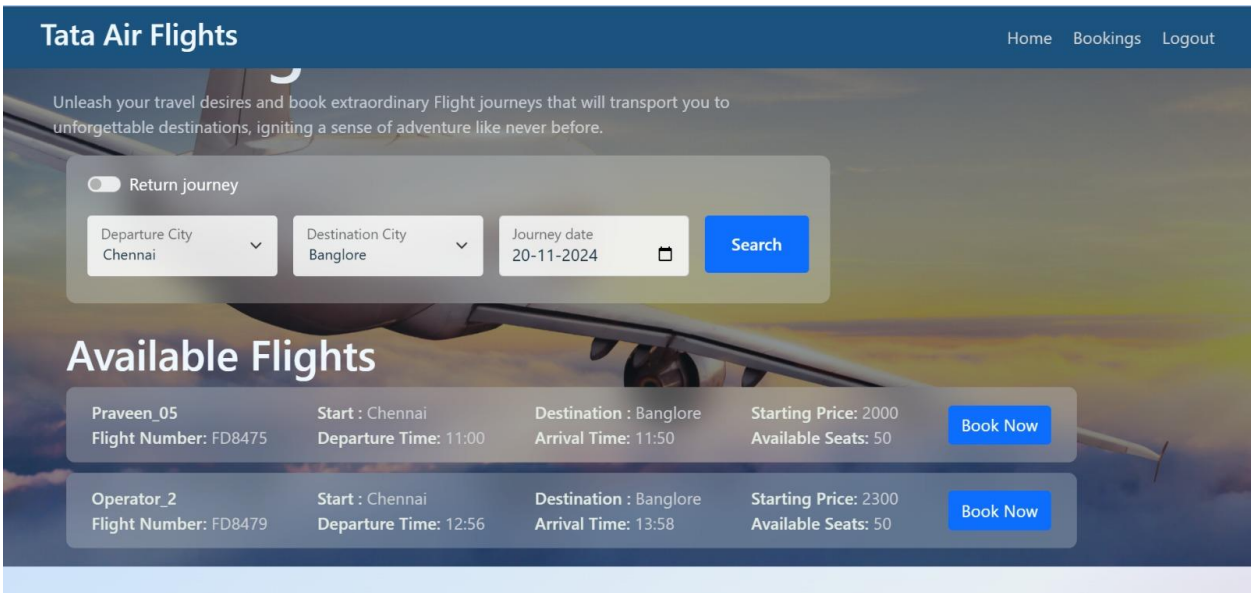

Tata Air Flights(Admin)
Home Users Bookings Flights Logout
Users
7
View all
Bookings
4
View all
Flights
9
View all
New Operator Applications
No new requests..

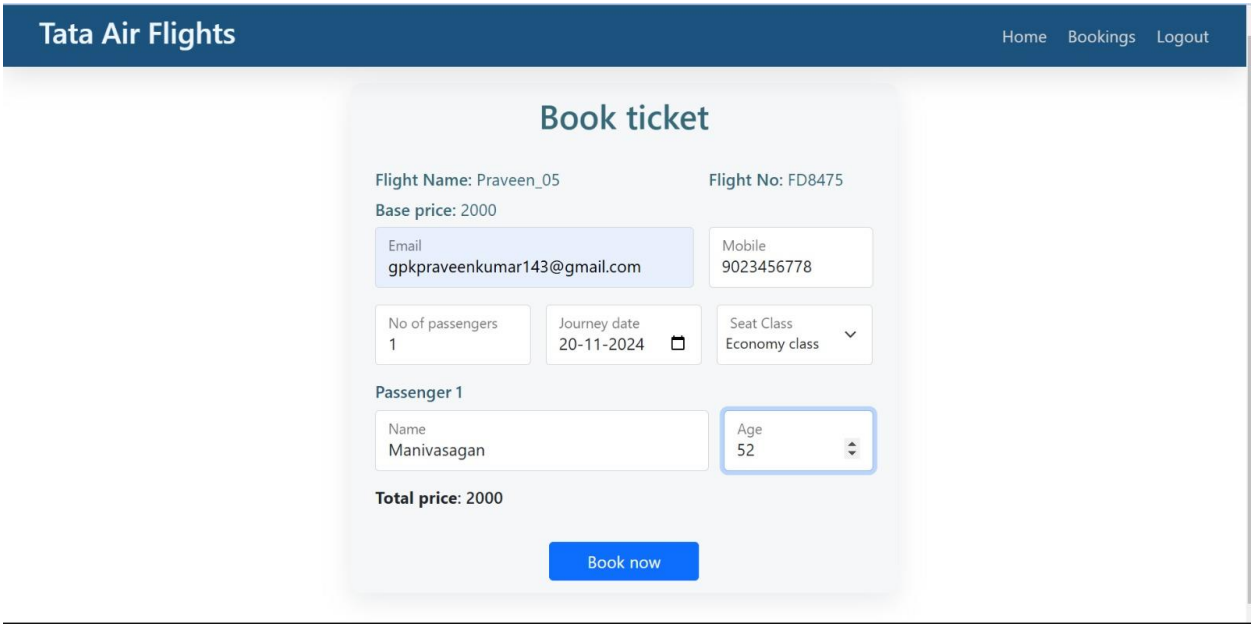

Tata Air Flights(Admin)
Home Users Bookings Flights Logout
All Users
UserId 67319e6be23871c1ec11edea Username User_1 Email user1@gmail.com
UserId 67319eace23871c1ec11edef Username User_2 Email user2@gmail.com
UserId 67319ed3e23871c1ec11edf2 Username User_3 Email user3@gmail.com
Flight Operators
Id 67319d9fe23871c1ec11edce Flight Name Praveen_05 Email praveenoperator@gmail.com
Id 67319f84e23871c1ec11edf5 Flight Name Operator_2 Email operator@gmail.com

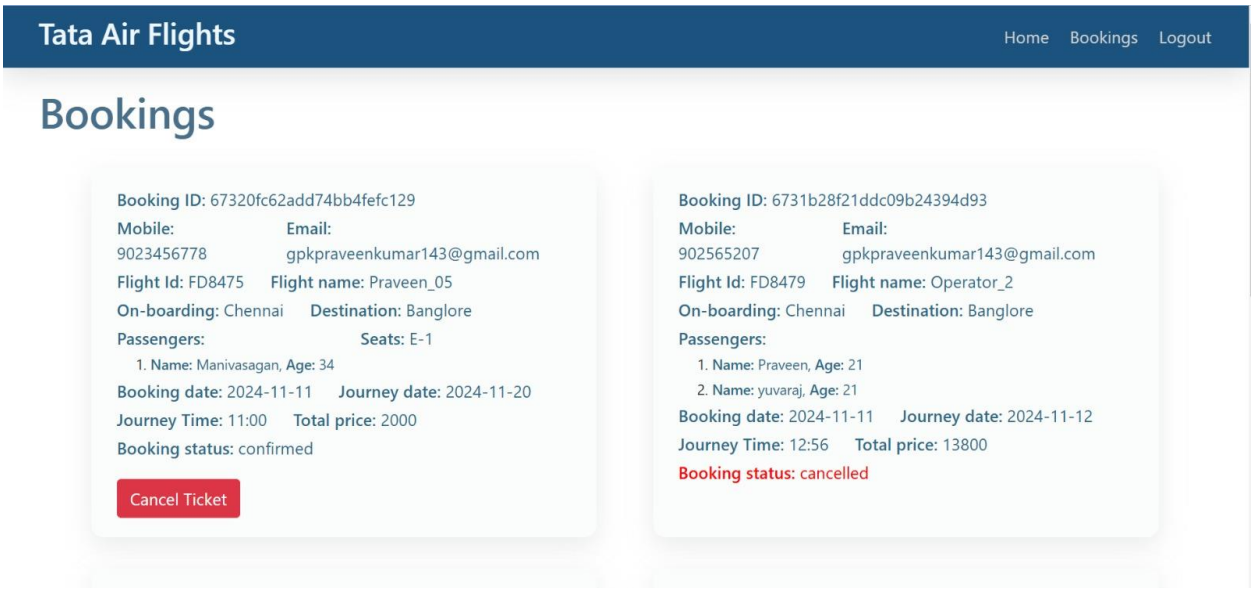

Tata Air Flights(Admin)
Booking status: confirmed
Cancel Ticket
Booking status: cancell
Booking ID: 6731b0e45e7c883a76816906
Mobile: 902565207
Email: gpkpraveenkumar143@gmail.com
Flight Id: FD8445
Flight name: Praveen_05
On-boarding: Chennai
Destination: Trivendrum
Passengers:
Seats: P-1, P-2
1. Name: Praveen, Age: 22
2. Name: Prasanth, Age: 22
Booking date: 2024-11-11
Journey date: 2024-11-12
Journey Time: 11:48
Total price: 10000
Booking status: confirmed
Cancel Ticket
Booking ID: 6731ab5d
Mobile: 902565207
Flight Id: FD8475
On-boarding: Chennai
Passengers:
1. Name: Prasanth Trish
Booking date: 2024-11
Journey Time: 11:00
Booking status: cancell

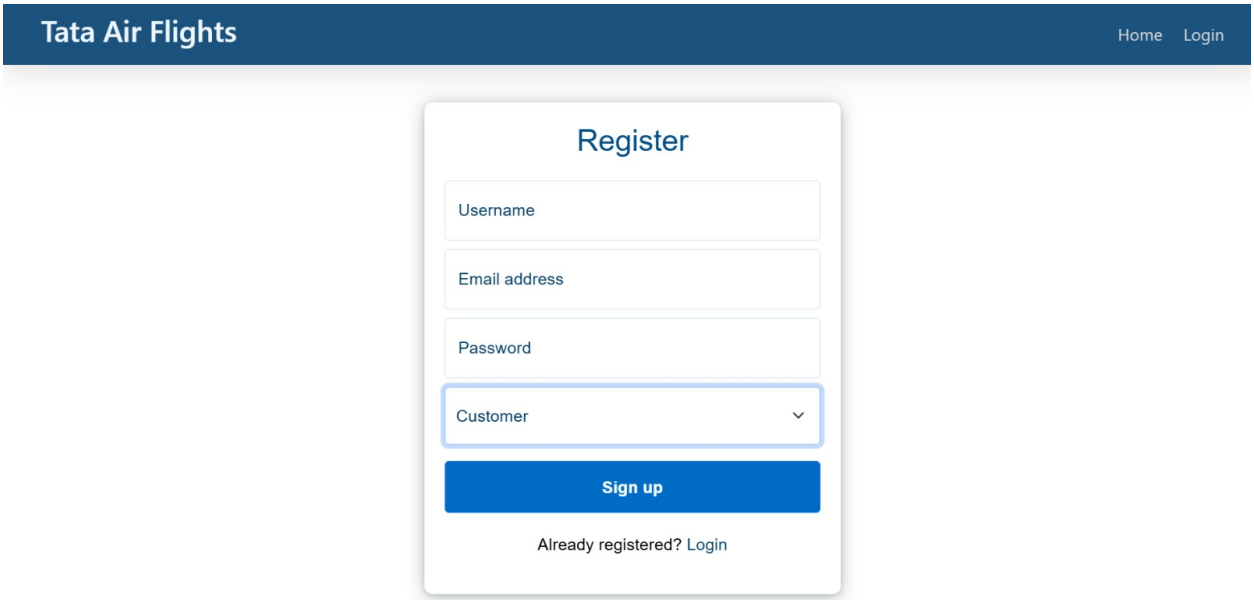

Tata Air Flights
Home
Login
Register
Username
Email address
Password
Admin
Sign up
Already registered? Login

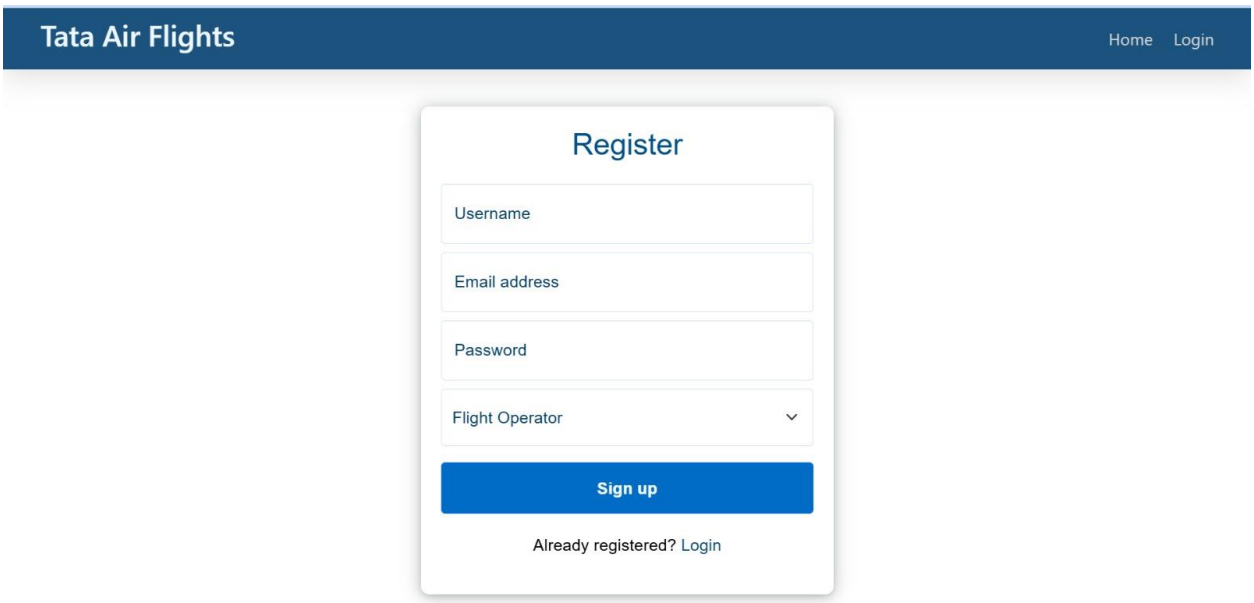

Tata Air Flights(Admin)
Home
Users
Bookings
Flights
Logout
Users
9
View all
Bookings
4
View all
Flights
9
View all
New Operator Applications
Operator name:
operator12
Operator email:
operator123@gmail.com
Approve
Reject
Operator name:
operator154
Operator email:
operator154@gmail.com
Approve
Reject

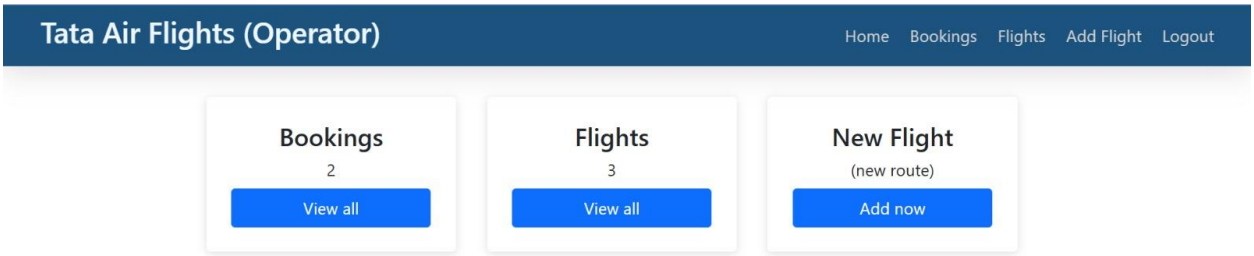

Tata Air Flights
Home
Bookings
Logout
Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.
Return journey
Departure City
Chennai
Destination City
Banglore
Journey date
20-11-2024
Search
Available Flights
Praveen_05
Flight Number: FD8475
Start : Chennai
Departure Time: 11:00
Destination : Banglore
Arrival Time: 11:50
Starting Price: 2000
Available Seats: 50
Book Now
Operator_2
Flight Number: FD8479
Start : Chennai
Departure Time: 12:56
Destination : Banglore
Arrival Time: 13:58
Starting Price: 2300
Available Seats: 50
Book Now

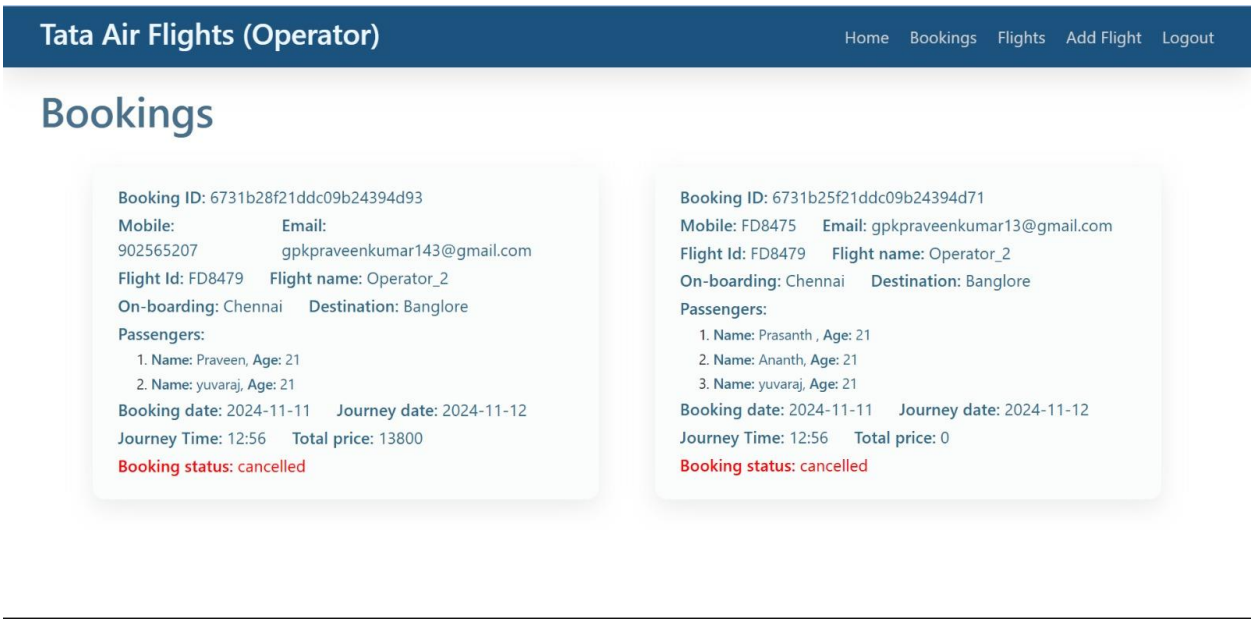

Tata Air Flights
Home
Bookings
Logout
Book ticket
Flight Name: Praveen_05
Flight No: FD8475
Base price: 2000
Email
gpkpraveenkumar143@gmail.com
Mobile
9023456778
No of passengers
1
Journey date
20-11-2024
Seat Class
Economy class
Passenger 1
Name
Manivasagan
Age
52
Total price: 2000
Book now

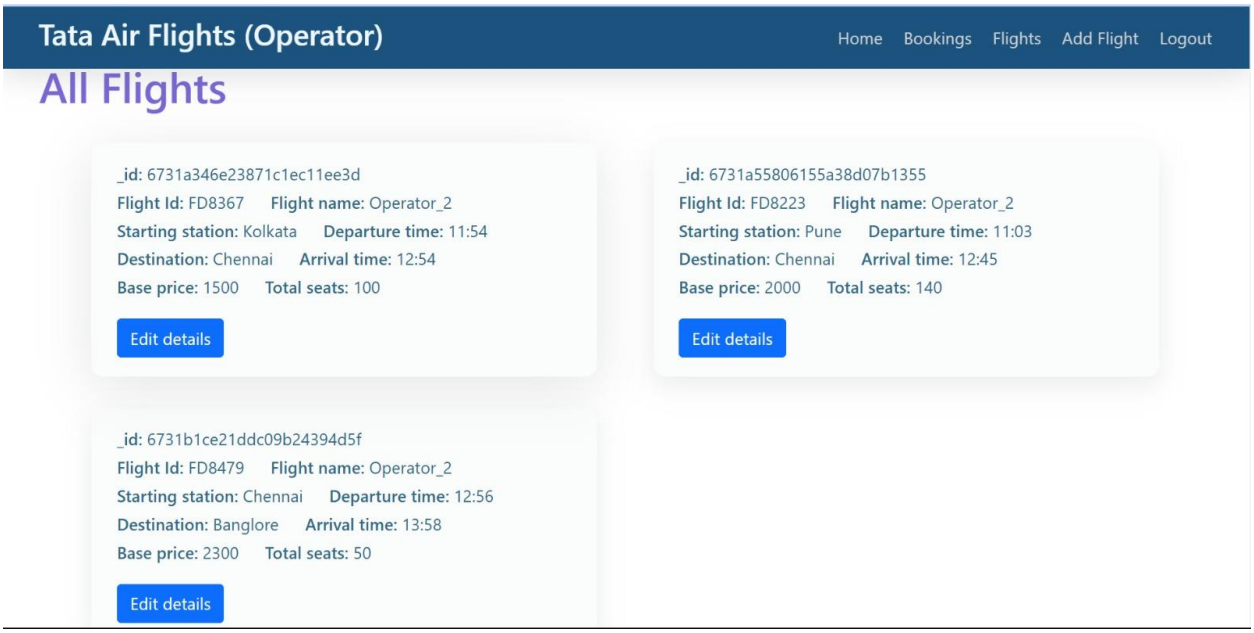

Tata Air Flights
Home Bookings Logout
Bookings
Booking ID: 67320fc62add74bb4fefc129
Mobile: 9023456778
Email: gpkpraveenkumar143@gmail.com
Flight Id: FD8475 Flight name: Praveen_05
On-boarding: Chennai Destination: Banglore
Passengers: Seats: E-1
1. Name: Manivasagan, Age: 34
Booking date: 2024-11-11 Journey date: 2024-11-20
Journey Time: 11:00 Total price: 2000
Booking status: confirmed
Cancel Ticket
Booking ID: 6731b28f21ddc09b24394d93
Mobile: 902565207
Email: gpkpraveenkumar143@gmail.com
Flight Id: FD8479 Flight name: Operator_2
On-boarding: Chennai Destination: Banglore
Passengers:
1. Name: Praveen, Age: 21
2. Name: yuvaraj, Age: 21
Booking date: 2024-11-11 Journey date: 2024-11-12
Journey Time: 12:56 Total price: 13800
Booking status: cancelled

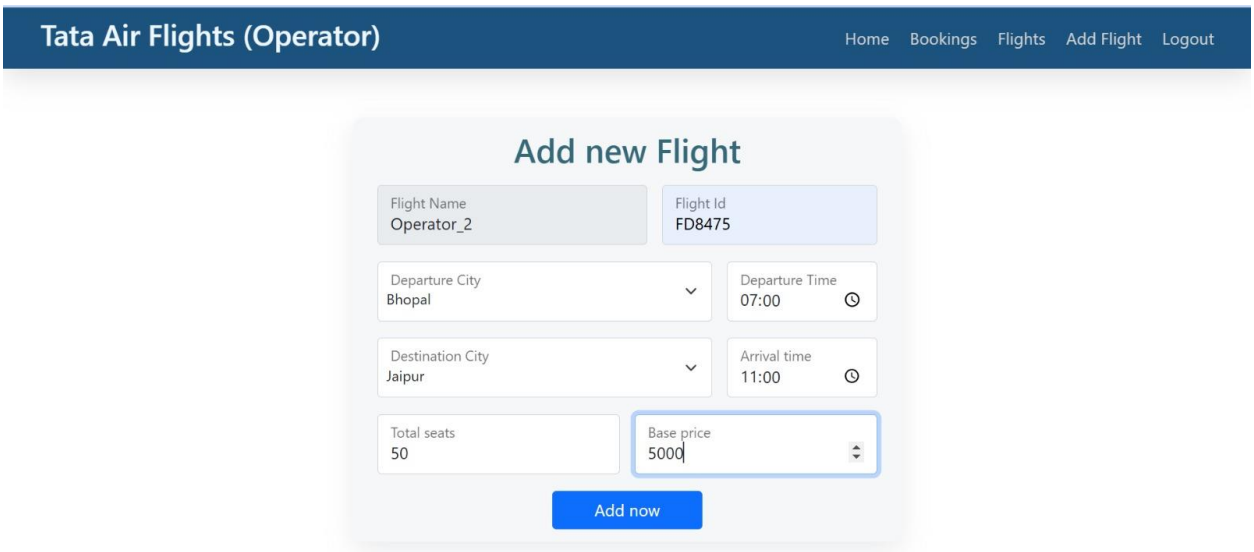

Tata Air Flights
Home Login
Register
Username
Email address
Password
Customer
Sign up
Already registered? Login

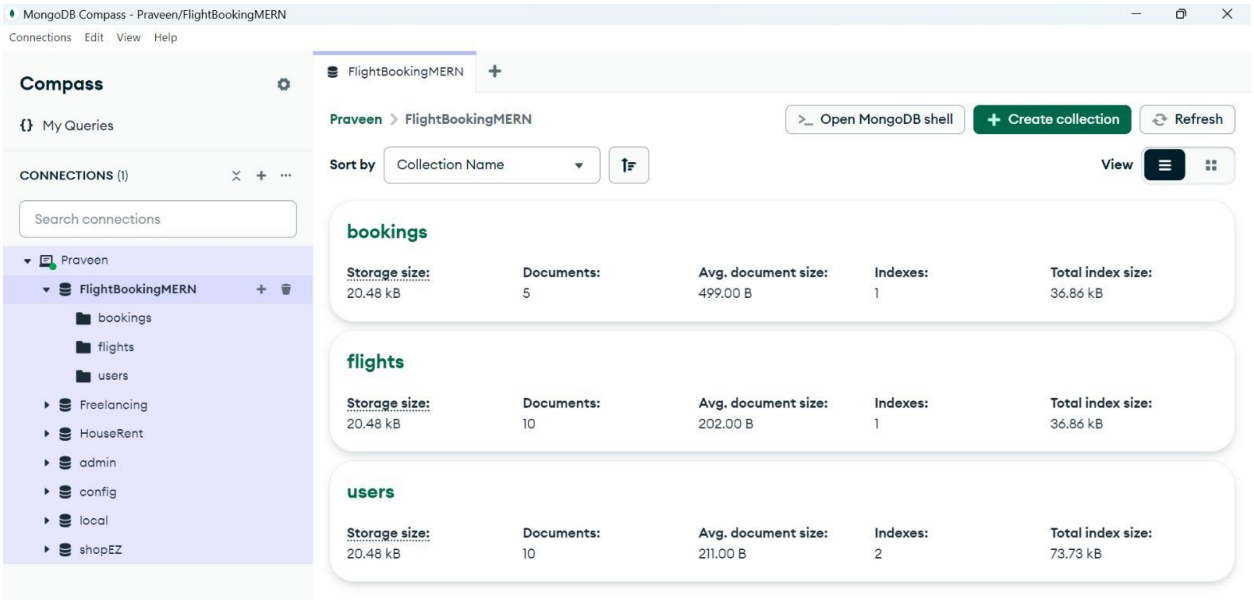

Tata Air Flights
Home
Login
Register
Username
Email address
Password
Flight Operator
Sign up
Already registered? Login

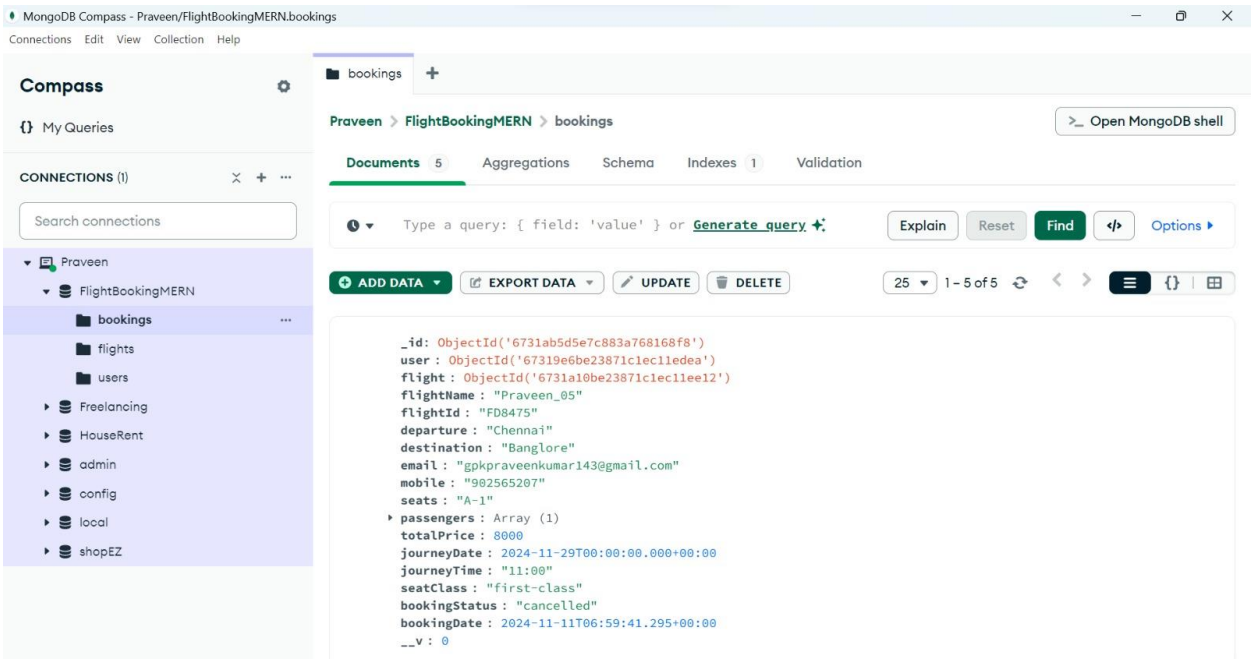

Tata Air Flights (Operator)
Home
Bookings
Flights
Add Flight
Logout
Bookings
2
View all
Flights
3
View all
New Flight
(new route)
Add now

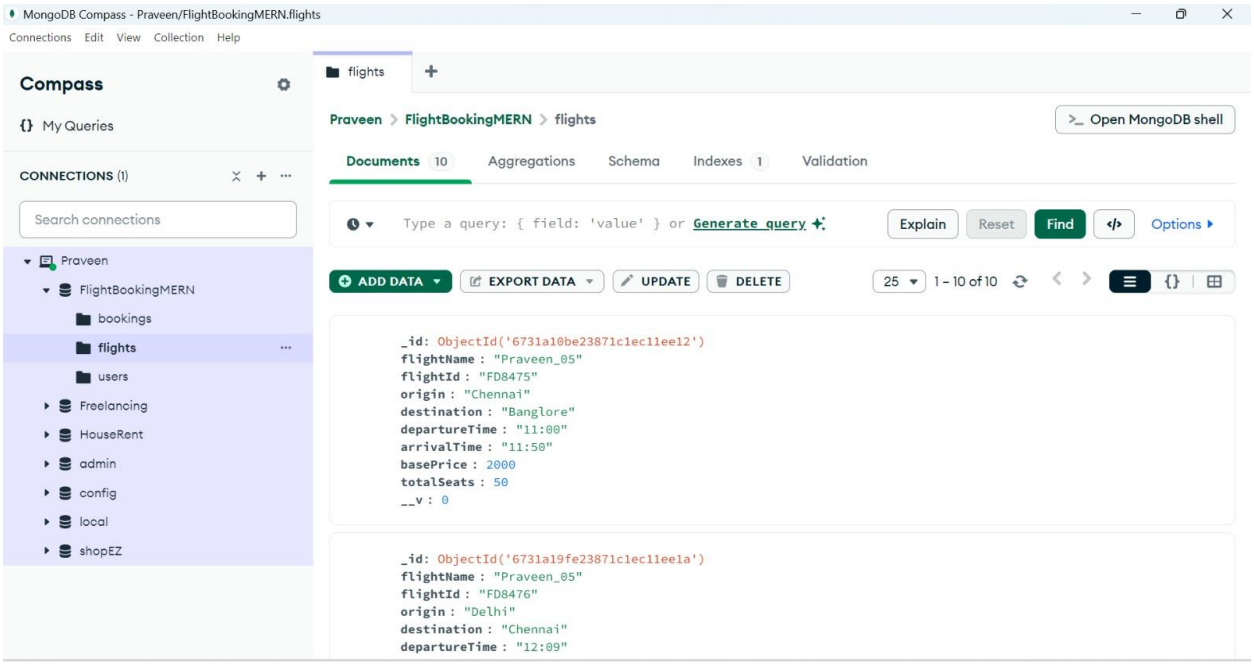

Tata Air Flights (Operator)
Home Bookings Flights Add Flight Logout
Bookings
Booking ID: 6731b28f21ddc09b24394d93
Mobile: 902565207 Email: gpkpraveenkumar143@gmail.com
Flight Id: FD8479 Flight name: Operator_2
On-boarding: Chennai Destination: Banglore
Passengers:
1. Name: Praveen, Age: 21
2. Name: yuvaraj, Age: 21
Booking date: 2024-11-11 Journey date: 2024-11-12
Journey Time: 12:56 Total price: 13800
Booking status: cancelled
Booking ID: 6731b25f21ddc09b24394d71
Mobile: FD8475 Email: gpkpraveenkumar13@gmail.com
Flight Id: FD8479 Flight name: Operator_2
On-boarding: Chennai Destination: Banglore
Passengers:
1. Name: Prasanth , Age: 21
2. Name: Ananth, Age: 21
3. Name: yuvaraj, Age: 21
Booking date: 2024-11-11 Journey date: 2024-11-12
Journey Time: 12:56 Total price: 0
Booking status: cancelled

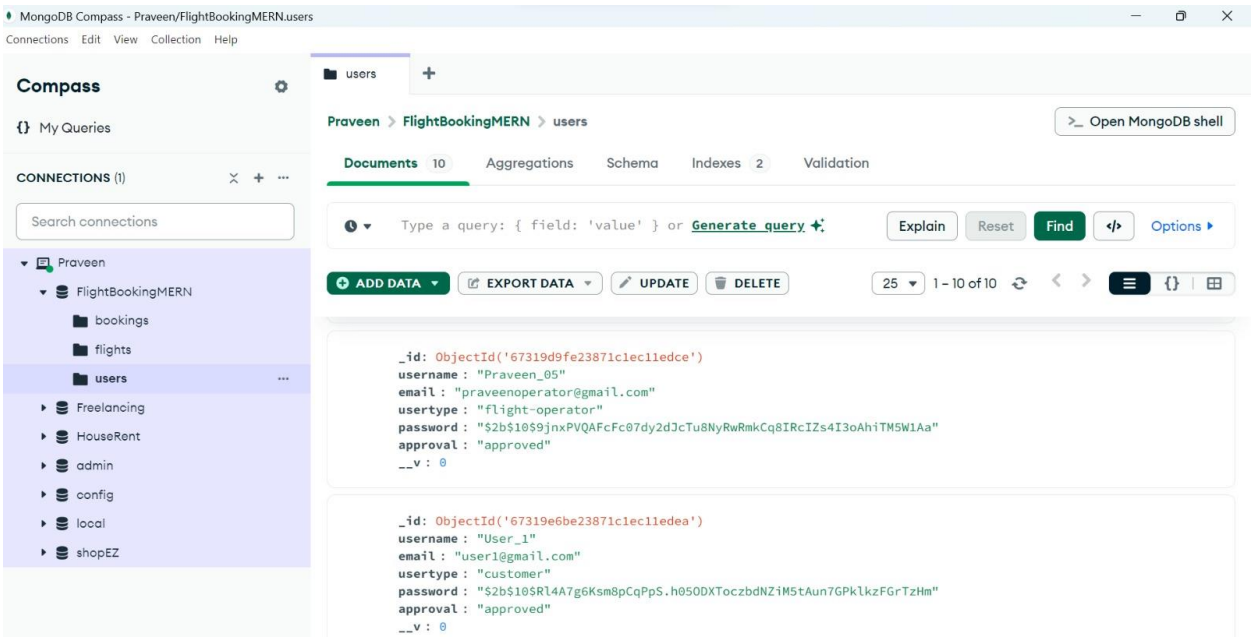

Tata Air Flights (Operator)
Home Bookings Flights Add Flight Logout
All Flights
_id: 6731a346e23871c1ec11ee3d
Flight Id: FD8367 Flight name: Operator_2
Starting station: Kolkata Departure time: 11:54
Destination: Chennai Arrival time: 12:54
Base price: 1500 Total seats: 100
Edit details
_id: 6731a55806155a38d07b1355
Flight Id: FD8223 Flight name: Operator_2
Starting station: Pune Departure time: 11:03
Destination: Chennai Arrival time: 12:45
Base price: 2000 Total seats: 140
Edit details
_id: 6731b1ce21ddc09b24394d5f
Flight Id: FD8479 Flight name: Operator_2
Starting station: Chennai Departure time: 12:56
Destination: Banglore Arrival time: 13:58
Base price: 2300 Total seats: 50
Edit details


Tata Air Flights (Operator)
Home
Bookings
Flights
Add Flight
Logout
Add new Flight
Flight Name
Operator_2
Flight Id
FD8475
Departure City
Bhopal
Departure Time
07:00
Destination City
Jaipur
Arrival time
11:00
Total seats
50
Base price
5000
Add now


MongoDB Compass - Praveen/FlightBookingMERN
Connections
Edit
View
Help
Compass
My Queries
CONNECTIONS (1)
Search connections
Praveen
FlightBookingMERN
bookings
flights
users
Freelancing
HouseRent
admin
config
local
shopEZ
FlightBookingMERN
Praveen > FlightBookingMERN
Open MongoDB shell
Create collection
Refresh
Sort by
Collection Name
View
bookings
Storage size: 20.48 kB
Documents: 5
Avg. document size: 499.00 B
Indexes: 1
Total index size: 36.86 kB
flights
Storage size: 20.48 kB
Documents: 10
Avg. document size: 202.00 B
Indexes: 1
Total index size: 36.86 kB
users
Storage size: 20.48 kB
Documents: 10
Avg. document size: 211.00 B
Indexes: 2
Total index size: 73.73 kB


MongoDB Compass - Praveen/FlightBookingMERN.bookings
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (1)
Search connections
Praveen
FlightBookingMERN
bookings
flights
users
Freelancing
HouseRent
admin
config
local
shopEZ
bookings
Praveen > FlightBookingMERN > bookings
Open MongoDB shell
Documents 5
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 5 of 5
_id: ObjectId('6731ab5d5e7c883a768168f8')
user : ObjectId('67319e6be23871c1ec11edea')
flight : ObjectId('6731a10be23871c1ec11ee12')
flightName : "Praveen_05"
flightId : "FD8475"
departure : "Chennai"
destination : "Banglore"
email : "gpkpraveenkumar143@gmail.com"
mobile : "902565207"
seats : "A-1"
passengers : Array (1)
totalPrice : 8000
journeyDate : 2024-11-29T00:00:00.000+00:00
journeyTime : "11:00"
seatClass : "first-class"
bookingStatus : "cancelled"
bookingDate : 2024-11-11T06:59:41.295+00:00
__v : 0


MongoDB Compass - Praveen/FlightBookingMERN.flights
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (1)
Search connections
Praveen
FlightBookingMERN
bookings
flights
users
Freelancing
HouseRent
admin
config
local
shopEZ
flights
Praveen > FlightBookingMERN > flights
Open MongoDB shell
Documents 10
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 10 of 10
_id: ObjectId('6731a10be23871c1ec11ee12')
flightName : "Praveen_05"
flightId : "FD8475"
origin : "Chennai"
destination : "Banglore"
departureTime : "11:00"
arrivalTime : "11:50"
basePrice : 2000
totalSeats : 50
__v : 0
_id: ObjectId('6731a19fe23871c1ec11ee1a')
flightName : "Praveen_05"
flightId : "FD8476"
origin : "Delhi"
destination : "Chennai"
departureTime : "12:09"



10.CONCLUSION:

The conclusion of this project highlights the successful development of a full-stack flight booking application. This application integrates a React front end with a Node.js and Express back end, connected to a MongoDB database. Through a modular code structure, secure authentication, and efficient data handling, this project delivers a reliable platform that allows users to search for flights, book tickets, and manage reservations. By utilizing tools and libraries such as Axios for HTTP requests, Multer for managing file uploads, JWT for secure user authentication, and Mongoose for database interactions, the application provides a smooth and user-friendly experience.

In addition, essential security measures like bcryptjs for password hashing and CORS for cross-origin resource sharing enhance the platform's security and data integrity. Overall, this project demonstrates an effective implementation of a scalable and interactive flight booking system using modern web development practices and technologies. It lays a solid foundation for further improvements, including expanding features, refining the user interface, or optimizing performance for handling high traffic in a production environment.

Github Project Link: https://github.com/Praveen8603/Flight_booking_application.git