## Flight Booking Application Documentation

### Abstract

The flight booking application enables users to book flight tickets through an automated web interface. Users can log in with a username and password, search for flights, select their preferred flights, choose seats, and specify the number of tickets. The system displays the booking details and provides a printable form or a digital confirmation for the user to present at the airport. This project aims to provide a convenient and efficient way for customers to purchase flight tickets online, minimizing the need for manual processing by airline staff.

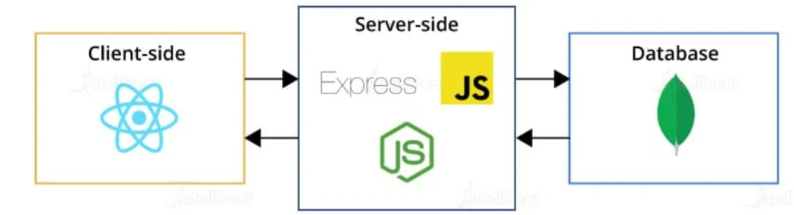
### **Introduction**

The flight booking application streamlines the process of booking airline tickets by offering an easy-to-use online plattform. Users can explore available flights, choose their desired itineraries, and secure their reservations from the comfort of their homes. Utilizing a robust client-server framework built on Express.js and MongoDB, the application ensures smooth data handling and real-time updates. Key functionalities include user authentication, flight search, seat selection, booking management, and real-time seat availability. This documentation provides a comprehensive guide to setting up, developing, and understanding the technical architecture and features of the application.

### **Scenario**

Imagine Sarah, a frequent traveler, who needs to book a last-minute flight for an urgent business meeting. With a busy schedule, she prefers the convenience of booking her flight tickets online. Accessing the FlightMate app, Sarah easily navigates through the intuitive interface, searches for available flights, and selects the most suitable option. The app provides a visual seating chart, allowing Sarah to choose her preferred seat. After finalizing her selection, she completes the booking with a secure payment portal and promptly receives a booking confirmation with all the necessary details for her upcoming flight. As the departure date approaches, Sarah effortlessly retrieves her booking information via the app, ensuring a smooth and stress-free travel experience.

### Technical Architecture



The flight booking application uses a client-server architecture, with a clear division between frontend and backend components:

#### Frontend (Client)

* **UI and Interaction**: The frontend handles the user interface and interactions, providing a responsive and intuitive experience.
* **Axios**: Utilizes the Axios library for making HTTP requests to the backend.
* **Functionality**: Manages user inputs, displays flight information, and communicates with the server via RESTful APIs.

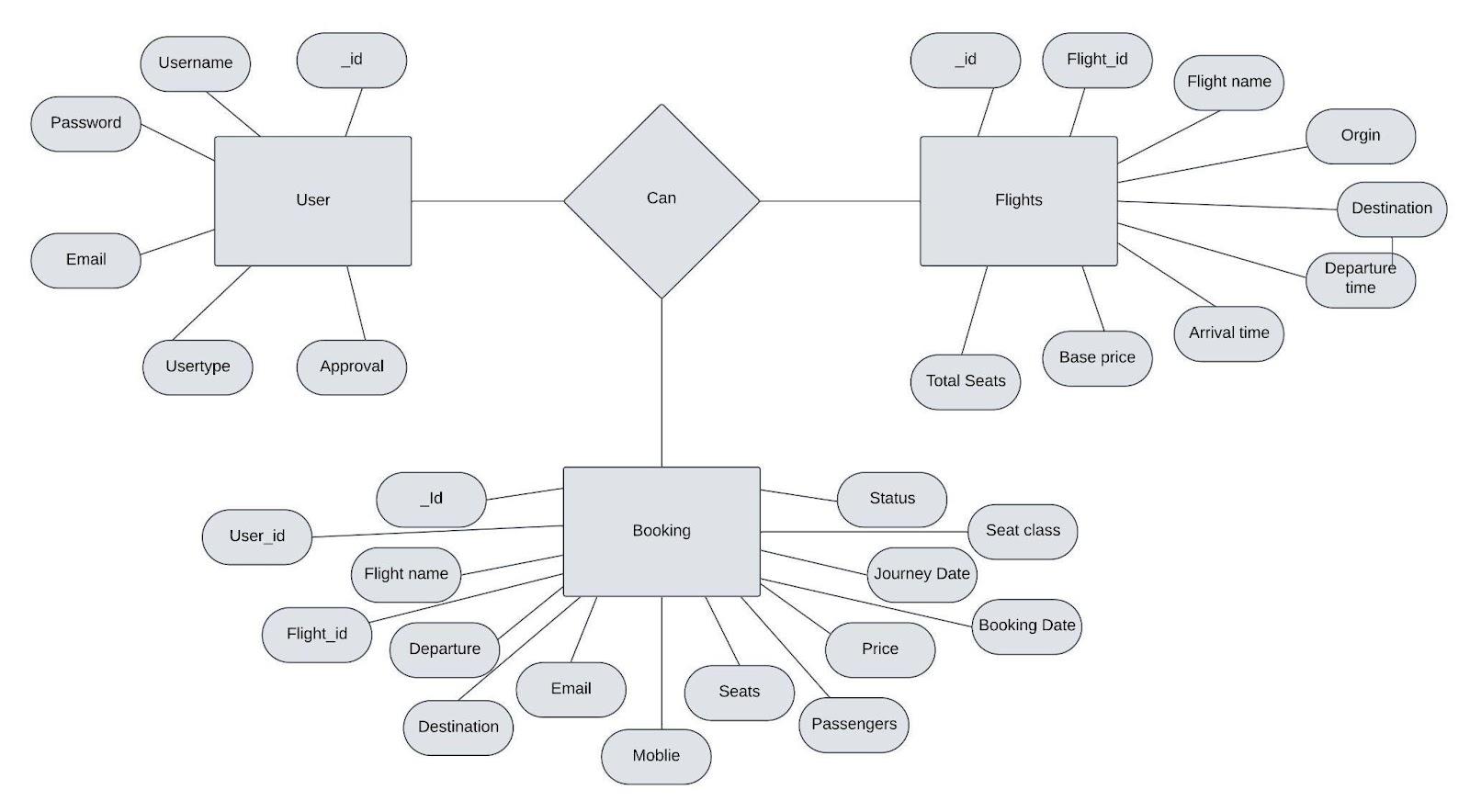
#### Backend (Server)

* **Express.js**: The server side uses the Express.js framework to manage routing, middleware, and HTTP request handling.
* **RESTful APIs**: Exposes APIs for operations such as retrieving flight details, processing bookings, and managing user profiles.

#### Database

* **MongoDB**: A NoSQL database used to store and manage data such as user profiles, flight information, and bookings.
* **Flexibility and Scalability**: Supports dynamic schema and horizontal scaling, allowing for efficient data handling and growth.

**ER DIAGRAM**



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

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

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

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

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

**Key Features**

### 1. User Authentication and Profiles:

* Secure user login and registration.
* User profiles to manage personal details, preferences, and booking history.

### 2. Flight Search and Booking:

* Search functionality to filter flights by destination, date, price, airline, etc.
* Real-time flight availability and pricing updates.
* Detailed flight information, including duration, stops, and layover times.
* Seat selection with a visual seat map.

### 3. Booking Management:

* Easy booking and payment process with multiple payment options.
* Booking confirmation via email or SMS.
* Option to view, modify, or cancel bookings.

### 4. Real-time Updates:

* Live updates on flight status, delays, and cancellations.
* Notifications for changes in flight schedules.

### 5. Admin and Operator Panel:

* Admin dashboard to manage flights, bookings, users, and promotions.
* Approval and management of airline operators and their flights.

### 6. Promotions and Discounts:

* Integration of promo codes and special discounts for users.
* Loyalty programs and reward points system.

### 7. Customer Support Integration:

* Integrated chat support or contact options for customer service.
* FAQ section to help users with common queries.

### 8. Security Features:

* Data encryption for secure transactions and user data protection.
* Two-factor authentication (2FA) for enhanced account security.

### 9. Scalability and Performance:

* Optimized for high performance to handle large volumes of traffic.
* Scalable architecture to support future growth and additional features.

### 10. Mobile Responsiveness:

* Fully responsive design to provide a seamless experience on mobile devices and tablets.

### 11. Multilingual and Multi-currency Support:

* Support for multiple languages and currencies to cater to a global audience.

### 12. Analytics and Reporting:

* Tools to generate reports on bookings, user activity, and revenue.
* Insightful analytics to track and improve system performance.

### 13. Integration with Other Services:

* Integration with third-party services for travel insurance, car rentals, and hotel bookings.
* API integration for airline schedules and availability.

### 14. Customizable Notifications:

* Customizable notifications and reminders for upcoming flights and check-in times.

These features help create a comprehensive and efficient flight booking system that meets the needs of both users and administrators.

**PRE-REQUISITES**

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

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

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

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

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

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

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

\* Open your command prompt or terminal and run the following command : npm install express

**React.js**: React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

To install React.js, a JavaScript library for building user interfaces, follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

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

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

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

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

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

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

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

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

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

**To run the existing Flight Booking App project downloaded from github:**

Follow below steps:

**Clone the repository:**

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

**Git clone:** <https://github.com/harsha-vardhan-reddy-07/Flight-Booking-App-MERN>

**Install Dependencies:**

* Navigate into the cloned repository directory:

**cd Flight-Booking-App-MERN**

* Install the required dependencies by running the following command:

**npm install**

**Start the Development Server:**

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

**npm run dev or npm run start**

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

**Access the App:**

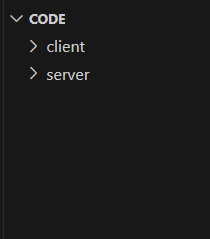
• Open your web browser and navigate to [http://localhost:3000](http://localhost:3000/)

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

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

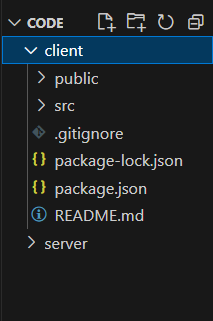
### **PROJECT STRUCTURE**

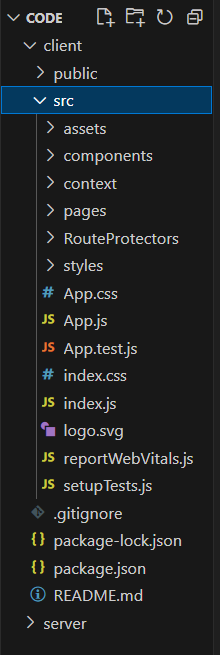
* Inside the Flight Booking app directory, we have the following folders



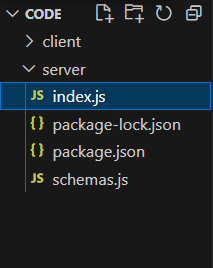
* **Client directory:**

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



* 
* **Server directory:**

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



### **APPLICATION FLOW**

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

### **Project setup and configuration****s:**

**Folder setup:**

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

* client
* Server

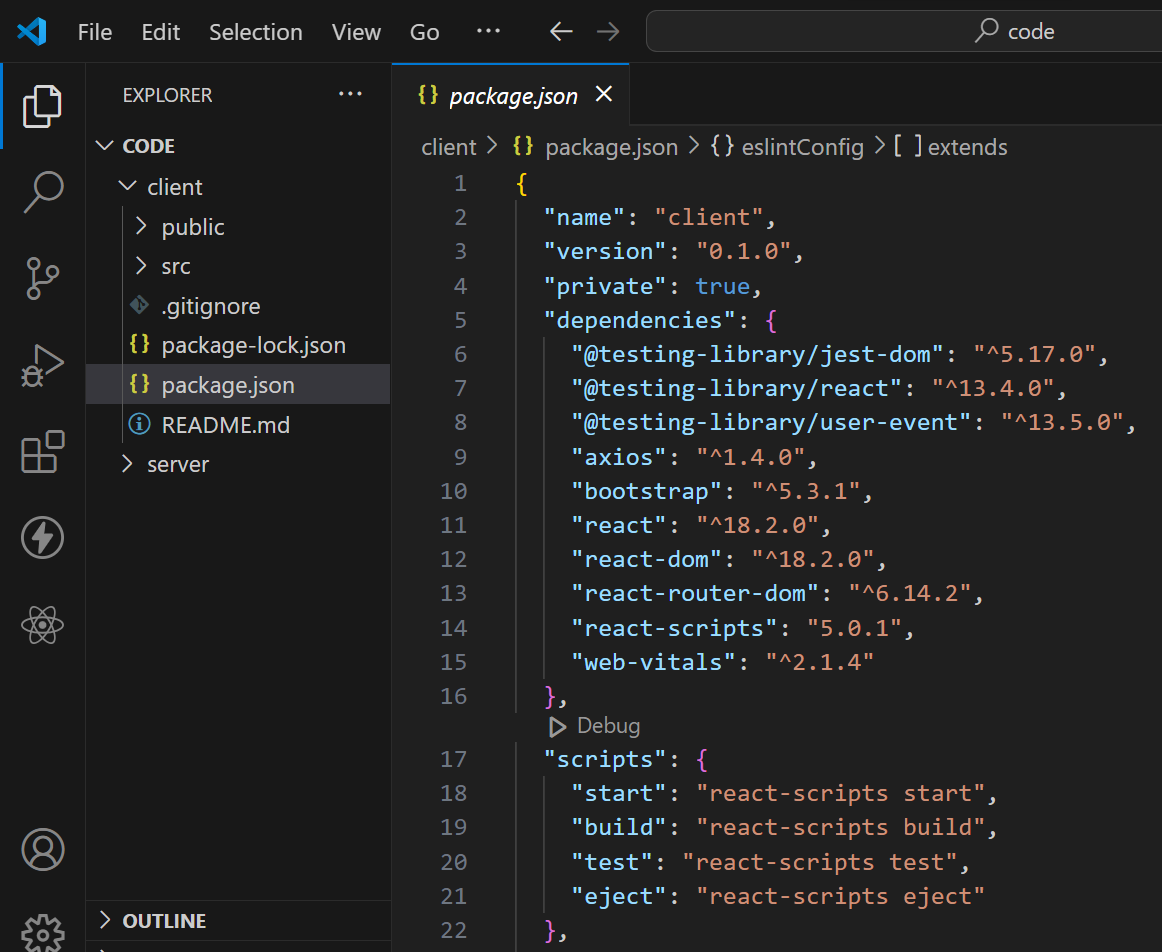
**Installation of required tools:**

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

For frontend, we use:

* React Js
* Bootstrap
* Axios

After instaling all the requird libraries, we’ll be seeing the package.json file similar to the one below.

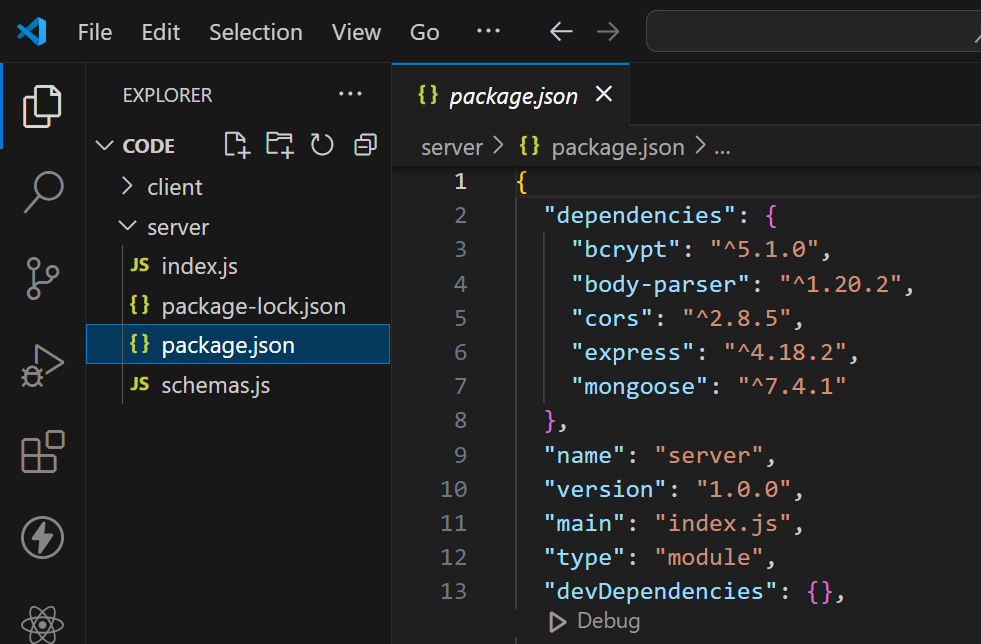


Now, open the backend folder to install all the necessary tools that we use in the backend.

For backend, we use:

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

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



### **Backend Development**

**1. Database Configuration:**

* Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.
* Create a database and define the necessary collections for flights, users, bookings, and other relevant data.

1. **Create Express.js Server:**

* Set up an Express.js server to handle HTTP requests and serve API endpoints.
* Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

1. **Define API Routes:**

* Create separate route files for different API functionalities such as flights, users, bookings, and authentication.
* Define the necessary routes for listing flights, handling user registration and login managing bookings, etc.
* Implement route handlers using Express.js to handle requests and interact with the database.

1. **Implement Data Models:**

* Define Mongoose schemas for the different data entities like flights, users, and bookings.
* Create corresponding Mongoose models to interact with the MongoDB database. Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

1. **User Authentication:**

* Create routes and middleware for user registration, login, and logout.
* Set up authentication middleware to protect routes that require user authentication

.

1. **Handle new Flights and Bookings:**

* Create routes and controllers to handle new flight listings, including fetching flight data from the database and sending it as a response.
* Implement booking functionality by creating routes and controllers to handle booking requests, including validation and database updates.

1. **Admin Functionality:**

* Implement routes and controllers specific to admin functionalities such as adding flights, managing user bookings, etc.
* Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

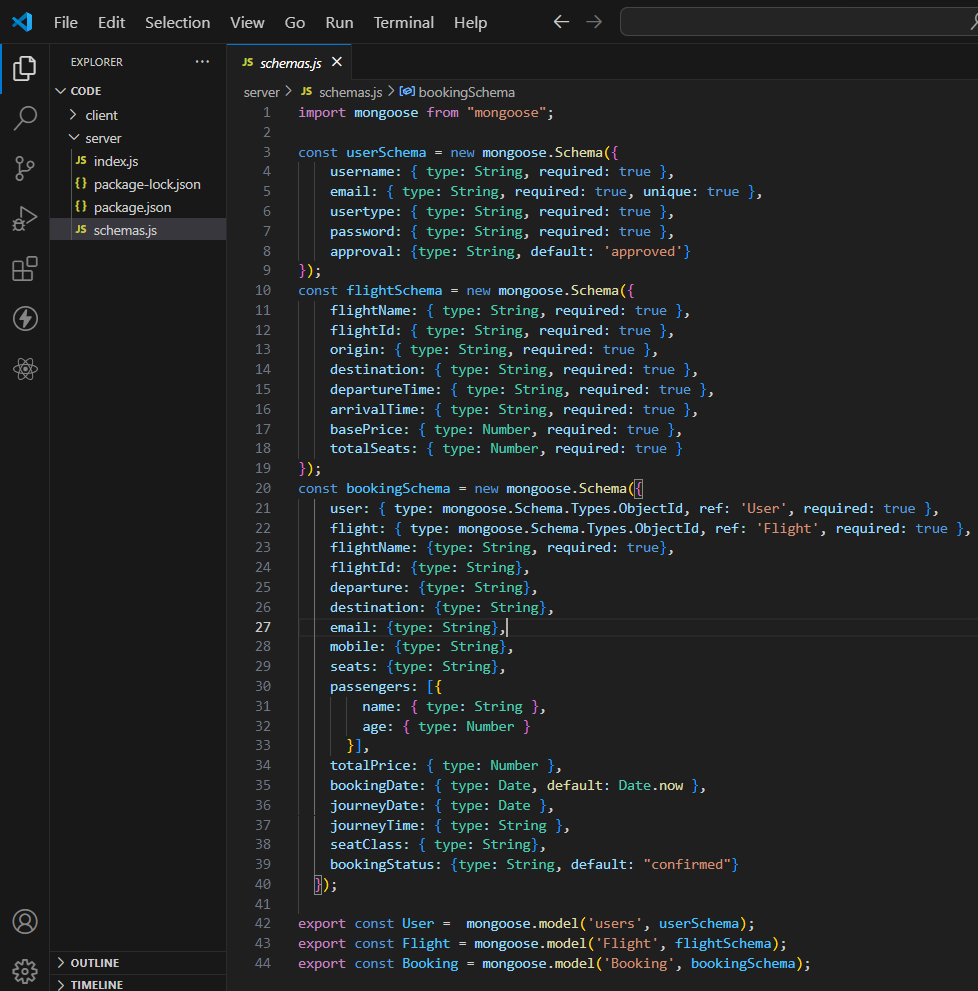
1. **Error Handling:**

* Implement error handling middleware to catch and handle any errors that occur during the API requests.
* Return appropriate error responses with relevant error messages and HTTP status codes.

### **Database development**

**Configure schema**

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



* **Connect database to backend**

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



### **Frontend development**

1. **Login/Register**

* Create a Component which contains a form for taking the username and password.
* If the given inputs matches the data of user or admin or flight operator then navigate it to their respective home page

1. **Flight Booking (User):**

* In the frontend, we implemented all the booking code in a modal. Initially, we need to implement flight searching feature with inputs of Departure city, Destination, etc.,
* Flight Searching code: With the given inputs, we need to fetch the available flights. With each flight, we add a button to book the flight, which redirects to the flight-Booking page.

1. **Fetching user bookings:**

* In the bookings page, along with displaying the past bookings, we will also provide an option to cancel that booking.

1. **Add new flight(Admin):**

* Now, in the admin dashboard, we provide functionality to add new flights.
* We create a html form with required inputs for the new flight and then send an httprequest to the server to add it to the database.

1. **Update Flight:**

* Here, in the admin dashboard, we will update the flight details in case if we want to make any edits to it
* Along with this, implement additional features to view all flights, bookings, and users in the admin dashboard.

**Web development:**

1. Setup React Application:

• Create a React app in the client folder.

• Install required libraries

• Create required pages and components and add routes.

2.Design UI components**:**

• Create Components.

• Implement layout and styling.

• Add navigation.

3.Implement frontend logic:

• Integration with API endpoints.

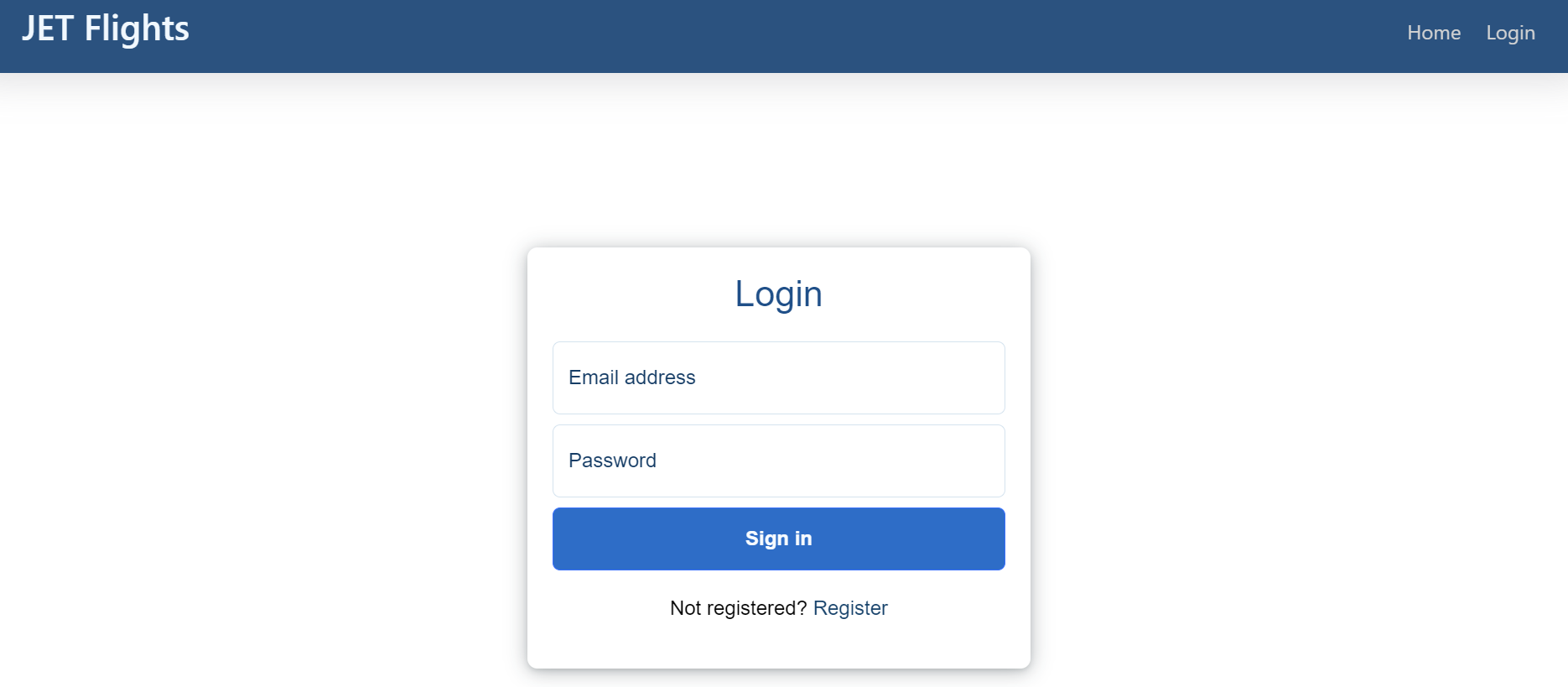
• Implement data binding.

**Reference Article Link:**

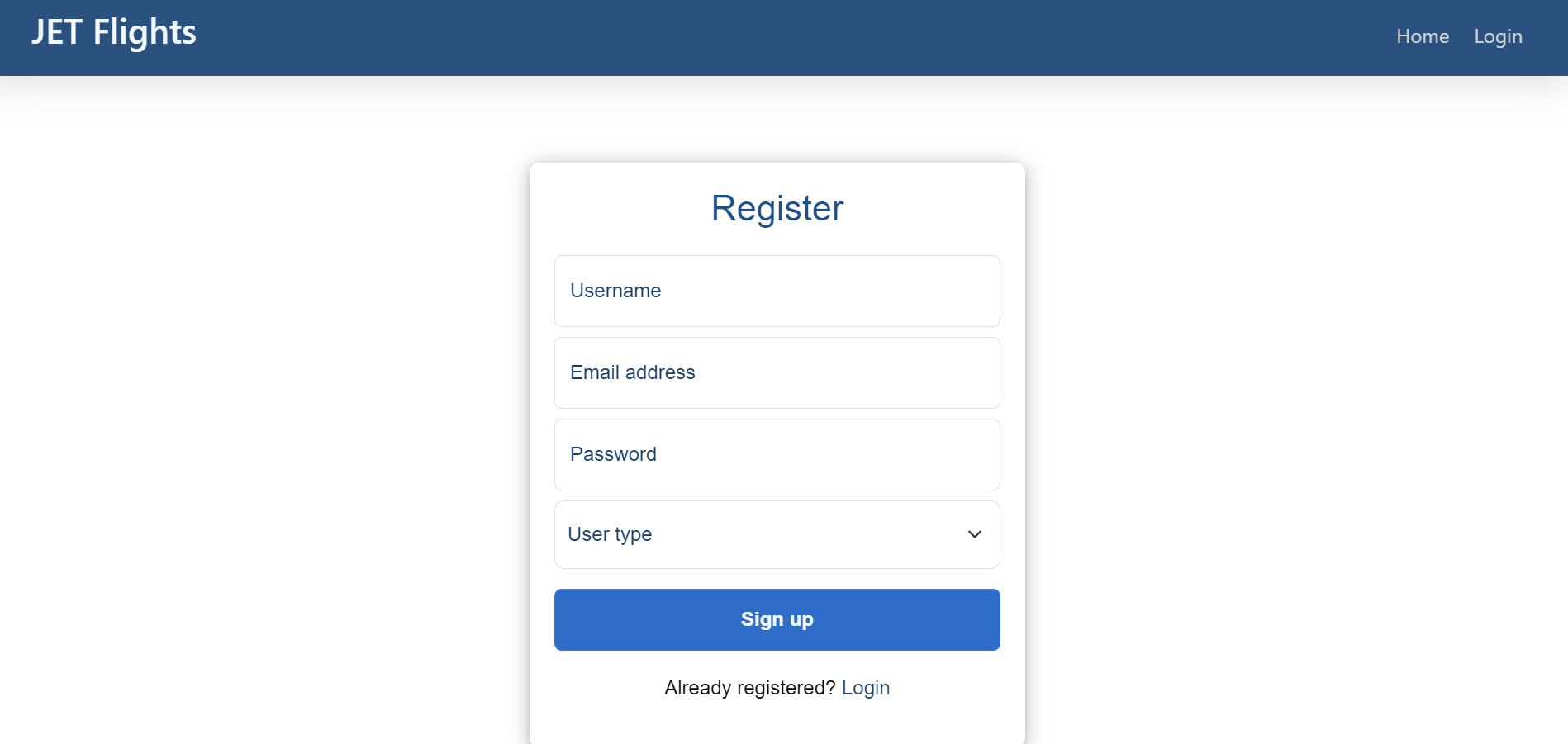
https://www.w3schools.com/react/react\_getstarted.asp

**Output Screenshots**

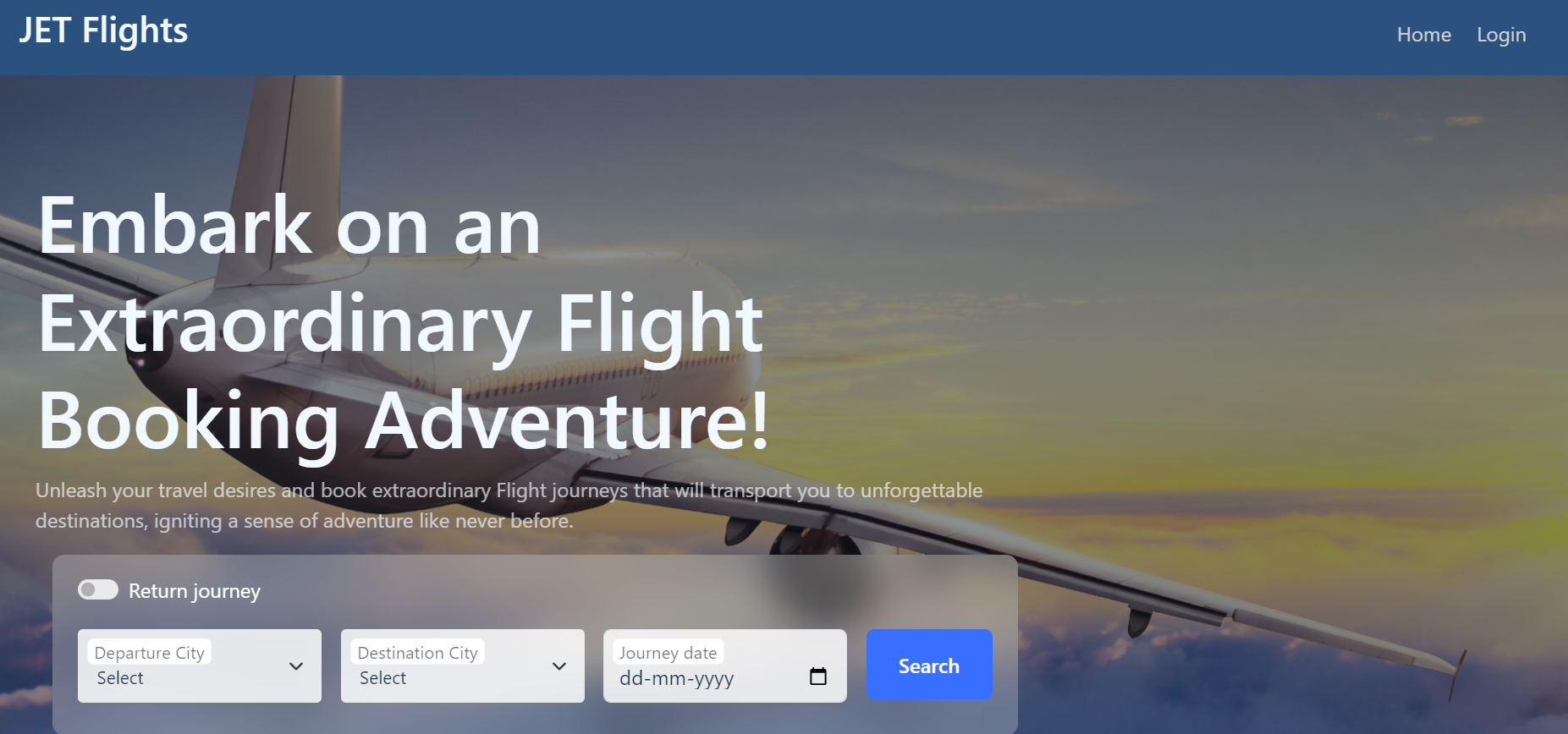
Login page



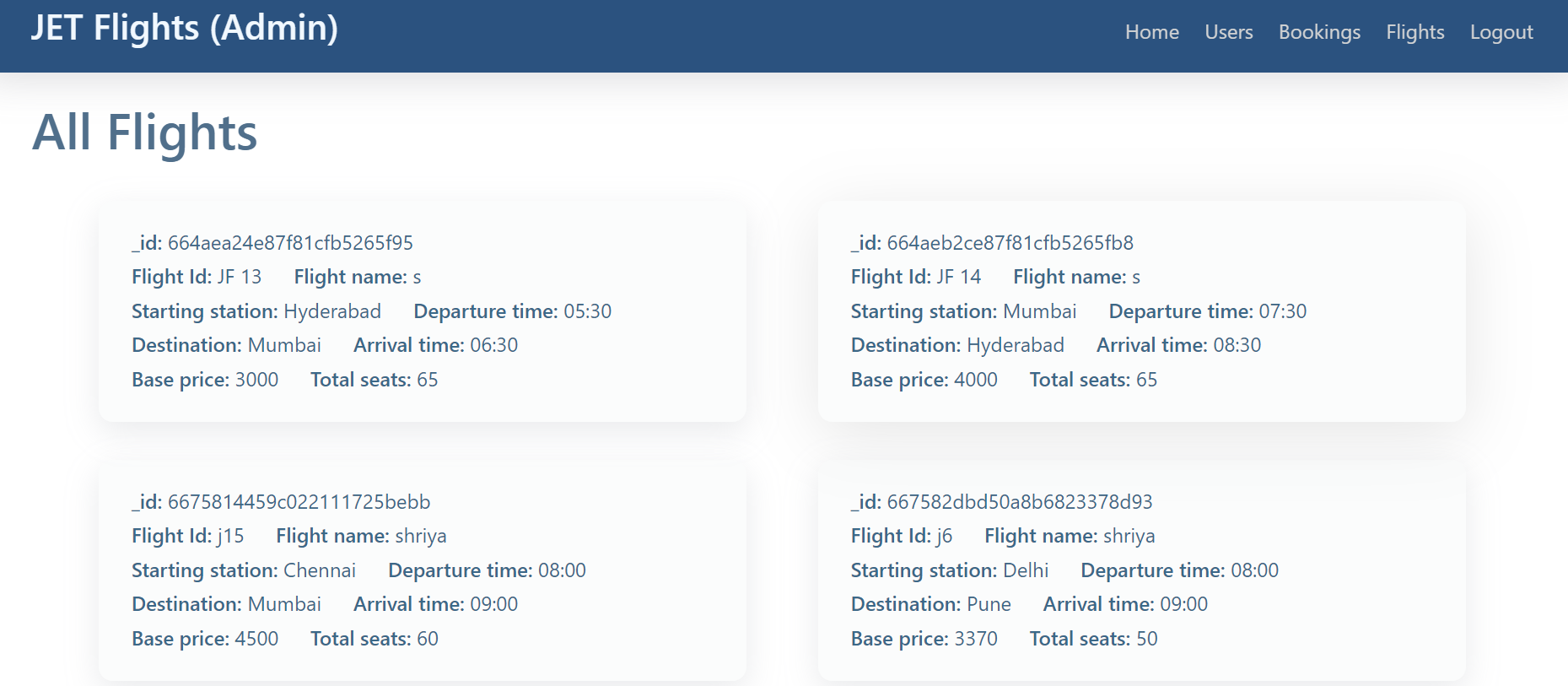
Register page



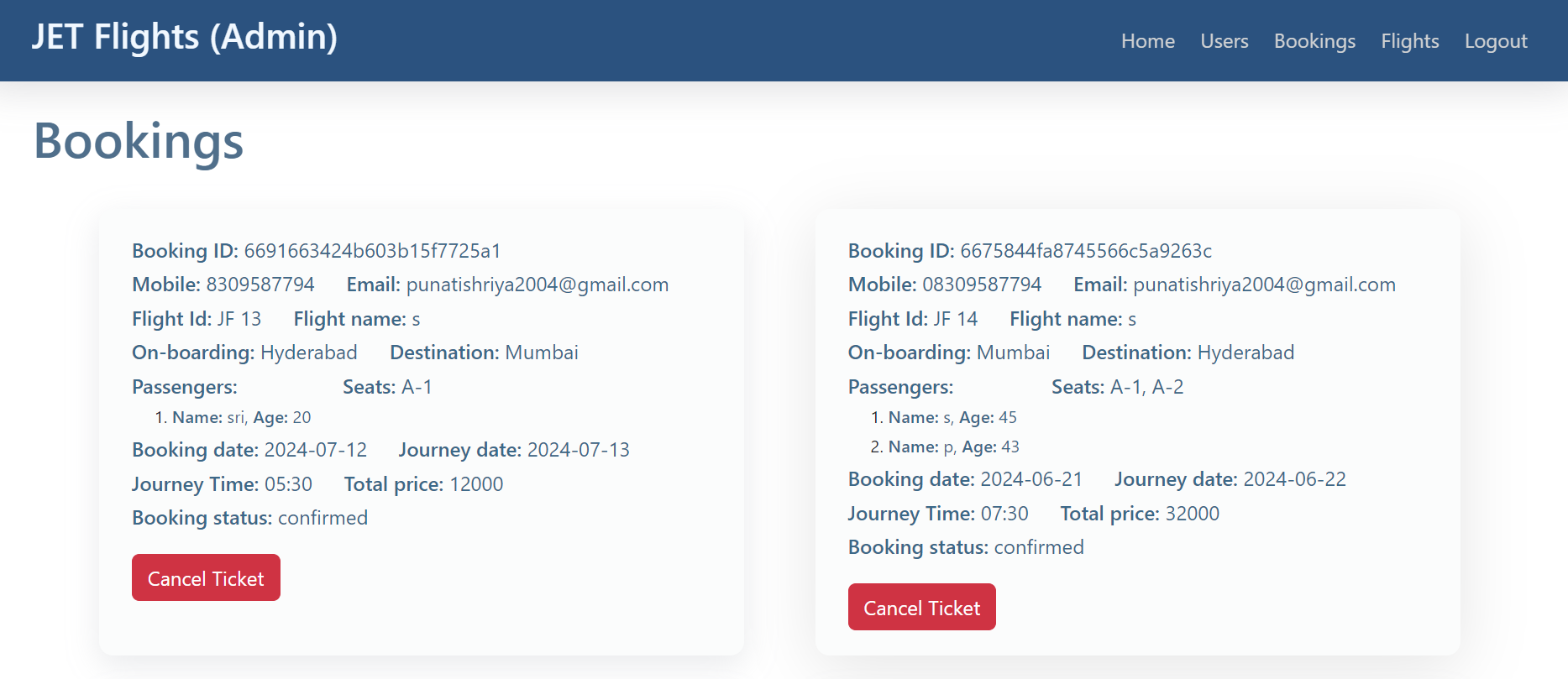
Home page



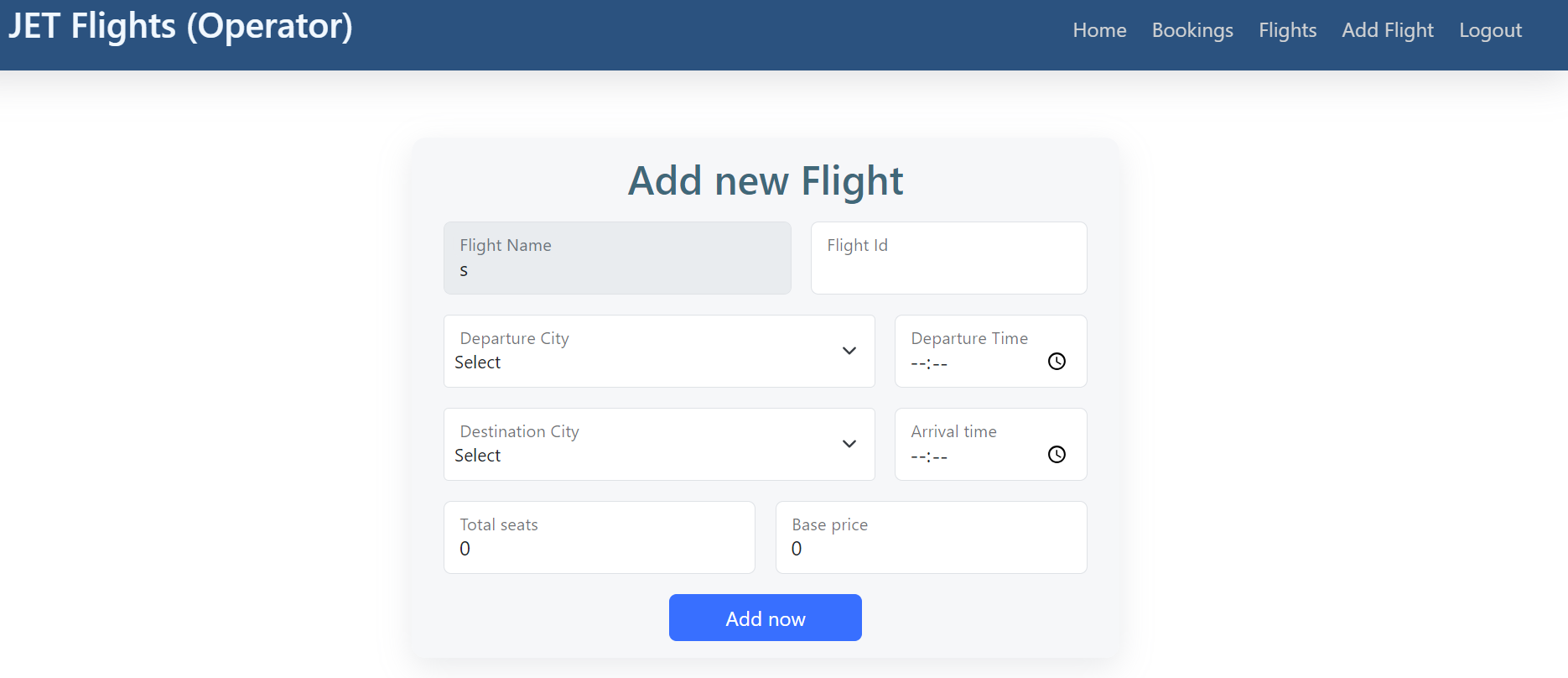
All Flights



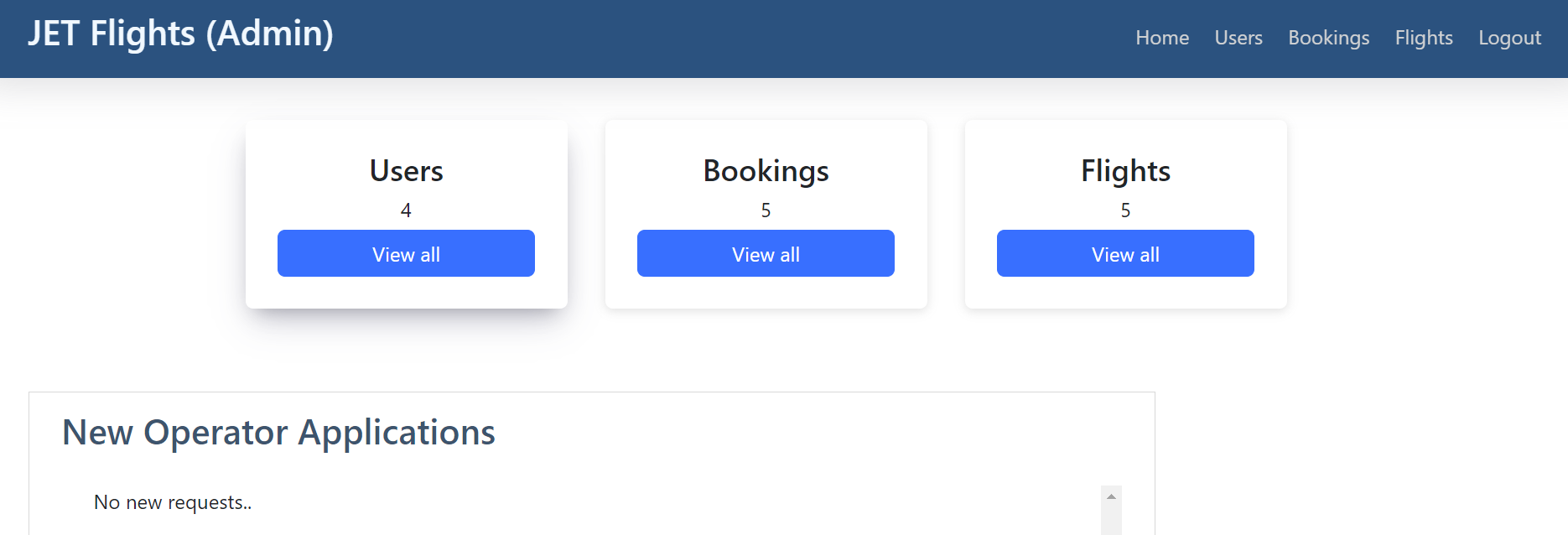
Bookings



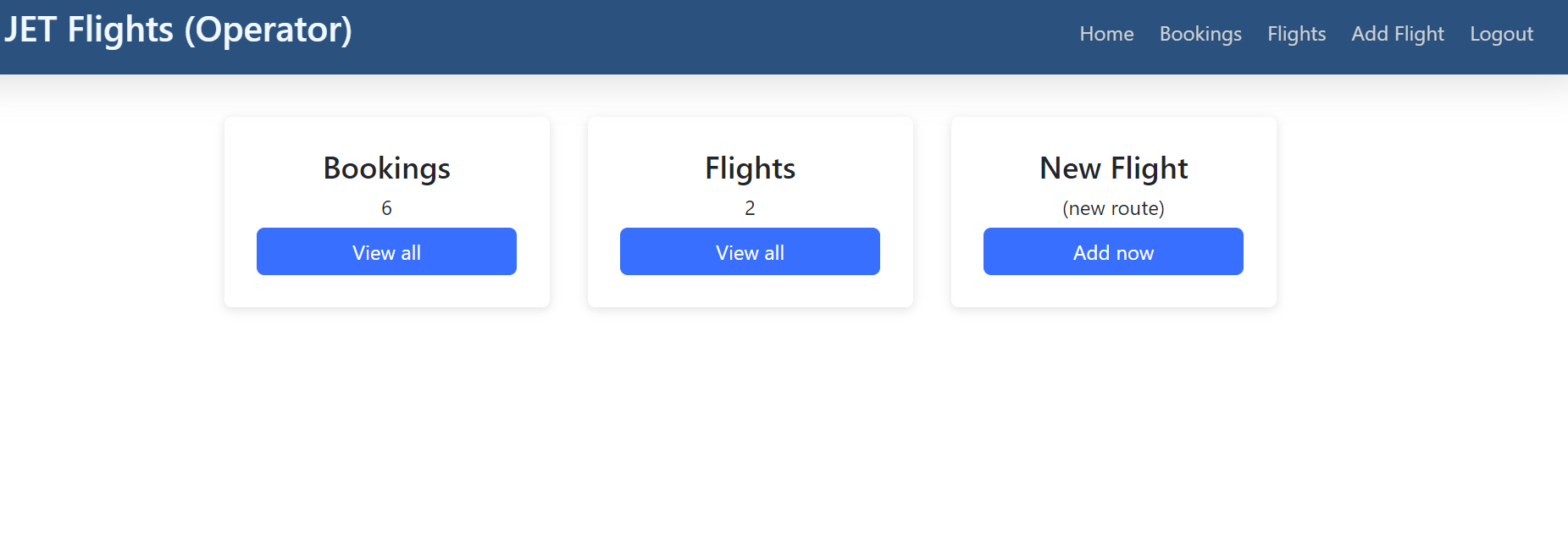
Add New Flight



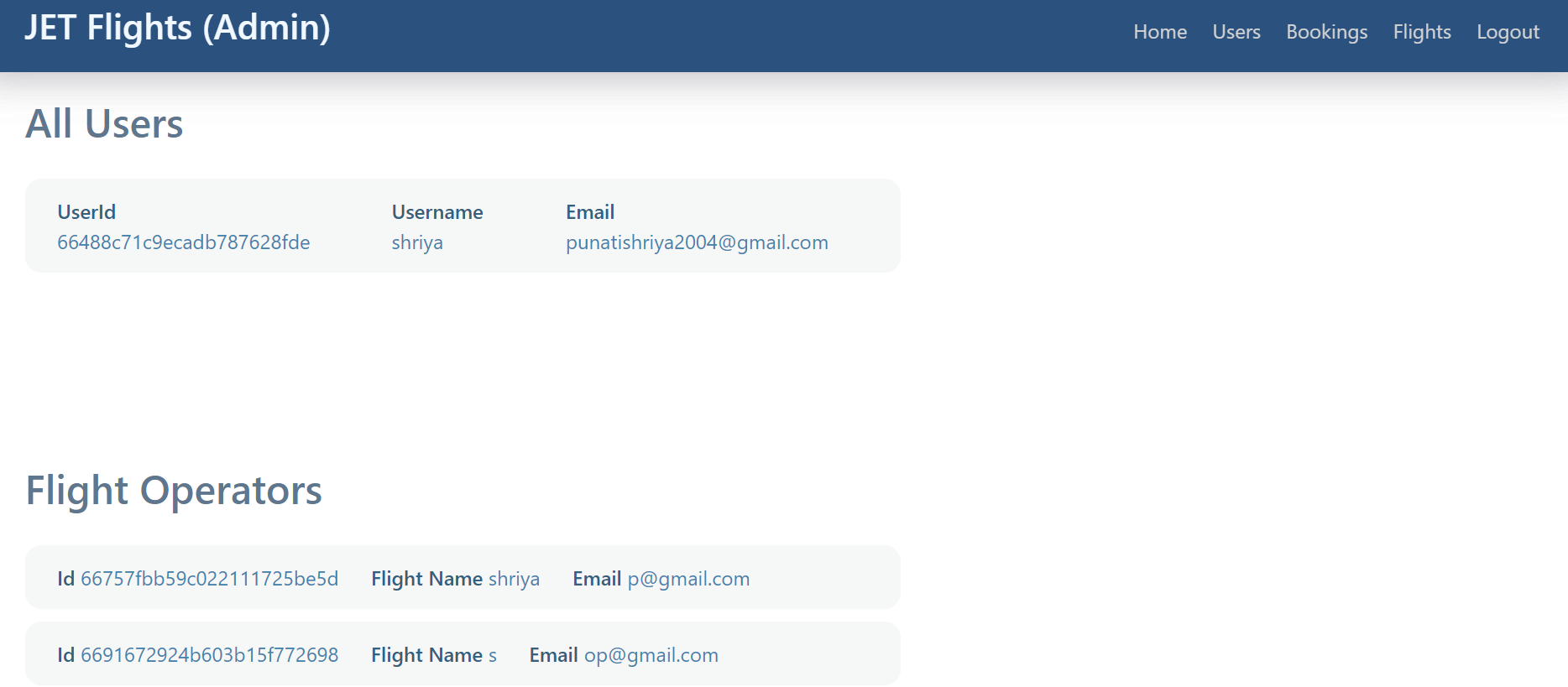
Admin Page



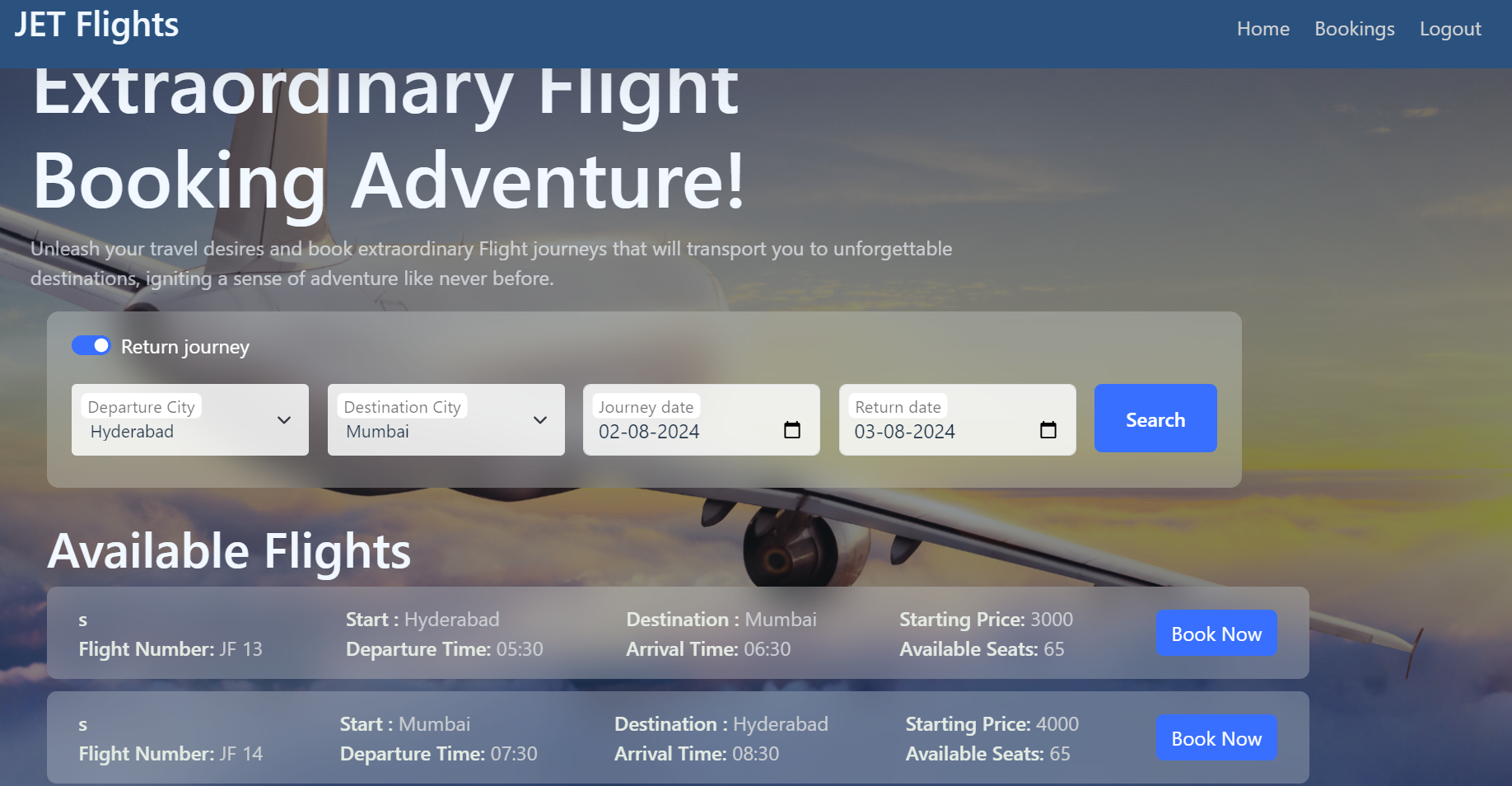
Operator Page



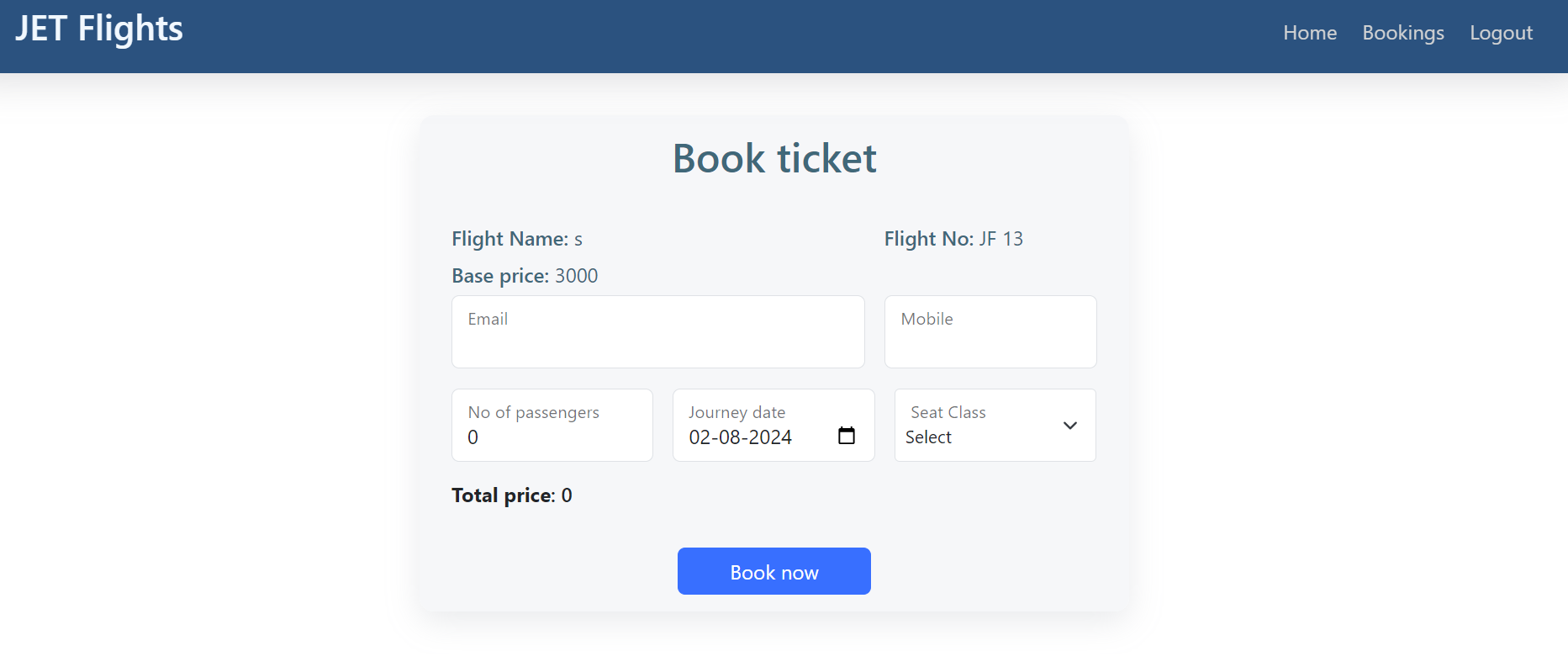
Users



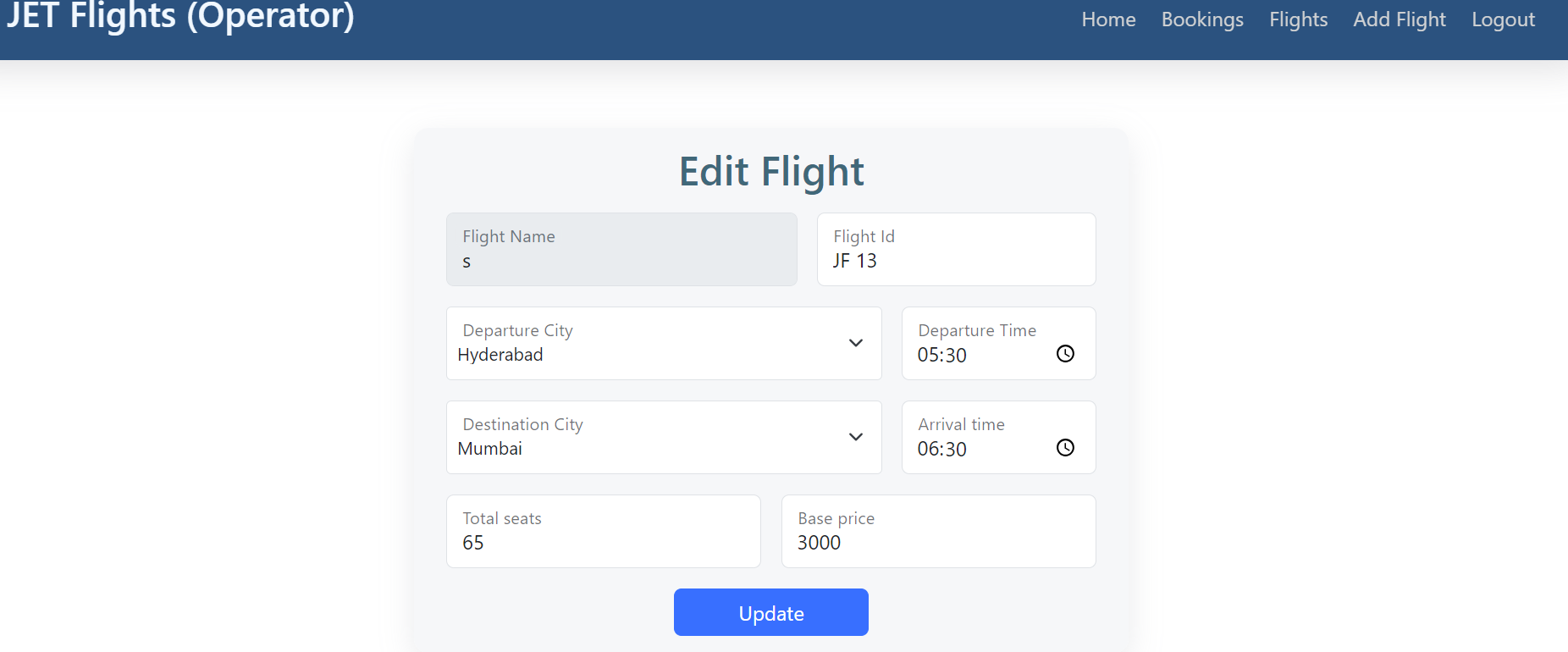
Available Flights



Book Ticket



Edit Flight Details



Demo Link

<https://drive.google.com/file/d/1NViO9BFnQPFjmAKHSJj6vsrliUmvA-Vc/view?usp=drive_link>

**Conclusion**

**In conclusion, the flight booking system transforms the air travel experience by offering a seamless and personalized way for users to book flights online. With its user-friendly interface, travelers can effortlessly search for flights, select preferred schedules, and book tickets from the comfort of their homes. The system's robust client-server architecture, built on Express.js and MongoDB, ensures efficient data handling and real-time updates, greatly enhancing the overall user experience.**

**By integrating essential functionalities such as user authentication, flight management, and real-time seat availability tracking, the system provides convenience and satisfaction to users. The automated nature of the platform reduces the workload on airline staff, as booking and ticket management are handled entirely through the system. This scalability and adaptability make the flight booking system suitable for a wide range of applications, promising to redefine the traditional flight booking process.**

**Overall, the flight booking system sets a new standard for online travel platforms, showcasing the potential of technology to innovate and improve the way people plan and manage their journeys.**