

# FLIGHT BOOKING APPLICATION

## INTRODUCTION:

A flight booking application is a versatile and user-centric software platform designed to simplify the process of planning and booking air travel. It empowers individuals to seamlessly search for available flights, select and book their preferred options, and compare routes and prices to find the most cost-effective solutions for their journey. With its robust features, the application ensures that users can make informed decisions to optimise their travel plans.

The primary purpose of such an application is to enhance the overall travel experience by offering convenience and flexibility. Users can explore a wide range of flight options from various airlines, filter results based on their preferences (such as price, duration, or layovers), and finalise bookings with ease. Additionally, the application can incorporate advanced search algorithms to identify the most affordable flight routes while considering user satisfaction.

Beyond the booking process, these applications aim to promote a healthier and more comfortable journey. This can include features like personalised recommendations, real-time notifications about flight status, and customer support to address queries or concerns promptly. By streamlining the planning phase and offering comprehensive support, a flight booking application ensures that travellers can focus on enjoying their journey without the stress of logistics.

## KEY FEATURES:

### User Registration

- This feature allows users to create personalised accounts, ensuring their data and booking history are securely saved for future reference. It simplifies repeated bookings and provides a seamless experience by maintaining continuity across sessions.

### Tracker and Notifier

- Users can track real-time updates about their flights, including schedule changes, delays, or cancellations. Notifications ensure that users stay informed about aviation events and their flight details, enhancing planning and reducing uncertainty.

### Robust Environment

- The application fosters a user-friendly interface that allows individuals to resolve queries effortlessly. With intuitive navigation and clear instructions, users can access features and services without any complications.

### Administration Management

- A strong administrative backend manages user accounts efficiently, ensuring smooth application operation. This feature also streamlines processes such as booking confirmations, payment handling, and user inquiries while maintaining system robustness and reliability.

### Enhanced Security

- Advanced security measures protect users' sensitive data, including personal details and payment credentials. The application employs encryption, multi-factor authentication, and protection against cyber threats like hacking and malware to safeguard information.

## BASIC DESCRIPTION:

A flight booking application is an user friendly application that has been designed to search,see and the book the required flight for the passenger in order to travel his journey in an safer and enjoyable manner.an real time example has been provided for the better understanding

## REAL TIME EXAMPLE:

Rahul,a wanderer and explorer who is willing to go to Auckland for his vacation from his home town India.The following are the tasks that have been done by rahul.

### 1.User registration and login:

Rahul visits the flight booking application and then creates an account by giving his credentials and signing up.He fills out the name,age,account etc to create the user account.After completing the required process he may receive the required verification mail through his mail id.Then he logs into the system by using his email and password.

### 2.Searching:

After creating his account he may go to the search bar and then put his details such as name,Date,age,country and destination etc.After he enters these things the application may show the actual flights that are scheduled for his destination on that particular time.

### 3.Tracking and notification:

After he booked his flight to auckland he may receive the live notification and tracking of the booked flight through his mail.So that he may know the actual scenario of his flight and he will reach the place to catch the flight without any complications.

### 4.User friendly interaction:

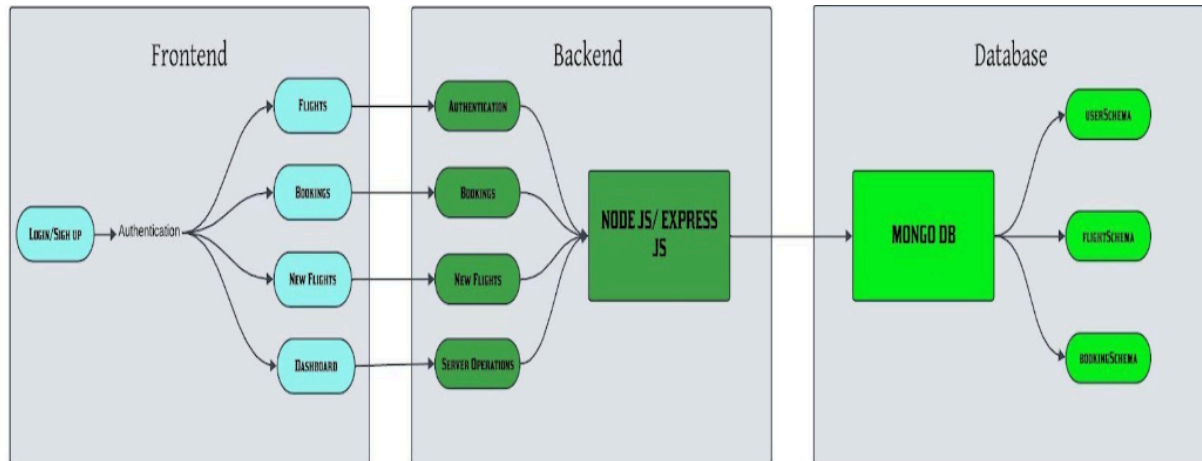
Rahul can be able to ask the questions regarding the flights and his queries can be solved in a short period of time.

### 5.Robust security:

This is a secured application where the details and credentials which have been entered by the rahul will not be hacked by the other third parties at any cost.

## TECHNICAL ARCHITECTURE:

The diagrammatic representation of the flight booking app is given as follows,

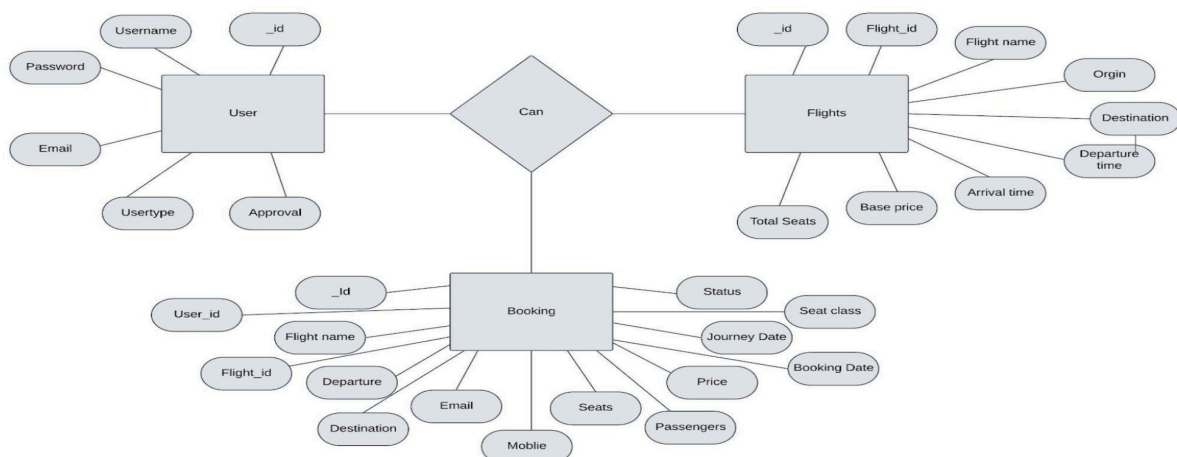


Here you can see the technical architecture of the flight booking application.

- The front end of the application consists of the login/signup page and then it is followed by the flights, bookings and new flights and a dashboard.
- The backend of the application has been created using the node js/express js and it consists of the authentications, bookings, flights, new flights and the server operator.
- The database has been created with the help of the Mongo db and it can consists of the user schema, flight schema and booking schema

## ER DIAGRAM:

The ER diagram for the flight application has been given as follows



The ER diagram is the one which shows the interaction of the client and the application

## PRE-REQUISITES:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

### Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

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

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

### Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, 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-like format. It provides high performance, horizontal scalability, and

seamless integration with Node.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/>

### 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.

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

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

Front-end Framework: Utilise Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and bootstrap.

### 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://git-scm.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>

To run the existing Video Conference 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/haseenabegum05/NM2024TMID01157-Flight-Booking-APP>

### Install Dependencies:

- Navigate into the cloned repository directory:  
cd Flight-Booking-App-MERN
- Install the required dependencies by running the following command:  
npm install

### Start the Development Server:

- To start the development server, execute the following command:  
npm run dev or npm run start
- The e-commerce app will be accessible at <http://localhost:3000> by default.  
You can change the port configuration in the .env file if needed.

### Access the App:

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

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

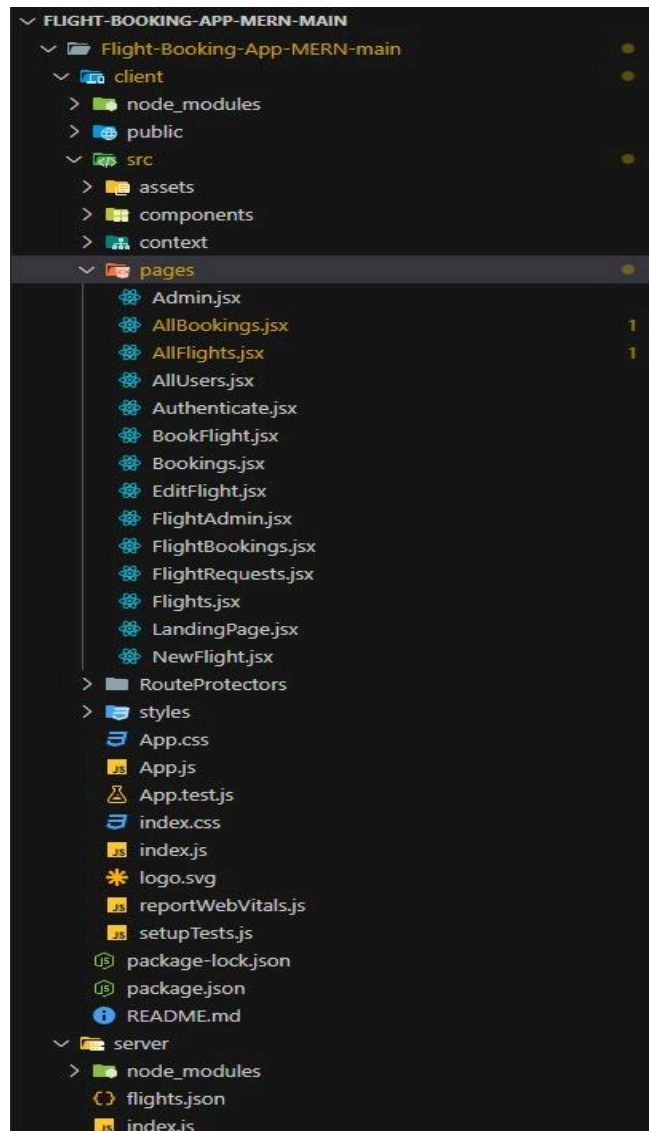
You have successfully installed and set up the flight booking app on your local machine. You can now proceed with further customization, development, and testing as needed.

## PROJECT STRUCTURE:

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

### 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 (back end) 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

### Folder setup:

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

- **client**
- **Server**

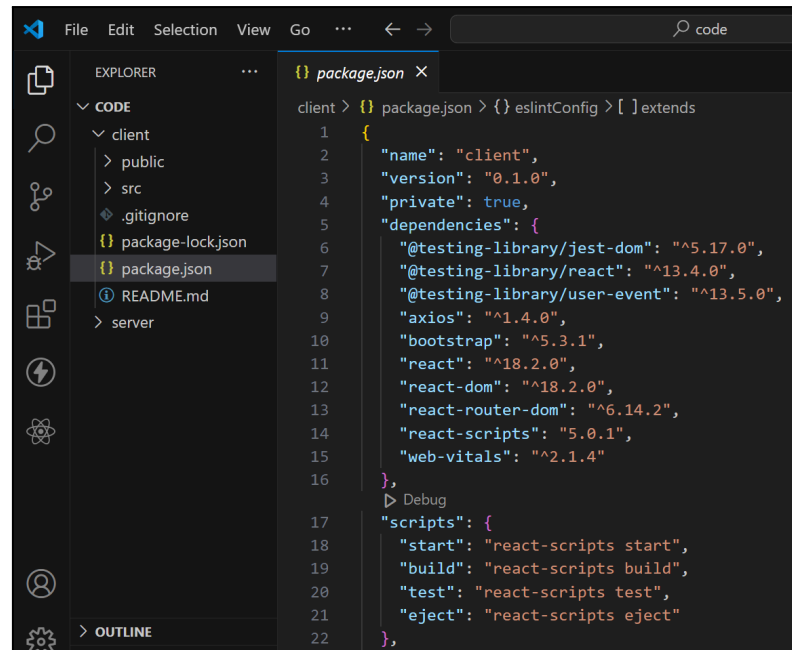
### Installation of required tools:

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

- **React Js**

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





The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'client' folder is expanded, showing files like public, src, .gitignore, package-lock.json, package.json, and README.md. The 'package.json' file is selected and its content is displayed in the main editor. The package.json file is for a client application and includes dependencies for testing libraries, axios, bootstrap, react, react-dom, react-router-dom, react-scripts, and web-vitals. It also includes a 'scripts' section with commands for start, build, test, and eject.

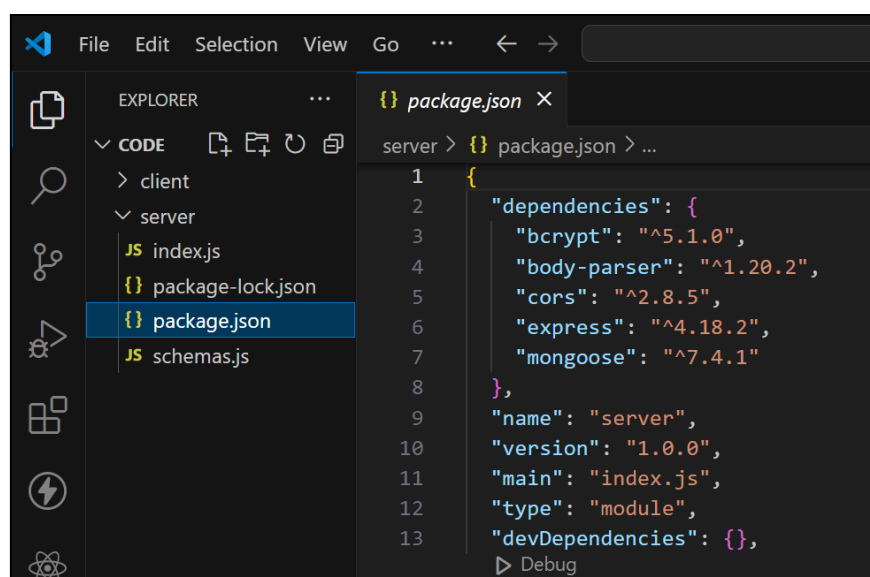
```
1 {
2   "name": "client",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.17.0",
7     "@testing-library/react": "^13.4.0",
8     "@testing-library/user-event": "^13.5.0",
9     "axios": "^1.4.0",
10    "bootstrap": "^5.3.1",
11    "react": "^18.2.0",
12    "react-dom": "^18.2.0",
13    "react-router-dom": "^6.14.2",
14    "react-scripts": "5.0.1",
15    "web-vitals": "^2.1.4"
16  },
17  "scripts": {
18    "start": "react-scripts start",
19    "build": "react-scripts build",
20    "test": "react-scripts test",
21    "eject": "react-scripts eject"
22  },
```

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.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'server' folder is expanded, showing files like index.js, package-lock.json, package.json, and schemas.js. The 'package.json' file is selected and its content is displayed in the main editor. The package.json file is for a server application and includes dependencies for bcrypt, body-parser, cors, express, and mongoose. It also includes a 'main' field pointing to index.js and a 'type' field set to 'module'.

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

## BACKEND DEVELOPMENT

### 1. Database Configuration:

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

### 2. Create Express.js Server:

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

### 3. Define API Routes:

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

### 4. Implement Data Models:

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

### 5. User Authentication:

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

### 6. Handle new Flights and Bookings:

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

### 7. Admin Functionality:

- Implement routes and controllers specific to admin functionalities such as adding flights, managing user bookings, etc.

- Add necessary authentication and authorization checks to ensure only authorised admins can access these routes.

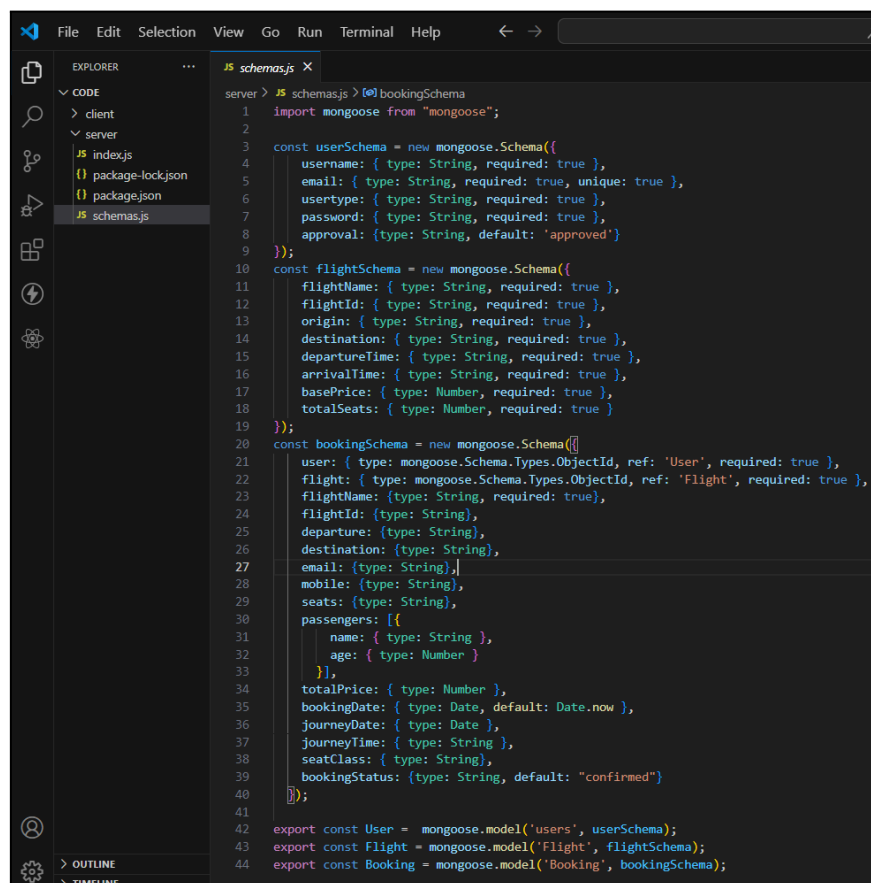
## 8. Error Handling:

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

## 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.



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

## 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.

```
//  
const PORT = process.env.PORT || 6001;  
mongoose.connect(process.env.MONGO_URL, {  
  useNewUrlParser: true,  
  useUnifiedTopology: true  
}).then(()=>{  
  
  server.listen(PORT, ()=>{  
    console.log(`Running @ ${PORT}`);  
  });  
  
}).catch((err)=>{  
  console.log("Error: ", err);  
})
```

## 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

### 2. Flight Booking (User):

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

### 3. Fetching user bookings:

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

### 4. Add new flight(Admin):

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

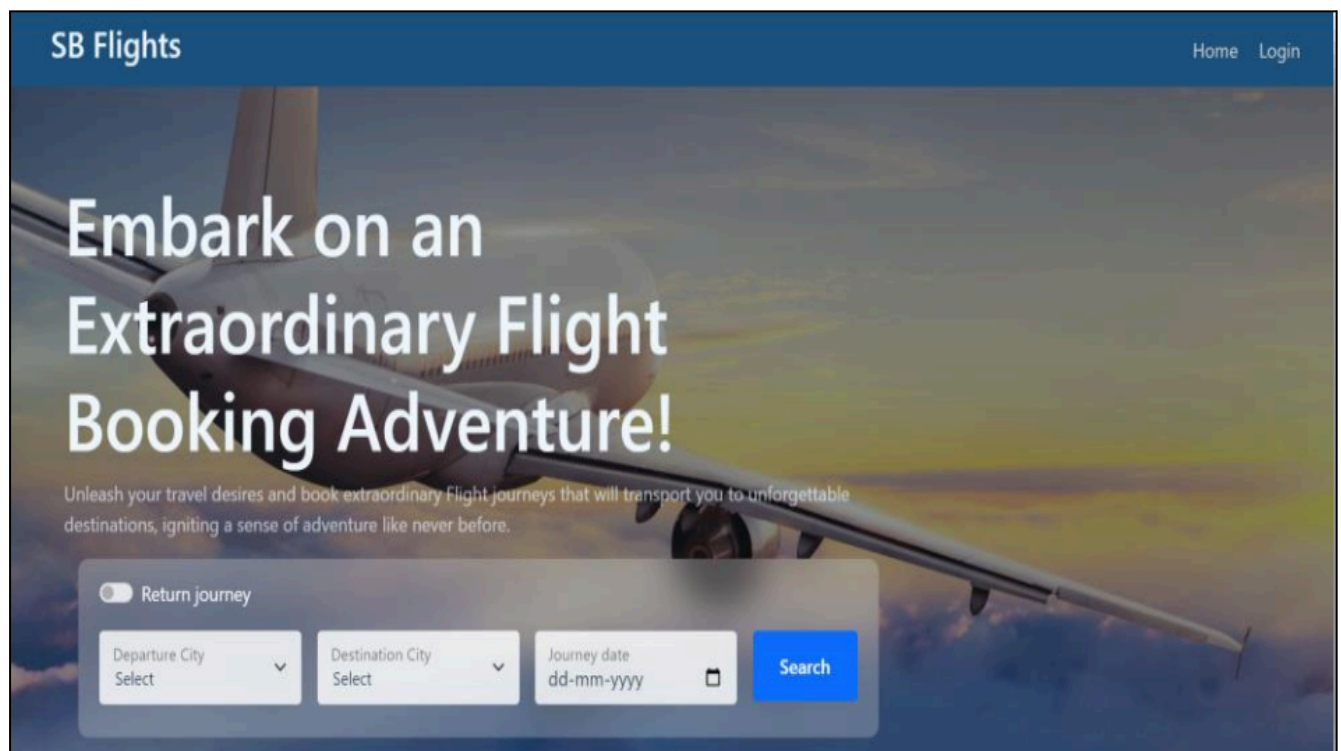
#### 5. Update Flight:

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

### Project Implementation

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

### Landing page UI



## Authentication

SB Flights

HomeLogin

Login

Email address

Password

Sign in

Not registered? Register

## User Registrations

Register

Username

| I

Email address

Password

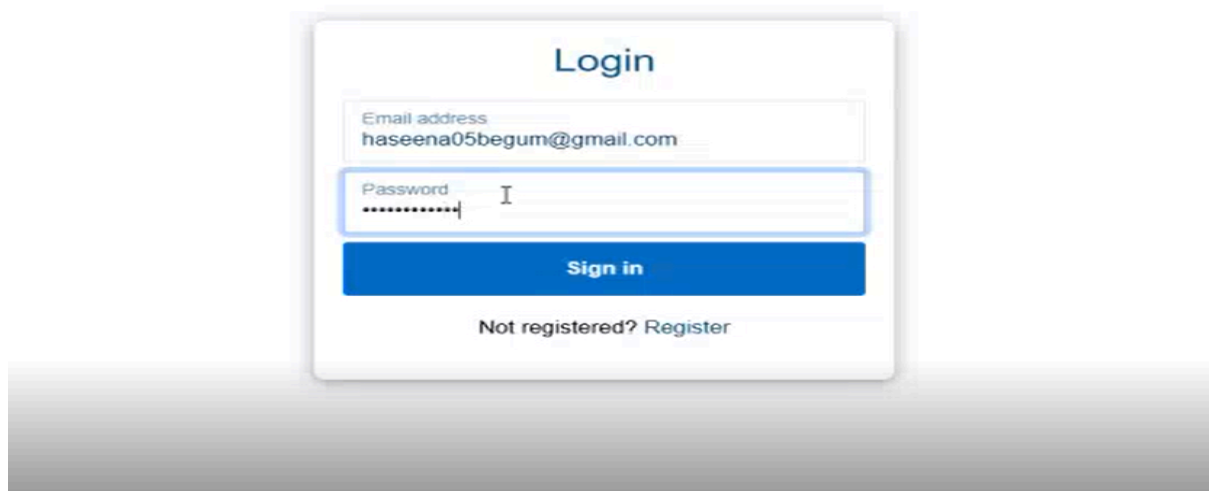
User type

▼

Sign up

Already registered? Login

## Login Page



The login form is centered on a light gray background. It features a title 'Login' in a dark blue font. Below the title are two input fields: 'Email address' with the value 'haseena05begum@gmail.com' and 'Password' with masked characters '.....'. A blue 'Sign in' button is positioned below the password field. At the bottom, there is a link 'Not registered? Register'.

**Login**

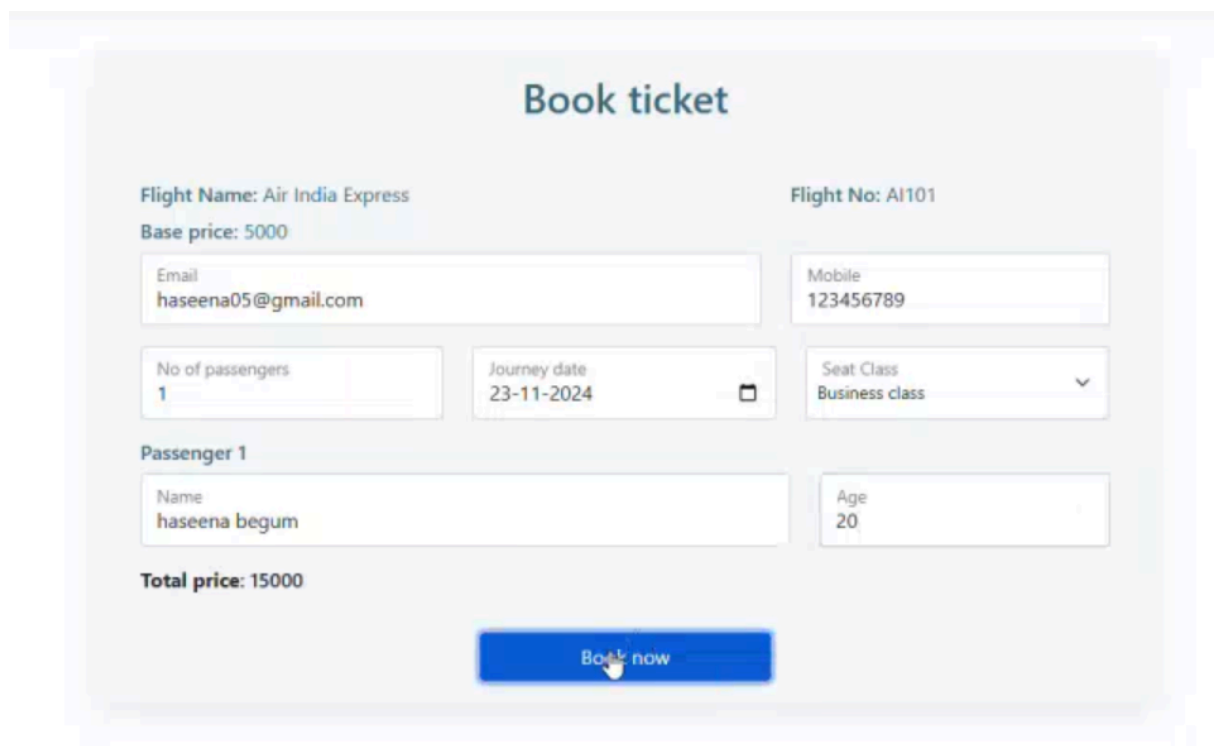
Email address  
haseena05begum@gmail.com

Password  
.....

**Sign in**

Not registered? [Register](#)

## Flight Booking



The flight booking form is titled 'Book ticket' in a dark blue font. It displays flight details: 'Flight Name: Air India Express' and 'Flight No: AI101'. The 'Base price: 5000' is shown. The form includes several input fields: 'Email' (haseena05@gmail.com), 'Mobile' (123456789), 'No of passengers' (1), 'Journey date' (23-11-2024), 'Seat Class' (Business class), 'Name' (haseena begum), and 'Age' (20). The 'Total price: 15000' is displayed at the bottom left. A blue 'Book now' button is at the bottom center.

**Book ticket**

Flight Name: Air India Express      Flight No: AI101

Base price: 5000

Email  
haseena05@gmail.com

Mobile  
123456789

No of passengers  
1

Journey date  
23-11-2024

Seat Class  
Business class

Passenger 1

Name  
haseena begum

Age  
20

Total price: 15000

**Book now**

Generated Ticket

Bookings

Booking ID: 673c34ff2f464a07168d477

Mobile: 123456789    Email: haseena05@gmail.com

Flight Id: AI101    Flight name: Air India Express

On-boarding: Delhi    Destination: Mumbai

Passengers:    Seats: 8-1

1. Name: haseena begum, Age: 20

Booking date: 2024-11-19    Journey date: 2024-11-23

Journey Time: 2024-11-20T10:30:00    Total price: 15000

Booking status: confirmed

Cancel Ticket

Booking ID: 673c35002f464a07168d489

Mobile: 123456789    Email: haseena05@gmail.com

Flight Id: AI101    Flight name: Air India Express

On-boarding: Delhi    Destination: Mumbai

Passengers:    Seats: 8-4

1. Name: haseena begum, Age: 20

Booking date: 2024-11-19    Journey date: 2024-11-23

Journey Time: 2024-11-20T10:30:00    Total price: 15000

Booking status: confirmed

Cancel Ticket

Booking ID: 673c35002f464a07168d490

Mobile: 123456789    Email: haseena05@gmail.com

Flight Id: AI101    Flight name: Air India Express

On-boarding: Delhi    Destination: Mumbai

Passengers:    Seats: 8-5

1. Name: haseena begum, Age: 20

Booking date: 2024-11-19    Journey date: 2024-11-23

Journey Time: 2024-11-20T10:30:00    Total price: 15000

Booking status: confirmed

Cancel Ticket

Booking ID: 673c34ff2f464a07168d482

Mobile: 123456789    Email: haseena05@gmail.com

Flight Id: AI101    Flight name: Air India Express

On-boarding: Delhi    Destination: Mumbai

Passengers:    Seats: 8-2

1. Name: haseena begum, Age: 20

Booking date: 2024-11-19    Journey date: 2024-11-23

Journey Time: 2024-11-20T10:30:00    Total price: 15000

Booking status: confirmed

Cancel Ticket



