Flight Booking App Report

1. Introduction

Nm id- 97527A86DC61652956B49B360F7A3217

Project Title: Flight Booking App

Team Members:

- **Ravikant Tiwari**– Project Leader And Data Administrator(MongoDB)
- **Priyanshu Kumar**– Backend Developer (Express.js)
- **Ritesh Kumar** –Frontend Developer(React.js)
- **Munna Kumar**- Backend Developer (node.js)

---

2. Project Overview

Purpose: The Flight Booking App is a web application designed to allow users to search for flights, book tickets, and manage their bookings in an intuitive and seamless manner. The project aims to streamline the flight booking process with easy navigation, secure payment options, and a responsive UI.

Goals:

- Enable users to search for flights based on departure and destination cities, dates, and number of passengers.
- Allow users to create an account, log in, and manage bookings.
- Provide an admin panel for managing flight schedules, prices, and booking records.
- Implement secure user authentication and authorization.

Features:

- **Flight Search:** Users can search for available flights by entering flight details.
- **User Registration & Login:** Secure authentication system with JWT tokens.
- **Flight Booking:** After selecting a flight, users can proceed to book tickets and make payments.
- **Booking History:** Users can view their past and upcoming bookings.
- **Admin Panel:** Admins can add, modify, or remove flight schedules.
- **Responsive UI:** The frontend is built with React, ensuring compatibility with mobile and desktop devices.

---

3. Architecture

Frontend(React)

- **React**: The frontend is developed using React.js, which allows for a component-based architecture and efficient rendering. The application is structured using functional components with hooks for managing state and effects.
- **React Router**: Used for navigation between pages such as the home page, flight search page, booking page, and user dashboard.
- **State Management**: We use React.js Context API for global state management to handle authentication status and user detail.

Backend (Node.js & Express)

- **Node.js**: The backend is built using Node.js for non-blocking, asynchronous operations, which is ideal for handling multiple user requests concurrently.
- **Express.js**: The backend API is developed using Express.js to handle HTTP requests and route them to appropriate handlers.
- **Middleware**: JWT-based authentication middleware is used for secure user access. Other middleware handle logging and error management.

Database (MongoDB)

- **MongoDB**: MongoDB is used to store flight and user data in a flexible, document-based format.
- **Database Schema**:
    - **User Schema**: Contains user details such as name, email, password (hashed), and booking history.
    - **Flight Schema**: Stores flight details including flight number, origin, destination, departure time, and price.
    - **Booking Schema**: Links users to specific flights they have booked, along with the number of passengers and booking date.

Sample Flight Schema:

Javascript

```
Const flightSchema = new mongoose.Schema({
 flightNumber: String,
 origin: String,
 destination: String,
 departureTime: Date,
```

arrivalTime: Date

price: Number});

---

4. Setup Instructions

Prerequisites:

- **Node.js** (version 14 or later)
- **MongoDB** (local or cloud, such as MongoDB Atlas)

Installation:

1. Clone the repository:

https://github.com/ravikantiwari88/Flight_Booking_App_Naan_Mudhalvan

2. **Frontend Setup:**
    - Navigate to the client directory:

      cd client

      npm start

    - Install dependencies:

      npm install
    - Create a .env file and add the necessary environment variables such as the API URL.
    - Start the React development server:

      cd server

      node index.js
3. **Backend Setup:**
    - Navigate to the server directory:

      cd server
    - Install dependencies:

      npm install
    - Create a .env file and add your MongoDB URI, JWT secret, and other necessary environment variables.
    - Start the backend server:

4. **Database Setup:**
    - Ensure MongoDB is running locally or configure MongoDB Atlas and connect it via the .env file.

---

5. Folder Structure

Client (React Frontend):

```
/client
 ├── /src
 |   ├── /components      # Reusable UI components (Buttons, Forms, etc.)
 |   ├── /pages           # Pages for different routes (Home, Search, Booking, etc.)
 |   ├── /context         # Context API for global state management
 |   ├── /services        # API calls to the backend
 |   ├── /utils           # Helper functions and constants
 |   └── App.js           # Main entry point for React app
```

Server (Node.js Backend):

```
/server
 ├── /controllers        # Logic for handling API requests
 ├── /models             # MongoDB models (User, Flight, Booking)
 ├── /routes             # API routes (User, Flight, Booking)
 ├── /middleware         # Authentication and authorization middleware
 ├── /config             # Configuration files (database, JWT secret)
 └── server.js           # Entry point for the backend server
```

---

6. Running the Application

Frontend:

1. Navigate to the client directory:

2. Start the React development server:

npm start

Backend:

1. Navigate to the server directory:

   cd server

2. Start the Node.js backend server:

   node index.js

---

7. API Documentation

Base URL: http://localhost:3000/api/

Endpoints:

- **POST /auth/register**
  - Registers a new user.
  - Request: { username, email, password }
  - Response: { message: "User registered successfully." }
- **POST /auth/login**
  - Logs a user in and returns a JWT token.
  - Request: { email, password }
  - Response: { token: "jwt-token" }
- **GET /flights**
  - Retrieves available flights.
  - Request: { origin, destination, date, passengers }
  - Response: [{ flightNumber, origin, destination, departureTime, price }, ...]
- **POST /bookings**
  - Books a flight for the user.
  - Request: { userId, flightId, passengers }
  - Response: { message: "Booking successful", bookingDetails }

---

8. Authentication

Authentication is handled using JWT (JSON Web Tokens):

- **Login:** Users log in using their email and password. Upon successful authentication, a JWT token is issued.
- **Authorization:** The JWT token is sent as a Bearer token in the Authorization header for any protected routes (e.g., booking a flight, viewing bookings).
- **Session Management:** The token is stored in the browser's local storage.

---

9. User Interface

Here are some key screenshots of the UI:

- **Home Page:**
- **Flight Search Results:**
- **Booking Page:**

---

10. Testing

Testing is done using **Jest** and **Supertest** for unit and integration tests:

- **Jest**: Used for testing the backend logic, such as user registration and flight booking.
- **Supertest**: Used for testing the API endpoints to ensure they respond correctly to HTTP requests.

---

11. Screenshots or Demo

- https://github.com/ravikantiwari88/Flight_Booking_App_Naan_Mudhalvan
- Screenshots showcasing the booking process and UI flow are included in the previous section.

---

12. Known Issues

- **Flight Date Filter:** The flight date filter sometimes returns incorrect results when the date format is not consistent with the user's locale.
- **Payment Gateway Integration:** The payment system is not yet integrated (future feature).

---

13. Future Enhancements

- **Payment Gateway Integration**: Implement secure payment gateways like Stripe or PayPal for flight booking payments.
- **Multi-language Support**: Add support for multiple languages.
- **Push Notifications**: Notify users about flight status changes, booking confirmations, etc.

- **Admin Features**: Add more advanced features for admin users, such as managing user accounts and tracking sales data.