The Air Ticket Booking: Documentation

### 1. Project Overview

The Air Ticket Booking project is a web application designed to facilitate the booking of flights online. It allows users to sign up, sign in, view available flights, make bookings, manage their bookings, and provides administrative capabilities through a dashboard.

### 2. Architecture

#### Frontend Architecture

The frontend of the application is built using React.js, a popular JavaScript library for building user interfaces. It utilizes modern practices such as component-based architecture and client-side routing for a seamless user experience. Tailwind CSS is used for styling, providing utility-first CSS classes for rapid UI development.

##### Key Files and Folders:

- \*\*`index.html`\*\*: Entry point of the application, includes the main `<div>` container and references to necessary scripts.

- \*\*`App.jsx`\*\*: Root component managing routing using React Router. Defines routes for different pages like Home, Available Flights, Schedule, Login, Signup, and more.

- \*\*`main.jsx`\*\*: Renders the `<App>` component within the Redux `<Provider>` and `<PersistGate>` for state management and persistence.

- \*\*`App.css` and `index.css`\*\*: Style sheets defining the layout, styling, and animations used throughout the application.

#### Backend Architecture

The backend is developed using Node.js and Express.js, providing a robust server-side framework. MongoDB is used as the database with Mongoose for data modeling and interaction. This setup allows for efficient handling of flight data, user authentication, booking management, and administrative functionalities.

##### Key Files and Folders:

- \*\*`index.js`\*\*: Entry point of the server, sets up Express middleware, connects to MongoDB using Mongoose, and defines routes for flight, user authentication, and booking operations.

- \*\*`controllers`\*\*: Contains modules handling business logic for different routes such as user authentication (`auth.controller.js`), flight operations (`flight.controller.js`), and booking management (`booking.controller.js`).

- \*\*`models`\*\*: Defines MongoDB schemas (`booking.model.js`, `flight.model.js`) using Mongoose, specifying the structure and validation for bookings and flights.

### 3. APIs

The backend exposes RESTful APIs for interacting with flights, user authentication, and booking management:

- \*\*Authentication APIs\*\*:

- `/api1/auth/signup`: Registers a new user with email and password.

- `/api1/auth/signin`: Authenticates user credentials and returns a JWT token for authorization.

- \*\*Flight APIs\*\*:

- `/api1/flight`: Retrieves all flights.

- `/api1/flight/getFlightById/:id`: Retrieves a specific flight by ID.

- `/api1/flight/filteredFlights`: Retrieves flights filtered by departure and arrival airports.

- \*\*Booking APIs\*\*:

- `/api1/booking/new`: Makes a new booking.

- `/api1/booking/user`: Retrieves bookings made by the authenticated user.

- `/api1/booking/:id`: Retrieves a specific booking by ID.

- `/api1/booking/flight`: Retrieves booked seats for a specific flight.

- `/api1/booking/`: (Admin) Retrieves all bookings.

- `/api1/booking/:id`: (Admin) Deletes a specific booking by ID.

- `/api1/booking/update`: (Admin) Updates a booking's seat number and date.

### 4. Usage Instructions

To use the Air Ticket Booking application:

1. \*\*Sign Up and Sign In\*\*: Create a new account using `/signup` and authenticate using `/signin` endpoints.

2. \*\*View Flights\*\*: Navigate to `/AvailableFlights` to view all available flights or filter them using `/filteredFlights`.

3. \*\*Make a Booking\*\*: Use the `/new` endpoint to book a seat on a flight, specifying the user ID, flight ID, date, and seat number.

4. \*\*Manage Bookings\*\*: Users can manage their bookings through `/ManageBookings`, viewing, updating, or deleting bookings as needed.

5. \*\*Admin Dashboard\*\*: Administrators can access the `/Dashboard` to view all bookings and perform administrative tasks.

### 5. Development Process

The project was developed iteratively, following these steps:

- \*\*Planning and Design\*\*: Defined project requirements, created wireframes, and established the frontend and backend architecture.

- \*\*Implementation\*\*: Implemented frontend components, backend logic, database integration, and API development.

- \*\*Testing\*\*: Conducted unit tests for controllers, models, and API endpoints using tools like Postman and Jest.

- \*\*Deployment\*\*: Deployed the application on a development server, ensuring functionality across different environments.

- \*\*Monitoring and Maintenance\*\*: Monitored application performance, addressed bugs, and implemented feature enhancements based on user feedback.

### 6. Challenges Faced

- \*\*Authentication and Authorization\*\*: Implementing secure authentication and authorization mechanisms using JWT tokens and role-based access control.

- \*\*Database Operations\*\*: Efficiently managing complex data relationships and ensuring data integrity in MongoDB.

- \*\*Error Handling\*\*: Implementing robust error handling across frontend and backend components to provide a seamless user experience.

### 7. Conclusion

The Air Ticket Booking project demonstrates the integration of modern web technologies to create a scalable and user-friendly flight booking application. By leveraging React.js, Node.js, Express.js, and MongoDB, the application provides a responsive and efficient platform for users to book flights and manage their travel itineraries effectively.