MERN Stack Assignment: "Flight Booking System"

Objective:

To build a full-stack web application that allows users to search for flights, view available options, and make bookings. The application should include user authentication, a flight search engine, and a booking management system.

Requirements:

1. **User Authentication:**
    - Implement user registration, login, and logout functionality.
    - Use JWT (JSON Web Token) for authentication.
    - Hash passwords using bcrypt.
    - Include different user roles (e.g., User, Admin).
2. **Frontend:**
    - Create a React application with the following pages:
        - **Home Page:** Welcome page with an overview of the system and a search bar for flights.
        - **Search Results Page:** Displays available flights based on search criteria (origin, destination, date).
        - **Flight Details Page:** Shows detailed information about a selected flight (airline, duration, price, available seats).
        - **Booking Page:** Allows users to enter personal details and confirm a booking.
        - **User Profile Page:** Displays the user's profile information, booking history, and allows editing personal details.
        - **Admin Dashboard:** Allows admin users to add, update, and delete flights, as well as view all bookings.
        - **Login and Registration Pages:** For users to log in or register.
    - Implement routing using `react-router-dom`.
    - Use state management (React Context API or Redux) for managing user authentication, flight data, and booking data.
    - Ensure the UI is responsive and user-friendly.
3. **Backend:**
    - Set up a RESTful API using Node.js and Express.js with the following endpoints:
        - `POST /api/register`: Register a new user.
        - `POST /api/login`: Authenticate a user and provide a JWT token.
        - `GET /api/flights`: Retrieve a list of all available flights.
        - `GET /api/flights/search`: Retrieve flights based on search criteria (origin, destination, date).
        - `GET /api/flights/:id`: Retrieve details of a specific flight.

        - `POST /api/bookings`: Create a new booking for a flight (requires authentication).
        - `GET /api/bookings/user/:id`: Retrieve all bookings for a specific user (requires authentication).
        - `GET /api/bookings`: Retrieve all bookings (Admin only).
        - `PUT /api/bookings/:id`: Update a booking (Admin only).
        - `DELETE /api/bookings/:id`: Delete a booking (Admin only).
        - `POST /api/flights`: Add a new flight (Admin only).
        - `PUT /api/flights/:id`: Update flight details (Admin only).
        - `DELETE /api/flights/:id`: Delete a flight (Admin only).
    - Use Mongoose to interact with MongoDB for data storage.
    - Implement input validation and error handling for all endpoints.
4. **Database:**
    - Design a MongoDB schema with the following collections:
        - **Users:** Stores user information (username, email, hashed password, role, etc.).
        - **Flights:** Stores flight details (flight number, airline, origin, destination, date, time, price, available seats, etc.).
        - **Bookings:** Stores booking details (userId, flightId, number of seats, total price, booking status, etc.).
5. **Deployment:**
    - Deploy the backend API to a cloud platform (e.g., Heroku, Render, or any preferred service).
    - Deploy the frontend React app to a platform like Vercel, Netlify, or any preferred service.
    - Use environment variables for managing sensitive information (e.g., database connection strings, JWT secret).
6. **Additional Features:**
    - Implement a flight booking confirmation email using a service like SendGrid or Nodemailer.
    - Add pagination to the flight search results.
    - Allow users to filter search results by price, duration, airline, or available seats.
    - Implement a date picker for selecting the travel date.
    - Allow users to cancel their bookings within a certain time frame.
    - Add error pages for 404 and 500 status codes.

Bonus Tasks:

- Integrate a third-party API (like Amadeus API) to fetch real-time flight data.
- Use a React library such as Material-UI or Tailwind CSS for enhanced UI components.
- Implement unit and integration testing for the backend API using tools like Jest or Mocha.
- Add an interactive map feature showing the flight route.

- Implement state management in the frontend using Redux Toolkit for better state handling.
- Add an admin analytics dashboard showing metrics like the total number of flights, bookings, most popular destinations, etc.

Submission:

- The code should be version controlled using Git.
- Submit the GitHub repository link containing the complete project (frontend and backend).
- Include a README file in the repository with instructions on how to set up and run the project locally.
- Provide the URLs for the deployed frontend and backend applications.

Evaluation Criteria:

- Code quality and adherence to best practices.
- Proper use of the MERN stack technologies.
- Functionality of all required features.
- Responsiveness and usability of the frontend.
- Handling of edge cases and errors.
- Completion of additional features and bonus tasks.