Full-Stack Engineering Assignment: Flight Booking System

Overview

Design, build, and deploy a web-based Flight Booking System that allows users to search, book, and manage flight reservations. The system should support both one-way and round-trip bookings. This assignment evaluates your skills in frontend development, backend architecture, database design, API development, authentication, automated testing, and deployment.

Requirements

Functional Requirements

User Management

- Authentication using Supabase Auth
- User profile management with personal and payment information
- Booking history for registered users

Flight Search

- Search functionality with the following filters:
  - Origin and destination airports
  - Departure date
  - Return date (for round-trip)
  - Number of passengers (adults, children, infants)
  - Cabin class (Economy, Premium Economy, Business, First)
- Support for one-way and round-trip bookings
- Display available flights with details:
  - Flight number
  - Airline
  - Departure and arrival times
  - Duration
  - Price
  - Available seats

Booking Management

- Passenger information collection
- Booking confirmation with e-ticket generation
- Booking modification and cancellation
- Real-time flight status updates using Server-Sent Events (SSE)
- Email notifications for booking confirmation and updates

Technical Requirements

Frontend

- Build a responsive SPA using React, NextJS or similar modern framework
- Implement state management (Redux, Context API, etc.)
- Create reusable UI components with proper documentation
- Implement form validation
- Build interactive data visualizations for the admin dashboard
- Implement error handling and loading states
- Optimize for mobile and desktop experiences
- Use IndexedDB for offline data persistence and caching of flight search results
- Implement Web Workers for performance-intensive tasks (like filtering and sorting large flight datasets)

Backend

- Utilize Supabase as the primary backend service for:
    - Database (PostgreSQL)
    - Authentication
    - Storage
    - Realtime data updates
- Develop supplementary APIs for business logic not covered by Supabase
- Implement Server-Sent Events (SSE) for real-time flight updates
- Implement proper error handling and logging
- Write unit and integration tests

Deployment

- Deploy the application to a cloud platform of your choice (AWS, GCP, Azure, Vercel, Netlify, etc.)
- Provide a public URL to access the working application
- Document the deployment process

Technical Constraints

- The frontend must be built using React
- Supabase must be used for backend database and authentication
- Implement Server-Sent Events (SSE) for real-time updates
- Use IndexedDB for offline data persistence
- Implement Web Workers for CPU-intensive operations
- Code must include automated tests (unit) (Good to have)
- Include documentation for API endpoints (Use Swagger)

Development Tools

- You are encouraged to use Cursor or Windsurf IDE to accelerate development
- These AI-powered coding tools can help with code generation, debugging, and optimization

Deliverables

Source Code

- Frontend and backend code repositories with clear READMEs
- Database schema definition
- Docker configuration files

Documentation

- System architecture diagram
- API documentation
- Setup and deployment instructions
- Test coverage report

Deployment

- Working application URL
- Deployment documentation

Presentation

- 15-minute demonstration of the application
- Discussion of technical decisions and trade-offs
- Explanation of architecture and design patterns used

Evaluation Criteria

Candidates will be evaluated on the following:

Code Quality

- Clean, maintainable code structure
- Proper error handling
- Application of design patterns
- Code reusability

System Design

- Architecture decisions
- Database schema design
- API design
- Scalability considerations

Functionality

- Meeting all requirements
- Edge case handling (e.g., flight cancellations, overbooking)
- Performance
- User experience

Testing

- Test coverage
- Testing approach
- Quality of tests

Deployment

- Successfully deployed application
- Documentation quality

Bonus Points

- Implementing a fare calendar to display prices across dates
- Implementing a recommendation engine for alternative flights
- Supporting multi-city flight bookings
- Adding loyalty program functionality
- Implementing progressive web app (PWA) features

DevOps (Good to Have)

- Containerize the application using Docker
- Set up CI/CD pipeline
- Implement basic monitoring and logging
- Infrastructure as code

Time Expectation

Candidates should complete this assignment within 3 days. Quality is valued over quantity, so focus on delivering a well-designed system with core functionality rather than implementing all features poorly.

Submission Guidelines

- Provide GitHub repositories for the frontend and backend code
- Include a link to the deployed application
- Submit all documentation as Markdown files in the repositories
- Schedule a presentation and code review session after submission

Good luck with your implementation!