Software Requirements Specification (SRS) for SKY Airlines Airline Reservation System

1. Introduction

The **SKY Airlines Airline Reservation System (ARS)** is a user-friendly web application that simplifies flight booking, seat reservations, and user management. Whether you're a passenger planning a trip or an airline administrator overseeing operations, the ARS aims to enhance your experience.

1.1 Description

The ARS seamlessly connects passengers with available flights, allowing them to explore options, select seats, and finalize bookings. For administrators, it streamlines flight management tasks, ensuring efficient seat allocation and accurate records.

1.2 Purpose

- **Passenger Convenience**: Imagine a traveler searching for a convenient flight. The ARS provides an intuitive interface, allowing passengers to find suitable flights, compare prices, and secure their seats hassle-free.
- **Operational Efficiency**: Behind the scenes, airline staff benefit from automated processes. The ARS handles reservations, updates seat availability, and maintains passenger data.
- **Data Integrity**: Trustworthy data storage ensures that bookings remain accurate and confidential.

2. Intended Audience

The ARS caters to different user groups:

- **Passengers**: Individuals seeking seamless flight booking experiences.
- **Administrators**: Airline staff responsible for managing flights and reservations.
- **Developers**: Those maintaining and enhancing the system.

3. Intended Use

The ARS empowers users to:

- **Passengers**:
    - Search for flights based on preferences (dates, destinations).
    - Reserve seats with confidence.
    - Access booking details anytime.
- **Administrators**:
    - Manage flight information (add, update, delete).
    - Monitor seat availability.
    - Ensure smooth operations.

4. Scope

The ARS covers essential features:

- **Flight Reservation and Seat Allocation**:
  - Passengers can explore flight options, view seat layouts, and choose seats.
  - The system allocates seats efficiently.
- **User Account Management**:
  - Secure login for passengers and administrators.
  - Profile management (view bookings, update personal details).
- **Responsive Design**:
  - The ARS adapts seamlessly to various devices (desktop, tablet, mobile).
- **GitHub API Integration**:
  - Passport.js ensures secure authentication via GitHub credentials.

5. Assumptions and Dependencies

- **Assumptions**:
  - Users are familiar with basic web interfaces.
  - The GitHub API remains accessible for authentication.
- **Dependencies**:
  - **C++**: Powers backend logic and data processing.
  - **SQL**: Manages the database.
  - **Next.js**: Facilitates server-side rendering and smooth navigation.

6. Overall Description

6.1 System Architecture

- The ARS follows a layered approach:
  - **Presentation Layer**: Next.js components handle user interfaces (think of it as the "look and feel").
  - **Application Layer**: C++ manages business logic (the system's brain).
  - **Data Layer**: SQL stores flight data, reservations, and user profiles.
- **GitHub API Integration**:
  - Passport.js ensures secure authentication.
- **TailwindCSS Styling**:
  - TailwindCSS provides consistent, visually appealing styles.

6.2 Deployment

- Deploy the system on web servers (e.g., Apache, Nginx).
- Ensure scalability, load balancing, and failover mechanisms.

6.3 Key Aspects

Let's explore the critical aspects of the ARS:

Performance

- The ARS aims for responsiveness:
    - Quick page loads (within a couple of seconds).
    - Efficient handling of concurrent user requests.

Reliability

- The system strives for reliability:
    - Minimal downtime.
    - Graceful error handling.

Security

- Security measures protect user data:
    - Secure APIs (HTTPS).
    - Encryption for sensitive information (passwords, personal details).

Availability

- The ARS ensures availability:
    - Backup servers for failover.
    - Continuous monitoring.

Maintainability

- Regular updates and enhancements:
    - Bug fixes.
    - System improvements.

7. Functional Requirements

7.1 Login Panel (Administrator)

- Admin Authentication:
    - Secure login using GitHub credentials.
    - Access control for administrators.
- Flight Management:
    - Add, update, and delete flight details.
    - Monitor seat availability.

7.2 User Interfaces (Software)

- **Passenger Interface**:
    - Intuitive search for flights.
    - Seat selection.
    - Access to booking history.
- **Responsive Design**:

- Consistent experience across devices.

Certainly! Let's dive deeper into the functional requirements for the **SKY Airlines Airline Reservation System**. I'll provide additional details and expand upon the features. Here's an extended version:

Functional Requirements for SKY Airlines Airline Reservation System

1. Travel Search Feature

- **Description**:
  - Passengers should be able to search for flights based on various criteria, including:
    - Origin and destination cities.
    - Preferred travel dates (departure and return).
    - Number of passengers (adults, children, infants).
    - Class (economy, business, first).
  - The system should retrieve relevant flight options matching the search parameters.

2. Result Browsing Feature

- **Description**:
  - After performing a search, passengers can browse through the search results.
  - The system should display essential information for each flight, such as:
    - Flight number and airline.
    - Departure and arrival times.
    - Layovers (if any).
    - Available seat classes and prices.
  - Passengers can filter and sort results based on preferences (e.g., price, duration).

3. Flight Booking Feature

- **Description**:
  - Passengers can select a specific flight from the search results.
  - The system should allow passengers to:
    - Choose their preferred seat(s).
    - Provide passenger details (names, contact information).
    - Confirm the booking and proceed to payment.
  - Upon successful booking, passengers receive a confirmation with a unique booking reference.

4. Change or Cancellation Feature

- **Description**:
  - Passengers should have the option to modify or cancel their bookings.
  - Change requests may include:
    - Rescheduling flights (within policy limits).
    - Updating passenger information.
    - Adding or removing passengers.
  - Cancellation requests should follow refund policies and generate appropriate notifications.

5. Administrative Features

- **Description**:
  - **Administrator Login**:
    - Secure login for airline staff.
    - Access control based on roles (e.g., superadmin, flight manager).
  - **Flight Management**:
    - Add, update, or delete flight details (routes, schedules, aircraft types).
    - Manage seat availability and block seats for special cases (e.g., passengers with disabilities).
  - **Booking Management**:
    - View and modify passenger bookings.
    - Handle special requests (e.g., meal preferences, extra baggage).
  - **Reporting and Analytics**:
    - Generate reports on flight occupancy, revenue, and popular routes.
    - Monitor system performance and identify trends.

6. User Experience Enhancements

- **Responsive Design**:
  - Ensure the system works seamlessly on desktops, tablets, and mobile devices.
- **User-Friendly Interface**:
  - Intuitive navigation.
  - Clear instructions and error messages.
  - Visual cues for seat selection and booking steps.
- **Customer Support Integration**:
  - Provide chat support or a call center for passenger inquiries and assistance.

Feel free to adapt and elaborate on these functional requirements further. If you have any specific features in mind or need additional details, feel free to ask! 🛫

1. Unit Testing: The Microscopic Inspectors

Purpose:

Unit testing is like using a magnifying glass to examine tiny components—individual functions, methods, or classes. Its purpose is to:

- **Catch Bugs Early**: Imagine spotting a loose rivet on an aircraft before it causes turbulence. Unit tests do just that—they identify issues at the smallest level.
- **Ensure Component Reliability**: Each function or method is tested independently to ensure it performs as expected.

How It Works:

1. **Test Cases Creation**:
    - Developers write small test cases for each component.
    - These cases cover various scenarios, including normal behavior, edge cases, and error conditions.
2. **Automated Execution**:
    - Automated testing tools execute these test cases.
    - If a component fails, alarms sound, and we rush to fix it.
3. **Regression Guard**:
    - As the system evolves, unit tests act as guardians against regression—ensuring that new changes don't break existing features.

2. Integration Testing: The Symphony of Components

Purpose:

Integration testing ensures that different components play harmoniously together. Its purpose is to:

- **Orchestrate Harmony**: Just like an orchestra, where violins and trumpets must blend seamlessly, integration tests verify that components interact without discord.
- **Detect Interaction Issues**: We want our components to dance gracefully, not step on each other's toes.

How It Works:

1. **Component Assembly**:
    - We assemble the orchestra (components).
    - Conductors (test scripts) orchestrate complex scenarios.
2. **Scenario Exploration**:
    - We simulate interactions—data flowing between components, APIs being called, and services collaborating.
    - If the violins clash with the trumpets, we fine-tune the score.

3. User Acceptance Testing (UAT): Real Passengers, Real Smiles

Purpose:

UAT is like inviting real passengers on board to evaluate our system. Its purpose is to:

- **Validate from a User's Perspective**: Passengers (users) take the system for a spin, ensuring it meets real-world requirements.
- **Fine-Tune Based on Feedback**: If they frown, we tweak the seat cushions.

How It Works:

1. **Real Users, Real Tasks**:
    - Passengers perform tasks: booking flights, modifying bookings, and exploring features.
    - We observe their expressions—joy, frustration, or confusion.
2. **Feedback Loop**:
    - Their insights guide us. If they struggle, we adjust course.
    - UAT ensures our passengers have a smooth journey.