Software Requirements Specification (SRS) for Airline Reservation System

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for the Airline Reservation System (ARS), a web-based application designed to simplify airline booking processes. The system will enable users to search for flights, book tickets, manage bookings, and provide administrative tools for airline staff to manage flight schedules and availability. This SRS reflects a microservices-based architecture for better scalability and modularity.

1.2 Scope

The ARS is designed to cater to customers, administrators, and airline staff. The system will:

- Allow customers to search, book, and cancel flights.
- Provide secure payment integration.
- Enable airline administrators to manage flight schedules, pricing, and seat availability.
- Be highly scalable, available, and secure, leveraging a microservices architecture deployed on Oracle Cloud or similar platforms.

Each functionality will be encapsulated within independent microservices communicating through REST APIs and/or asynchronous messaging systems.

1.3 Definitions and Acronyms

- **CRUD**: Create, Read, Update, Delete.
- **REST API**: Representational State Transfer Application Programming Interface.
- **JWT**: JSON Web Token.
- **SLA**: Service Level Agreement.
- **GDPR**: General Data Protection Regulation.

1.4 References

- Oracle Cloud documentation: https://docs.oracle.com/en/cloud/
- Spring Boot documentation: https://spring.io/projects/spring-boot

---

2. Overall Description

2.1 Product Perspective

The ARS will be a distributed microservices-based system with integrations to payment gateways, notification services (email/SMS), and third-party APIs for flight data. The architecture will feature an API Gateway for handling client requests and a messaging system (e.g., RabbitMQ/Kafka) for asynchronous communication between services.

2.2 Product Features

1. **Flight Search**:
    - Independent service to search flights based on source, destination, and date.
    - Filter results by price, duration, and airline.
2. **Booking Management**:
    - Secure booking and ticket generation via a dedicated Booking Service.
    - Integration with a Payment Service.
3. **Profile Management**:
    - Manage customer profiles with booking history.
4. **Admin Features**:
    - CRUD operations for flight schedules, pricing, and seat inventory.
    - Reporting and analytics.
5. **Notification System**:
    - Email/SMS notifications for booking confirmations, cancellations, and updates.

2.3 User Classes and Characteristics

1. **Customers**: End-users who search and book flights.
2. **Administrators**: Manage flights and monitor the system.
3. **Airline Staff**: Handle operational aspects like seat availability.

2.4 Operating Environment

- **Frontend**: ReactJS hosted on Railway or Oracle Cloud.
- **Backend**: Spring Boot microservices communicating via REST APIs and message queues.
- **Database**: PostgreSQL, with independent databases for each service.
- **Containerization**: Docker with Kubernetes for orchestration.
- **Messaging**: RabbitMQ/Kafka for asynchronous communication.

2.5 Design and Implementation Constraints

- Compliant with GDPR for user data.
- Ensure high availability (99.9% uptime).

- Payment gateway integration meeting PCI DSS standards.

2.6 Assumptions and Dependencies

- External services like payment gateways and email APIs are operational.
- Oracle Cloud or similar infrastructure services are reliable.

---

3. Functional Requirements

3.1 Flight Search Service

- **Inputs**: Source, destination, date.
- **Outputs**: List of flights with details (price, duration, seats available).
- **Process**:
  - Query the Flight Database for flights matching input criteria.
  - Apply filters for sorting and selection.

3.2 Booking Service

- **Inputs**: Flight ID, passenger details, payment information.
- **Outputs**: Booking confirmation, ticket.
- **Process**:
  - Validate flight and seat availability through the Flight Search Service.
  - Coordinate with the Payment Service.
  - Update the database with booking details.

3.3 Notification Service

- **Inputs**: Booking details, contact information.
- **Outputs**: Email/SMS notifications.
- **Process**:
  - Generate notifications for booking confirmations, cancellations, and updates.

3.4 Admin Management Service

- **Inputs**: Flight schedules, pricing, inventory updates.
- **Outputs**: Confirmation of updates.
- **Process**:
  - Enable CRUD operations on flight data.
  - Generate reports using analytics tools.

---

4. Non-Functional Requirements

4.1 Performance

- System must respond to API calls within 2 seconds.

4.2 Scalability

- Scale services independently to handle 10,000+ concurrent users.

4.3 Security

- HTTPS for all communications.
- Encrypt sensitive data.
- OAuth2 for authentication and authorization.

4.4 Availability

- 99.9% uptime using load balancing and failover mechanisms.

4.5 Maintainability

- Centralized logging and monitoring (e.g., ELK stack).
- Modular codebase for easy updates.

---

5. Appendices

- **Glossary**: Definitions of terms and acronyms.
- **References**: Links to relevant documentation.