Software Requirement Specifications

[Airline Reservation System]

Version: [4.0]

Project Code	
Supervisor	*Syeda Rubab Jaffar*
Co Supervisor	
Project Team	*Qasim Ali* *Jahanzaib Mirza* *Muhammad Maaz Khan*
Submission Date	*05-07-2025*

Document History

Version	Name of Person	Date	Description of change
1.0	Jahanzaib Mirza	01-May-2025	Created initial document structure and template
1.5	Muhammad Maaz Khan	02-May-2025	Added introduction, purpose, and intended audience sections
2.0	Qasim Ali	03-May-2025	Drafted overall system description
2.5	Jahanzaib Mirza	04-May-2025	Added external interface requirements
3.0	Muhammad Maaz Khan	05-May-2025	Added functional and non-functional requirements
3.5	Qasim Ali	06-May-2025	Finalized use cases and performance/security specs
4.0	Jahanzaib Mirza	07-May-2025	Reviewed and compiled final version for submission

Distribution List

Name	Role

Document Sign-Off

Version	Sign-off Authority	Sign-off Date

Table of Contents

1. Introduction

1.1. Purpose of Document

The purpose of this document is to outline the Software Requirements Specification (SRS) for the SkyNest Airline Reservation System. It defines the system's functionality, design constraints, user interactions, and other relevant details to ensure a clear understanding among developers, testers, stakeholders, and future maintainers. This document serves as a reference throughout the software development lifecycle.

1.2. Intended Audience

This document is intended for:

- **Project Developers**: To understand and implement system functionality.
- **Project Supervisors and Evaluators**: To assess the completeness and alignment of the project with the requirements.
- **Testers/QA Engineers**: To verify and validate that the system meets the documented specifications.
- **Future Maintenance Teams**: For future upgrades or debugging.
- **End Users (indirectly)**: To ensure that their needs are effectively captured and addressed.

1.3 Abbreviations

Abbreviation	Description
SRS	Software Requirements Specification
UI	User Interface
DB	Database
SQL	Structured Query Language
API	Application Programming Interface
UX	User Experience

1.4 Document Convention

Font: Arial
Font size: 16 for section headings, 12 for section subheadings, 10 for section bodies.

2. Overall System Description

2.1. Project Background

In today's fast-paced world, the demand for digitalized, efficient, and user-friendly travel solutions is growing rapidly. Traditional airline booking systems often suffer from poor user experience, limited accessibility, and a lack of transparency. The SkyNest Airline Reservation System was conceptualized to bridge this gap by offering an end-to-end online solution for both passengers and airline administrators. The project was initiated to address the need for a responsive and scalable airline reservation system built using modern technologies.

2.2. Project Scope

SkyNest is a web-based airline reservation system developed using Spring Boot for the backend, React.js for the frontend, and MySQL as the database. The system offers two modules: Admin and User. Key functionalities include user registration/login, flight booking, payment and cancellation request handling, ticket generation, review submission, and an AI-based chatbot for assistance. Admins can manage flights, handle user requests, analyze statistics, and view user feedback. The scope of the project covers complete functionality required to manage airline reservations, improve service delivery, and enhance user engagement.

2.3. Not In Scope

- Integration with third-party payment gateways (dummy flows are used instead).
- Real-time flight tracking or integration with airline APIs.
- Mobile application version of the system.
- Multi-language support (currently limited to English).
- Biometric authentication or advanced security mechanisms like 2FA.

2.4. Project Objectives

- Provide an intuitive platform for users to search, book, and manage flight reservations.
- Enable administrators to efficiently manage flights and handle booking or cancellation requests.
- Offer transparency in the booking process and communication through a notification system.
- Improve decision-making with admin-side analytics and visual data representations.
- Enhance user support through a built-in chatbot that provides real-time guidance.

2.5. Stakeholders

- **End Users (Passengers)**: Individuals who register on the platform to book flights.
- **Administrators**: Authorized personnel responsible for managing flight data, reviewing requests, and overseeing system operations.
- **Developers**: Team responsible for designing and implementing the system.
- **Project Supervisors/Instructors**: Academic stakeholders overseeing the development process and evaluating outcomes.
- **Testers**: Individuals responsible for verifying that the system meets requirements and is bug-free.

2.6. Operating Environment

- **Frontend**: React.js (runs on any modern web browser: Chrome, Firefox, Edge)
- **Backend**: Spring Boot (Java-based REST APIs)
- **Database**: MySQL

- **Server**: Hosted on localhost or cloud platforms (e.g., Heroku, AWS – if deployed)
- **Operating System**: Windows/Linux for development; browser-agnostic for end users
- **Network**: Internet connectivity required for access

2.7. System Constraints

- **Software Constraints**: System is limited to web browser usage only; no support for native mobile platforms.
- **Hardware Constraints**: Requires a system capable of running modern browsers and server applications.
- **Cultural Constraints**: System language is restricted to English only.
- **User Constraints**: Target audience is digitally literate users familiar with web interfaces.
- **Academic Constraints**: Development was bound by semester deadlines and resource availability.
- **Legal Constraints**: No actual personal or payment data is stored or transmitted due to privacy and ethical limitations of academic projects.

2.8. Assumptions & Dependencies

- It is assumed that users and admins have stable internet connectivity.
- The system is assumed to run in a browser with JavaScript enabled.
- Admins are assumed to manually validate aircraft availability before flight creation.
- The project depends on the MySQL server being available and properly configured.
- Chatbot functionality depends on predefined intents and does not support natural language AI integration.

3. External Interface Requirements

This section outlines how the SkyNest Airline Reservation System interfaces with external systems, hardware, and communication technologies to ensure seamless operation. A high-level context diagram is described below (a visual diagram can be inserted later during document finalization).

3.1. Hardware Interfaces

SkyNest is a web-based system and has minimal direct interaction with hardware components. However, the system relies on the following hardware interfaces:

- User Device: Personal computers or laptops (Windows/Linux/macOS) with a modern browser.
- Admin Device: Same as above, with access to administrative features.
- Server Hardware: A local machine capable of running a Spring Boot backend and MySQL database.

3.2. Software Interfaces

The SkyNest system depends on the following software components:

- Frontend Framework: React.js
    - Manages the user interface and interacts with backend APIs using HTTP/HTTPS.
- Backend Framework: Spring Boot (Java)
    - Hosts RESTful APIs that handle business logic, authentication, and database interaction.
- Database: MySQL 8.x
    - Stores user data, flight records, bookings, reviews, and more.
    - Communicates with the backend via JDBC.
- Chatbot Module:
    - Integrated within the frontend; uses scripted responses for predefined queries.
    - Interacts with a JSON-based data store or internal state (not an external AI API).
- Development Tools:
    - IntelliJ IDEA / VS Code for development.
    - Postman for API testing.
    - Git for version control.
- Browser Compatibility:
    - Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions).

3.3. Communications Interfaces

Web Communication:

- All communication between frontend and backend occurs over HTTP or HTTPS protocols.
- RESTful API endpoints are used for CRUD operations related to bookings, flights, users, etc.

Data Format:

- JSON is used as the standard format for data exchange between frontend and backend.

Security and Authentication:

- Basic JWT-based token authentication for secure sessions (if implemented).
- HTTPS can be used during deployment to encrypt communication.

Network Environment:

- Requires stable internet connection for users to access the system.
- Backend server must be reachable from the frontend via IP or domain name.

Email Notifications (optional/future scope):

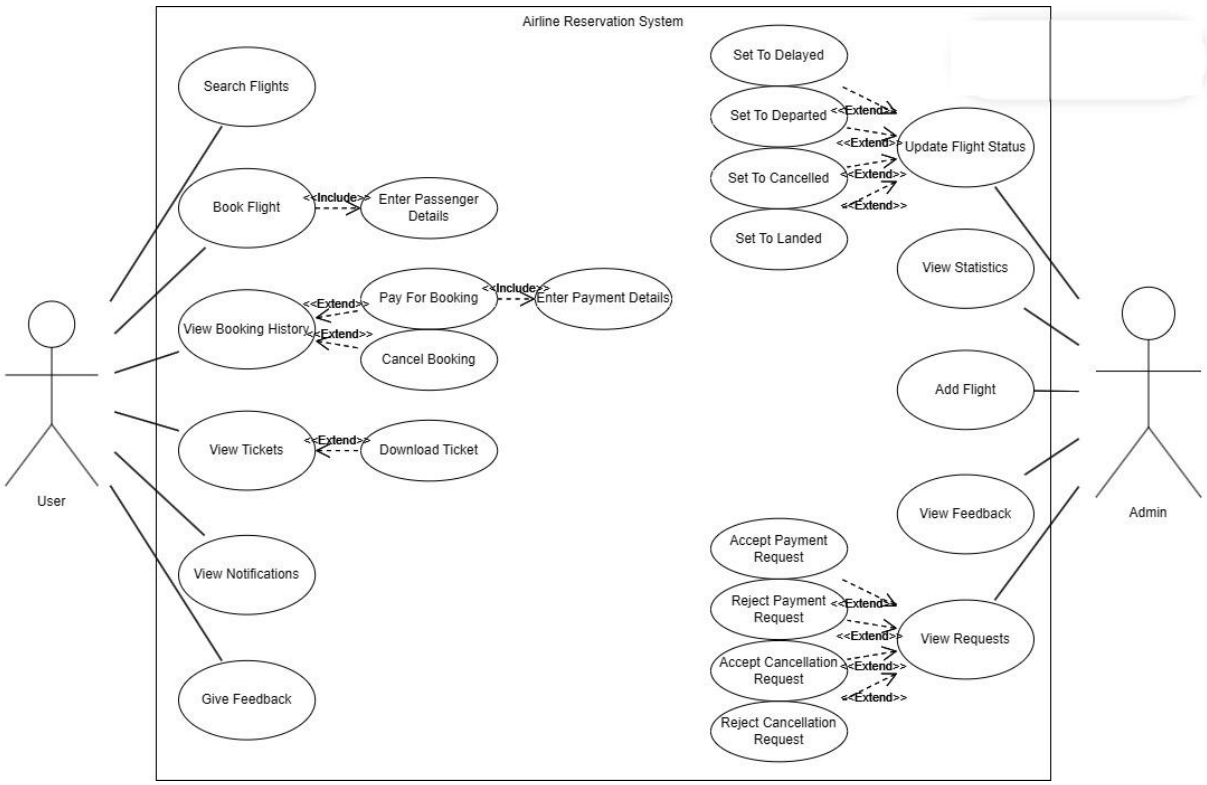
- Email support may be added for sending booking confirmations or request statuses.
- SMTP integration is not currently in scope.

4. Functional Requirements

4.1. Functional Hierarchy

- User Management
    - User Registration
    - User Login/Logout
    - Profile Management
    - Password Recovery
- Flight Management
    - Search Flights
    - View Flight Details
    - Filter Flights (by destination, time, airline, etc.)
- Booking Management
    - Select Flight
    - Enter Passenger Details
    - Confirm Booking
    - Generate Ticket
    - View Booking History
    - Cancel Booking
- Ticket Management
    - View Ticket Details
    - Download/Print Ticket
    - Check-in (optional)
- Feedback and Help
    - Submit Feedback
    - View Help/FAQ
- Admin Panel (if applicable)
    - Manage Flights
    - View Bookings
    - View Feedback
    - Manage Users

4.1.1. Diagram



4.1.2. Use cases

UC01 – Book Flight		
Use case Id:	*UC01*	
Actors: *Registered user*		
Feature: *Book flight*		
Pre-condition:	*User has logged in a searched for flights, and has a list of flights to choose from*	
Scenarios		
Step#	***Action***	***Software Reaction***
1.	*User clicks 'Book' on a selected flight.*	*System opens passenger details form.*
2.	*User enters passenger details.*	*System validates form input.*
Alternate Scenarios: ***2a:*** *Missing/invalid passenger information → System requests user to re-enter passenger details.*		
Post Conditions		
Step#	***Description***	
1.	*Booking details are generated*	
2.	*Booking details are displayed on screen*	
Use Case Cross referenced	*UC02, UC03*	

UC02 – Manage Booking		
Use case Id:	UC02	
Actors: Registered user		
Feature: Manage booking		
Pre-condition:	User is logged in and has at least one booking	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User navigates to 'Booking history'.	System fetches user bookings.
2.	User selects a booking to pay.	System opens payment gateway.
3.	User enters payment details.	System validates payment details.
Alternate Scenarios: **1a:** No bookings exist → System shows appropriate message. **2a:** User selects a booking to cancel → System sends a cancel request to admin and redirects user. **3a:** Invalid payment details → System displays error message.		
Post Conditions		
Step#	**Description**	
1.	System shows success message upon validation of payment details and sends a payment request to admin.	
Use Case Cross referenced	UC01, UC04	

UC03 – Add Flight		
Use case Id:	UC03	
Actors: Admin		
Feature: Add flight		
Pre-condition:	Admin is logged in to dashboard.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	Admin navigates to 'Add Flight' form.	System displays input fields for flight details.
2.	Admin enters flight data and submits.	System validates input and adds record to the database.
Alternate Scenarios: **2a:** Duplicate or invalid flight info → System shows validation error.		
Post Conditions		
Step#	**Description**	
1.	Message informing success is displayed.	
Use Case Cross referenced	UC01	

UC04 – Manage Requests

Use case Id:	*UC04*	
Actors: *Admin*		
Feature:	*Manage requests*	
Pre-condition:	*Admin is logged in to dashboard.*	
Scenarios		
Step#	***Action***	***Software Reaction***
1.	*Admin opens the 'Requests' panel.*	*System fetches pending requests*
2.	*Admin chooses 'Accept' or 'Reject' on a request.*	*System processes the booking and sends a notification to the user.*
Alternate Scenarios: ***1a:*** *No requests found → System shows appropriate message.*		
Post Conditions		
Step#	***Description***	
1.	*Request accepted/rejected message shown in place of the request.*	
2.	*Booking status updated.*	
3.	*Notification sent to user.*	
Use Case Cross referenced	*UC02*	

5. Non-functional Requirements

5.1. Performance Requirements

- The system should load the homepage within 3 seconds on a standard 10 Mbps connection.
- The backend should be able to handle **50 concurrent users** without significant performance degradation.
- Database queries must return results in under 1 second for typical flight search operations.
- The system should maintain **99% uptime** (excluding maintenance periods).
- Booking transactions must be processed with a success rate of **≥ 99.5%**.

5.2. Safety Requirements

- The system should prevent duplicate bookings through validation and database-level constraints.
- All forms must have input validation to prevent erroneous or malicious input.
- Admin-level functionalities should be protected to avoid accidental deletion or modification of critical data.
- In the event of a failure, users should be informed gracefully and not lose entered data where possible.

5.3. Security Requirements

- All user accounts must be authenticated using a secure login system.
- Passwords must be stored in hashed form.
- Only authorized users should be able to access their own data.
- Admin APIs must be protected using role-based access controls.
- Frontend-backend communication should use HTTPS in production environments.
- Input sanitization must be applied to prevent SQL injection, XSS, and other attacks.

5.4. User Documentation

The following user documentation will be delivered with the system:

- **User Manual**: Covers system usage, from login to booking and feedback.
- **Admin Guide**: Covers system management, flight entry, and user oversight.
- **Online Help**: Tooltip-based guidance and FAQ section embedded within the UI.
- **Walkthrough Tutorial**: A step-by-step flow for first-time users on booking flights.

6. References

- IEEE Std 830-1998 – *IEEE Recommended Practice for Software Requirements Specifications*
- Oracle MySQL Documentation – https://dev.mysql.com/doc
- Spring Boot Documentation – https://spring.io/projects/spring-boot
- React Documentation – https://reactjs.org/docs
- GitHub – https://github.com/Q-A-S-I-M/Airline-Reservation-System
- Web Security Guidelines – OWASP Top 10 2023 https://owasp.org

7. Appendices

This section provides additional supporting material relevant to the airline reservation system ("SkyNest") that supplements the main content of this document.

Glossary

- Booking ID – A unique identifier assigned to each confirmed flight reservation.
- Ticket ID – A unique code assigned per passenger per flight, generated upon successful booking.
- Passenger – An individual whose travel information is used to create a booking and issue tickets.
- Flight Search – A system functionality allowing users to query available flights based on input criteria.
- Registered User – A user who has created an account and can perform authenticated actions like booking, canceling, and viewing history.
- Guest User – A non-logged-in user who can only search for flights and view general information.