Software Requirements Specification

for

<Airline Management System>

Version <1.0>

Prepared by

Group Name: ***<Dev Ops>***

<Syed Muhammad Waiz>	**<F2023376117>**	**<f2023376117@umt.edu.pk>**
<Hamza Halim>	**<F2023376113>**	**<f2023376113@umt.edu.pk>**

Instructor: **<Junaid Nasir>**

Course: **<Software Engineering>**

Lab Section: ***<A9>***

Date: **<1/18/2025>**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Syed Muhammad waiz rizvi, Hamza Halim	First version to be created	01/18/25

1 Introduction

1.1 Document Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive description of the functional and non-functional requirements for the Airline Management System (AMS). This document serves as a reference for stakeholders, developers, testers, and project managers, ensuring a clear understanding of the system's objectives, features, and constraints.

1.2 Product Scope

The Airline Management System is designed to streamline and automate various operations involved in airline management, including flight scheduling, ticket booking, passenger management, and crew allocation. The system aims to enhance operational efficiency, improve customer satisfaction, and reduce manual errors. The AMS will cater to airlines of varying sizes, offering scalability and flexibility to meet diverse requirements.

1.3 Intended Audience and Document Overview

This document is intended for the following stakeholders:

- **Project Managers**: To plan and manage the development process.
- **System Developers**: To design and implement the system features.
- **Test Engineers**: To validate the functionality and performance of the system.
- **Airline Staff**: To understand the system's capabilities and provide feedback.
- **Regulatory Authorities**: To ensure the system complies with aviation industry standards.

Document Overview

- **Section 1**: Introduction to the document's purpose, scope, and audience.
- **Section 2**: Overall system description, including features and constraints.
- **Section 3**: Detailed functional requirements.
- **Section 4**: Non-functional requirements and design constraints.
- **Section 5**: Appendices, including definitions and references.

1.4 Definitions, Acronyms and Abbreviations

- **AMS**: Airline Management System
- **API**: Application Programming Interface
- **GUI**: Graphical User Interface
- **IATA**: International Air Transport Association
- **CRM**: Customer Relationship Management

1.5 Document Conventions

- **Bold Text**: Highlights critical terms or sections.
- ***Italic Text***: Emphasizes important notes or examples.
- **Monospaced Text**: Represents code snippets or system commands.
- Requirements are numbered sequentially for easy reference (e.g., R1, R2).

1.6 References and Acknowledgments

- **References**:
    - IATA Regulations and Guidelines (https://www.iata.org)
    - Software Engineering Body of Knowledge (SWEBOK)
    - Industry standards for airline management systems
- **Acknowledgments**:
        - Airline industry professionals for their insights and feedback
        - Development team for their technical contributions

2 Overall Description

2.1 Product Overview

The Airline Management System (AMS) is a software platform developed to manage and optimize various airline operations. It integrates functionalities such as flight scheduling, ticket reservations, passenger check-in, baggage handling, crew management, and reporting. The system provides a user-friendly interface and robust backend to meet the needs of airlines and enhance passenger experience.

2.2 Product Functionality

The key functionalities of the AMS include:

- **Flight Management**: Scheduling and rescheduling flights, tracking delays, and monitoring flight statuses.
- **Reservation System**: Managing ticket bookings, cancellations, and modifications.
- **Passenger Management**: Handling check-ins, seat allocations, and boarding processes.
- **Crew Management**: Assigning and managing crew schedules, monitoring duty hours, and ensuring regulatory compliance.
- **Baggage Tracking**: Managing and tracking passenger baggage to minimize loss or mishandling.
- **Reporting and Analytics**: Providing insights into operational efficiency, customer trends, and financial performance.

2.3 Design and Implementation Constraints

- **Regulatory Compliance**: The system must adhere to aviation industry regulations, such as those mandated by IATA and local aviation authorities.
- **Scalability**: The system should handle varying loads, including peak travel seasons and sudden surges in demand.
- **Security**: The AMS must ensure data security and privacy, complying with standards such as GDPR.

- **Integration**: The system must seamlessly integrate with existing airline infrastructure, including hardware and third-party software.
- **Performance**: The system should provide real-time responses and minimal downtime to ensure smooth operations.

2.4 Assumptions and Dependencies

- Airlines will provide the necessary infrastructure, including servers and network connectivity, for system deployment.
- The system will rely on accurate and up-to-date data from external sources such as weather APIs and airport databases.
- Users (airline staff and passengers) will have basic familiarity with digital platforms.
- The AMS development team will have access to detailed requirements and feedback from airline stakeholders.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interfaces for the AMS are designed to ensure ease of use and accessibility for various types of users. Key interface components include:

- **Web Interface**: A browser-based GUI for passengers to book flights, check in, and manage bookings.
- **Mobile App**: An intuitive interface available for iOS and Android to enable passengers to access all AMS features on the go.
- **Check-in Kiosk Interface**: A touch-based interface with straightforward menus to guide passengers through check-in and boarding pass generation.
- **Admin Dashboard**: A detailed control panel for airline staff to manage schedules, flights, and passenger data efficiently.

3.1.2 Hardware Interfaces

The Airline Management System (AMS) interacts with the following hardware components:

1. **Self-Service Kiosks**:
   - Used for passenger check-in, seat selection, and boarding pass printing.
   - The system communicates with the kiosks via a secure interface to fetch booking details and process seat allocations.
2. **Baggage Handling Systems**:
   - Automates the tracking and management of passenger baggage.
   - The AMS integrates with conveyor systems and RFID scanners to ensure accurate baggage handling.
3. **Mobile Devices (Smartphones and Tablets)**:
   - Enables passengers and airline staff to interact with the system through mobile applications.
   - Hardware features include GPS for location services and barcode scanners for ticket verification.
4. **Workstations for Airline Staff**:
   - Used by staff to manage flight schedules, check-ins, and crew assignments.
   - Requires a robust interface to handle large amounts of data efficiently.
5. **Biometric Scanners**:
   - Facilitates passenger authentication during check-ins and boarding processes.
   - Interfaces with the AMS for real-time identity verification and status updates.
6. **Printers**:
   - Used for printing boarding passes, baggage tags, and operational reports.
   - The AMS communicates with networked or standalone printers to generate outputs.

7. **Sensors and Tracking Devices**:
    - Includes weather sensors, GPS trackers, and altitude monitors.
    - Provides real-time data to the AMS for flight operations and safety measures.
8. **Network Routers and Switches**:
    - Facilitates communication between the AMS and other external systems.
    - Ensures smooth data flow between hardware components and the AMS backend

3.1.3 Software Interfaces

The AMS communicates with other software systems to provide seamless functionality. Key interfaces include:

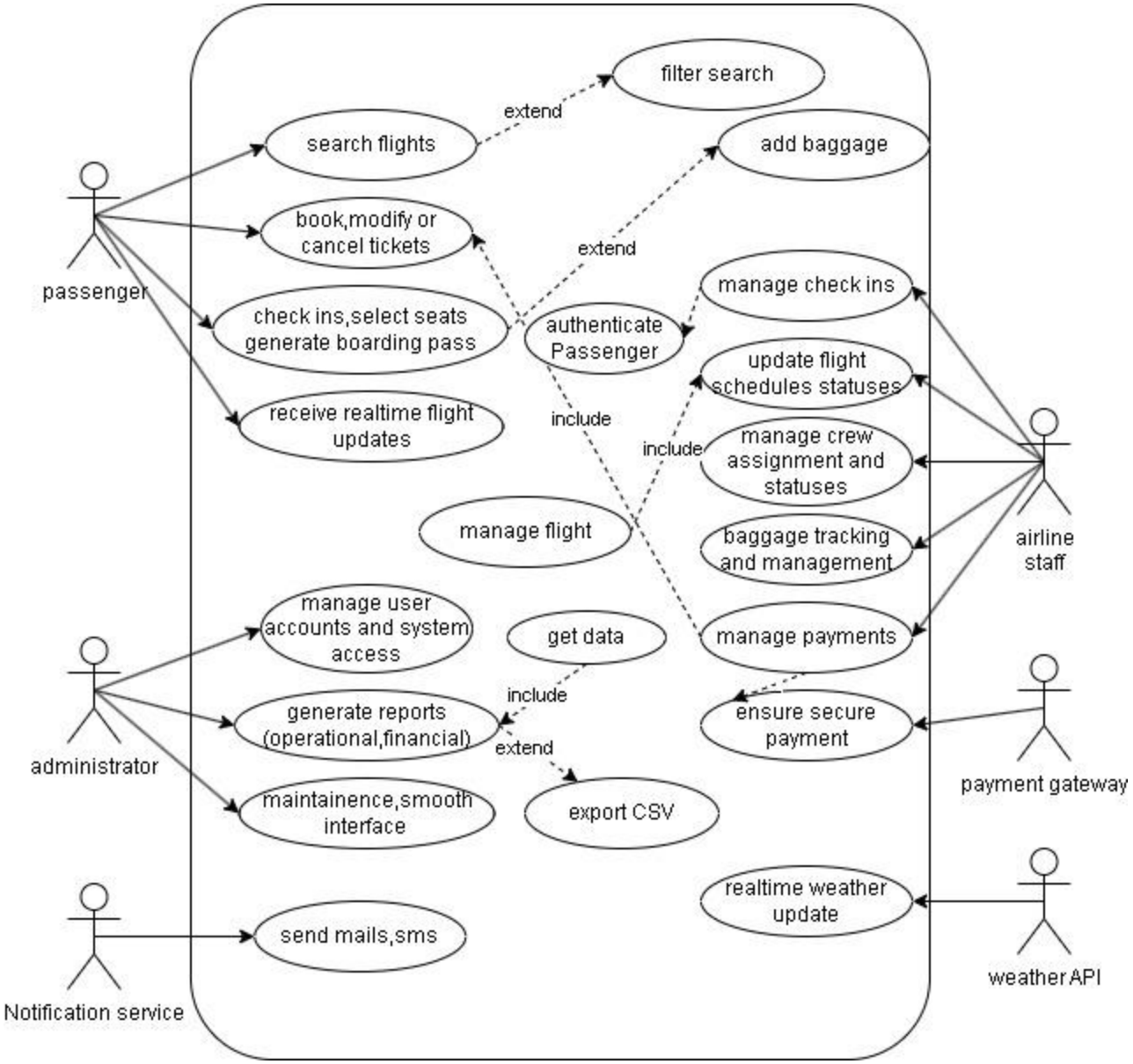
1. **Mobile App Integration**:
    - Allow passengers to manage bookings, receive notifications, and check flight statuses.
    - Supports secure commands and data exchanges with the AMS backend.
2. **Payment Gateway**:
    - Ensures secure processing of ticket transactions via credit cards, PayPal, and other payment options.
3. **Weather API**:
    - Fetches real-time weather updates to inform passengers and optimize flight schedules.
4. **Airport Database Integration**:
    - Syncs flight schedules, gate information, and other operational data.
5. **Notification Services**:
    - Enables communication with passengers via SMS, email, and push notifications.

3.2 Functional Requirements

The functional requirements for the AMS include:

- **R1**: The system allows users to search for flights based on origin, destination, and date.
- **R2**: The system shall enable passengers to book, modify, and cancel tickets.
- **R3**: The system should provide real-time flight status updates.
- **R4**: The system shall support check-in processes, including seat selection and boarding pass generation.
- **R5**: The system shall manage crew schedules and assignments efficiently.
- **R6**: The system shall generate detailed reports on operational metrics, such as flight punctuality and ticket sales.
- **R7**: The system shall ensure secure payment processing for ticket transactions.

3.3 Use Case Model



3.3.1 Use Case: Search Flights

- ***Author**: Syed Muhammad Waiz Rizvi*
- ***Purpose**: Allow passengers to search for flights based on parameters like destination, date, and time.*
- ***Requirements Traceability**: Linked to the requirement for a dynamic flight search feature (R1.1).*
- ***Priority**: High*
- ***Preconditions**: User must have access to the search interface (web or mobile).*
- ***Postconditions**: A list of available flights is displayed.*
- ***Actors**: Passenger*
- ***Extends**: None*
- ***Flow of Events**:*
    1. ***Basic Flow**: User enters search criteria; system retrieves matching flights.*
    2. ***Alternative Flow**: No flights match; system prompts the user to modify criteria.*
    3. ***Exceptions**: System fails to retrieve flight data due to network issues.*
- ***Includes**: "Filter Search"*
- ***Notes/Issues**: Ensure the search feature is optimized for speed and accuracy.*

3.3.2 Use Case: Book, Modify, or Cancel Tickets

- ***Author**: Syed Muhammad Waiz Rizvi*
- ***Purpose**: Provide functionality for passengers to manage bookings.*
- ***Requirements Traceability**: Linked to ticket management (R1.2).*
- ***Priority**: High*
- ***Preconditions**: User must log in and authenticate.*
- ***Postconditions**: Ticket is booked, modified, or canceled, with updates reflected in the database.*
- ***Actors**: Passenger*
- ***Extends**: None*
- ***Flow of Events**:*
    1. ***Basic Flow**: User selects a flight and performs desired action (book, modify, cancel).*
    2. ***Alternative Flow**: Selected flight is unavailable; system prompts for other options.*
    3. ***Exceptions**: Payment fails or session timeout.*
- ***Includes**: "Manage Payments," "Ensure Secure Payment"*
- ***Notes/Issues**: Payment integration should be secure and error-free.*

3.3.3 Use Case: Authenticate Passenger

- ***Author**: Syed Muhammad Waiz Rizvi*
- ***Purpose**: Verify passenger identity during check-ins and other critical actions.*
- ***Requirements Traceability**: Security requirement (R1.3).*
- ***Priority**: High*
- ***Preconditions**: Passenger must have valid credentials.*
- ***Postconditions**: Passenger is authenticated successfully.*
- ***Actors**: Passenger, System*
- ***Extends**: None*
- ***Flow of Events**:*

1. ***Basic Flow**: Passenger provides credentials; system verifies and grants access.*
2. ***Alternative Flow**: Credentials are incorrect; system prompts for retry.*
3. ***Exceptions**: Authentication service is down.*

- ***Includes**: None*
- ***Notes/Issues**: Biometric authentication should be considered for enhanced security.*

3.3.4 Use Case: Manage Check-Ins

- ***Author**: Hamza Halim*
- ***Purpose**: Enable passengers to check in and select seats.*
- ***Requirements Traceability**: Linked to passenger check-in process (R1.4).*
- ***Priority**: Medium*
- ***Preconditions**: Passenger must have a valid booking.*
- ***Postconditions**: Check-in is completed, and boarding pass is generated.*
- ***Actors**: Passenger*
- ***Extends**: None*
- ***Flow of Events**:*
  1. ***Basic Flow**: Passenger checks in, selects a seat, and receives a boarding pass.*
  2. ***Alternative Flow**: Preferred seat unavailable; system suggests alternatives.*
  3. ***Exceptions**: Check-in system unavailable or ticket invalid.*
- ***Includes**: "Authenticate Passenger"*
- ***Notes/Issues**: Ensure seamless integration with kiosks and mobile apps.*

3.3.5 Use Case: Manage Flights

- ***Author**: Hamza Halim*
- ***Purpose**: Provide airline staff tools to manage flight schedules and statuses.*
- ***Requirements Traceability**: Flight management system (R1.5).*
- ***Priority**: High*
- ***Preconditions**: Staff must log in with appropriate credentials.*
- ***Postconditions**: Flight data is updated in the system.*
- ***Actors**: Airline Staff*
- ***Extends**: None*
- ***Flow of Events**:*
  1. ***Basic Flow**: Staff updates flight schedules or statuses (e.g., delays).*
  2. ***Alternative Flow**: Data entry error; system prompts correction.*
  3. ***Exceptions**: Database connection issue.*
- ***Includes**: "Update Flight Schedules Statuses," "Manage Crew Assignments and Statuses"*
- ***Notes/Issues**: Ensure role-based access control.*

3.3.6 Use Case: Baggage Tracking and Management

- ***Author**: Hamza Halim*
- ***Purpose**: Track and manage passenger baggage during travel.*
- ***Requirements Traceability**: Baggage management requirement (R1.6).*
- ***Priority**: Medium*
- ***Preconditions**: Baggage must be tagged and entered the system.*

- ***Postconditions****: Baggage location and status are updated.*
- ***Actors****: Passenger, System*
- ***Extends****: None*
- ***Flow of Events****:*
    1. ***Basic Flow****: System scans baggage tag and updates its location.*
    2. ***Alternative Flow****: Tag unreadable; staff manually input details.*
    3. ***Exceptions****: System cannot locate baggage.*
- ***Includes****: None*
- ***Notes/Issues****: Consider implementing real-time tracking.*

3.4 Sequence diagram

A sequence diagram visually represents how different components interact with each other in the context of a specific scenario. For the Airline Management System (AMS), here's a sequence diagram for a common process: **Flight Booking**.
In this scenario, a **Passenger** searches for flights, selects a flight, and books a ticket. The system interacts with multiple components, including the **AMS backend**, **Payment Gateway**, and **Notification Service**.

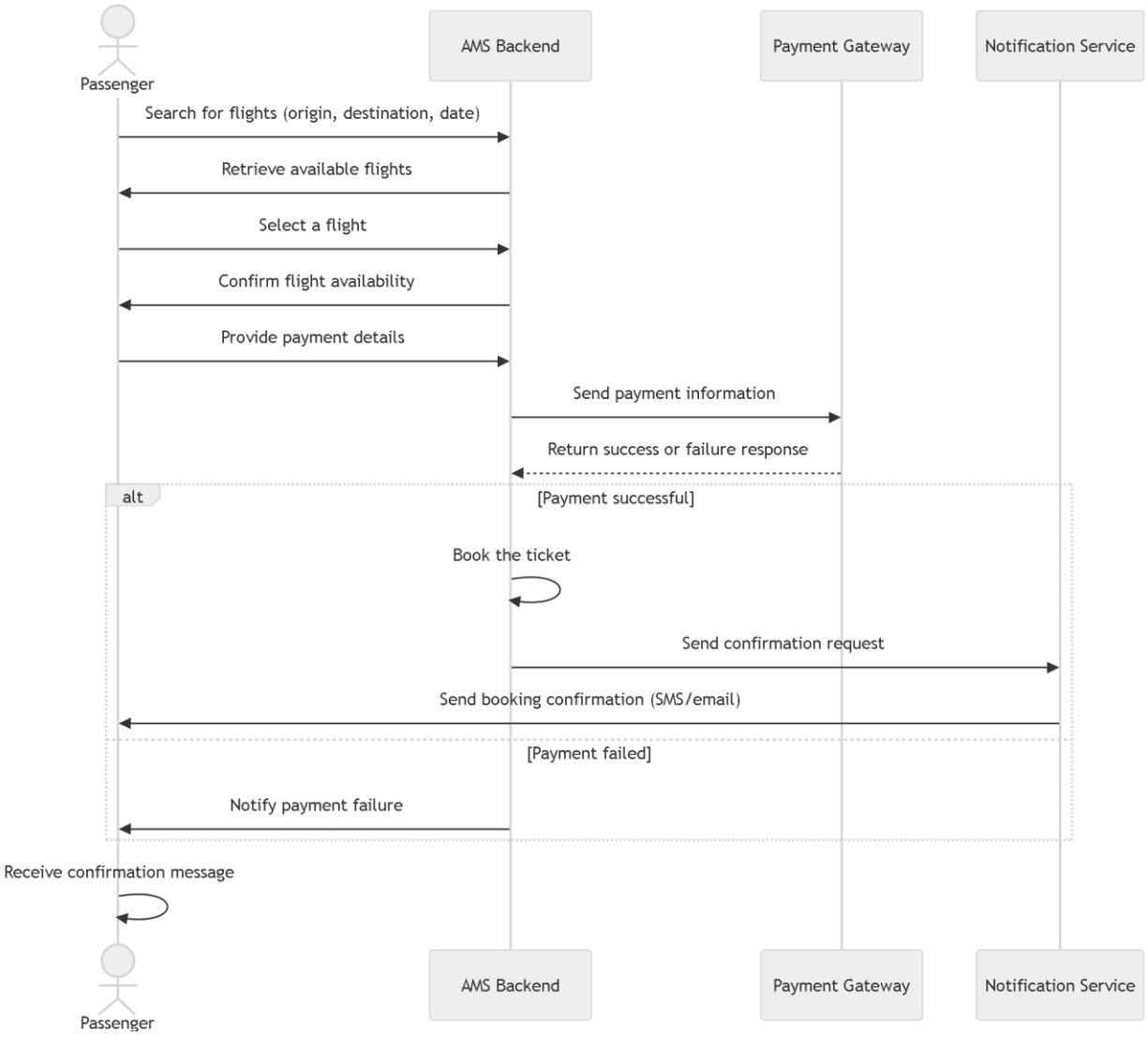
- **Sequence Diagram: Flight Booking**

Actors:

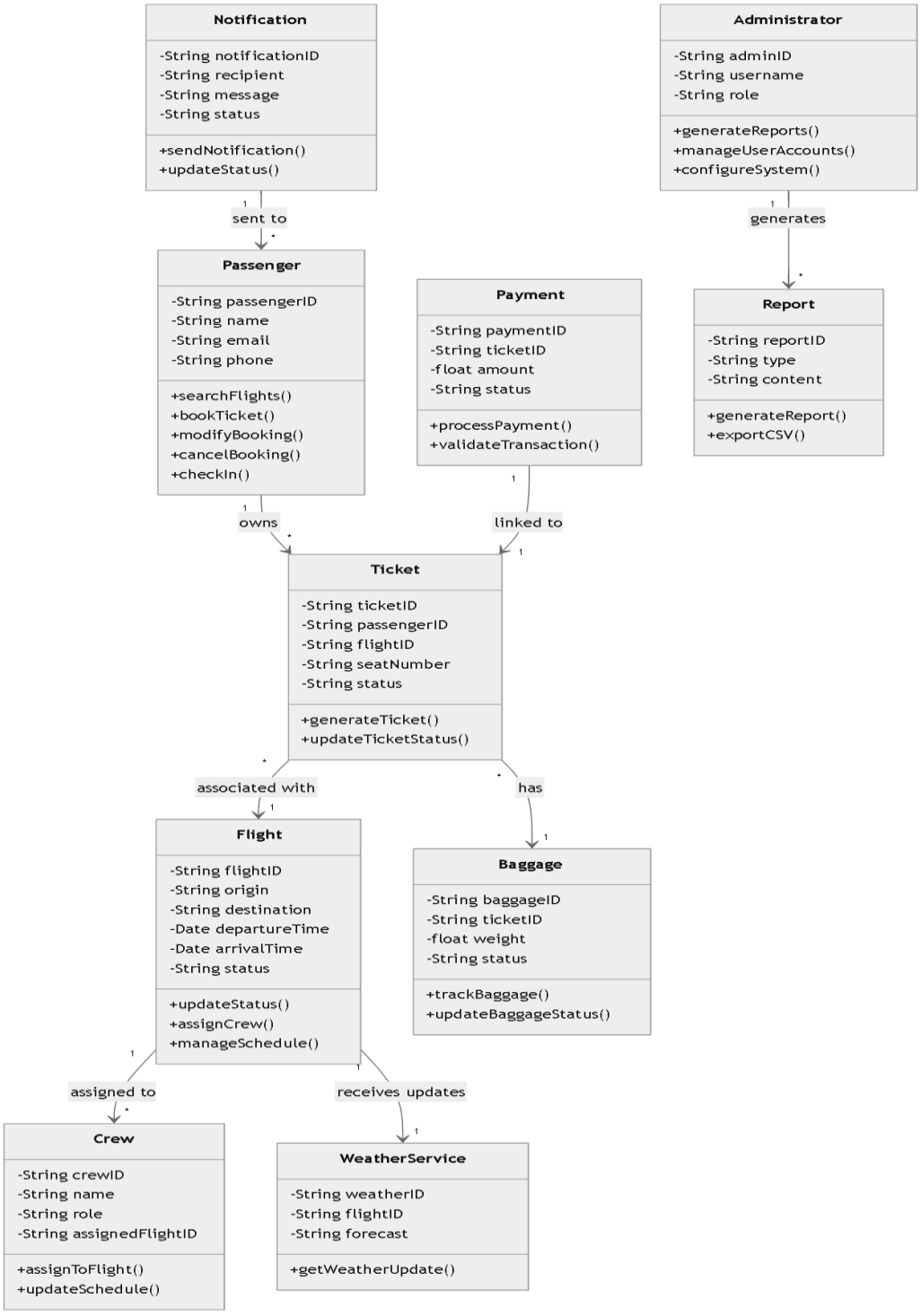
- Passenger
- AMS Backend
- Payment Gateway
- Notification Service

Flow:

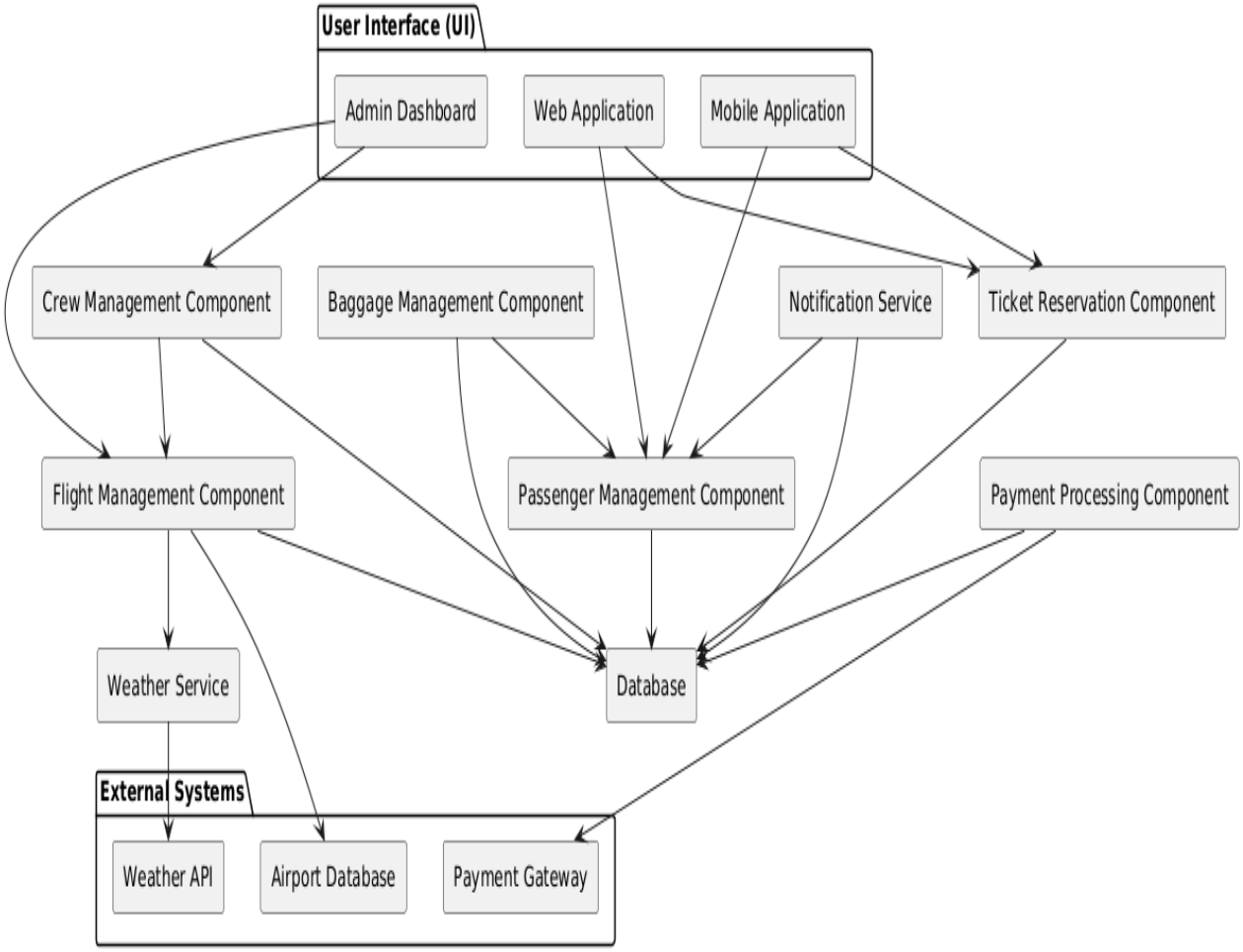
1. **Passenger** *initiates a flight search by providing the origin, destination, and date.*
2. **AMS Backend** *processes the request and retrieves available flights.*
3. **Passenger** *selects a flight.*
4. **AMS Backend** *confirms flight availability and prompts the passengers to proceed with booking.*
5. **Passenger** *provides payment details.*
6. **AMS Backend** *sends the payment information to the* ***Payment Gateway*** *for transaction processing*.
7. **Payment Gateway** *returns a success or failure response.*
8. *If the payment is successful, the* **AMS Backend** *books the ticket and stores the details.*


Passenger
AMS Backend
Payment Gateway
Notification Service
Search for flights (origin, destination, date)
Retrieve available flights
Select a flight
Confirm flight availability
Provide payment details
Send payment information
Return success or failure response
alt
[Payment successful]
Book the ticket
Send confirmation request
Send booking confirmation (SMS/email)
[Payment failed]
Notify payment failure
Receive confirmation message
Passenger
AMS Backend
Payment Gateway
Notification Service

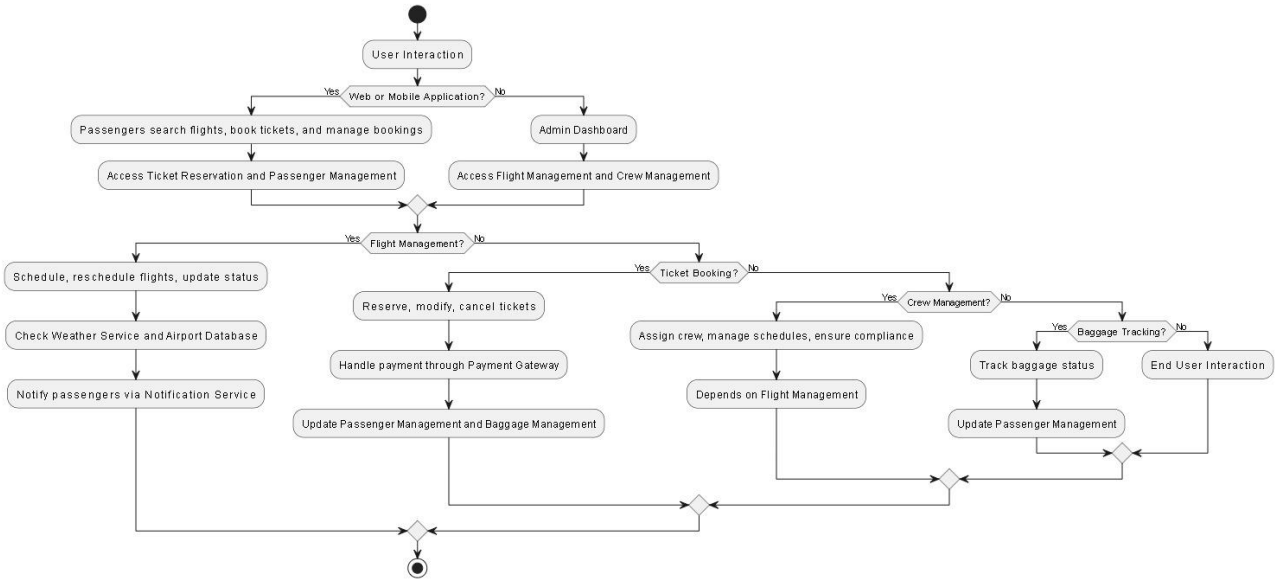
3.5 Class diagram



3.6 Compenent diagram



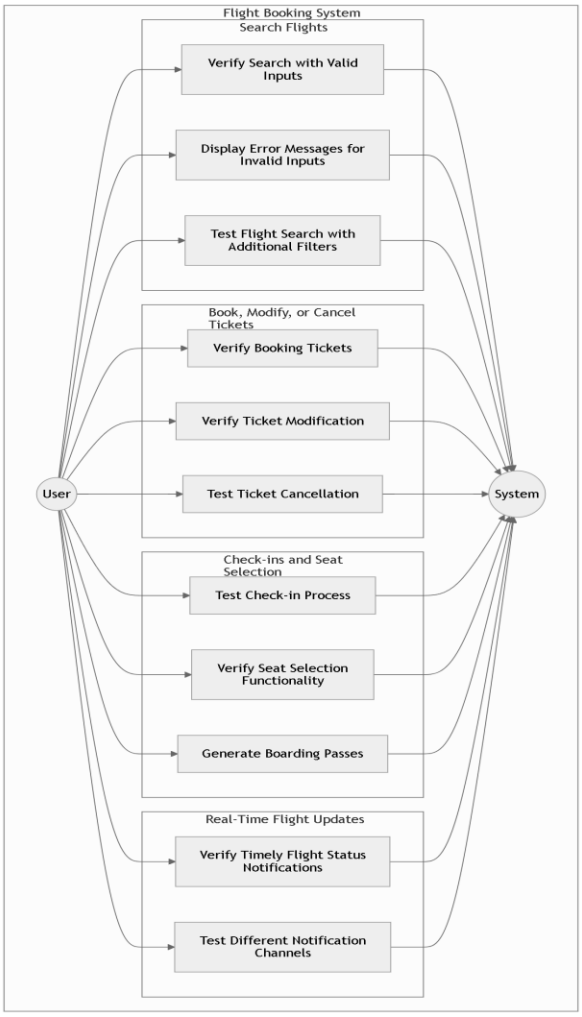
3.7 Flowchart



3.8 Test cases

3.8.1 Passenger Test Cases

1. **Search Flights**
   - Verify the search functionality with valid inputs (e.g., source, destination, date).
   - Verify the system displays error messages for invalid search inputs.
   - Test flight search with additional filters like baggage or class preference.
2. **Book, Modify, or Cancel Tickets**
   - Verify booking tickets for different scenarios (e.g., single trip, round trip).
   - Verify ticket modification (e.g., date, passenger details, seat selection).
   - Test ticket cancellation and ensure proper refund processing.
3. **Check-ins and Seat Selection**
   - Test the check-in process with valid ticket details.
   - Verify seat selection functionality and restrictions (e.g., premium seats).
   - Ensure the system generates boarding passes accurately.
4. **Real-Time Flight Updates**
   - Verify passengers receive timely flight status notifications (e.g., delay, cancellation).
   - Test different notification channels (e.g., SMS, email).



3.8.2 Airline Staff Test Cases

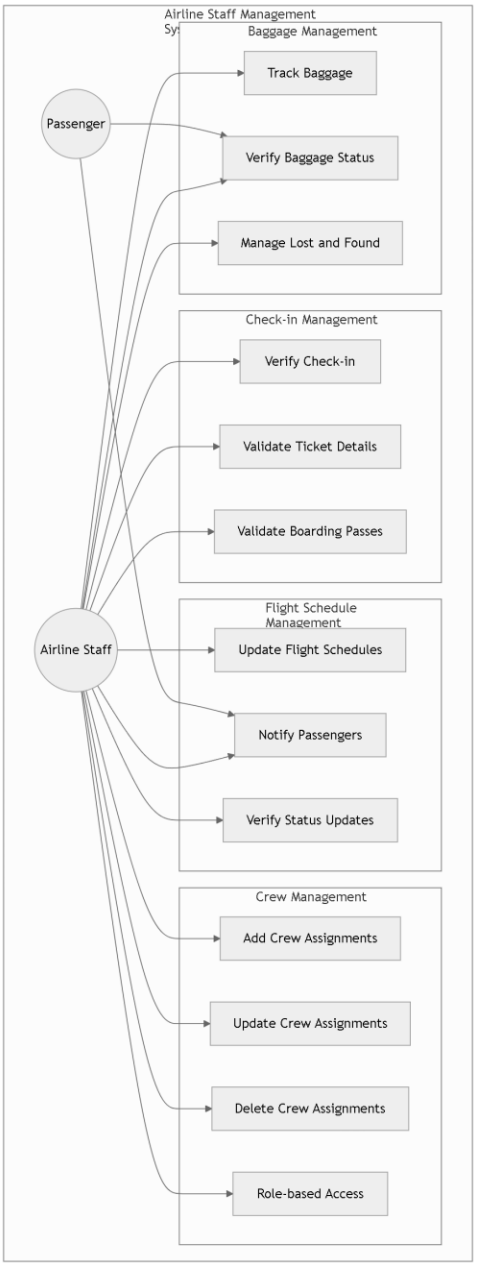
1. **Manage Check-ins**
   - Verify the ability to check in passengers.
   - Ensure proper validation of ticket details and boarding passes.
2. **Update Flight Schedules and Statuses**
   - Test schedule updates and confirm passengers receive real-time notifications.
   - Verify status updates (e.g., flight delayed, on-time, canceled).
3. **Manage Crew Assignments**
   - Verify adding, updating, and deleting crew assignments.
   - Ensure role-based access for crew management.
4. **Baggage Tracking and Management**
   - Test baggage tracking functionality (e.g., lost and found cases).
   - Verify baggage status updates to passengers.

3.8.3 Administrator Test Cases

1. **User Account and System Access Management**
   - Test user account creation, updates, and deletion.
   - Verify role-based access control for different user types (e.g., admin, staff).
2. **Generate Reports**
   - Verify operational and financial report generation.
   - Test exporting reports in different formats (e.g., CSV, PDF).
3. **System Maintenance**
   - Verify system performance under different loads.
   - Test system smoothness during routine maintenance activities.

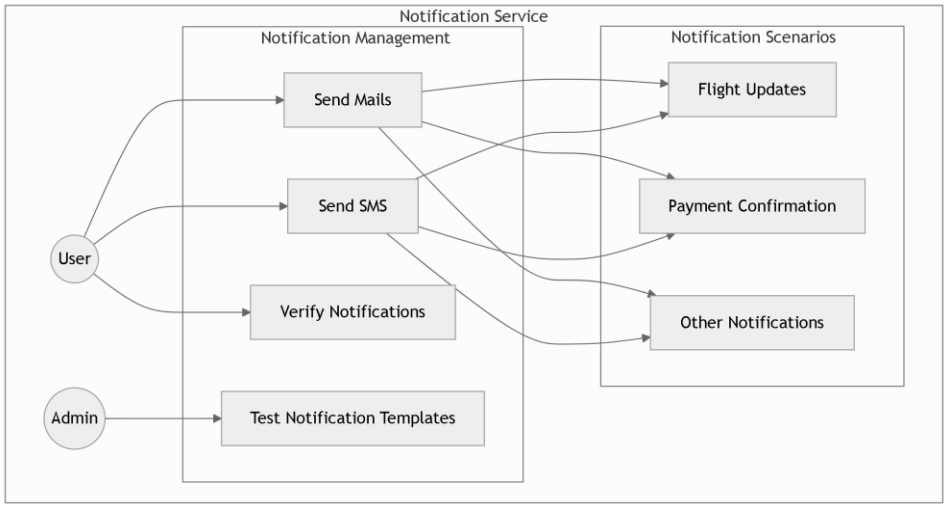
5. Payment Gateway Test Cases

1. **Payment Management**
   - Verify successful payment processing.
   - Test for payment failures and refunds.
2. **Secure Payment**
   - Test secure payment integration using encryption protocols.
   - Verify compliance with payment regulations (e.g., PCI-DSS).



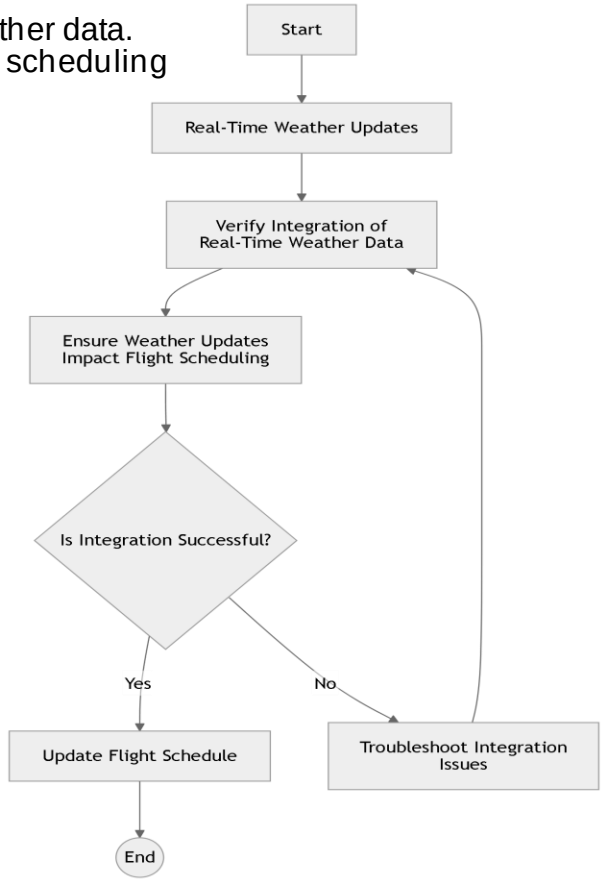
3.8.4 Notification Service Test Cases

1. **Send Mails and SMS**
   - Verify email and SMS notifications for various scenarios (e.g., flight updates, payment confirmation).
   - Test different notification templates.



3.8.5 Weather API Test Cases

1. **Real-Time Weather Updates**
   - Verify the integration of real-time weather data.
   - Ensure weather updates impact flight scheduling appropriately



4 Other Non-functional Requirements

4.1 Performance Requirements

The performance requirements for the Airline Management System (AMS) include:

• The system must process flight search queries within **2 seconds** for **95%** of user requests.

• **AMS** should handle up to 500 transactions per second **(TPS)** during peak operations with a response time of less than **1 second**.

• Real-time updates for flight status and notifications should be propagated to users within **5 seconds** of any change.

• The system shall maintain a **99.9%** uptime annually, ensuring minimal disruption to operations.

• For large-scale operations, the system should support up to **100,000** concurrent users without significant degradation in performance.

4.2 Safety and Security Requirements

Safety and security are critical for the AMS to protect sensitive users and operational data. The requirements are:

• All data transmission must be encrypted using **TLS 1.3** or higher to prevent unauthorized access.

• The system shall implement **multi-factor authentication (MFA)** for all administrative users.

• User data, including **Personally Identifiable Information (PII),** must be stored in compliance with the **General Data Protection Regulation (GDPR)** and other relevant regulations.

• The system shall maintain an audit trail of all user and administrative actions, storing logs for at least one year for forensic purposes.

• **Biometric authentication** must be used for passenger identity verification during check-ins and boarding processes to enhance security.

• The system must implement **role-based access control (RBAC)** to restrict access to sensitive functions based on user roles.

• **Regular vulnerability assessments** and **penetration tests** should be conducted quarterly to ensure the system remains secure against emerging threats.

4.3 Software Quality Attributes

The AMS shall adhere to the following software quality attributes:

4.3.1 **Reliability**:

The system must exhibit fault-tolerant behavior, ensuring that no single point of failure can disrupt operations. In case of failure, the **recovery time objective (RTO**) should not exceed 5 minutes.

4.3.2 **Maintainability**:

The system must allow for modular updates and upgrades, enabling changes to individual components without affecting overall functionality.

4.3.3 **Scalability**:

The AMS should be scalable to accommodate increased traffic, such as during peak travel seasons, without requiring major architectural changes.

4.3.4 **Usability**:

The user interfaces must be intuitive and accessible, meeting standards such as **WCAG** 2.1 for web accessibility.

4.3.5 **Interoperability**:

The system must seamlessly integrate with external systems like payment gateways, airport databases, and third-party **CRM** software via standardized **APIs.**

4.3.6 **Efficiency**:

The AMS backend must optimize resource usage to minimize operational costs, with server load never exceeding **80%** of capacity under normal conditions.

4.3.7 **Adaptability**:

The system must support easy configuration to comply with updates in industry regulations or airline policies.

5 Other Requirements

In addition to functional and non-functional requirements, the following requirements are specified:

- **Legal and Regulatory Compliance**:
  The AMS must comply with international and regional aviation regulations, such as those stipulated by the International Air Transport Association (IATA).
  All payment processing must adhere to PCI-DSS standards to ensure secure handling of credit card transactions
- **Database Requirements**:
  The system should use a distributed database architecture to ensure high availability and redundancy.
  Real-time backups must be implemented, with daily snapshots stored in secure offsite locations.
  The database must support ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure transactional integrity.
- **Disaster Recovery and Business Continuity**:
  The AMS must include a disaster recovery plan to ensure minimal downtime during catastrophic events, with a recovery point objective (RPO) of 15 minutes.
  A secondary data center must be available to take over operations in the event of a primary data center failure.
- **Localization and Internationalization**:
  The system must support multiple languages and currencies to cater to international users.
  Regional date, time, and measurement formats must be configurable based on user preferences.
- **User Training and Documentation**:
  Comprehensive user manuals and training materials must be provided for airline staff and passengers.
  An online help system with FAQs, tutorials, and support contact information must be integrated into the platform.
- **Environmental Considerations**:
  The system must be energy-efficient, utilizing cloud infrastructure that adheres to green computing standards.

Appendix A – Data Dictionary

Data Dictionary A detailed description of the data elements used in the AMS is as follows:
• **Passenger ID**: A unique alphanumeric identifier for each passenger.
• **Flight Number**: A unique identifier for each flight, following the IATA standard format.
• **Booking Reference**: A unique code assigned to each booking, used for retrieval and modification.
• **Baggage Tag ID**: A unique RFID or barcode identifier for tracking passenger baggage.

• **Flight Status**: An enumerated field representing the status of a flight (e.g., Scheduled, Delayed, Canceled, In-flight, Landed).
• **Payment Status**: An indicator of whether a payment has been completed (e.g., Pending, Successful, Failed).

Appendix B - Group Log

Meeting Log:
• **Meeting 1 (01/10/2025):** Initial discussion on project scope, objectives, and team roles.
• **Meeting 2 (01/12/2025):** Finalized functional requirements and system architecture.
• **Meeting 3 (01/15/2025):** Reviewed use cases and sequence diagrams for key operations.
• **Meeting 4 (01/17/2025):** Addressed non-functional requirements and integration constraints.

Individual Contributions:
•**Syed Muhammad Waiz Rizvi**: Authored use cases, defined external interface requirements.
• **Hamza Halim**: Drafted performance and security requirements, prepared appendices.