National University of Computer & Emerging Sciences

"Software Requirements Specification (SRS)"

"Flight Management System"

Instructor:

Miss Fizza Mansoor

Group Members:

ARSH (23K-0078)

MIR AHMED (23I-0142)

Table of Contents

1. Introduction

1.1 Purpose of Document

This document defines the functional and non-functional requirements for Flighty, a Flask-based web application designed to manage flight bookings, user profiles, and administrative tasks for airlines and airports.

1.2 Intended Audience

- Project supervisors
- Software development team
- Testing and QA team
- Airline stakeholders (admins and passengers)

1.3 Abbreviations

- **DBMS**: Database Management System
- **UI**: User Interface
- **RBAC**: Role-Based Access Control
- **API**: Application Programming Interface
- **Font**: Times New Roman
- **Font Size**: 12 pt (body), 14 pt (headings)
- **Bold headings** for sections.

1.4 Document Convention

1. **Font**: Times New Roman
2. **Font Size**:
    a. Normal Text: 18 pt
    b. Headings: 20 pt
3. **Headings**: Bold formatting for all section titles

2. Overall System Description

2.1 Project Background

Airlines and passengers face challenges in managing flight bookings, cancellations, and administrative tasks. Flighty simplifies these processes through a centralized platform for users and administrators.

2.2 Project Scope

- User registration and authentication.
- Flight search, booking, and cancellation.
- Admin management of flights, airports, and airlines.
- User profile management and transaction history.
- Session management and authorization.

2.3 Not in Scope

- Real-time flight tracking.
- Multi-airline ticket integration.
- Seat selection during booking.

2.4 Project Objectives

- Simplify flight booking for passengers.
- Enable admins to manage flights and airports efficiently.
- Ensure secure user authentication and session management.
- Provide transaction history and cancellation options.

2.5 Stakeholders

- Passengers
- Airline administrators

- Developers and testers

2.6 Operating Environment

- **Frontend**: HTML/CSS/JS
- **Backend**: Flask (Python)
- **Database**: MySQL
- **Deployment**: Local development server (localhost:5000).

2.7 System Constraints

- **Software**: Flask, MySQL, Python 3.x.
- **Hardware**: Local server or machine with internet access.
- **Cultural**: English language support only.
- **Security**: Password hashing and session encryption.

2.8 Assumptions & Dependencies

- Users have basic internet access.
- MySQL server is locally hosted.
- Admin credentials are predefined (`admin:admin`).

3. External Interface Requirements

3.1 Hardware Interfaces

- Laptops/desktops with modern browsers (Chrome, Firefox).

3.2 Software Interfaces

- **Frontend**: HTML templates, CSS, JavaScript.
- **Backend**: Flask framework.
- **Database**: MySQL connector for Python.

3.3 Communications Interfaces

- HTTP/HTTPS protocols for web requests.
- RESTful API endpoints for data interaction.

4. Functional Requirements

4.1 Functional Hierarchy

- **User Authentication**
    - Signup, login, logout.
- **Flight Management**
    - Search flights by source/destination.
    - Book/cancel tickets.
- **Admin Functions**
    - Add/remove flights, airports, airlines.
    - View all users.
- **Profile Management**
    - Update user details.
    - View booking history.

4.2 Use Cases

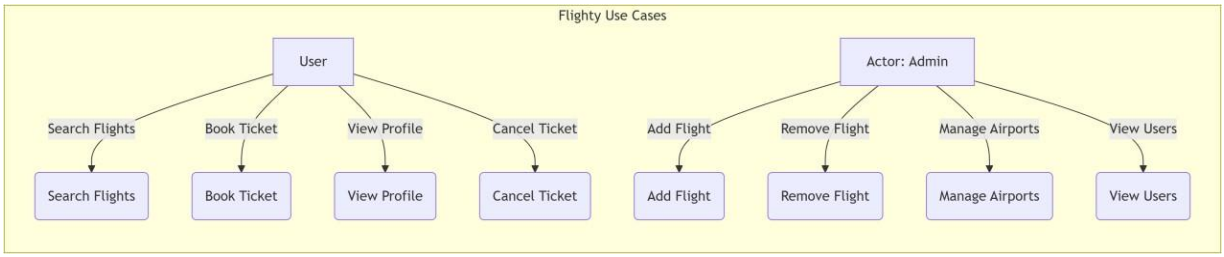
Use Case 1: Book a Flight

- **Actors**: Passenger
- **Pre-condition**: User is logged in.
- **Steps**:
    - Passenger selects source and destination.
    - System displays available flights.
    - Passenger selects flight and confirms booking.

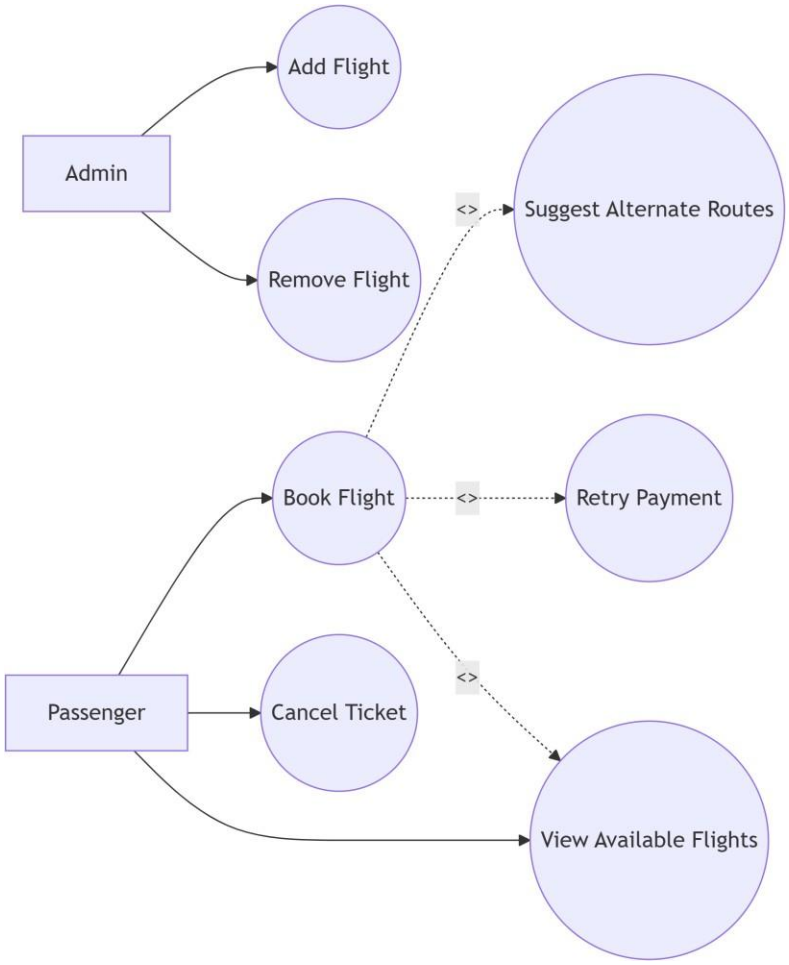
- System generates ticket and updates database.

- **Alternate Scenarios**:
  - No flights available → System suggests alternative routes.
  - Payment failure → Retry or cancel booking.

1. **Use Case Diagram**



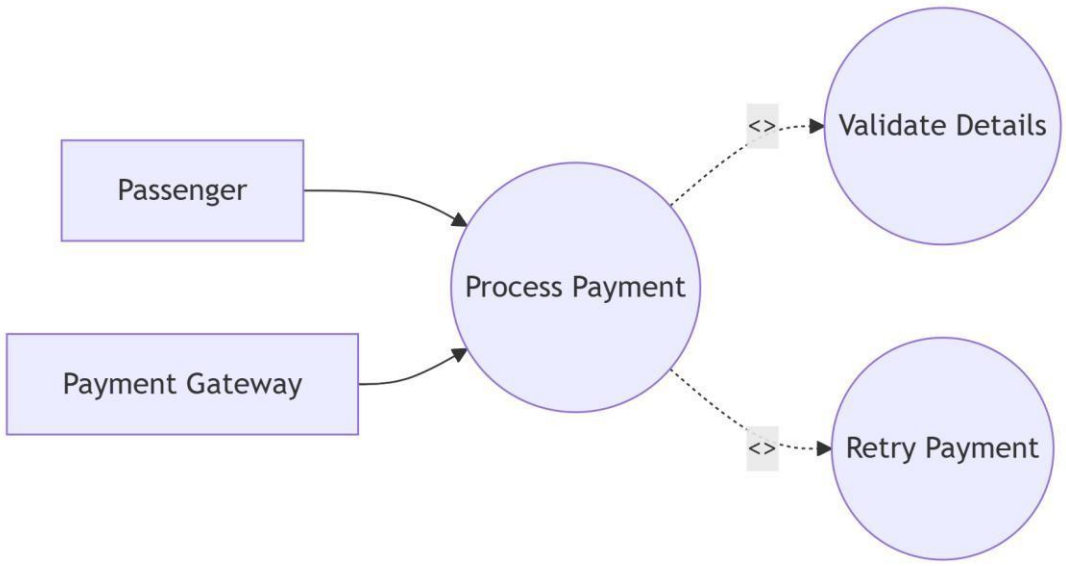
2. **Diagram for booking flight**



Use case Id: 2		
Use case ID:	2 (Book Flight)	
Actors: Passenger		
Feature: Flight Booking Management		
Pre-condition:	User is logged in	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	Passenger selects source/destination	System displays airports for selection
2.	Passenger views available flights	System queries database and lists flights.
3.	Passenger selects flight	System checks seat availability.
4.	Passenger confirms booking	System reserves seat and generates ticket
Alternate Scenarios:		
1a: No flights available → System suggests alternative routes/dates.		

3a: Seat unavailable → System recommends next available flight. | 4a: Payment failure → Retry payment or cancel booking.	
Post Conditions	
Step#	**Description**
1	Ticket is saved in the database
2	Passenger receives confirmation email
Use Case Cross referenced	Managing Appointments
User Authentication, Payment Processing	

3. **Diagram for Payment**



Use case Id: 3		
Use case ID:	3 (Process Payment)	
Actors: Passenger, Payment Gateway		
Feature: Payment Processing		
Pre-condition:	Flight booking initiated	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	Passenger enters payment details	System forwards details to gateway
2.	Payment gateway processes transaction	System validates payment status
3.	Gateway returns approval	System records payment and updates booking
4.	Gateway returns decline	System notifies user of failure
Alternate Scenarios:		
2a: Payment declined → System prompts retry or allows cancellation.		
Post Conditions		
Step#	**Description**	
1.	Payment record is saved in the database	
2.	Ticket is confirmed and confirmation email sent	
Use Case Cross referenced	Flight Booking Management, User Authentication	

References

1. **Flask Documentation**

   Pallets Projects. (2023). *Flask Web Framework*. [Online]. Available: https://flask.palletsprojects.com/

2. **MySQL Documentation**

   Oracle Corporation. (2023). *MySQL 8.0 Reference Manual*. [Online]. Available: https://dev.mysql.com/doc/

3. **Werkzeug Security**

   Pallets Projects. (2023). *Werkzeug: Password Hashing*. [Online]. Available: https://werkzeug.palletsprojects.com/en/3.0.x/utils/#module-werkzeug.security

4. **Role-Based Access Control (RBAC)**

   Ferraiolo, D. F., & Kuhn, D. R. (1992). *Role-Based Access Control*. NIST. [Online]. Available: https://csrc.nist.gov/projects/role-based-access-control

5. **Payment Gateway Integration**

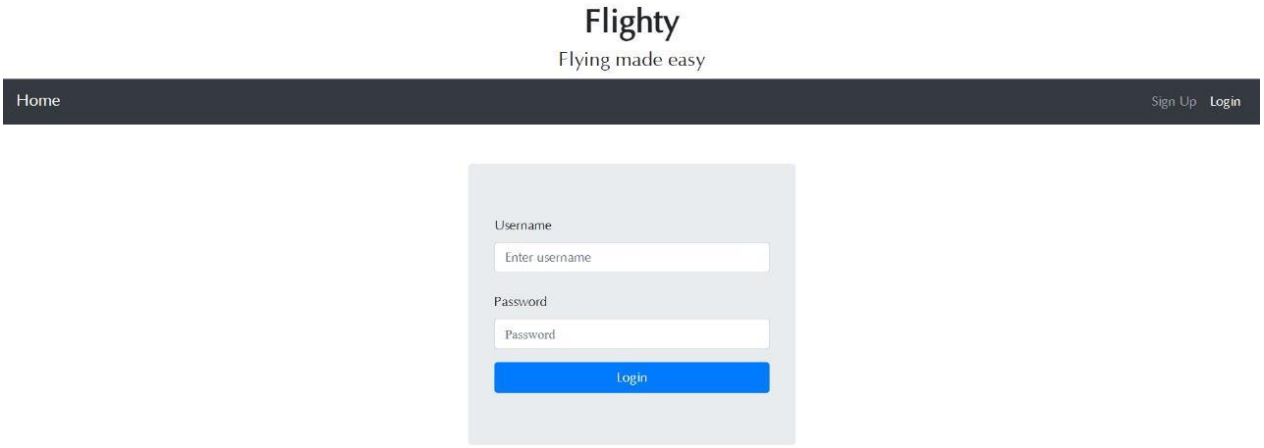
   Stripe Inc. (2023). *Stripe API Documentation*. [Online]. Available: https://stripe.com/docs/api

6. **Web Development Best Practices**

   MDN Web Docs. (2023). *HTML, CSS, and JavaScript Guides*. [Online]. Available: https://developer.mozilla.org/

Appendices:

LOGIC PAGE



REGISTRATION PAGE