Software Requirements Specification for Flight Management System

Version 1.0

Prepared By:

Hassan Adel Hassan

Ahmed Ihab Mahmoud

Mariam Alaa-ElDin Mohamed

Hossam Mustafa El Sayed

Khaled Tareq

Table of Contents

1. Introduction

Survey on the topic

Managing flight bookings, customer details, and ticket records manually is inefficient, time-consuming, and error-prone. The proposed system automates these tasks, providing a fast, secure, and reliable platform for booking flights, managing schedules, and handling customer interactions.

Problem Definition

The need for the system stems from manual processes involved in airline ticket bookings, which are time-consuming and prone to errors. The system will automate the booking and reduce redundant data entry, and ensure better record management.

Scope of the system

The Flight Management System is a web application that allows customers to book, view, and manage their flight tickets, while administrators manage the database. The system includes:

- Secure and reliable online ticket booking.

- Flight and schedule management for users and administrators.

- Unique identification for each user, ticket, and flight.

2. Similar Systems

Comparison with existing systems

Most existing online booking systems from major airlines offer similar functionality. However, they often lack flexibility for administrators in managing flight routes and user data. The proposed system focuses on a user-friendly interface for both administrators and users.

Differences

This system emphasizes the following unique aspects:

- **User-Friendly Interface**: The system features a clean and intuitive interface that allows users to search for flights, view availability, and book tickets seamlessly.
- **Comprehensive Reservation Management**: The system integrates a detailed reservations feature, enabling users to view flight details, including airline names, departure and arrival times, and seat numbers, through secure session-based access.
- **Flexible Seat Allocation**: The platform dynamically assigns seat numbers during the booking process, ensuring an efficient and error-free experience.
- **Secure Authentication**: By leveraging password hashing, the system ensures secure user registration and login functionality.

- **Administrative Efficiency**: Administrators benefit from improved tools for managing flight data, including real-time updates to available seats and streamlined database management, allowing for precise control over airline operations.
- **Data Validation**: The system incorporates validation mechanisms for key user inputs such as passport numbers, email addresses, and usernames to minimize errors during registration.

These features ensure a secure, efficient, and user-focused booking experience while empowering administrators to manage the platform effectively.

3. Glossary

- SQL: Structured Query Language, used for managing relational databases.

- DBMS: Database Management System.

- Administrator: A user with full access to the database to manage flights and users.

- User: A customer who can book and manage flight tickets.

4. User Requirements

Functional Requirements

1. Allow users to sign up and create accounts.

2. Authenticate users via login to access the booking platform.

3. Search for available flight schedules.

4. Book flights with no conflicting schedules.

5. Facilitate flight booking by ensuring real-time seat availability updates.

6. Store user accounts and flight reservations.

7. View flight reservations.

8. Generate tickets automatically for each new reservation.

9. Contact web application admin for support.

10. Allow administrators to add new flight schedules to the system.

11. Allow administrators to update flight details.

12. Allow administrators to delete flights.

Non-Functional Requirements

1. Ensure data security and privacy by implementing secure authentication mechanisms (e.g., password hashing) and encrypting sensitive user data.

2. Provide an easy-to-use and user-friendly interface with intuitive navigation and clear instructions.

3. Support high availability and scalability to handle an increasing number of users and reservations without performance degradation.

4. Maintain session integrity and user authentication to prevent unauthorized access to user accounts.

5. Enable error handling and fault tolerance by providing clear error messages.

5. Constraints

Operating Environment

The system will run on a web server, with the frontend accessible via modern web browsers, and the backend interacting with a SQL database.

Hardware Constraints

No specific hardware constraints, apart from standard server and user-end device requirements (e.g., PC, tablet, mobile).

Software Constraints

The system will use a SQL-based DBMS and must support HTTP communication protocols.

6. System Users

Administrators

Can manage flights and perform administrative functions like adding or removing flights and managing schedules.

Users

Can register on the platform, login, book flights, and view booking history.

7. System Interfaces

User Interfaces

The web interface allows users to register, and book flights. Administrators and airline staff will use a dashboard to manage flight schedules and user data.

Hardware Interfaces
The system will interact with server hardware for data storage and retrieval, requiring no specialized hardware.

Software Interfaces
The system will connect with a SQL-based DBMS for storing user, flight, and ticket data, and communicate with the backend using HTTP.

Communication Interfaces
The system will use HTTP protocols to communicate between the client-side frontend and the server-side backend. Data transfer will be encrypted for security.

8. Software Requirements and Specifications

Use Case 1: Account Creation and Authentication
Actors**:** User

Description: A user creates an account by providing a username, email, passport number, and password. The system validates the input, ensures no duplicates exist, securely hashes the password, and stores the account details in the database. Users can log in using their email and password for access to the system's features.

Use Case 2: Booking a Flight
Actors: User.

Description: The user logs into the system, searches for available flights by specifying origin, destination, and departure date, and selects a desired flight. The system ensures real-time seat availability and prevents conflicting bookings. Upon confirmation, the system assigns a seat number, generates a ticket, and updates seat availability for the selected flight.

Use Case 3: Viewing a Flight
Actors: User

Description: The user logs into their account and views a list of previously booked flights. The system retrieves and displays details such as departure and arrival times, seat numbers, and ticket prices.

Use Case 4: Searching for Flights
Actors**:** User

Description**:** The user inputs flight criteria, including origin, destination, and departure date. The system retrieves and displays matching flights, including airline name, seat availability, ticket prices, and departure/arrival times.

Use Case 5: Managing Flight Schedules
Actors: Administrator, Airline Staff

Description: The administrator or airline staff adds, removes, or updates flight schedules based on operational needs.

Use Case 6: Reservation Viewing by Administrators

Actors**:** Administrator

Description**:** Administrators can access and view user reservations, including ticket details, seat numbers, and associated user accounts.

9. Future Work

Adding real-time flight tracking. Integration with third-party travel services. Expanding payment options (e.g., mobile payments, cryptocurrencies).

10. Work Plan

Work Breakdown Structure (WBS)

User Interface Design

- Design and implement the frontend templates for the user experience.
- Develop pages for account registration (`signup.html`), login (`registration.html`), flight booking (`book.html`), and flight reservation (`reservations.html`).
- Ensure that pages like `availableFlights.html` show flight availability dynamically based on user input.

Database Design and Schema Development

- Design and create database tables for `User`, `Flight`, `Ticket`, `Airport`, `Airline`, and `Reservation`.
- Implement relationships between tables, ensuring that data like user accounts, flight schedules, and reservations are properly linked.
- Establish a query structure to handle the following:
    - User registration, login, and account management (`insert_user`, `check_user`).
    - Flight availability checks and ticket reservations (`available_seats`, `update_seats`, `add_ticket`).
    - Reservation retrieval for logged-in users (`get_reservations`).

Backend Logic Development

- **Signup Functionality:**
    - Implement user registration where users can sign up by providing a valid username, email, passport number, and password (`signup_post`).

    - Validate user inputs such as unique email and username, passport ID format, and hash passwords for security using `bcrypt` (`validate_email`, `validate_username`, `validate_passport_id`).
- **Login Functionality:**
    - Enable user login where users provide their email and password. The system checks credentials and creates a session if valid (`my_form_post`).
- **Flight Search and Booking:**
    - Implement functionality for users to search for available flights based on origin, destination, and departure date (`search`).
    - Display flight results and handle booking, including checking seat availability and assigning seats (`available_seats`, `ticket_price`, `add_ticket`).
- **Flight Management (for Administrators):**
    - Enable flight schedule management by allowing the administrator to add, update, or remove flights (`add_flight`, `update_flight`, `delete_flight`).
- **Reservation Management:**
    - Allow users to view their reservations (`reservations`, `get_reservations`).
    - Allow administrators to manage flight reservations made by users.

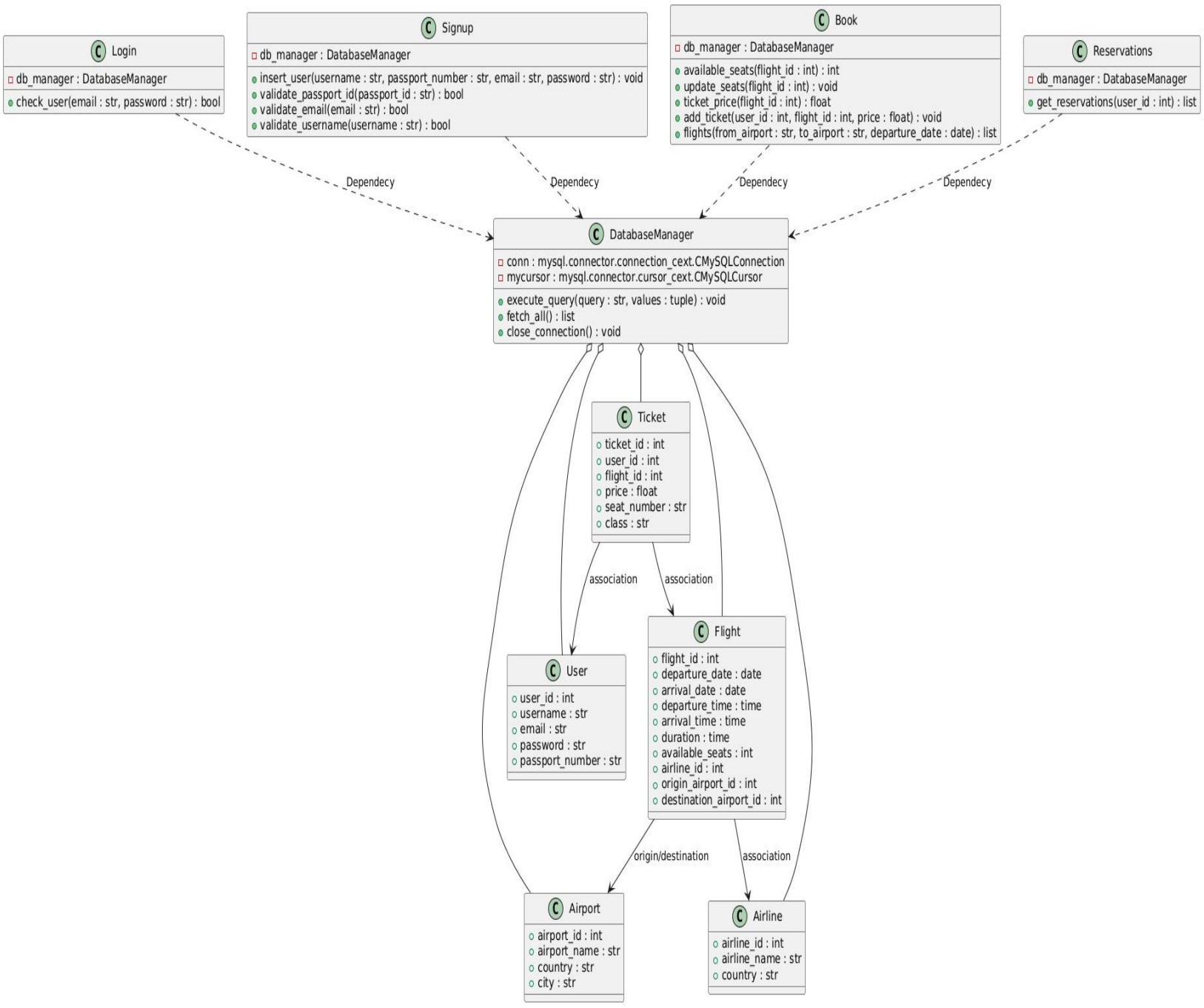
Testing and Bug Fixing

- Test each feature and ensure the system functions as expected:
    - Verify user registration and login functionality.
    - Ensure flight search and seat booking logic works correctly.
    - Test the display of user reservations and booking confirmation.
- Debug and fix issues related to form validation, session management, and flight availability.

11. System Modeling

Class Diagram

Entities: Users, Tickets, Flights, Airlines. Relationships: The diagram will represent how these entities are interconnected (e.g., a User can book multiple Tickets, which are linked to Flights).



Use Case Diagram

A use case diagram visually represents the interactions between actors and the system to achieve specific goals. It helps identify system functionalities and user roles, ensuring clear communication of system behavior.