# 

**SOFTWARE REQUIREMENTS SPECIFICATION (SRS)**

**Project Title:** Airline Reservation System

# Submitted by: Hazal Group

|  |  |
| --- | --- |
| **Unbreen Fatima** | **SU92-BSSEM-F24-351** |
| **Ukasha Abid** | **SU92-BSSEM-F24-147** |
| **M. Saqib Imran** | **SU92-BSSEM-F24-109** |
| **M. Umar** | **SU92-BSSEM-F24-088** |
| **M. Ahmad Naveed** | **SU92-BSSEM-F24-149** |

# Submitted To: Ma’am Ayesha Karim

# Course Name: Object Oriented Programming

# Course Code: CS-1206

# Date of Submission: 05/05/2025

Table of Contents

[1. Introduction 3](#_Toc198580589)

[1.1 Purpose 3](#_Toc198580590)

[1.2 Scope 3](#_Toc198580591)

[1.3 Intended Audience 3](#_Toc198580592)

[1.4 Definitions 3](#_Toc198580593)

[2. Overall Description 3](#_Toc198580594)

[2.1 Product Perspective 3](#_Toc198580595)

[2.2 Product Functions 3](#_Toc198580596)

[2.3 User Characteristics 3](#_Toc198580597)

[2.4 Constraints 4](#_Toc198580598)

[3. Specific Requirements 4](#_Toc198580599)

[3.1 Functional Requirements 4](#_Toc198580600)

[3.2 Non-Functional Requirements 4](#_Toc198580601)

[4. System Features 4](#_Toc198580602)

[4.1 Passenger Registration 4](#_Toc198580603)

[4.2 Admin Management 4](#_Toc198580604)

[4.3 Flight Booking 4](#_Toc198580605)

[4.4 Cancellation & View 4](#_Toc198580606)

[5. Console UI Design 5](#_Toc198580607)

[5.2 Passenger: - Menu Upon successful login, passengers are presented with the following menu: 5](#_Toc198580608)

[5.3 Admin Menu: - After successful login with predefined credentials, the admin is presented with the following menu: 5](#_Toc198580609)

[6. Appendix 6](#_Toc198580610)

[7. Class Diagram / Class Design 6](#_Toc198580611)

[7.1 User (Abstract Base Class) 6](#_Toc198580612)

[7.2 Passenger (Derived from User) 6](#_Toc198580613)

[7.3 Admin (Derived from User) 7](#_Toc198580614)

[7.4 Flight 7](#_Toc198580615)

[7.5 Ticket 8](#_Toc198580616)

[7.6 AirlineSystem 8](#_Toc198580617)

[8. Conclusion 10](#_Toc198580618)

## 1. Introduction

### 1.1 Purpose

The purpose of this project is to develop an Airline Reservation System that allows passengers to register, log in, book and cancel tickets, and view flight information. Administrators can add flights and view all bookings.

### 1.2 Scope

This system is designed for managing domestic or international airline ticketing services through a command-line interface. It supports two user roles: Admin and Passenger.

### 1.3 Intended Audience

• University faculty evaluating the project  
• Students learning software engineering  
• Developers maintaining or expanding the system

### 1.4 Definitions

• Flight: Scheduled aircraft journey from one destination to another.  
• Ticket: A digital confirmation of a flight booking.  
• Passenger: A user who can book flights.  
• Admin: A user who can manage flights and view all tickets.

## 2. Overall Description

### 2.1 Product Perspective

The system is a standalone application with no external dependencies apart from file handling (for saving tickets).

### 2.2 Product Functions

• Passenger registration and login  
• Admin login  
• Add/view flights (admin)  
• Book/cancel/view tickets (passenger)

### 2.3 User Characteristics

Users must have basic knowledge of using terminal or console applications. Admin must know credentials: admin/admin123

### 2.4 Constraints

• Runs on systems with C++ support  
• No GUI (CLI only)  
• File-based data persistence (not database-driven)

## 3. Specific Requirements

### 3.1 Functional Requirements

FR1: The system shall allow passengers to register with a unique username.  
FR2: The system shall authenticate passenger and admin logins.  
FR3: The system shall allow admins to add flights.  
FR4: The system shall allow passengers to book tickets for available flights.  
FR5: The system shall generate a unique ticket ID.  
FR6: The system shall allow passengers to cancel their tickets.  
FR7: The system shall persist ticket information in a text file.  
FR8: The system shall allow admins to view all flights and tickets.

### 3.2 Non-Functional Requirements

NFR1: The system shall be written in C++ and run in a console environment.  
NFR2: The system shall have a user-friendly text-based interface.  
NFR3: The system shall store ticket data in a `.txt` file.  
NFR4: The system shall respond to user inputs within 1 second.  
NFR5: The code shall follow modular OOP design using classes.

## 4. System Features

### 4.1 Passenger Registration

Allows new users to sign up with unique username, name, email, and password.

### 4.2 Admin Management

Admins can log in with fixed credentials and manage flight data and ticket records.

### 4.3 Flight Booking

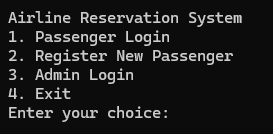
Passengers can select from available flights and book tickets. System generates unique ticket IDs.

### 4.4 Cancellation & View

Passengers can cancel their bookings and view ticket/flight details.

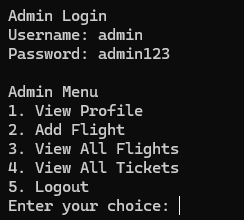
## 5. Console UI Design

**5.1 Main Menu: -** The Main Menu serves as the entry point of the Airline Reservation System. It provides users with four options:



### 5.2 Passenger: - Menu Upon successful login, passengers are presented with the following menu:

### 5.3 Admin Menu: - After successful login with predefined credentials, the admin is presented with the following menu:



## 6. Appendix

Files: tickets.txt  
Tools Used: C++, g++ compiler  
Interface: Console

# 7. Class Diagram / Class Design

## 7.1 User (Abstract Base Class)

**Purpose**: Represents a general user of the system (admin or passenger).

**Attributes:**

**username**: string — Unique login name(public)

**password:** string — Account password(public)

**name**: string — Full name(public)

**email**: string — Email address(public)

**Responsibilities:**

**User(uname: string, pwd: string, n: string, e: string) (Constructor)**

**virtual viewProfile() const : void (Virtual method, overridden in derived classes)**

## 7.2 Passenger (Derived from User)

**Purpose**: Represents a passenger who can book and cancel flights.

**Attributes:**

int bookings[MAX\_BOOKINGS]

int bookingCount

**Methods:**

Passenger(string uname="", string pwd="", string n="", string e="")

void viewProfile() const override

**Responsibilities**:

Passenger(uname: string, pwd: string, n: string, e: string) (Constructor - calls User constructor)

viewProfile() const : void (Overrides User::viewProfile)

## 7.3 Admin (Derived from User)

**Purpose**:Represents the system administrator.

**Attributes:**

Inherits all from User

**Responsibilities**:

Manages flights and views all tickets

Predefined credentials: admin / admin123

Overrides viewProfile()

(Constructor - calls User constructor with default admin details)

## 7.4 Flight

**Purpose**: Represents an available flight for booking.

**Attributes:**

**id**: int — Unique flight ID(public)

**name**: string — Flight/airline name(public)

**source**: string — Departure location(public)

**destination**: string — Arrival location(public)

**date**: string — Scheduled date(public)

**time**: string — Scheduled time(public)

**totalSeats**: int — Total seat capacity(public)

**availableSeats:** int— Remaining available seats(public)

**price**: float — Ticket price(public)

**Methods:**

Flight(int id=0, string name="", string src="", string dest="", string date="", string time="", int seats=0, float price=0)

## 7.5 Ticket

**Purpose**: Represents a booked flight ticket.

**Attributes:**

**ticketId:** string— Unique ticket code (auto-generated) (public)

**flightId:** int — ID of the booked flight(public)

**passengerUsername**: string — Username of the booking passenger(public)

**bookingDate**: string — Date of booking(public)

**totalPrice:** float — Total cost of ticket(public)

**bool isCancelled —** Marks whether the ticket has been cancelled (public)

**Methods:**

* Ticket(string tid="", int fid=0, string user="", string bDate="", float price=0.0)

**Responsibilities:**

* Maintains state of booking (active or cancelled)
* Enables logical deletion of tickets without erasing historical data

## 7.6 AirlineSystem

**Purpose:** Core manager of flights, users, and tickets.

**Attributes:**

* Passenger passengers[MAX\_PASSENGERS]
* int passengerCount
* Flight flights[MAX\_FLIGHTS]
* int flightCount
* Ticket tickets[MAX\_TICKETS]
* int ticketCount
* Passenger\* currentPassenger
* bool adminLoggedIn
* int flightIdCounter

**Public Methods:**

* void mainMenu()
* void adminLogin()
* void adminMenu()
* void registerPassenger()
* void passengerLogin()
* void passengerMenu()
* void bookTicket()
* void viewMyBookings()
* void cancelBooking()
* void addFlightMenu()
* void viewAllFlights()
* void viewAllTickets()
* void viewTicketDetails()
* void aboutUs()

**Private Helper Methods:**

* Passenger\* findPassenger(string username)
* Flight\* findFlight(int id)
* bool hasBookedFlight(int id)
* bool removeBooking(int flightId)
* string generateTicketId()
* string getCurrentDate()
* void displayAvailableFlights()
* void displayTicketDetails(const Ticket&)
* void saveTicketToFile(const Ticket&)
* void saveFlightsToFile()
* void loadFlightsFromFile()
* void savePassengersToFile()
* void loadPassengersFromFile()
* void saveTicketsToFile()
* void loadTicketsFromFile()

**Responsibilities:**

* Handles registration, login, and session management
* Manages flights, bookings, cancellations, and ticketing
* Handles file operations for persistent storage
* Tracks and generates unique IDs and dates

# 8. Conclusion

This Software Requirements Specification (SRS) outlines the functionalities and design of the Airline Reservation System. The document defines the system’s objectives, user roles, features, and console interface. With its clear separation of admin and passenger functionalities, the system aims to deliver an efficient and user-friendly ticket reservation experience through a command-line interface.