PROGRAMMING FUNDAMENTALS

AIRLINE RESERVATION MANAGEMENT SYSTEM REPORT

Submitted By:

- HASSAN MASOOD (2025-SE-487)
- AHMAD HASSAN SHAHID (2025-SE-474)
- HUSNAIN RAFIQUE (2025-SE-488)
- DAUD MASIH (2025-SE-480)

Submitted to:

- PROF SHANFA IRUM

UNIVERSITY OF ENGINEERING & TECHNOLOGY, LAHORE

NEW CAMPUS

Table of contents:

1. Project Objective:

The objective of this project is to design and implement a console-based **Airline Reservation Management System** in C++. The system is designed to streamline operations for two distinct user roles:

1) Administrators 2) **Passengers**.

- **For Administrators:** The system allows management of flight inventory (adding/removing flights), passenger oversight, and the generation of financial and utilization reports.

- **For Passengers:** The system provides functionality to search for flights, book seats, cancel reservations, and view booking history.

2. Data Structures (Structs):

The system utilizes two primary structures to organize data:

A. Structure: Flight

Used to manage the inventory of available flights.

- **int flightID**: Unique identifier for the flight.
- **string flightNumber**: The commercial flight number (e.g., EK-502).
- **string destination, fromLocation**: Route information.
- **string departureTime, arrivalTime**: Schedule details.
- **int totalSeats**: Maximum capacity of the flight.

- **int reservedSeats**: Counter for booked seats.
- **int seatMap[50][6]**: A 2D array representing the seating layout (rows and columns).
- **string* passengerNames**: A pointer used as a dynamic array to store the names of passengers on a specific flight.

B. Structure: Passenger

Used to manage user accounts and their specific bookings.

- **string userID, userPass**: Authentication credentials.
- **string fullName**: Passenger's personal name.
- **int flightID**: ID of the flight the passenger has booked.
- **int seats**: Number of seats reserved by this user.
- **string* bookedPassengerNames**: A pointer used to store the names of multiple passengers booked under one account.
- **string bookingStatus**: Tracks current status (e.g., "Booked", "No Booking").
- **double totalFare**: The total calculated cost of the reservation.

Global Variables

- **Flight flights[maxFlights]**: A static array storing all flight objects.
- **Passenger* passenger**: A pointer used as a dynamic array to store all registered users. This allows the system to resize the array when more users sign up.
- **int flightCount, passengerCount**: Trackers for the current number of records.
- **string adminUser, adminPass**: Hardcoded credentials for admin access.

3. Key Functions

Admin Functions:

- **adminloginPanel( )**: Authenticates the administrator
- **addFlight( ) / removeFlight( )**: Manages the flight inventory array
- **updatePassenger( ) / updateFlight( )**: Modifies existing records
- **removePassenger( )**: Modifies existing records

- **genflightsReport(**ostream& out**)**: Calculates income, seat utilization, and inventory status

Passenger Functions:

- **passengerSignUp( )**: Dynamically resizes the passenger array to register new users / Also used by admin to add new passenger
- **bookFlight(**int passengerIndex**)**: Handles seat selection, fare calculation (Economy/Business), and updates the seatMap
- **cancelReservation(**int passengerIdx**)**: Frees up reserved seats and resets passenger booking data
- **viewFlights( )**: Displays the schedule and status (Available/Full)
- **savePassengerReport( )**: Writes current passenger data to PassengerReport.txt
- **searchPassengerReport( )**: Reads from the file to find specific user history
- **genPassengerReport(**int userIdx**):** This function fulfills the requirement allowing passengers to generate personal booking reports for their own reservations

4. Technical Implementation

Arrays & Pointers

- **Arrays:**
    - Used a fixed list (flights) to store all flight details.
    - Used a grid (seatMap) to track which seats are taken and which are empty.
- **Pointers:**
    - Used dynamic memory for the passenger list so it can grow automatically when more people sign up.
    - Used pointers for names to save memory by only using the exact amount of space needed.

Loops

- **Do-While Loops:** Keep the main program running so it doesn't close until you choose "Exit".
- **For Loops:** Used to search through lists (like finding a flight) and display information.

- **While Loops:**
    - Used for error checking; they make the user try again until they enter the right information.

Control Structures

- **Switch Statements:**
    - Handle the menus (switching between Admin, Passenger, and Reports).
- **If-Else Logic:** Used for important decisions like
    - Checking if a password is correct.
    - Checking if seats are available.
    - Calculating prices (Business vs. Economy).

Input Validation

- **Duplicate Checks**: Stops users from creating IDs that already exist.
- **Range Checks**: Ensures numbers are correct (e.g., picking a seat row between 1 and 50 and col 1-5).
- **Availability**: Prevents booking a flight if it is already full.

File Handling

- **Saving Data** (ofstream): Writes booking history into text files so data isn't lost.
- **Reading Data** (ifstream): Reads the files back to show reports on the screen