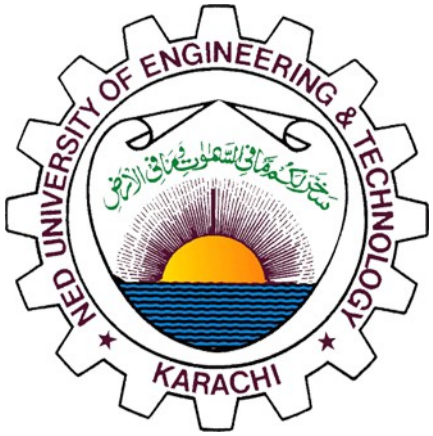
DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-115
Course Title: Computer Programming
Complex Engineering Problem
FE Batch 2025, Fall Semester 2025
Grading Rubric

Group Members:

Student No.	Name	Roll No.
S1	Muhammad Umer	CS-141
S2	Subhan Khan	CS-95
S3	Syeda Aamnah Alam	CS-105

CRITERIA AND SCALES				Marks Obtained		
				S1	S2	S3
Criterion 1: Are the motivation and relevance of the application clearly stated? (CPA-1, CPA-3)						
1	2	3	4			
The motivation & relevance are unclear.	The motivation is clear.	Application relevance is clear.	Both the motivation and relevance are clearly defined.			
Criterion 2: How well is the design (flowchart and algorithms) of the application? (CPA-1, CPA-3)						
1	2	3	4			
The design is difficult to read.	The design is readable only to someone who knows what it is supposed to be doing.	Some part of the design is well organized, while some part is difficult to follow.	The design is well organized and very easy to follow.			
Criterion 3: How did the student perform individually and as a team member? (CPA-2, CPA-3)						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			
Criterion 4: How diverse are shown test cases? (CPA-2, CPA-3)						
1	2	3	4			
The test cases are repeated and simple.	The test cases are distinct but simple.	The test cases are repeated but complex.	The test cases are diverse and complex.			
Criterion 5: Does the report adhere to the given format and requirements?						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially, but is formatted well.	The report contains all the required information, but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

Teacher's Signature



NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY, KARACHI

(Department of Computer System Engineering and Technology)

SECTION: C

COURSE: Computer Programming

COURSE CODE: CS-115

COURSE INCHARGE: Hameeza Ahmed

PROJECT TITLE: Flight Reservation System

ACADEMIC SESSION: FALL-2025

GROUP MEMBERS:

1. MUHAMMAD UMER (CS-141)
2. SUBHAN KHAN (CS-95)
3. SYEDA AAMNAH ALAM (CS-105)

Contents

1.0 Project Contribution

Member 1: Muhammad Umer (CS-142)

This member was primarily responsible for developing the core structure of the program. Their work included setting up the main code framework, ensuring that the menu navigation and overall program flow were functioning smoothly. They also implemented the **Add Flight** module, which involved taking input for new flights, validating the data, and storing it into the file system. In addition to feature development, this member handled the compilation, debugging, proof reading of the report and integration of different parts of the project, ensuring that all components worked together without errors.

Member 2: Subhan Khan (CS-95)

This member worked extensively on the documentation and final presentation of the project. They organized the entire report, structured the chapters according to the required format, and ensured proper styling, numbering, and layout. Along with report compilation, they contributed to coding by implementing the **Delete Flight** functionality and the **View Flights** feature for the admin side. Their role included maintaining clarity and accuracy in both the program's output and the written report.

Member 3: Syeda Aamnah Alam (CS-105)

This member was responsible for developing the user-related functions of the system. Their work included designing and implementing the **Book Ticket** and **Cancel Ticket** modules, as well as the **View Reservations** feature. They additionally worked on the **View Flights** option available to users, ensuring that flight information was properly displayed and updated according to bookings and cancellations. Apart from coding, this member also played a huge role in making the report, assisted in organizing test cases, verifying user-side outputs, and ensuring that the program behaved correctly under different scenarios.

Chapter 1 — Introduction

1.1 Problem Description:

A flight reservation system is a computer program that helps airlines manage every step of selling tickets. It keeps track of how many seats are left on each flight, checks prices, and stores passenger information like names, passport numbers, and contact details. When you search for a flight on a website or app, the system shows available options, calculates fares (including taxes and fees), and, once you choose, creates a booking record and issues a ticket number. It also handles changes, cancellations, seat assignments, and check in. Because it connects to global distribution systems (GDS) and online travel agencies, the same information is visible worldwide, making it possible for anyone including travel agents, airline staff or you to book a seat from anywhere.

1.2 Project Motivation, Relevance & Beneficiaries:

The motivation behind developing this system is to create a functional and practical application using fundamental C programming concepts. Flight reservation systems are widely used in the real world, making this project relevant to real-life scenarios. The beneficiaries include:

- Passengers, who can easily book and confirm flights.
- Airline staff, who can efficiently maintain organized records.
- Students, who gain hands-on experience with file handling and modular programming.
- Travel agencies, who can manage customer bookings systematically.
- Airport management teams, who benefit from streamlined reservation data.

1.3 Distinguishing Features of the Project:

Our Flight Reservation System has a few simple but practical features that make it useful:

- It has a clean, easy to understand menu so the user can move through options without confusion.
- Each major task (booking, searching, updating) is handled separately, which makes the program organized and easier to maintain.
- Passenger information is saved permanently using files, so records do not get lost when the program closes.
- The search option helps quickly find a booking based on name or flight number.
- The program checks for basic input errors, which helps avoid invalid entries.

1.4 Most Challenging Part of the Project:

- The most challenging part of this project was managing the data in a clean and error-free way using file handling. Since the system stores passenger information permanently, we had to make sure that every booking was written correctly, updated properly, and retrieved without mistakes. Handling updates inside the file, preventing duplicate entries, and keeping a fixed format for all records required careful planning.
- Another difficulty was dividing the code into functions so that each function worked smoothly with the others. Passing structures, handling strings, checking invalid inputs, and making sure the program didn't crash due to small mistakes took time and patience. Overall, the biggest challenge was keeping the project simple for users but still well-structured and stable on the technical side.

1.5 New Things Learnt in C:

While working on the Flight Reservation System, several new concepts and practical skills in C programming were learned:

- Understanding and using file handling functions such as fopen(), fclose(), fprintf(), fscanf(), and different file modes (read, write, append).
- Creating and using structures (struct) to store multiple pieces of information together in an organized way.
- Breaking the project into separate functions, passing parameters, and returning values properly.
- Working with strings, including reading full names, handling spaces, and avoiding buffer issues.
- Using loops and conditional statements to design a menu-driven program.
- Learning how to validate user input to avoid incorrect or missing data.
- Understanding the importance of modular programming, which makes the code cleaner and easier to debug.
- Learning how data stored in files behaves differently from data stored in memory.
- Managing pointer concepts indirectly through file pointers and string handling.
- Improving debugging skills by fixing logical errors, syntax errors, and issues in reading/writing data.

Chapter 2 – Methodology

2.1 Overview of Development Approach:

The development of the Flight Reservation System was carried out using a step-by-step approach. The goal was to design a simple, menu-driven program that allows users to enter passenger details and display the stored records. The project was developed entirely in the C programming language, following the basic programming structures taught in the course. Every feature was implemented using separate code blocks to keep the program organized, readable, and easy to debug.

To ensure proper record management, file handling was selected as the primary method of storing information. This allowed the program to save passenger details permanently, even after termination. The entire design of the project revolves around structured data input, validation, and file storage.

2.2 Step-by-Step Development Process:

2.21 Main Menu Creation:

- The first task was to design the main menu that gives two primary options: **Admin Login** and **User Login**.
- A third option, **Exit**, was added to allow users to close the program safely.
- The menu repeats in a loop until the user chooses to exit, ensuring continuous interaction.

2.22 Admin Menu Design:

- After selecting Admin Login, an **Admin Menu** is displayed with options:
    1. Add Flight
    2. Delete Flight
    3. View All Flights
- Each option corresponds to a separate function to maintain modularity and make the code organized.

2.23 User Menu Design

- After selecting User Login, a **User Menu** is displayed with options:
    1. View Flights
    2. Book Ticket
    3. Cancel Ticket
    4. View My Reservations
- Like the admin menu, each choice calls a separate function.

2.24 Adding a Flight (Admin Function)

- When **Add Flight** is chosen, the program collects flight details from the admin: Flight ID, source, destination, date, time, total seats, and ticket price.
- The number of available seats is initially set equal to total seats.
- The data is written to the file **Flights.txt** using **append mode**, ensuring previous records remain intact.
- A confirmation message is displayed after successfully adding a flight.

2.25 Viewing All Flights

- Both admin and user can view all flights.
- The program opens **Flights.txt** in read mode and reads each line using a loop.
- Each flight's details (ID, source, destination, date, time, total seats, available seats, price) are printed in a formatted way.
- If no records exist, a message is displayed indicating that no flights are available.

2.26 Deleting a Flight (Admin Function)

- The admin enters the Flight ID to delete.
- All flight records are first read into an array of structures.
- The program searches for the given Flight ID, removes the corresponding record, and then rewrites the updated flight list back to **Flights.txt**.
- A success or error message is displayed depending on whether the flight was found.

2.27 Booking a Ticket (User Function)

- The user enters the Flight ID and number of seats to book.
- The program checks if the flight exists and if enough seats are available.
- If valid, it reduces the available seats and updates the **Flights.txt** file accordingly.
- A confirmation message is displayed showing the number of seats booked.
- If seats are insufficient, an error message is shown.

2.28 Cancelling a Ticket (User Function)

- The user enters the Flight ID and number of seats to cancel.

- The program verifies the flight exists and that canceling the requested seats does not exceed total seats.
- Available seats are updated in **Flights.txt**, and a success message is displayed.
- If the request is invalid, an error message is shown.

2.29 Viewing Reservations (User Function)

- All reservations are stored in a separate file **Reservations.txt**.
- The program reads this file and prints the Reservation ID, Flight ID, Passenger Name, and Seats Booked.
- If no reservations exist, a message is displayed to indicate this.

2.3 File Handling and Data Integrity

- For all operations (add, delete, book, cancel), the program uses **file handling** to ensure that data is stored permanently.
- Before writing updates to the file, all records are read into arrays of structures to maintain consistency.
- Every modification rewrites the entire file to prevent data corruption.

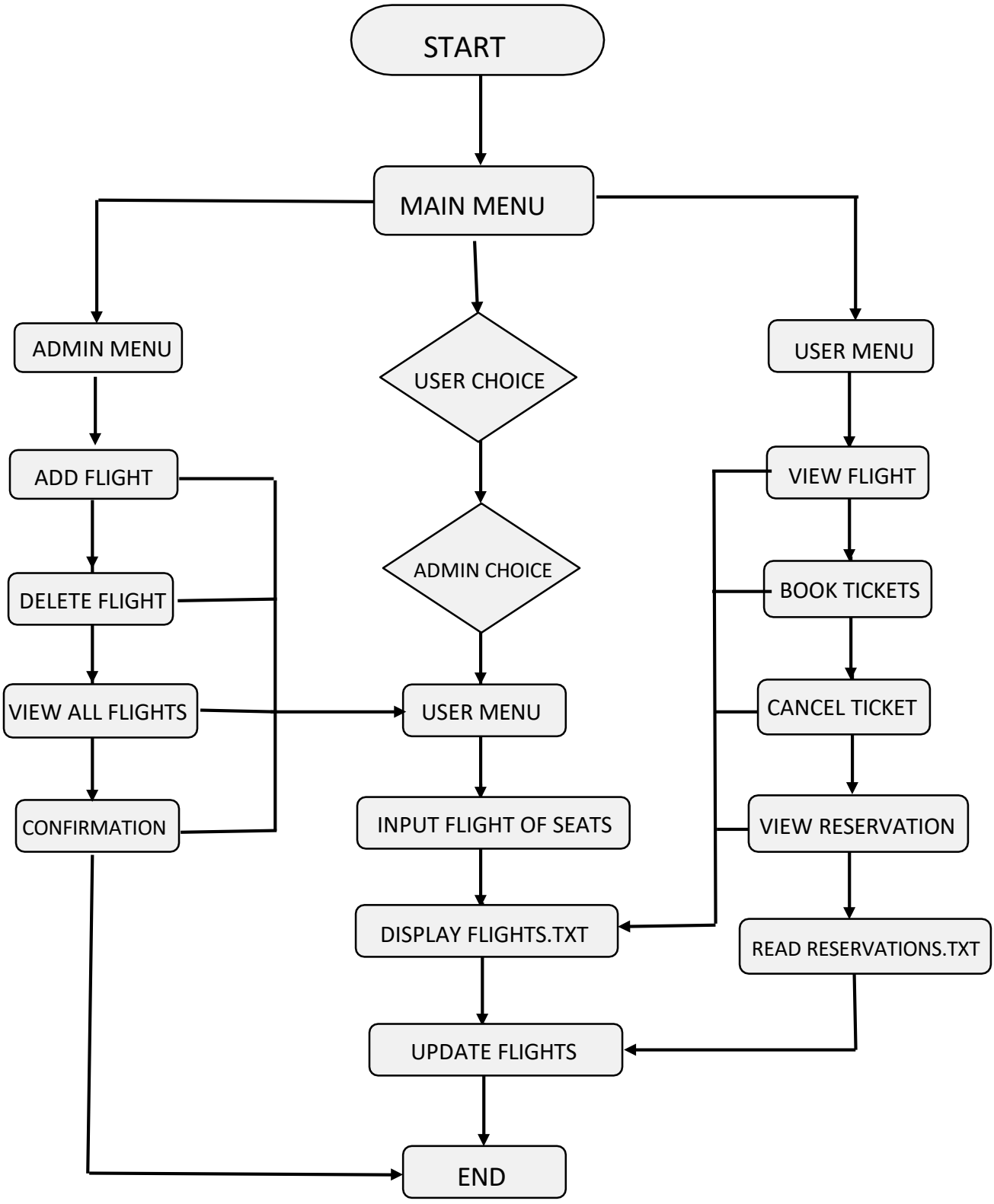
2.4 Error Handling and Validation

- The program checks for invalid menu selections.
- During booking or cancellation, seat availability and limits are verified.
- Messages are displayed to guide users in case of incorrect inputs.

2.5 Looping and User Interaction

- After each operation, the program returns to the main or respective menu, allowing multiple operations without restarting the program.
- This looping ensures continuous interaction until the user chooses to exit.

2.6 FLOWCHART


START
MAIN MENU
ADMIN MENU
USER CHOICE
USER MENU
ADD FLIGHT
VIEW FLIGHT
DELETE FLIGHT
ADMIN CHOICE
BOOK TICKETS
VIEW ALL FLIGHTS
USER MENU
CANCEL TICKET
CONFIRMATION
INPUT FLIGHT OF SEATS
VIEW RESERVATION
DISPLAY FLIGHTS.TXT
READ RESERVATIONS.TXT
UPDATE FLIGHTS
END

2.7 PSEUDOCODE OF FLIGHT RESERVATION SYSTEM:

```
BEGIN

DISPLAY "FLIGHT RESERVATION SYSTEM"

DO
DISPLAY Main Menu:
1. Admin Login
2. User Login
3. Exit

INPUT userChoice

IF userChoice == 1 THEN
DISPLAY Admin Menu:
1. Add Flight
2. Delete Flight
3. View All Flights
INPUT adminChoice

IF adminChoice == 1 THEN
INPUT flight details
OPEN Flights.txt in append mode
WRITE details to file
CLOSE file
PRINT "Flight Added"
ELSE IF adminChoice == 2 THEN
INPUT Flight ID to delete
READ all flights into array
REMOVE flight from array
WRITE updated flights to Flights.txt
PRINT "Flight Deleted"
ELSE IF adminChoice == 3 THEN
```

```
READ all flights from Flights.txt
DISPLAY flight details
ENDIF

ELSE IF userChoice == 2 THEN
DISPLAY User Menu:
1. View Flights
2. Book Ticket
3. Cancel Ticket
4. View Reservations
INPUT userChoice

IF userChoice == 1 THEN
READ Flights.txt
DISPLAY flights
ELSE IF userChoice == 2 THEN
INPUT Flight ID and seats to book
CHECK available seats
IF enough seats THEN
UPDATE available seats
WRITE back to Flights.txt
PRINT "Booking Successful"
ELSE
PRINT "Not enough seats"
ENDIF
ELSE IF userChoice == 3 THEN
INPUT Flight ID and seats to cancel
UPDATE available seats
WRITE back to Flights.txt
PRINT "Cancellation Successful"
ELSE IF userChoice == 4 THEN
READ Reservations.txt
DISPLAY reservation details
ENDIF
```

```
ELSE IF userChoice == 3 THEN
PRINT "Exiting Program"
END
ENDIF

WHILE program not exited

END
```

Chapter 3 -Tools and Platforms Used

This chapter provides an overview of the software tools, programming environment, and platforms utilized during the development of the *Flight Reservation System*. Selecting appropriate tools was essential for ensuring smooth development, easy debugging, structured documentation, and reliable execution of the program.

3.1 Programming Language

C Language:

The project was implemented entirely in **C**, as required in the course. C was chosen due to its high execution speed, efficient memory management, and suitability for structured programming. Its support for functions, file handling, arrays, and pointers allowed us to implement the reservation logic, store flight data, and manage records in a systematic way.

3.2 Development Environment / IDE

3.21 VS Code

VS Code was used to write, compile, and debug the program. It provided features such as:

- Syntax highlighting
- Automatic indentation

- Error and warning messages
- Project management support
- Integrated compiler

These features made the development process more efficient and reduced the time spent on debugging.

3.3 Compiler

3.31 GCC (GNU Compiler Collection)

The project was compiled using the GCC compiler, which is reliable, fast, and widely used for academic programming in C. The compiler ensured compatibility with standard C syntax and provided useful diagnostic information during compilation.

3.4 File Handling Tools / Text Files

The storage of data (such as flight records and user reservations) was managed through simple **.txt files**. These files served as the project's database and were updated each time a flight was added, deleted, booked, or cancelled. Text files were used because:

- They are easy to create and maintain
- They require no external libraries
- They work well with basic C file handling functions (fopen, fprintf, fscanf, fclose)

3.5 Hardware Platform

The software was developed and tested on a standard personal computer equipped with:

- A multi-core processor (Intel/AMD)
- 4GB or more RAM
- Windows operating system

This configuration was sufficient for developing, compiling, and executing the program without any performance issues.

3.6 Documentation Tools

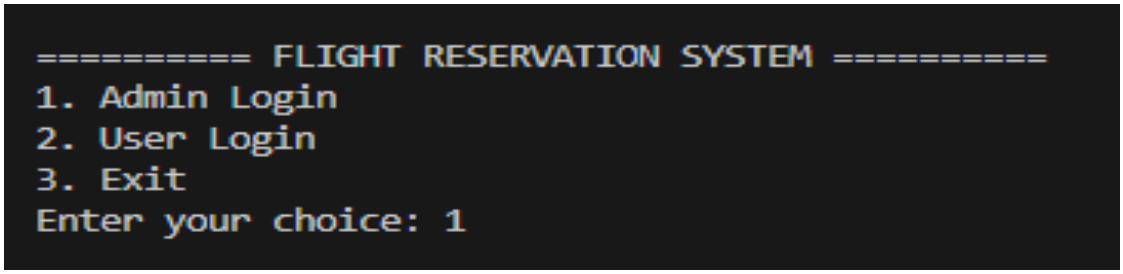
3.61 Microsoft Word

Microsoft Word was used to prepare the project report. Its features such as automatic table of contents generation, section formatting, styles, page numbering, and image insertion helped maintain a clean and professional structure throughout the report.

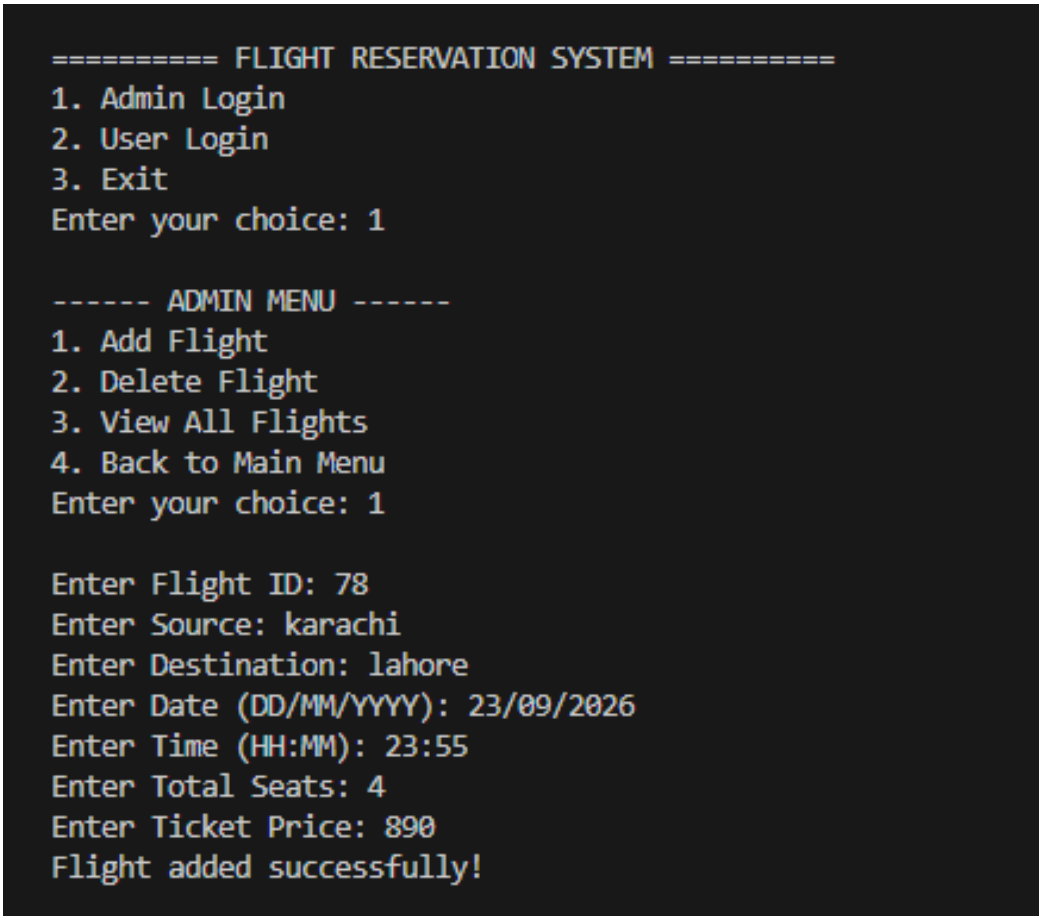
3.7 Additional Utilities

- **Flowchart/Diagram Maker (e.g Canva):** Used to design the flowchart illustrating the overall program flow.
- **Screenshot Tool (Snipping Tool / Windows Screenshot):** Used to capture test case outputs, menus, and program execution results for Chapter 4.

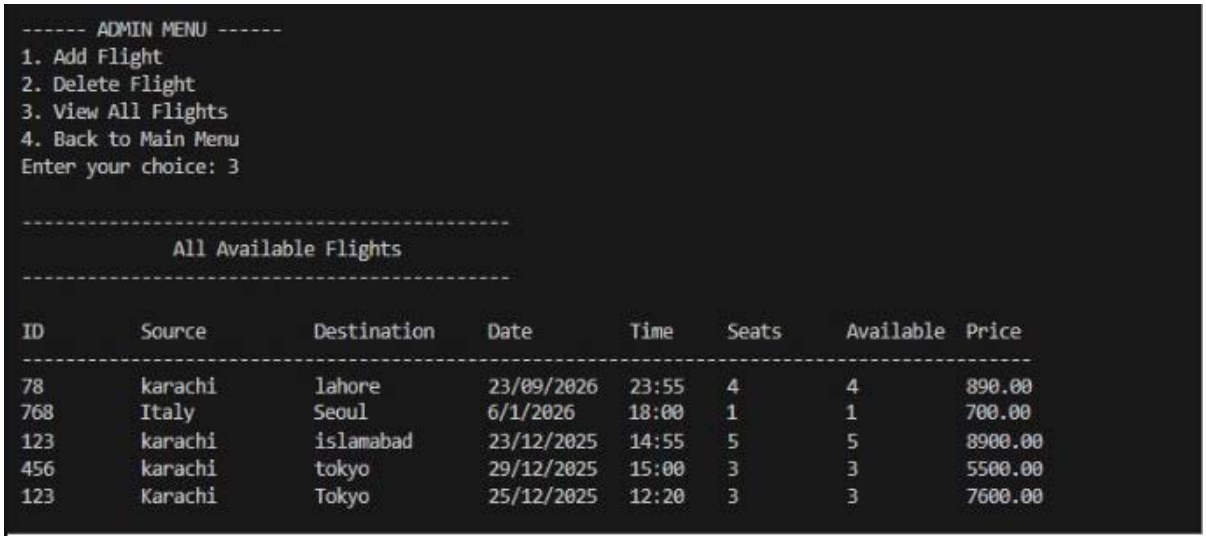
Chapter 4 – Test Cases



This screenshot shows the first screen that appears when the program runs. The system displays three main options: **Admin Login**, **User Login**, and **Exit**. The user is asked to enter a number to choose an option. This menu acts as the starting point of the whole program and directs the user to the desired section.



In this part of the program, the Admin Menu is displayed after selecting the Admin Login option. The screenshot shows the process of adding a new flight. The system asks the admin to enter all required details such as Flight ID, source, destination, date, time, total seats, and ticket price. Once the information is entered, the program saves the flight record into the file.



```
------ ADMIN MENU ------
1. Add Flight
2. Delete Flight
3. View All Flights
4. Back to Main Menu
Enter your choice: 3

-----------------------------------------
            All Available Flights
-----------------------------------------

ID        Source         Destination    Date        Time    Seats     Available  Price
-------------------------------------------------------------------------------------
78        karachi        lahore         23/09/2026  23:55   4         4          890.00
768       Italy          Seoul          6/1/2026    18:00   1         1          700.00
123       karachi        islamabad      23/12/2025  14:55   5         5          8900.00
456       karachi        tokyo          29/12/2025  15:00   3         3          5500.00
123       Karachi        Tokyo          25/12/2025  12:20   3         3          7600.00
```

This screenshot of the program displays the option for viewing all flights. When selected, the system reads the stored flight records from the file and shows details such as Flight ID, source, destination, date, time, seats, and price.

```
2. User Login
3. Exit
Enter your choice: 2

------ USER MENU ------
1. View Flights
2. Book Ticket
3. Cancel Ticket
4. View My Reservations
5. Back to Main Menu
Enter your choice: 2

Enter Flight ID to book: 123

Flight found!
Source: karachi, Destination: islamabad
Available Seats: 5
Price per ticket: 8900.00
Enter number of seats to book: 2

================================
2 seat(s) booked successfully!
Total Price: 17800.00
Remaining Seats on Flight: 3
================================
Enter Passenger Name for reservation: Aamnah
Reservation saved! Your Reservation ID is: 12303
```

Now the user selects the *User Menu* and chooses the option to book a flight. The system asks the user to enter their User ID and the desired Flight ID. After the details are provided, the flight is found and it asks user to enter number of seats and displays available seats and price and finally reservation is done.

```
------ ADMIN MENU ------
1. Add Flight
2. Delete Flight
3. View All Flights
4. Back to Main Menu
Enter your choice: 2

Enter Flight ID to delete: 768

 Flight with ID 768 deleted successfully!
```

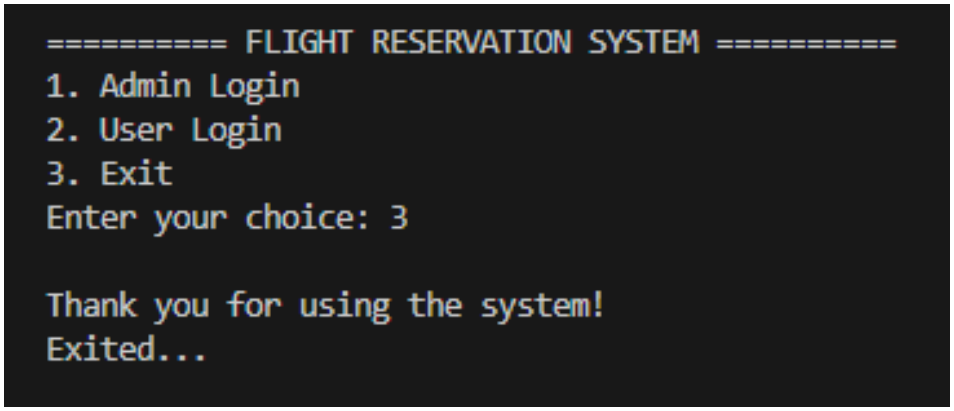
Now the admin wants to delete a flight, so the flight ID is asked and if it is found, it is deleted successfully.

```
------ ADMIN MENU ------
1. Add Flight
2. Delete Flight
3. View All Flights
4. Back to Main Menu
Enter your choice: 3

-----------------------------------------
            All Available Flights
-----------------------------------------

ID        Source         Destination    Date         Time    Seats     Available  Price
-------------------------------------------------------------------------------------
78        karachi        lahore         23/09/2026   23:55   4         4          890.00
123       karachi        islamabad      23/12/2025   14:55   5         3          8900.00
456       karachi        tokyo          29/12/2025   15:00   3         3          5500.00
123       Karachi        Tokyo          25/12/2025   12:20   3         3          7600.00
```

Finally after deleting the flight, the updated flights are viewed.



The user chooses option 3 and the program is exited with a thankyou message.

Chapter 5 - Conclusion and Future Work

5.1 Conclusion

- In this project, we developed a basic *Flight Reservation System* using the C programming language. The aim was to create a menu-driven program that allows an admin to manage flight records and enables users to view flights, book seats, cancel bookings, and check their reservations. Throughout the development, we applied concepts such as functions, arrays, file handling, conditional statements, and loops.
- Completing this project helped us understand how different programming components come together to form a complete application. We also learned how important it is to structure code properly, test each module separately, and ensure that the program handles common user inputs correctly. Working on this system also improved our teamwork and our ability to solve small technical issues that appeared during coding and debugging. Overall, the system works as intended and fulfills the basic requirements of a simple reservation program.

5.2 Future Enhancements

The current version of the system is functional, but it can be improved in several ways. Some possible future additions include:

5.21 Better User Interface

Right now the system runs in the console. In the future, a graphical interface could be added to make the program easier to use

5.22 User Accounts

Instead of booking without login, users could have their own accounts where they can view their booking history and personal details

5.23 Database Support

The system uses text files for storing data. Moving to a proper database would make searching and updating records faster and more reliable.

5.24 Online Functionality

The program could be expanded to work on a network so that multiple users can make reservations at the same time.

5.25 Seat Selection

A feature for selecting specific seats instead of entering only the number of seats could be added.

5.26 Better Admin Features

More options for the admin, such as viewing reports, booking statistics, or revenue summaries, could be included later.

References

1. Let Us C
2. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, 2nd Edition, Prentice Hall, 1988.
3. Tutorialspoint. “C Programming Language Tutorial.” https://www.tutorialspoint.com/cprogramming/index.htm
4. GeeksforGeeks. “C Programming Basics.” https://www.geeksforgeeks.org/c-programming-language/
5. Programiz. “C Programming Guide.” https://www.programiz.com/c-programming