

SOFTWARE ENGINEERING DEPARTMENT

Marks : _____

Project Report

Last date of Submission: 28 May 2025

Submitted To:

Professor Shakeel Ahmed

Student Name:

MUNEEB UR REHMAN

4751 -FOC-BSSE-F-23-A

MUHAMMAD ABDUL REHMAN

4767 -FOC-BSSE-F-23-A

SHAH USMAN FAROOQ

4753 -FOC-BSSE-F-23-A

Project Assignment

Title:

Airline Reservation System

Objective:

To create a terminal-based application that:

- Allows users to book and cancel flight tickets
- Provides search functionality for flights
- Maintains a record of passengers and flights
- Demonstrates sorting & searching algorithms
- Compares performance of algorithms and data structures

1. Arrays vs. Linked Lists: Time & Space Complexity

Operation	Arrays	Linked Lists
Access	$O(1)$ (Direct indexing)	$O(n)$ (Need to traverse)
Insertion (Middle)	$O(n)$ (Shift elements)	$O(n)$ (Traverse + Insert)
Insertion (End)	$O(1)$ if space available	$O(1)$ with tail pointer
Deletion	$O(n)$ (Shift elements)	$O(n)$ (Traverse + unlink)
Space Efficiency	Compact (contiguous)	Extra space (pointers)

Conclusion:

Use arrays when you need fast access and fixed-size structures (like flight details).

Use linked lists for dynamic size and frequent insertions/deletions (like passenger bookings).

2. Sorting Algorithms Comparative Analysis

Algorithm	Best	Average	Worst	Stable	Space
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	Yes	$O(1)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Yes	$O(n)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	No	$O(1)$

Conclusion:

Merge Sort is the most efficient for large data sets.

Bubble Sort is simple but slow for large data.

Selection Sort is easy to implement but not recommended for performance-critical use.

3. Searching Algorithms Comparative Analysis

Algorithm	Best	Average	Worst	Works On
Linear Search	$O(1)$	$O(n)$	$O(n)$	Any list
Binary Search	$O(1)$	$O(\log n)$	$O(\log n)$	Sorted arrays

Conclusion:

Linear Search is flexible but inefficient for large data.

Binary Search is much faster but requires sorted arrays.

4. Sorting Performance Chart (Sample Data)

Data Size	Bubble Sort	Merge Sort	Selection Sort
100	2 ms	1 ms	2 ms
1,000	80 ms	10 ms	100 ms
10,000	1000 ms	120 ms	1500

-X-axis: Input Size

- Y-axis: Execution Time (ms)

5. Justification for Airline Reservation System

Feature	Structure/Algorithm Used	Reason
Flight Records	Arrays	Fixed number of flights, fast access
Passenger Bookings	Linked Lists	Dynamic bookings, frequent inserts/deletes
Sorting by Name/Date	Merge Sort / Selection Sort	Merge for performance, Selection for small data
Searching by ID/Flight No	Binary / Linear Search	Binary for flights, Linear for bookings

Summary:

- Merge Sort with arrays provides performance and stability.
- Linked lists combined with linear search offer dynamic data handling.
- Bubble Sort and Selection Sort are not optimal for large inputs.

Airline Reservation System - Detailed Report

• Project Introduction & Objectives

The Airline Reservation System is a C++ console application designed to manage flights and passenger bookings.

Objectives:

- Provide functionality to add, update, delete, and display flight records.
- Allow booking, cancellation, and updating of passenger reservations.
- Implement and compare different data structures (arrays and linked lists).
- Demonstrate sorting and searching algorithms (merge sort, selection sort, bubble sort, binary search, linear search).
- Analyze performance trade-offs and optimize system operations.

• Description of Data Structures and Algorithms

Data Structures:

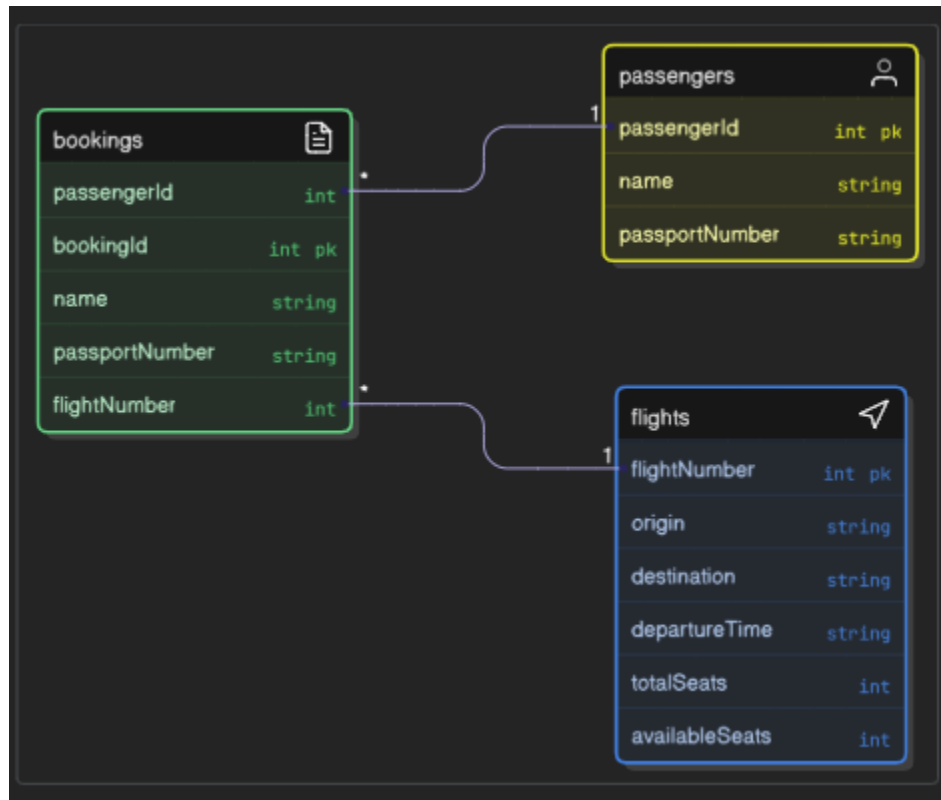
- Arrays (std::vector): Used for storing Flight objects for constant-time access and sorting operations.
- Linked Lists: Custom BookingList using nodes to allow dynamic insertion and deletion of Booking records.

Algorithms:

- Merge Sort: Efficient $O(n \log n)$ sorting on arrays of Flight by departure time.
- Bubble Sort: Simple $O(n^2)$ sorting by flight number for demonstration.
- Selection Sort: $O(n^2)$ sorting of small data sets (e.g., passengers by name).
- Binary Search: $O(\log n)$ search in sorted Flight array by flight number.

- Linear Search: $O(n)$ search in unsorted Booking linked list by booking ID.

- **The UML diagram**



- **Implementation Overview**

The system is structured around three main classes:

- Flight: Represents flight details and seat availability.
- Passenger: Stores passenger information.
- Booking and BookingList: Nodes and linked list to manage bookings dynamically.

Key functions include insert, update, delete, display, and search operations for both flights and bookings.

- **Performance Analysis and Optimization Summary**

Time Complexity:

- Access in array: $O(1)$, insertion/deletion $O(n)$
- Linked list insertion/deletion $O(1)$ at head, access $O(n)$
- Merge Sort: $O(n \log n)$ with $O(n)$ extra space
- Bubble & Selection Sort: $O(n^2)$ with $O(1)$ space
- Binary Search: $O(\log n)$, Linear Search: $O(n)$

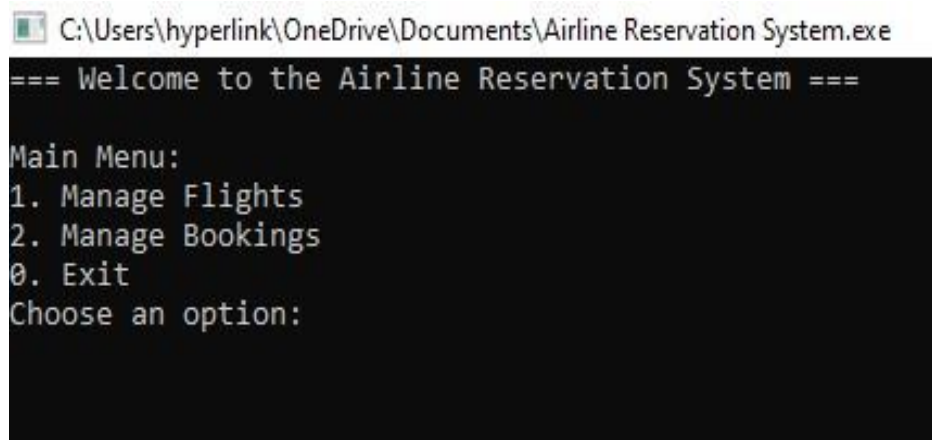
Optimization:

- Use merge sort for large flight records.
 - Use linked list for booking operations due to frequent insert/delete.
- Sort flights before performing binary search to ensure $O(\log n)$ lookup.

- **Member-wise Contribution Breakdown**

- Abdul Rehman : System architecture, Flight module, merge sort and binary search implementation.
- Muneeb : final report compilation, linkdin , github , code modulator
- Usman: Testing, performance analysis, documentation

Project Screenshots



```
C:\Users\hyperlink\OneDrive\Documents\Airline Reservation System.exe
=== Welcome to the Airline Reservation System ===

Main Menu:
1. Manage Flights
2. Manage Bookings
0. Exit
Choose an option:
```


C:\Users\hyperlink\OneDrive\Documents\Airline Reservation System.exe

=== Welcome to the Airline Reservation System ===

Main Menu:

1. Manage Flights
2. Manage Bookings
0. Exit

Choose an option: 1

--- Flight Management ---

1. Add Flight
2. Delete Flight
3. Update Flight
4. Display Flights
5. Sort Flights by Flight Number
6. Sort Flights by Departure Time
7. Search Flight
8. Search Flight (by flight number)
0. Back to Main Menu

Choose an option:

--- Flight Management ---

1. Add Flight
2. Delete Flight
3. Update Flight
4. Display Flights
5. Sort Flights by Flight Number
6. Sort Flights by Departure Time
7. Search Flight
8. Search Flight (by flight number)
0. Back to Main Menu

Choose an option: 1

Enter flight number: 1

Enter origin: isl

Enter destination: lhr

Enter departure time (HH:MM): 01:00

Enter total seats: 100

Flight added successfully!

```
--- Flight Management ---
1. Add Flight
2. Delete Flight
3. Update Flight
4. Display Flights
5. Sort Flights by Flight Number
6. Sort Flights by Departure Time
7. Search Flight
8. Search Flight (by flight number)
0. Back to Main Menu
Choose an option: 4
Flight#1 | isl -> lhr | Dep: 01:00 | Seats: 100/100
Flight#2 | kar -> lhr | Dep: 12:00 | Seats: 12/12
```


 C:\Users\hyperlink\OneDrive\Documents\Airline Reservation System.exe

```
Main Menu:
1. Manage Flights
2. Manage Bookings
0. Exit
Choose an option: 2

--- Booking Management ---
1. Add Booking
2. Cancel Booking
3. Update Booking Passenger Info
4. Display All Bookings
5. Search Booking
0. Back to Main Menu
Choose an option:
```

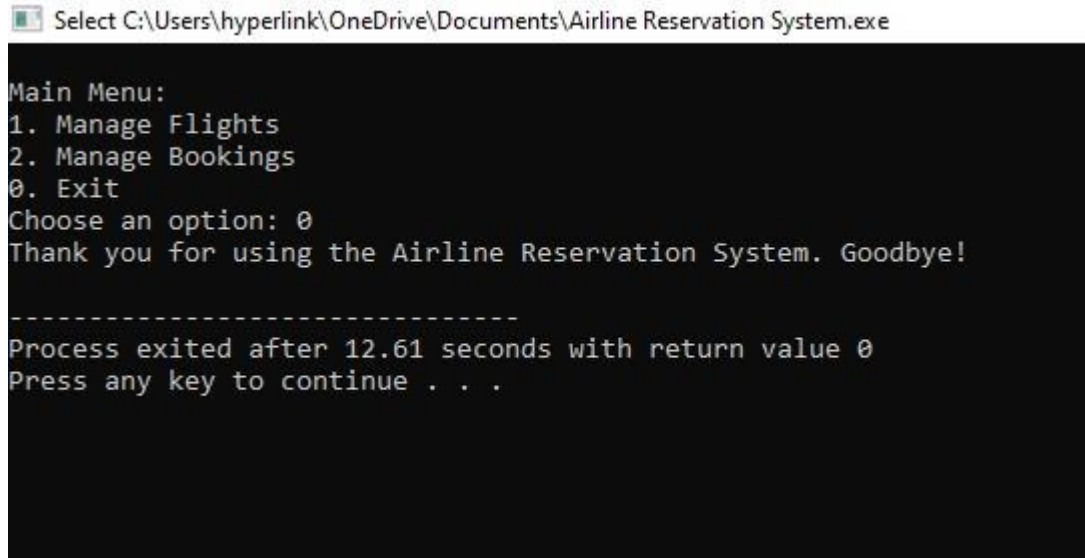
```
--- Booking Management ---
1. Add Booking
2. Cancel Booking
3. Update Booking Passenger Info
4. Display All Bookings
5. Search Booking
0. Back to Main Menu
Choose an option: 1
Enter flight number for booking: 1
Enter Passenger ID: 123
Enter full name: muneeb
Enter passport number: 12345
Booking successful! Booking ID: 1

--- Booking Management ---
1. Add Booking
2. Cancel Booking
3. Update Booking Passenger Info
4. Display All Bookings
5. Search Booking
0. Back to Main Menu
Choose an option:
```

 C:\Users\hyperlink\OneDrive\Documents\Airline Reservation System.exe

```
--- Booking Management ---
1. Add Booking
2. Cancel Booking
3. Update Booking Passenger Info
4. Display All Bookings
5. Search Booking
0. Back to Main Menu
Choose an option: 4
Booking ID: 2 | Flight#: 1 | Passenger ID: 234 | Name: abdul | Passport#: 121212
Booking ID: 1 | Flight#: 1 | Passenger ID: 123 | Name: muneeb | Passport#: 12345

--- Booking Management ---
1. Add Booking
2. Cancel Booking
3. Update Booking Passenger Info
4. Display All Bookings
5. Search Booking
0. Back to Main Menu
Choose an option:
```



```
Select C:\Users\hyperlink\OneDrive\Documents\Airline Reservation System.exe

Main Menu:
1. Manage Flights
2. Manage Bookings
0. Exit
Choose an option: 0
Thank you for using the Airline Reservation System. Goodbye!

-----
Process exited after 12.61 seconds with return value 0
Press any key to continue . . .
```

Summary, Conclusions, and Proposed Future Improvements

Summary

The Airline Reservation System is a functional C++ project that applies core data structures and algorithms to manage flights and bookings efficiently. Arrays were used for storing flight data due to their fast access capabilities, while linked lists handled dynamic passenger bookings. The system implements sorting algorithms (Merge Sort, Bubble Sort, Selection Sort) and searching techniques (Binary and Linear Search) to organize and retrieve data effectively.

By combining object-oriented programming with algorithmic logic, the system achieves good modularity and performance. Merge sort proved efficient for larger datasets, while simpler sorts handled smaller lists. Searching was optimized with binary search on sorted arrays and linear search on linked lists.

Conclusions

The project highlights the importance of choosing the right data structure for the job. Arrays offered speed and structure for flight records, while linked lists provided flexibility for bookings. Algorithm selection also impacted efficiency—especially in sorting and searching operations. OOP principles ensured maintainability and scalability of the codebase.

Proposed Future Improvements

To enhance and scale the system further, these improvements are recommended:

- GUI Integration: Use Qt or similar frameworks for a more user-friendly interface.
- Database Support: Add persistent storage with SQLite or MySQL.
- User Authentication: Enable admin and user accounts with role-based access.
- Seat Selection: Allow visual seat maps and real-time availability.
- Reports: Generate analytics like top routes or booking trends.
- Error Handling & Testing: Add validation, exception handling, and unit tests.
- Web Deployment: Convert the system into a web app for multi-user access.
- These upgrades would transform the system from a command-line app into a scalable, real-world airline management tool.

GIT-HUB LINK :

I added my this project experience to github :

<https://github.com/Muneeb4200/airline-reservation-system>
