

SOFTWARE ENGINEERING DEPARTMENT

Marks : _____

Project Assignment

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

Source code:

```
#include <iostream>
#include <string>
#include <vector>
#include <cctype> // for isdigit
#include <limits> // for std::numeric_limits
```

```

using namespace std;

// Forward declaration
class BookingList;

// Class to represent a flight
class Flight {
public:
    int flightNumber;
    string origin;
    string destination;
    string departureTime; // Format HH:MM
    int totalSeats;
    int availableSeats;

    Flight() : flightNumber(0), origin(""), destination(""), departureTime(""),
totalSeats(0), availableSeats(0) { }

    Flight(int number, const string& org, const string& dest, const string& depTime,
int seats)
        : flightNumber(number), origin(org), destination(dest),
departureTime(depTime), totalSeats(seats), availableSeats(seats) { }

    void display() const {
        cout << "Flight#" << flightNumber << " | " << origin << " -> " << destination
        << " | Dep: " << departureTime << " | Seats: " << availableSeats << "/" <<
totalSeats << endl;
    }
};

```

```

    }
};

// Array-based collection of flights
class FlightArray {
private:
    vector<Flight> flights;

public:
    // Insert a new flight
    void insertFlight(const Flight& flight) {
        flights.push_back(flight);
    }

    // Remove flight by flight number
    void removeFlight(int flightNumber) {
        for (auto it = flights.begin(); it != flights.end(); ++it) {
            if (it->flightNumber == flightNumber) {
                flights.erase(it);
                cout << "Flight#" << flightNumber << " removed successfully." << endl;
                return;
            }
        }
        cout << "Flight not found." << endl;
    }
}

```

```
// Update flight details
```

```
void updateFlight(int flightNumber, const string& newOrigin, const string&  
newDestination, const string& newDepTime, int newSeats) {
```

```
    for (auto& f : flights) {
```

```
        if (f.flightNumber == flightNumber) {
```

```
            f.origin = newOrigin;
```

```
            f.destination = newDestination;
```

```
            f.departureTime = newDepTime;
```

```
            int seatsDiff = newSeats - f.totalSeats;
```

```
            f.totalSeats = newSeats;
```

```
            f.availableSeats += seatsDiff;
```

```
            cout << "Flight#" << flightNumber << " updated successfully." << endl;
```

```
            return;
```

```
        }
```

```
    }
```

```
    cout << "Flight not found." << endl;
```

```
}
```

```
// Linear search for flight by number
```

```
Flight* linearSearch(int flightNumber) {
```

```
    for (auto& f : flights) {
```

```
        if (f.flightNumber == flightNumber)
```

```
            return &f;
```

```
    }
```

```
    return nullptr;
```

```
}
```

```
// Binary search for flight by number (flights must be sorted by flight number)
```

```
Flight* binarySearch(int flightNumber) {  
    int left = 0, right = (int)flights.size() - 1;  
    while (left <= right) {  
        int mid = left + (right - left) / 2;  
        if (flights[mid].flightNumber == flightNumber)  
            return &flights[mid];  
        else if (flights[mid].flightNumber < flightNumber)  
            left = mid + 1;  
        else  
            right = mid - 1;  
    }  
    return nullptr;  
}
```

```
// Bubble sort flights by flight number
```

```
void bubbleSortByFlightNumber() {  
    int n = (int)flights.size();  
    for (int i = 0; i < n - 1; i++) {  
        for (int j = 0; j < n - i - 1; j++) {  
            if (flights[j].flightNumber > flights[j + 1].flightNumber)  
                swap(flights[j], flights[j + 1]);  
        }  
    }
```

```

    }
}

// Quick sort flights by departure time (lexicographical order)
void quickSortByDepartureTime(int left, int right) {
    if (left < right) {
        int pi = partition(left, right);
        quickSortByDepartureTime(left, pi - 1);
        quickSortByDepartureTime(pi + 1, right);
    }
}

// Display all flights
void displayAll() {
    if (flights.empty()) {
        cout << "No flights available." << endl;
        return;
    }
    for (const auto& f : flights)
        f.display();
}

// Get number of flights (needed for quicksort index)
int getSize() {
    return (int)flights.size();
}

```

```
}
```

```
private:
```

```
// Partition function for quicksort (departureTime)
```

```
int partition(int left, int right) {
```

```
    string pivot = flights[right].departureTime;
```

```
    int i = left - 1;
```

```
    for (int j = left; j < right; j++) {
```

```
        if (flights[j].departureTime < pivot) {
```

```
            i++;
```

```
            swap(flights[i], flights[j]);
```

```
        }
```

```
    }
```

```
    swap(flights[i + 1], flights[right]);
```

```
    return i + 1;
```

```
}
```

```
};
```

```
// Class representing a passenger
```

```
class Passenger {
```

```
public:
```

```
    int passengerId;
```

```
    string name;
```

```
    string passportNumber;
```



```
Passenger() : passengerId(0), name(""), passportNumber("") { }
```

```
Passenger(int id, const string& n, const string& pass) : passengerId(id), name(n),  
passportNumber(pass) { }
```

```
void display() const {
```

```
    cout << "Passenger ID: " << passengerId << " | Name: " << name << " |  
    Passport#: " << passportNumber << endl;
```

```
}
```

```
};
```

```
// Booking node class for linked list
```

```
class Booking {
```

```
public:
```

```
    int bookingId;
```

```
    Passenger passenger;
```

```
    int flightNumber;
```

```
    Booking* next;
```

```
Booking() : bookingId(0), passenger(), flightNumber(0), next(nullptr) { }
```

```
Booking(int bId, const Passenger& p, int fNo): bookingId(bId), passenger(p),  
flightNumber(fNo), next(nullptr) { }
```

```
void display() const {
```

```
    cout << "Booking ID: " << bookingId << " | Flight#: " << flightNumber << " |  
    ";
```

```
    passenger.display();
```

```
}
```

```
};
```

```
// Linked list to manage bookings
```

```
class BookingList {
```

```
private:
```

```
    Booking* head;
```

```
    int bookingCount;
```

```
public:
```

```
    BookingList() : head(nullptr), bookingCount(0) {}
```

```
// Add a booking to the list
```

```
void insertBooking(const Passenger& p, int flightNumber) {
```

```
    bookingCount++;
```

```
    Booking* newBooking = new Booking(bookingCount, p, flightNumber);
```

```
    newBooking->next = head;
```

```
    head = newBooking;
```

```
    cout << "Booking successful! Booking ID: " << bookingCount << endl;
```

```
}
```

```
// Remove booking by booking ID
```

```
void removeBooking(int bookingId) {
```

```
    Booking* current = head;
```

```
    Booking* prev = nullptr;
```

```
    while (current != nullptr) {
```

```

    if (current->bookingId == bookingId) {
        if (prev == nullptr) {
            head = current->next;
        } else {
            prev->next = current->next;
        }
        delete current;
        cout << "Booking#" << bookingId << " cancelled successfully." << endl;
        return;
    }
    prev = current;
    current = current->next;
}
cout << "Booking ID not found." << endl;
}

```

// Update passenger info for a booking

```

void updateBooking(int bookingId, const Passenger& newPassenger) {
    Booking* current = head;
    while (current != nullptr) {
        if (current->bookingId == bookingId) {
            current->passenger = newPassenger;
            cout << "Booking#" << bookingId << " updated successfully." << endl;
            return;
        }
    }
}

```

```

        current = current->next;
    }
    cout << "Booking ID not found." << endl;
}

// Linear search booking by ID
Booking* searchBookingLinear(int bookingId) {
    Booking* current = head;
    while (current != nullptr) {
        if (current->bookingId == bookingId)
            return current;
        current = current->next;
    }
    return nullptr;
}

// Display all bookings
void displayAllBookings() const {
    if (head == nullptr) {
        cout << "No bookings available." << endl;
        return;
    }
    Booking* current = head;
    while (current != nullptr) {
        current->display();
    }
}

```

```

        current = current->next;
    }
}

// Destructor to free memory
~BookingList() {
    Booking* current = head;
    while (current != nullptr) {
        Booking* next = current->next;
        delete current;
        current = next;
    }
}

};

// Helper: read integer input with validation
int getIntInput(const string& prompt) {
    int val;
    while (true) {
        cout << prompt;
        cin >> val;
        if (!cin.fail()) {
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear input buffer
            return val;
        }
    }
}

```

```
    else {  
        cout << "Invalid input. Please enter a valid integer." << endl;  
        cin.clear();  
        cin.ignore(numeric_limits<streamsize>::max(), '\n');  
    }  
}  
}
```

// Helper: read string input

```
string getStringInput(const string& prompt) {  
    string val;  
    cout << prompt;  
    getline(cin, val);  
    return val;  
}
```

// Validate time string format HH:MM

```
bool isValidTimeFormat(const string& time) {  
    if (time.length() != 5) return false;  
    if (time[2] != ':') return false;  
    if (!isdigit(time[0]) || !isdigit(time[1]) || !isdigit(time[3]) || !isdigit(time[4]))  
return false;  
    int hh = stoi(time.substr(0, 2));  
    int mm = stoi(time.substr(3, 2));  
    if (hh < 0 || hh > 23 || mm < 0 || mm > 59) return false;  
    return true;  
}
```

```
}
```

```
// Menu for managing flights
```

```
void flightMenu(FlightArray& flightArray) {
```

```
    int choice;
```

```
    do {
```

```
        cout << "\n--- Flight Management ---" << endl;
```

```
        cout << "1. Add Flight" << endl;
```

```
        cout << "2. Delete Flight" << endl;
```

```
        cout << "3. Update Flight" << endl;
```

```
        cout << "4. Display Flights" << endl;
```

```
        cout << "5. Sort Flights by Flight Number (Bubble Sort)" << endl;
```

```
        cout << "6. Sort Flights by Departure Time (Quick Sort)" << endl;
```

```
        cout << "7. Search Flight (Linear Search)" << endl;
```

```
        cout << "8. Search Flight (Binary Search - requires sorted by flight number)"  
<< endl;
```

```
        cout << "0. Back to Main Menu" << endl;
```

```
        cout << "Choose an option: ";
```

```
        cin >> choice;
```

```
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear buffer
```

```
        switch (choice) {
```

```
            case 1: {
```

```
                int flightNo = getIntInput("Enter flight number: ");
```

```
                string origin = getStringInput("Enter origin: ");
```

```
                string dest = getStringInput("Enter destination: ");
```

```

string depTime;
do {
    depTime = getStringInput("Enter departure time (HH:MM): ");
    if (!isValidTimeFormat(depTime))
        cout << "Invalid time format, please try again." << endl;
} while (!isValidTimeFormat(depTime));

int seats = getIntInput("Enter total seats: ");
flightArray.insertFlight(Flight(flightNo, origin, dest, depTime, seats));
cout << "Flight added successfully!" << endl;
break;
}
case 2: {
    int flightNo = getIntInput("Enter flight number to delete: ");
    flightArray.removeFlight(flightNo);
    break;
}
case 3: {
    int flightNo = getIntInput("Enter flight number to update: ");
    string origin = getStringInput("Enter new origin: ");
    string dest = getStringInput("Enter new destination: ");

    string depTime;
    do {

```



```
        depTime = getStringInput("Enter new departure time (HH:MM): ");
        if (!isValidTimeFormat(depTime))
            cout << "Invalid time format, please try again." << endl;
    } while (!isValidTimeFormat(depTime));

    int seats = getIntInput("Enter new total seats: ");

    flightArray.updateFlight(flightNo, origin, dest, depTime, seats);
    break;
}
case 4:
    flightArray.displayAll();
    break;
case 5:
    flightArray.bubbleSortByFlightNumber();
    cout << "Flights sorted by flight number using Bubble Sort." << endl;
    break;
case 6:
    if (flightArray.getSize() > 0) {
        flightArray.quickSortByDepartureTime(0, flightArray.getSize() - 1);
        cout << "Flights sorted by departure time using Quick Sort." << endl;
    } else {
        cout << "No flights to sort." << endl;
    }
    break;
```

```
case 7: {
    int flightNo = getIntInput("Enter flight number to search: ");
    Flight* found = flightArray.linearSearch(flightNo);
    if (found) {
        cout << "Flight found:" << endl;
        found->display();
    } else {
        cout << "Flight not found." << endl;
    }
    break;
}
case 8: {
    cout << "Note: Flights must be sorted by flight number for Binary Search
to work correctly." << endl;
    int flightNo = getIntInput("Enter flight number to search: ");
    Flight* found = flightArray.binarySearch(flightNo);
    if (found) {
        cout << "Flight found:" << endl;
        found->display();
    } else {
        cout << "Flight not found." << endl;
    }
    break;
}
case 0:
    break;
```

```

        default:
            cout << "Invalid choice, please try again." << endl;
        }

    } while (choice != 0);
}

// Menu for managing bookings
void bookingMenu(BookingList& bookingList, FlightArray& flightArray) {
    int choice;
    do {
        cout << "\n--- Booking Management ---" << endl;
        cout << "1. Add Booking" << endl;
        cout << "2. Cancel Booking" << endl;
        cout << "3. Update Booking Passenger Info" << endl;
        cout << "4. Display All Bookings" << endl;
        cout << "5. Search Booking" << endl;
        cout << "0. Back to Main Menu" << endl;
        cout << "Choose an option: ";
        cin >> choice;
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear buffer

        switch (choice) {
            case 1: {
                int flightNo = getIntInput("Enter flight number for booking: ");

```

```

Flight* flight = flightArray.linearSearch(flightNo);
if (!flight) {
    cout << "Flight not found. Cannot create booking." << endl;
    break;
}
if (flight->availableSeats <= 0) {
    cout << "No available seats on this flight." << endl;
    break;
}
int passId = getIntInput("Enter Passenger ID: ");
string name = getStringInput("Enter full name: ");
string passport = getStringInput("Enter passport number: ");
Passenger p(passId, name, passport);

bookingList.insertBooking(p, flightNo);
flight->availableSeats--; // Decrease seat count
break;
}
case 2: {
    int bookingId = getIntInput("Enter booking ID to cancel: ");
    Booking* booking = bookingList.searchBookingLinear(bookingId);
    if (booking) {
        Flight* flight = flightArray.linearSearch(booking->flightNumber);
        if (flight)
            flight->availableSeats++; // Release seat
    }
}

```

```

        bookingList.removeBooking(bookingId);
    } else {
        cout << "Booking not found." << endl;
    }
    break;
}
case 3: {
    int bookingId = getIntInput("Enter booking ID to update passenger info:
");
    Booking* booking = bookingList.searchBookingLinear(bookingId);
    if (booking) {
        int passId = getIntInput("Enter new Passenger ID: ");
        string name = getStringInput("Enter new full name: ");
        string passport = getStringInput("Enter new passport number: ");
        Passenger p(passId, name, passport);
        bookingList.updateBooking(bookingId, p);
    } else {
        cout << "Booking not found." << endl;
    }
    break;
}
case 4:
    bookingList.displayAllBookings();
    break;
case 5: {
    int bookingId = getIntInput("Enter booking ID to search: ");

```

```
        Booking* booking = bookingList.searchBookingLinear(bookingId);
        if (booking) {
            cout << "Booking found:" << endl;
            booking->display();
        } else {
            cout << "Booking not found." << endl;
        }
        break;
    }
    case 0:
        break;
    default:
        cout << "Invalid choice, please try again." << endl;
    }
} while (choice != 0);
}


int main() {
    FlightArray flightArray;
    BookingList bookingList;

    cout << "=== Welcome to the Airline Reservation System ===" << endl;

    int choice;
    do {
```

```
    cout << "\nMain Menu:\n1. Manage Flights\n2. Manage Bookings\n0.  
Exit\nChoose an option: ";  
  
    cin >> choice;  
  
    cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear buffer  
  
    switch (choice) {  
        case 1:  
            flightMenu(flightArray);  
            break;  
        case 2:  
            bookingMenu(bookingList, flightArray);  
            break;  
        case 0:  
            cout << "Thank you for using the Airline Reservation System.  
Goodbye!" << endl;  
            break;  
        default:  
            cout << "Invalid option, please try again." << endl;  
    }  
    } while (choice != 0);  
  
    return 0;  
}
```

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 . . .