

# Project Report: Airline Booking System

Fournier Pierre-Antoine

## Contents

<b>1</b>	<b>Requirements Analysis</b>	<b>2</b>
<b>2</b>	<b>Detailed Module Design</b>	<b>2</b>
2.1	Module: Airline (airline.h, airline.cpp) . . . . .	2
2.2	Module: Booking (booking.h, booking.cpp) . . . . .	2
2.3	Module: Query (query.h, query.cpp) . . . . .	2
2.4	Main Application (main.cpp) . . . . .	2
2.5	Data Storage . . . . .	2
<b>3</b>	<b>Program Test Analysis and Operation Result Display</b>	<b>2</b>
<b>4</b>	<b>User Manual (Optional)</b>	<b>3</b>
4.1	. . . . .	3
<b>5</b>	<b>Conclusion</b>	<b>3</b>
<b>6</b>	<b>Appendix</b>	<b>4</b>

# 1 Requirements Analysis

The objective was to develop an airline booking system to manage flight information, reservations, and ticket refunds. The system includes:

- Input and management of flight data.
- Real-time updates of ticket availability upon booking or cancellation.
- Queries by flight ID, departure time, and terminal station.
- Ticket booking and refund capabilities.

## 2 Detailed Module Design

The project is modularized into clearly defined components:

### 2.1 Module: Airline (`airline.h`, `airline.cpp`)

- Stores and manages basic flight data (flight ID, departure and arrival stations, departure time, available seats).
- Functions for loading flight data from files.

### 2.2 Module: Booking (`booking.h`, `booking.cpp`)

- Manages ticket booking and refund operations.
- Updates available seats after each transaction.

### 2.3 Module: Query (`query.h`, `query.cpp`)

- Supports queries by flight ID, departure time, or terminal station.
- Provides detailed information about flights matching query parameters.

### 2.4 Main Application (`main.cpp`)

- Integrates all modules and provides a user-friendly console interface.
- Keeps the application active for multiple operations until explicitly terminated by the user.

### 2.5 Data Storage

Flight information stored persistently in text files (`flights`).

## 3 Program Test Analysis and Operation Result Display

The program was thoroughly tested for:

- **Flight information accuracy:** Verified correct loading and displaying of flights.
- **Real-time updates:** Validated seat availability updates after bookings and refunds.
- **Query functionality:** Checked accuracy and robustness of flight searches.
- **Booking and refund process:** Confirmed correct operation and data integrity during multiple bookings/refunds.

All tests passed successfully, demonstrating robustness and reliability.

## 4 User Manual (Optional)

### 4.1

Quick Usage Guide:

**Run the Program:**

```
make  
./airline.out
```

**Available Commands:**

- Enter flight information.
- Browse all flight data.
- Query flights by ID, departure time, or terminal.
- Book tickets.
- Refund tickets.
- Exit program with a specific command (e.g., "exit").

**File Structure:**

- Flight data stored in the file `flights`.

## 5 Conclusion

The Airline Booking System meets all specified requirements with clear modular design, robust functionality, and ease of use, making it suitable for real-world applications.

## 6 Appendix

```
> make
g++ -Wall -Wextra -g -c main.cpp -o main.o
g++ -Wall -Wextra -g -c airlines/airline.cpp -o airlines/airline.o
g++ -Wall -Wextra -g -c airlines/booking.cpp -o airlines/booking.o
g++ -Wall -Wextra -g -c query/query.cpp -o query/query.o
g++ -o airline.out -g main.o airlines/airline.o airlines/booking.o query/query.o
0 | main | ~/Documents/Github/Airport_Reservation
> 
```

Figure 1: Compiling the project

```
0 | main | ~/Documents/Github/Airport_Reservation
> ./airline.out
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.

```

Figure 2: Launching the project

```
> ./airline.out
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.
1
Enter Name of the compagny: Example
Enter the id of the flight: 45666
Enter the limit of passengers: 30000
Enter the destination: Shanghai
Enter the departure time: 10/05/25-12-00
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.
█
```

Figure 3: Adding a flight to the database

```
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.
2
Enter flight id:
123
Enter number of passengers:
10
Added 10
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.

```

Figure 4: Booking a flight

```
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.
3
Enter flight id:
123
Enter number of passengers:
5
Remove the passengers
```

Figure 5: Withdraw from a flight

```
Enter:
0 to exit,
1 to add a flight,
2 to book a flight,
3 to unbook a flight,
4 to look at a flight.
4
Choose the way you will check the flights:
1 for id,
2 for time of arrival,
3 for destination
1
Enter the id:
123
Name: AirFrance
ID: 123
Number of passengers: 5/200
Destination: Paris
Departure: 10/07/25-12-00
```

Figure 6: Checking the info of a flight



```
0 | main | ~/Documents/Github/Airport_Reservation  
> cat flights/flights  
AirFrance 123 5 200 10/07/25-12-00 Paris  
Emirates 456 0 120 11/07/25-12-00 Los-Angeles  
AmericanAirlines 879 0 500 12/07/20-12-00 New-York  
Lufthansa 963 0 150 13/07/25-12-00 Berlin  
Example 45666 0 30000 10/05/25-12-00 Shanghai  
0 | main | ~/Documents/Github/Airport_Reservation  
> 
```

Figure 7: The file containing the data