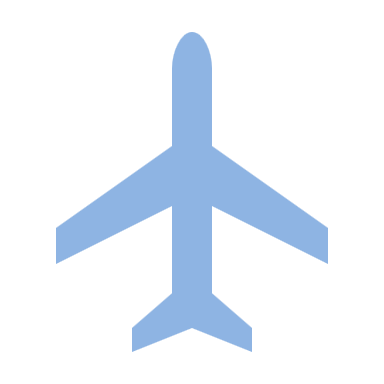


Lebanese University

Faculty of Engineering III

Electrical and Electronic Department

Object Oriented Programming

LUA

Lebanese University Airline

Dana Dagher 6436

Marwa CHEHAB 6485

https://[github](https://github.com/Dana-Dagher/c-_project.git).com/Dana-Dagher/c-\_project.git

2023- 2024

TABLE OF CONTENTS

[TABLE OF CONTENTS ii](#_Toc124986349)

[LIST OF FIGURES iii](#_Toc124986350)

[CHAPTER 1 INTRODUCTION 1](#_Toc124986351)

[CHAPTER 2 System Design and Implementation 2](#_Toc124986352)

[2.1 Introduction 2](#_Toc124986353)

2.2 Required Components ……………………………………………………...…...…..2

[2.3 System Design 2](#_Toc124986354)

[2.4 Implementation Steps 3](#_Toc124986355)

[CHAPTER 3 Conclusion and Future Work 8](#_Toc124986356)

[3.1 Conclusion 8](#_Toc124986357)

[3.2 Future Work 8](#_Toc124986358)

LIST OF FIGURES

Figure ‎2.3.1 Use Case diagram 3

Figure ‎2.3.2 Class diagram 4

# : INTRODUCTION

Objective:

The objective of this project is to develop a Flight Reservation System using C++. The system allows users to search for available flights based on their departure and arrival locations, desired date, and the number of seats needed. Users can then make reservations by providing passenger details and selecting a payment method.

Problem Statement:

Traditional flight reservation systems can be cumbersome, and users often face challenges in finding available flights that match their preferences. This project aims to address these issues by providing a user-friendly interface to search for and book flights efficiently.

Solution:

The Flight Reservation System implemented in C++ offers a streamlined solution for users to check available flights, book seats, and make payments. The system incorporates object-oriented programming principles to model entities such as flights, passengers, and bookings, enhancing the overall design and maintainability of the code.

# System Design and Implementation

**2.1 Introduction:**

In the design process of our Flight Reservation System, we followed a systematic approach to ensure an organized and efficient implementation. This involved identifying key components, defining their interactions, and detailing the steps to be taken for successful implementation. The design process aimed at creating a modular and scalable system to facilitate future enhancements.

**2.2 Required Components:**

The Flight Reservation System relies on several essential components to achieve its functionality. These components include:

* Date Class: Manages the representation and formatting of dates.
* Passenger Class: Stores information about passengers, including name, age, and contact details.
* PaymentMethod Class: Represents the chosen payment method for booking flights.
* Flight Class: Defines the structure of a flight, including details such as flight number, departure, arrival, date, available seats, and ticket price.
* Booking Class: Captures information about a booking, including the flight number, date, and the number of seats booked.
* BookingSystem Class: Orchestrates the interactions between flights, passengers, bookings, and payment processing.

**2.3 System Design:**

**2.3.1 Use Case diagram:**

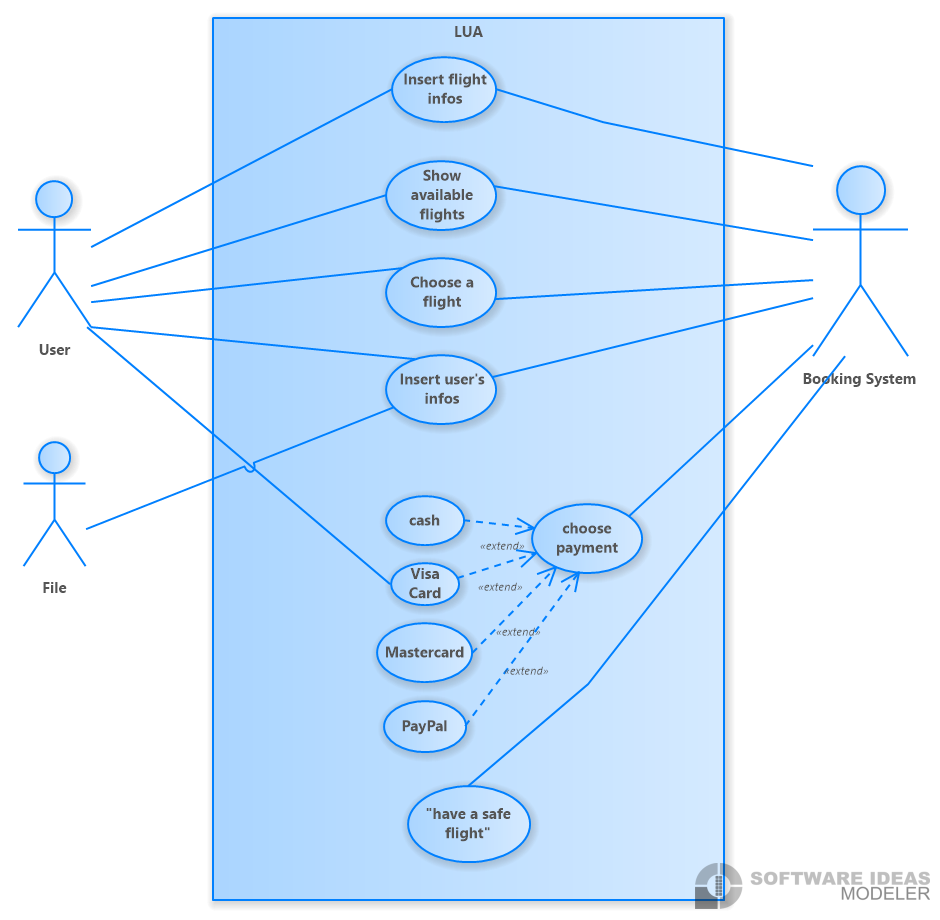
The Use Case Diagram illustrates the different interactions between actors and the system.

-User: Represents individuals interacting with the system.

-System: Encompasses the core functionalities of the Flight Reservation System.

Use Cases:

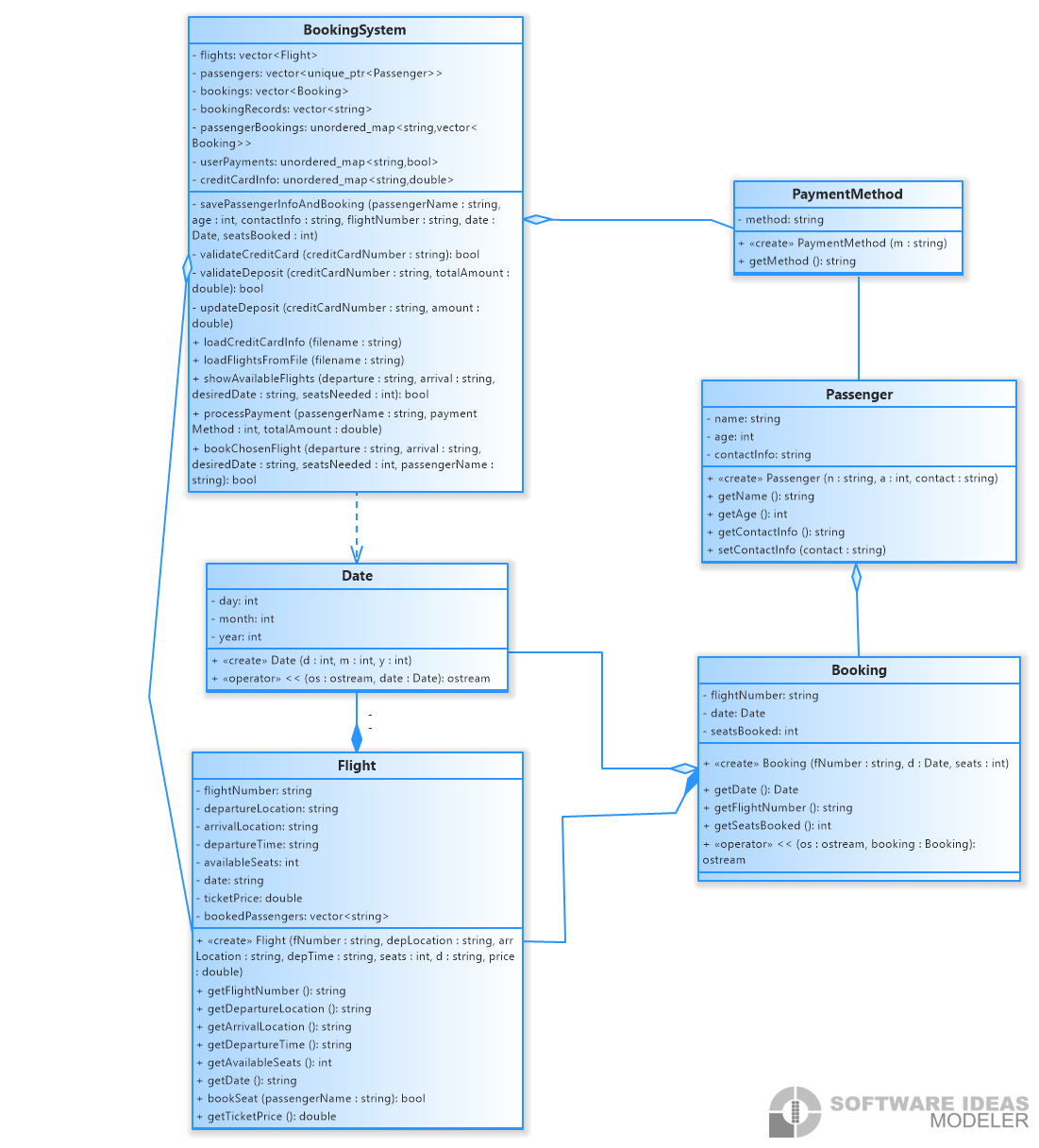
1. Search Flights: Allows users to search for available flights based on their preferences.
2. Book Seats: Enables users to reserve seats on a chosen flight.
3. Process Payment: Handles the payment process after seat reservation.

****

**2.3.2 Class Diagram:**

The Class Diagram provides a visual representation of the relationships between the key classes in the system.

Date, Passenger, PaymentMethod, Flight, Booking, BookingSystem: Core classes representing different aspects of the flight reservation process.



**2.4 Implementation Steps:**

The implementation of the Flight Reservation System involved a detailed process to ensure robust functionality, user-friendliness, and secure transactions. Below are the expanded details of the implementation steps:

**2.4.1 Loading Data :**

The system initiates by loading essential data from external files:

Code:

bookingSystem.loadFlightsFromFile("flights.txt");

bookingSystem.loadCreditCardInfo("bankFile.txt");

* Load Flights: The loadFlightsFromFile method reads flight information from the "flights.txt" file. Each line in the file represents a flight, and the system creates instances of the Flight class for each entry.
* Load Credit Card Info: The loadCreditCardInfo method retrieves credit card information from the "bankFile.txt" file. The file includes pairs of credit card numbers and corresponding deposit amounts. This information is stored in the creditCardInfo map for later validation.

**2.4.2 User Input and Criteria Selection:**

The system prompts the user to input details for each individual traveler:

Code:

cout << "Enter the number of users: ";

cin >> numUsers;

for (int i = 0; i < numUsers; ++i)

{

// User input for departure, arrival, desired date, seats needed, and passenger name.

// …

}

The loop iterates through each user, collecting information such as departure and arrival locations, desired travel date, the number of seats needed, and the passenger's name.

**2.4.3 Searching and Booking:**

The system then checks for available flights based on the user's criteria:

Code:

bool flightsAvailable = bookingSystem.showAvailableFlights

(departure, arrival, desiredDate, seatsNeeded);

Show Available Flights: The showAvailableFlights method filters flights based on the user's departure, arrival, and date preferences. If available flights meet the criteria, the user is presented with a list of choices.

Code:

if (flightsAvailable) {

bookingSystem.bookChosenFlight(departure, arrival, desiredDate, seatsNeeded, passengerName);

}

Book Chosen Flight: If flights are available, the user selects a specific flight, enters the number of seats to book, and provides passenger details. The bookChosenFlight method handles the booking process, ensuring seat availability, processing payments, and saving relevant information.

**2.4.4 Payment Processing:**

The system prompts the user to select a payment method and validates the payment details:

Code:

cout << "Select payment method:" << endl;

// Display payment options...

int paymentChoice;

cout << "Enter your choice (1-4): ";

cin >> paymentChoice;

switch (paymentChoice) {

// Case handling for different payment methods...

// ...

}

* Payment Options: The system presents the user with payment options, including PayPal, MasterCard, VisaCard, and Cash at Airport.

Code:

processPayment(name, to\_string(chosenFlight-> getTicketPrice()), totalAmount);

userPayments[passengerName] = true;

* Payment Processing: The processPayment method validates credit card information, checks for sufficient deposit amounts, and updates the deposit accordingly. The payment process ensures security and successfully concludes the booking.

**2.4.5 Data Persistence:**

Finally, passenger information and bookings are saved to a text file for future reference:

Code:

savePassengerInfoAndBooking(name, age, contactInfo, chosenFlight->getFlightNumber(), chosenDate, seatsToBook);

Save Passenger Information: The savePassengerInfoAndBooking method appends passenger information and booking details to the "passenger\_info\_and\_bookings.txt" file. This maintains a record of user activities for reference and future analysis.

**This detailed implementation process ensures a seamless and secure experience for users throughout the flight reservation system. Each step is carefully designed to handle user input, validate data, and maintain records for future reference.**

# Conclusion and Future Work

**3.1 Conclusion:**

In conclusion, the design and implementation of the Flight Reservation System reflect a meticulous process of incorporating object-oriented principles and systematic development. The system successfully achieves its objectives of providing users with a seamless experience in searching, booking, and processing payments for flights. The use of classes such as Flight, Passenger, and Booking enhances code organization and readability, fostering a maintainable and extensible system. The comprehensive analysis and representation of the system in the Use Case and Class Diagrams contribute to a thorough understanding of its functionality.

The Loading Data phase ensures that vital flight and credit card information is retrieved efficiently, while the User Input and Criteria Selection stage enables users to input their travel preferences. The Searching and Booking process optimally filters available flights based on user criteria, allowing for a tailored booking experience. The Payment Processing phase enhances the system's security and user experience, ensuring successful financial transactions. Finally, the Data Persistence step safeguards user information, maintaining a record of bookings for future reference.

**3.2 Future Work:**

Looking ahead, several avenues for enhancing the Flight Reservation System and expanding its utility in real-world scenarios can be explored:

* Integrated Travel Booking Platform: Extending the system to encompass hotel reservations would transform it into an integrated travel booking platform. Users could seamlessly plan their entire trip, booking both flights and accommodations in one place.
* Package Deals and Discounts: Collaborating with hotels could lead to the creation of bundled package deals, offering users discounts when booking flights and accommodations together. This strategy encourages users to explore and utilize both services.
* Multi-language Support: Incorporating multi-language support to cater to a global user base. Enabling users to interact with the system in their preferred language enhances accessibility and user satisfaction.
* Mobile Application Development: Developing a mobile application for the Flight Reservation System would cater to users who prefer to make bookings on their smartphones. This could include features such as push notifications and mobile-friendly interfaces.
* User Feedback and Analytics: Implementing a user feedback system and analytics tools would provide valuable insights into user behavior, preferences, and areas for improvement. This data could guide future enhancements and updates.
* Interactive Route Maps: Enhancing the user interface with interactive route maps that provide visual representations of flight paths, layovers, and destination information. This feature can make the booking process more engaging and informative.
* Social Media Integration: Allowing users to share their travel plans or booked flights on social media platforms. Integration with social media can enhance user engagement and serve as a marketing tool for the system. 😊

**These future updates aim to elevate the Flight Reservation System by introducing innovative features, improving user engagement, and staying aligned with industry trends. By embracing these enhancements, the system can remain competitive and provide a cutting-edge experience for users in the dynamic landscape of travel technology. Incorporating these future developments would further position the Flight Reservation System as a comprehensive and adaptable solution for both users and service providers, aligning it more closely with the evolving needs of the aviation industry and the expectations of modern travelers..**