**PROJECT REPORT**

***Submitted by***

**Krishiv chawda RA2211003011570**

**Jayvardhan singh tomar RA2211003011542**

**Vishesh palliwal RA2211003011546**

***Under the Guidance of***

**Dr. P. SARAVANAN**

**Assistant Professor, Department of Computing Technologies**

***In partial satisfaction of the requirements for the degree of***

## **BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**

**with specialization in CSE CORE**

## Logo, company name Description automatically generated

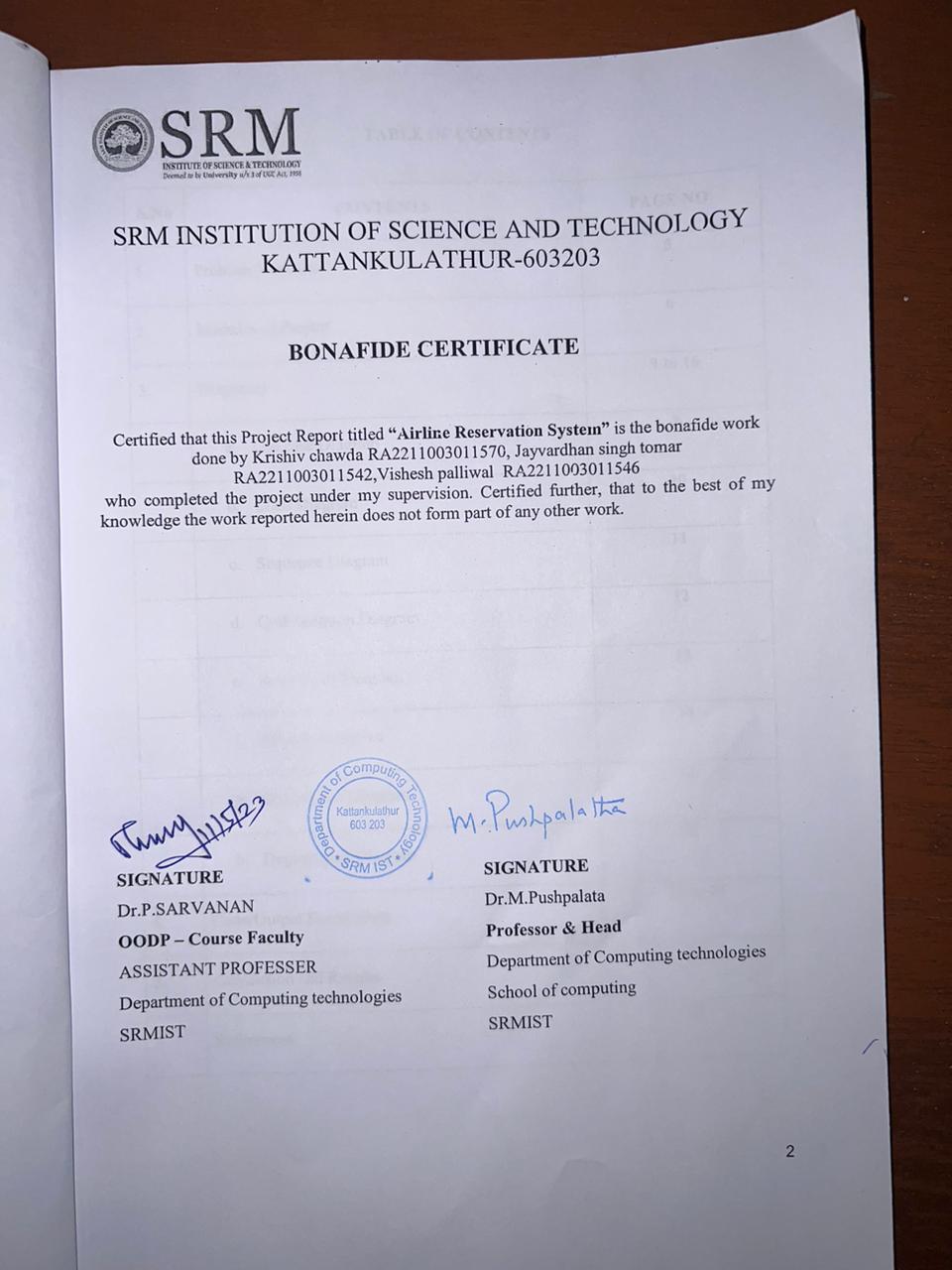
**SCHOOL OF COMPUTING**

# **COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**MAY 2023**



**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **CONTENTS** | **PAGE NO** |
| 1. | Problem Statement | **5** |
| 2. | Modules of Project | **6** |
| 3. | Diagrams | **9 to 16** |
|  | 1. Use case Diagram | **9** |
|  | 1. Class Diagram | **10** |
|  | 1. Sequence Diagram | **11** |
|  | 1. Collaboration Diagram | **12** |
|  | 1. State Chart Diagram | **13** |
|  | 1. Activity Diagram | **14** |
|  | 1. Component Diagram | **15** |
|  | 1. Deployment Diagram | **16** |
| 4. | Code/Output Screenshots | **17 to 29** |
| 5. | Conclusion and Results | **30** |
| 6. | References | **31** |

LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.

2. Identify Use Cases and develop the Use Case model.

3. Identify the conceptual classes and develop a domain model with UML Class diagram.

4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.

5. Draw relevant state charts and activity diagrams.

6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML: StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit**Problem Statement:**

The problem statement for an airline reservation system is to create a software system that enables customers to book flights, manage their reservations, and obtain necessary information about their flights, such as flight status, departure and arrival times, and flight details.

The system should be designed using Object-Oriented Design and Programming principles and should be modular, scalable, and extensible. It should be able to handle a large volume of users and transactions simultaneously and integrate with other systems, such as payment gateways, flight management systems, and airport information systems.

The system should also be user-friendly, secure, and able to handle unexpected events such as flight cancellations, delays, and changes in flight schedules. It should provide customers with real-time updates and options to manage their bookings accordingly.

Additionally, the system should have robust reporting capabilities to enable airlines to track their revenue, customer data, and other key metrics. The project should also consider the target audience or users of the system and design the user interface and functionality accordingly. Finally, the problem statement should outline any constraints or limitations of the project, such as time or budget constraints, and prioritize the key features and functionality to be developed.

Top of Form

**Modules of Project:**

The code is a flight booking system that allows users to make domestic and international flight bookings. Here's a detailed explanation of the modules used in the code:

1. `#include<iostream>`

This module allows input and output operations through standard streams such as `cin` and `cout`.

2. `#include<fstream>`

This module provides functionality to read from and write to files.

3. `#include<string.h>`

This module provides functions to manipulate strings.

4. `using namespace std;`

This line specifies that the code is using the standard namespace, which includes the standard C++ library.

5. `int glob=0;` and `int global=10;`

These lines define two global variables, `glob` and `global`, which can be accessed from any part of the code.

6. `class d\_booking`

This line defines a class called `d\_booking`, which is used for domestic flight bookings.

7. `protected:`

This keyword specifies that the member variables below it are protected, which means that they can only be accessed by the class itself and its subclasses.

8. `int pnr;`, `char f\_d[10],toja[7],tojd[7];`, `long int doj;`, `int choice,src,dest;`

These are member variables of the `d\_booking` class. They store the passenger's booking details, such as the PNR number, the departure and arrival locations, and the date of journey.

9. `public:`

This keyword specifies that the member functions below it are public, which means that they can be accessed from outside the class.

10. `void d\_pnr()`

This function generates a unique PNR number for each booking.

11. `int j\_detail()`

This function asks the user for their journey details, such as the date of journey, the source and destination locations, and then displays the available flights based on the source and destination locations. It then returns an integer value indicating the user's choice of flight.

12. `if`, `else if`, and `else` statements

These are conditional statements that check if a certain condition is true and then execute the code inside the corresponding block of code if it is. In this code, they are used to display the available flights based on the source and destination locations entered by the user.

13. `cout <<` and `cin >>` statements

These statements display output to the user and take input from the user, respectively. `cout` is used to display output to the console, and `cin` is used to take input from the user via the console.

Overall, these modules are used to create a flight booking system that allows users to make domestic and international flight bookings. The system takes input from the user, displays available flights, and generates a unique PNR number for each booking.

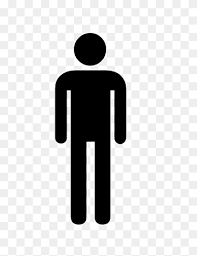
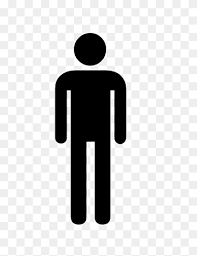
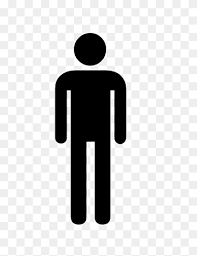
This code snippet is an implementation of some functions related to payment and flight ticketing system.

The payment class has protected member variables, such as choice1, bank, card, date, cvv, user\_id, and password. The public member function pay\_detail() is defined within this class. The function asks the user to select a payment method from debit card, credit card, or net banking. Depending on the choice, the user is prompted to provide relevant details like card number, date, cvv, bank, user id, and password. Finally, it displays a message for a successful transaction.

createfile() and createfilei() functions are used to create files for domestic and international flight bookings, respectively. Both functions accept an object of the passenger class, and the data is written to the corresponding file using the ofstream class.

The cancelticket() and cancelticketi() functions are used to cancel flight tickets for domestic and international bookings, respectively. The functions accept an integer argument, which represents the PNR number of the ticket to be canceled. The functions read data from the corresponding file and compare the PNR number with the given input. If the PNR number matches, the details of the corresponding passenger are displayed, and a refund of Rs. 1000 is issued. The original file is deleted, and a new file is created, containing all the original records except the one that was canceled.

The checkticket() and checkticketi() functions are used to display the details of a specific ticket for domestic and international bookings, respectively. These functions accept an integer argument, which represents the PNR number of the ticket to be checked. The functions read data from the corresponding file and compare the PNR number with the given input. If the PNR number matches, the details of the corresponding passenger are displayed. If the PNR number is not found, the function displays a message that the ticket was not found.

A. Use case Diagram****:

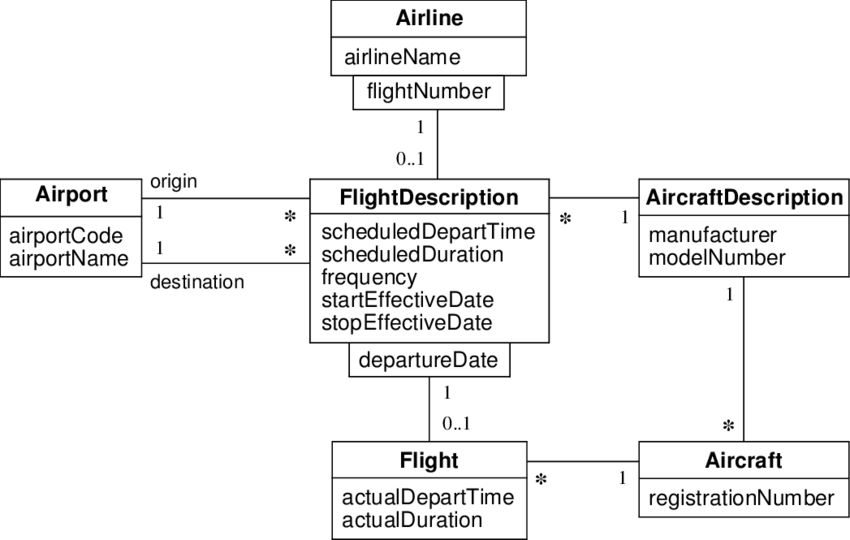
include

include

Bank payment portal

Passenger

Airline server

B. Class Diagram:

C. Sequence Diagram

Select flight ticket

Payment confirm

Ticket confirmed

Make payment

Ticket available

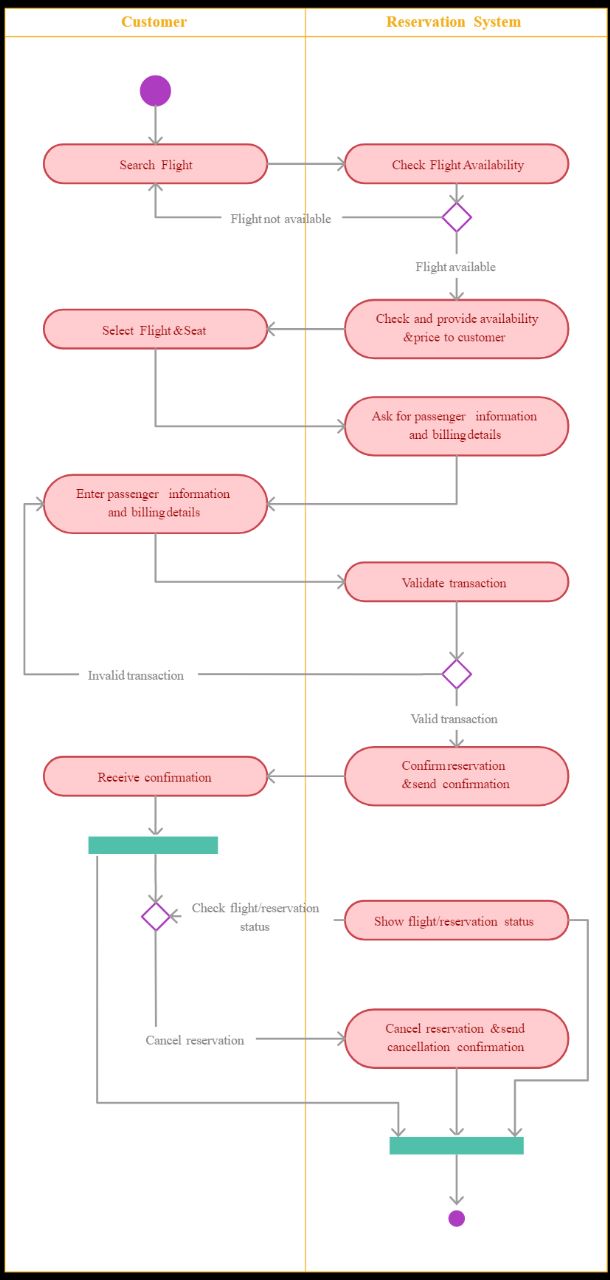
Search flight

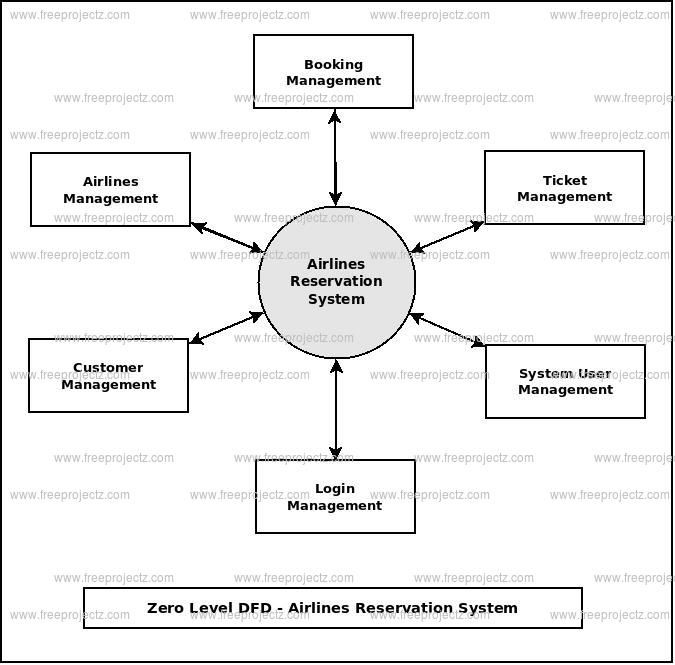
Payment

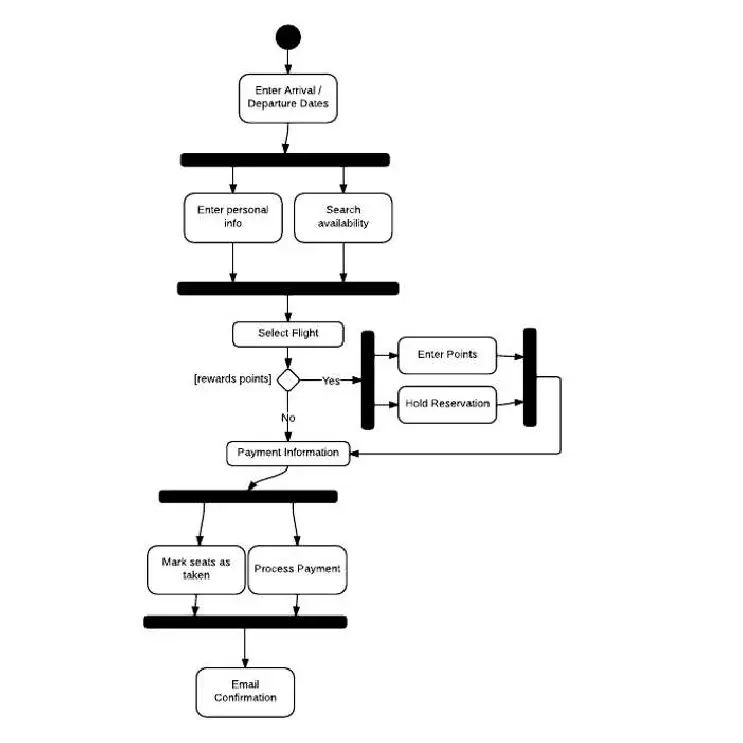
Flight

User

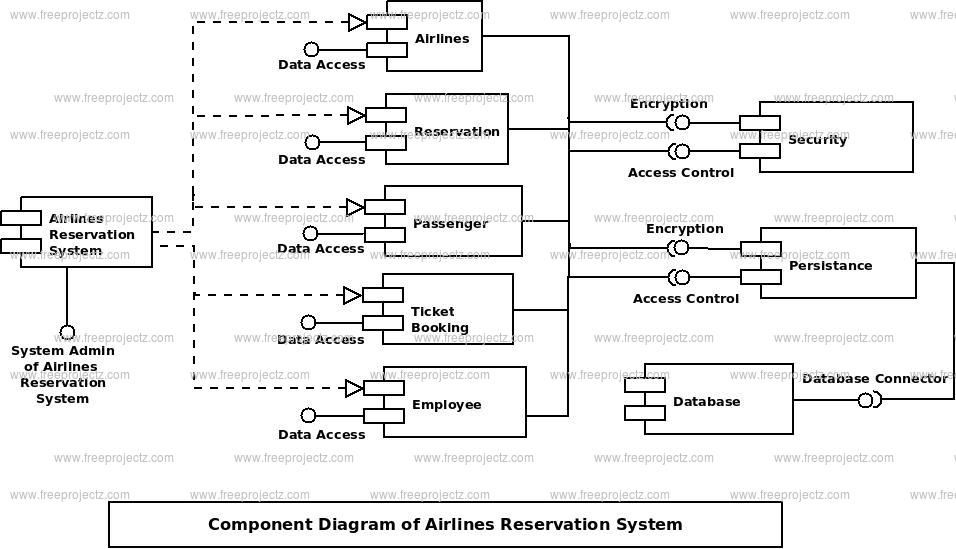
Booking

D. Collaboration diagram:

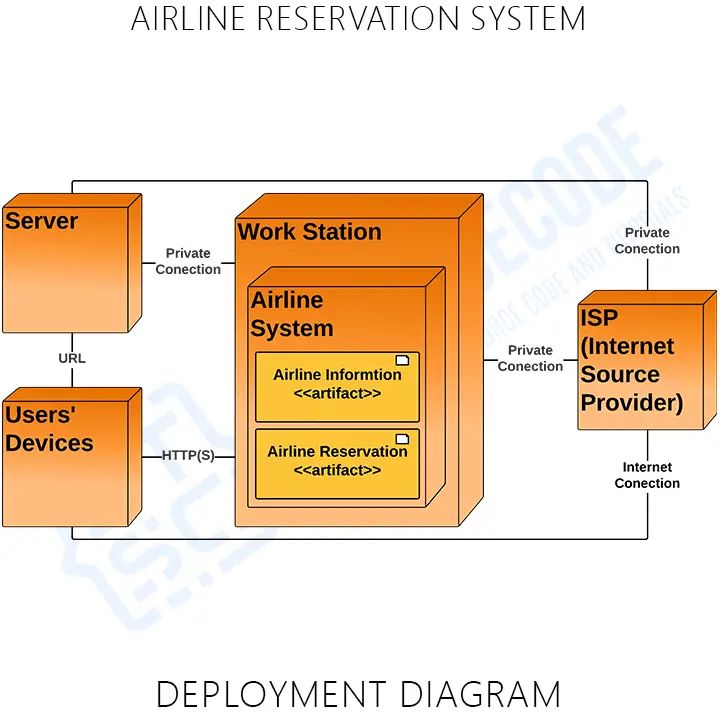
E. State Chart Diagram:

F. Activity Diagram:

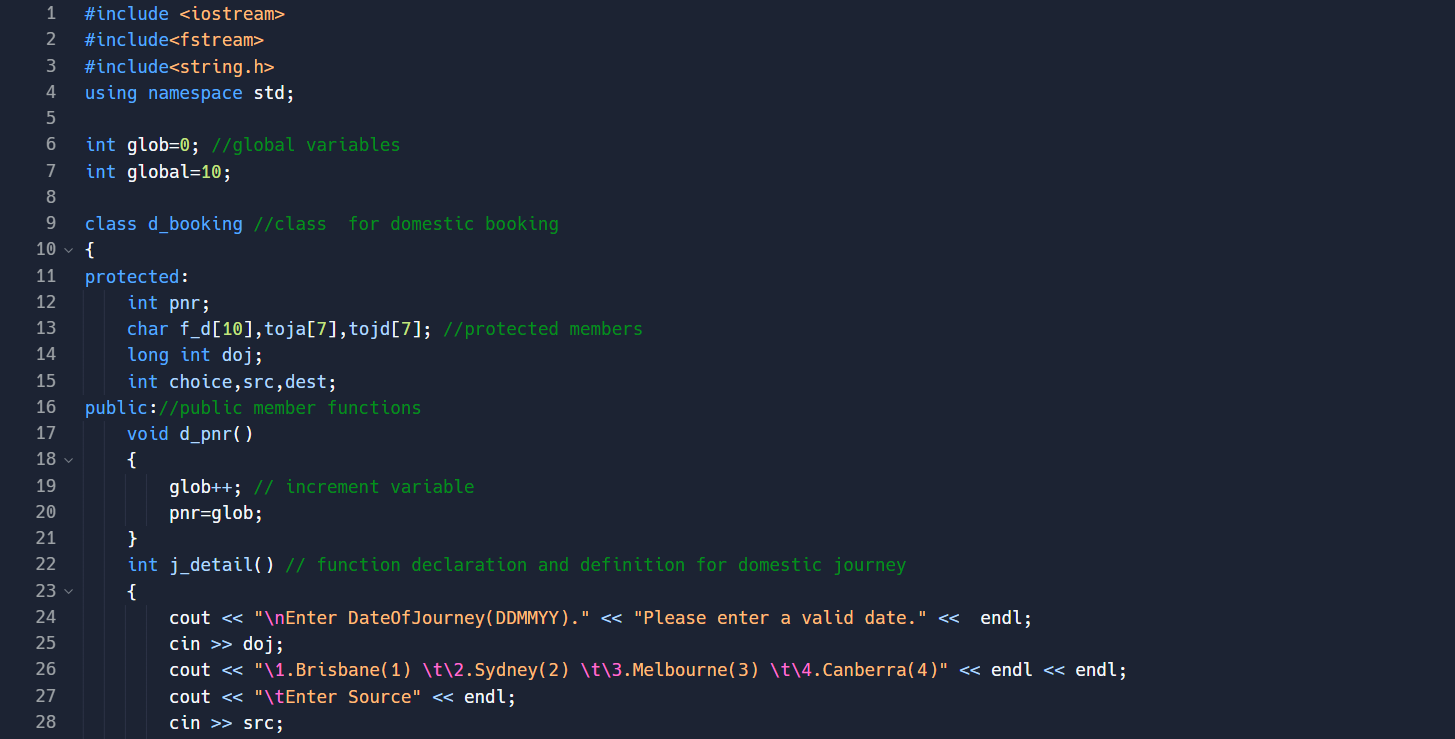
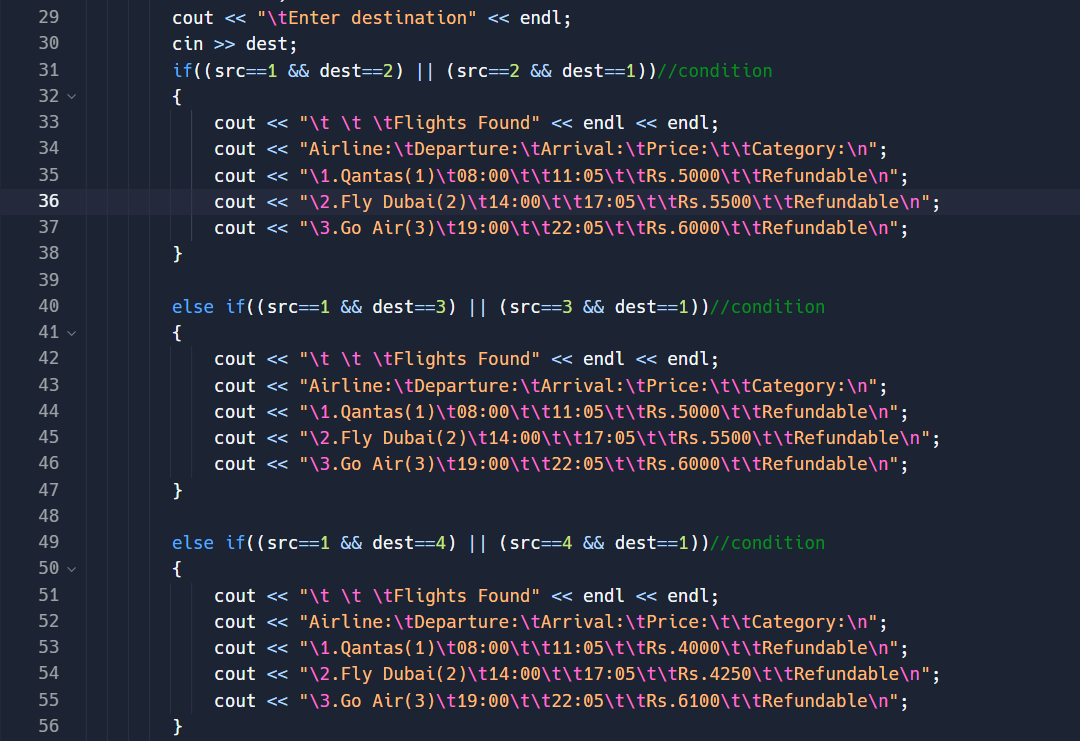
G. Component Diagram:

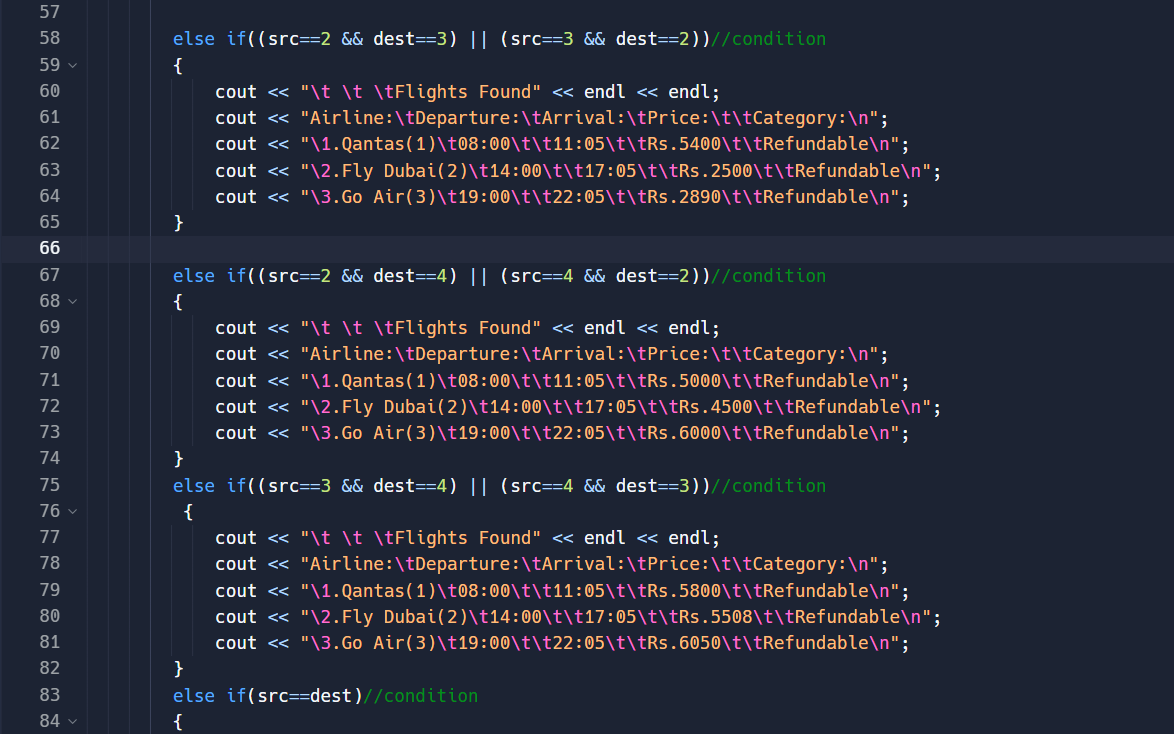


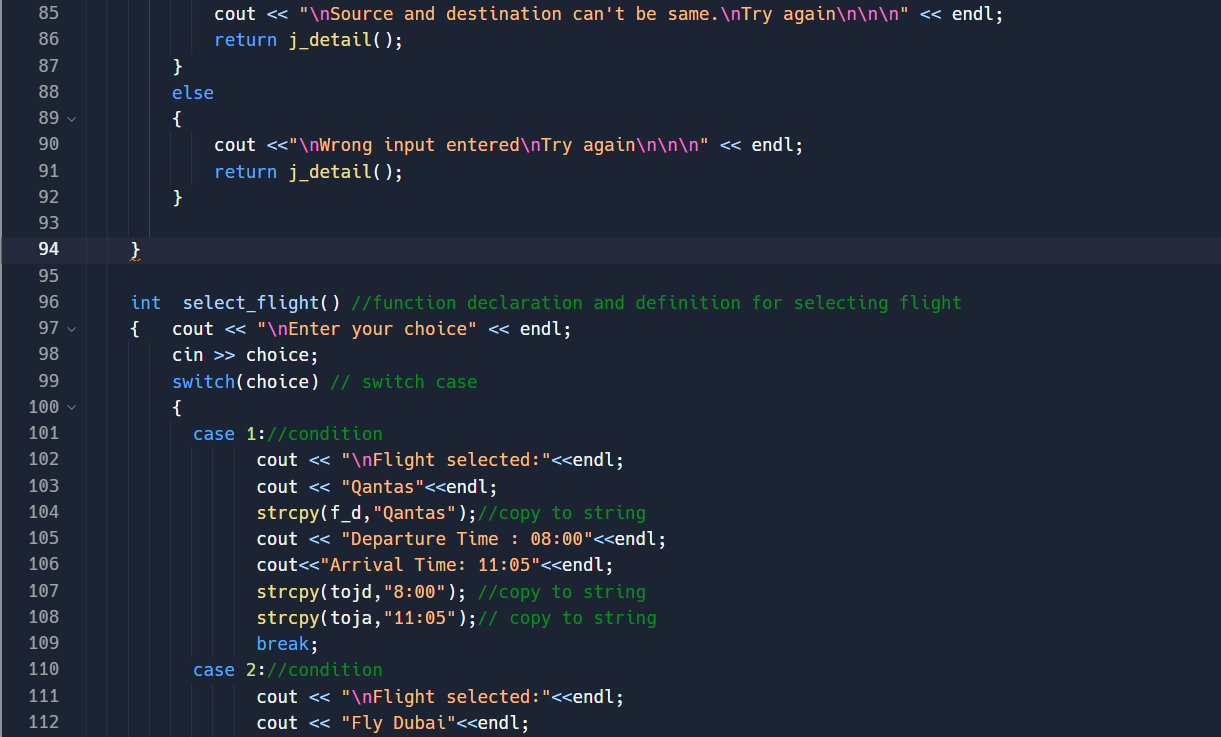
H. Deployment Diagram:

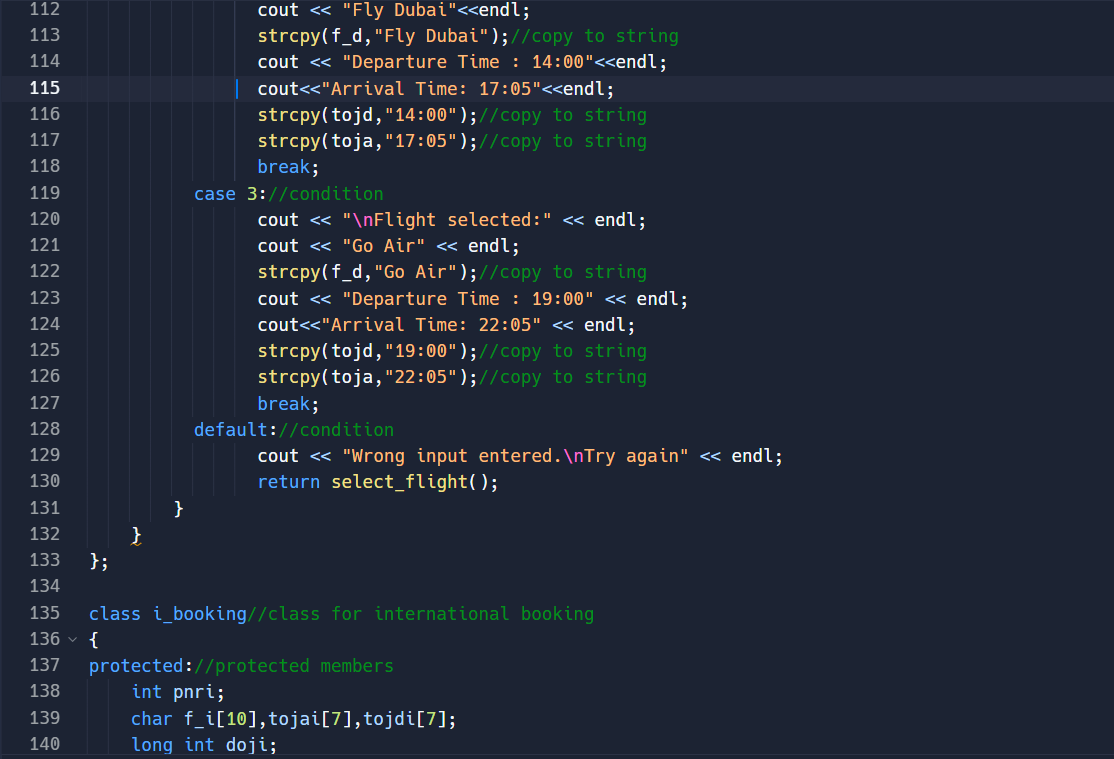


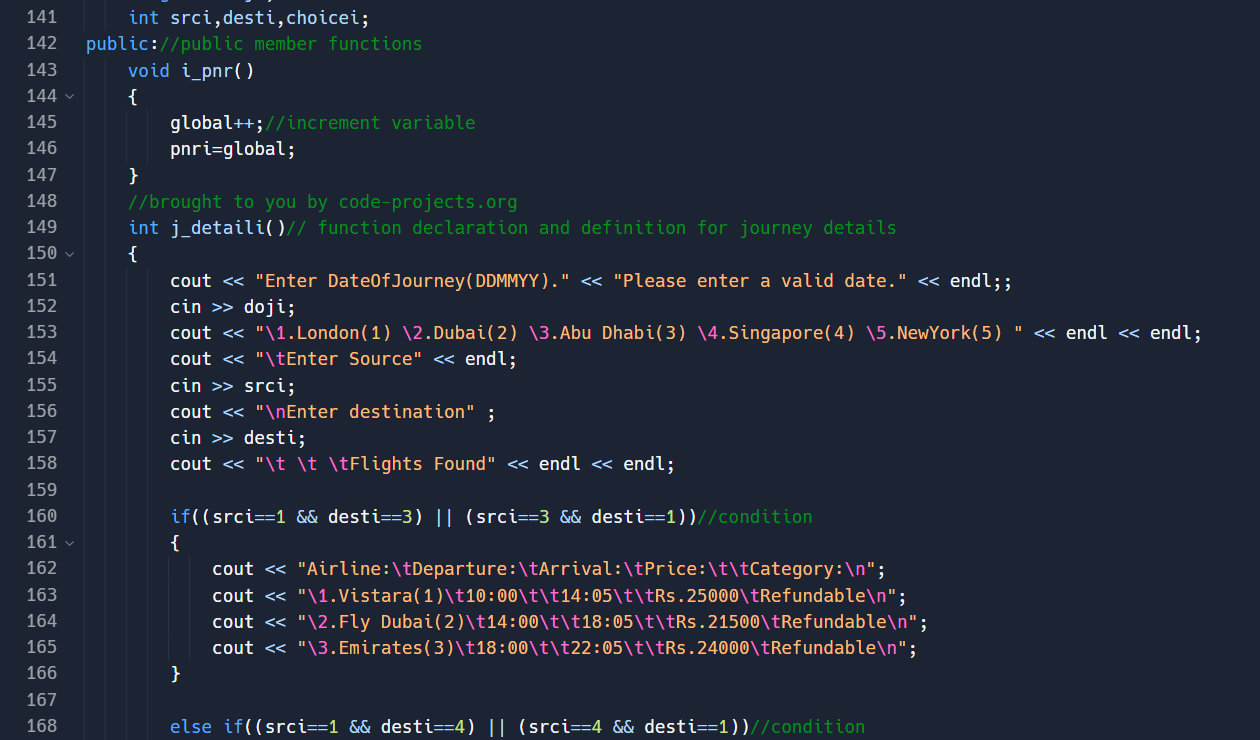
**Code Screenshots:**

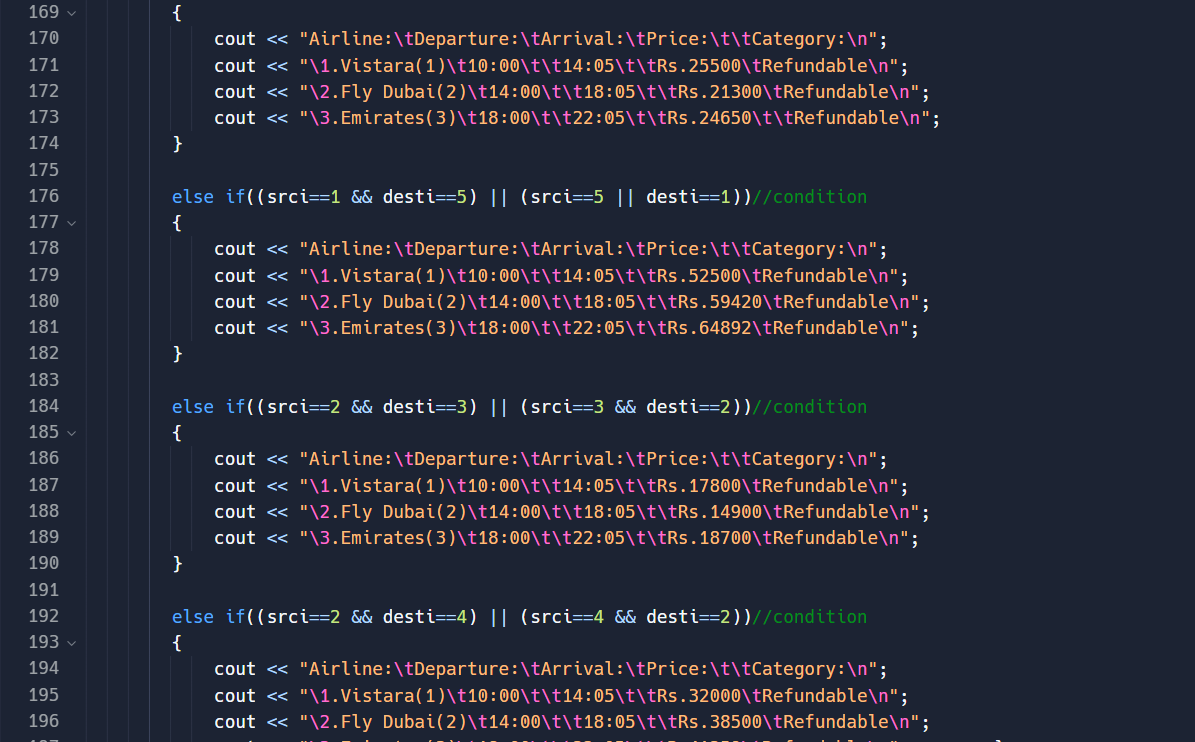


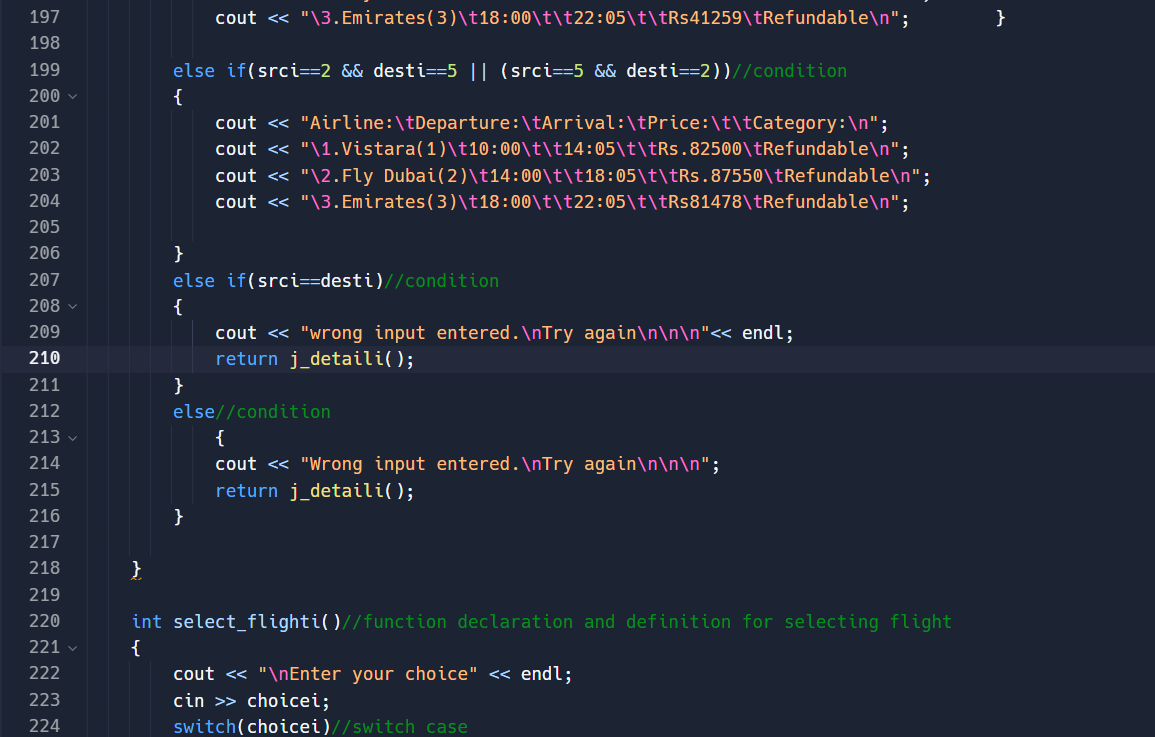
****

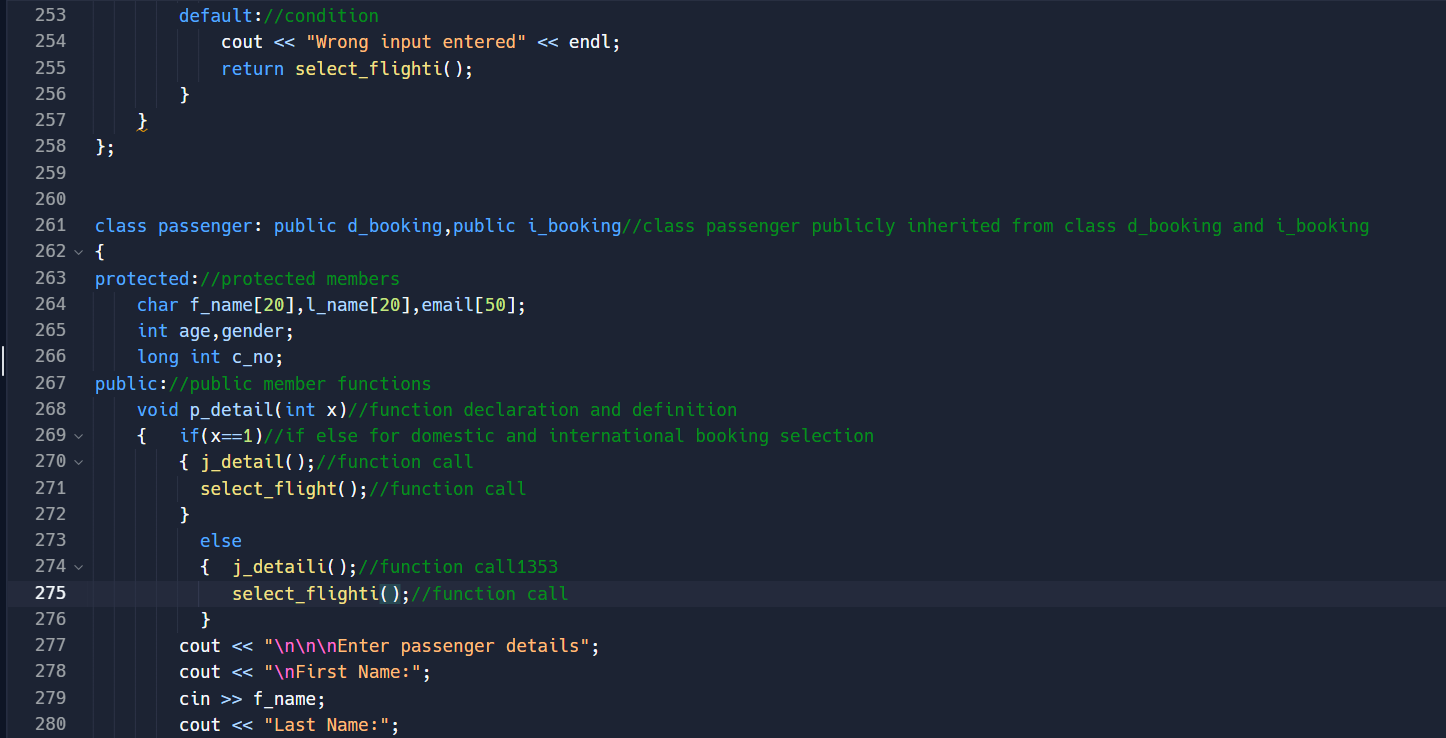
****

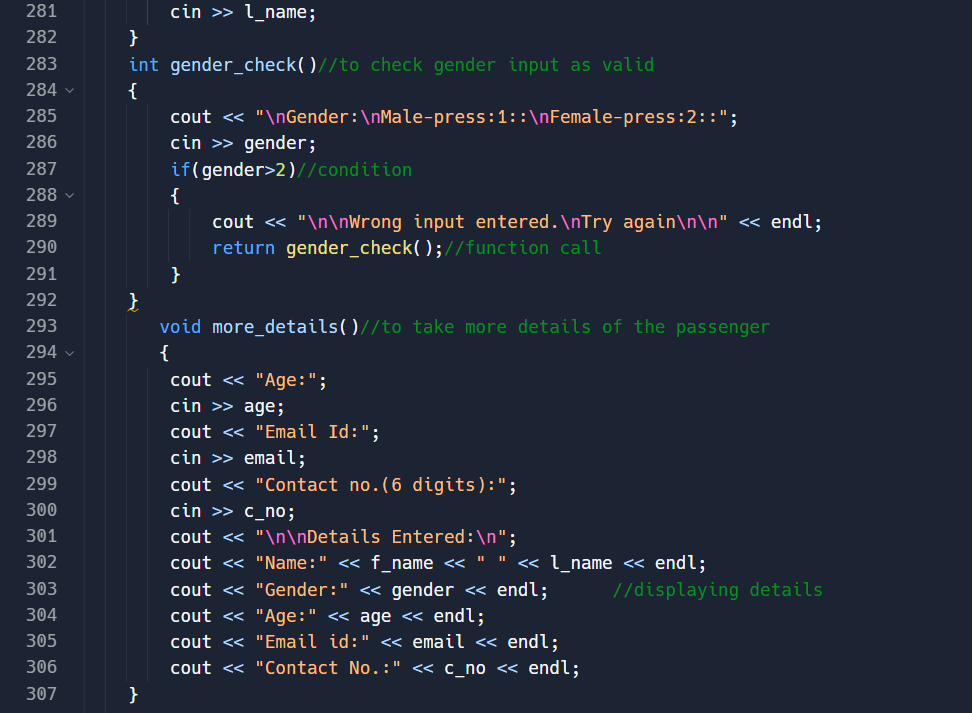
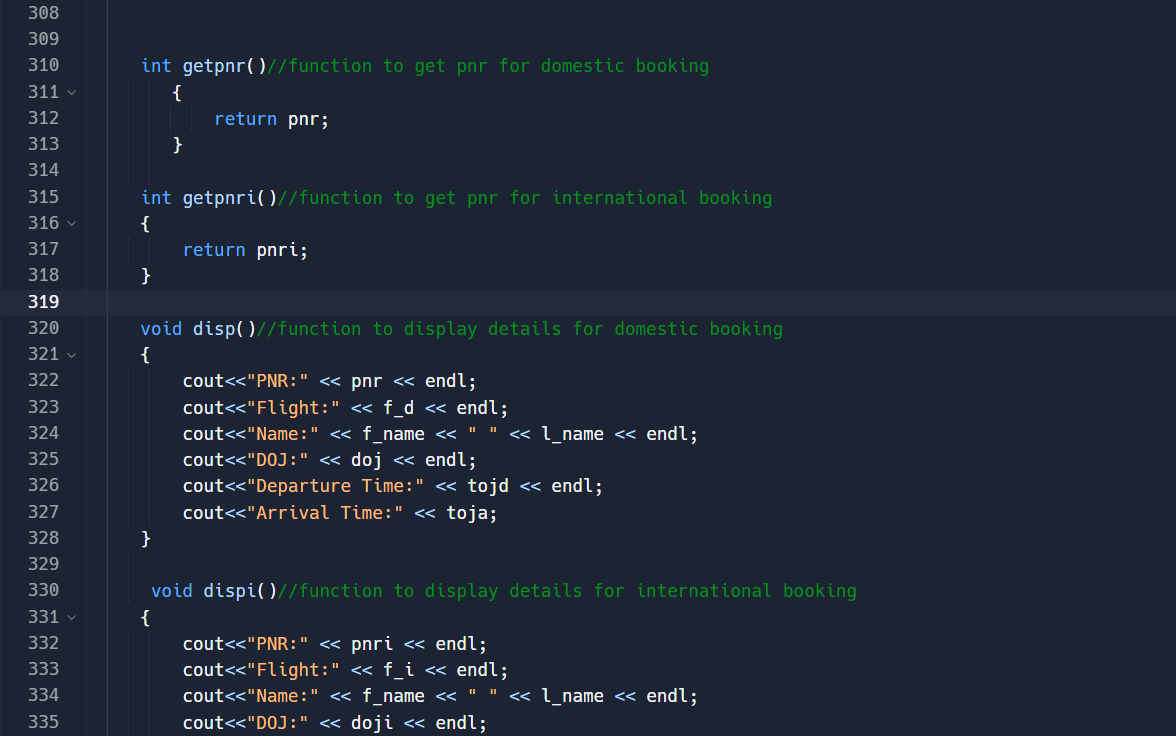
****

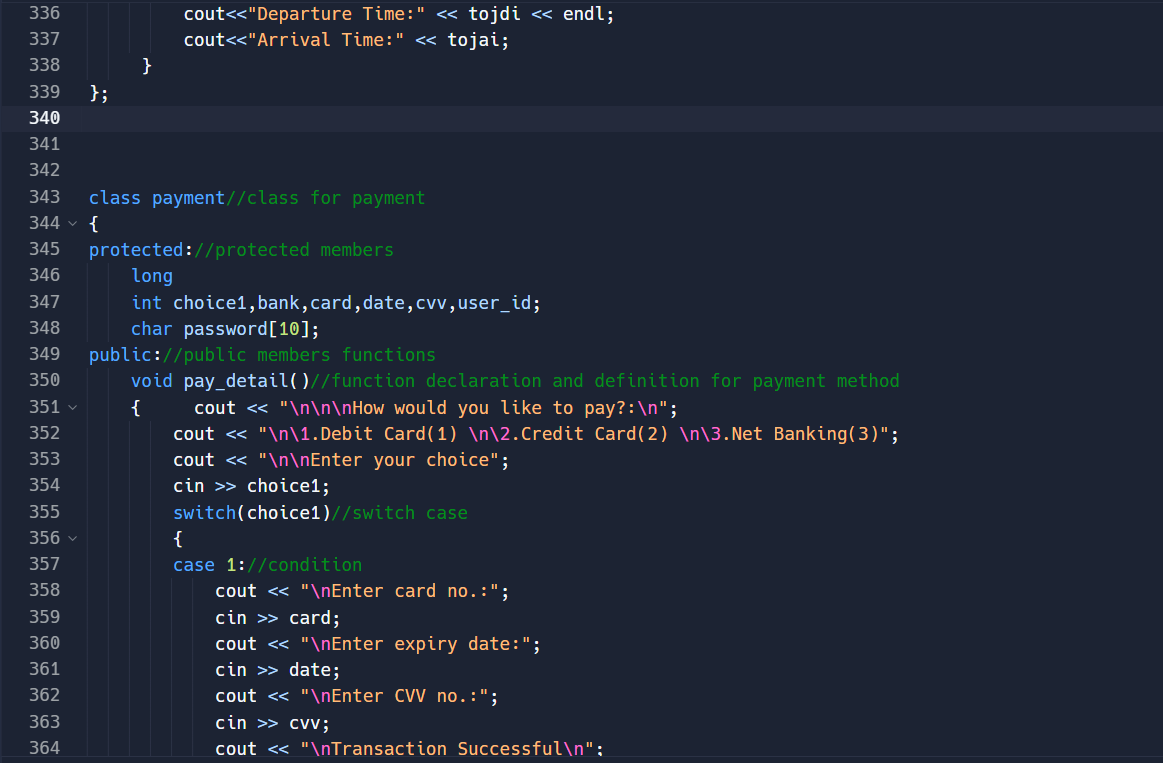
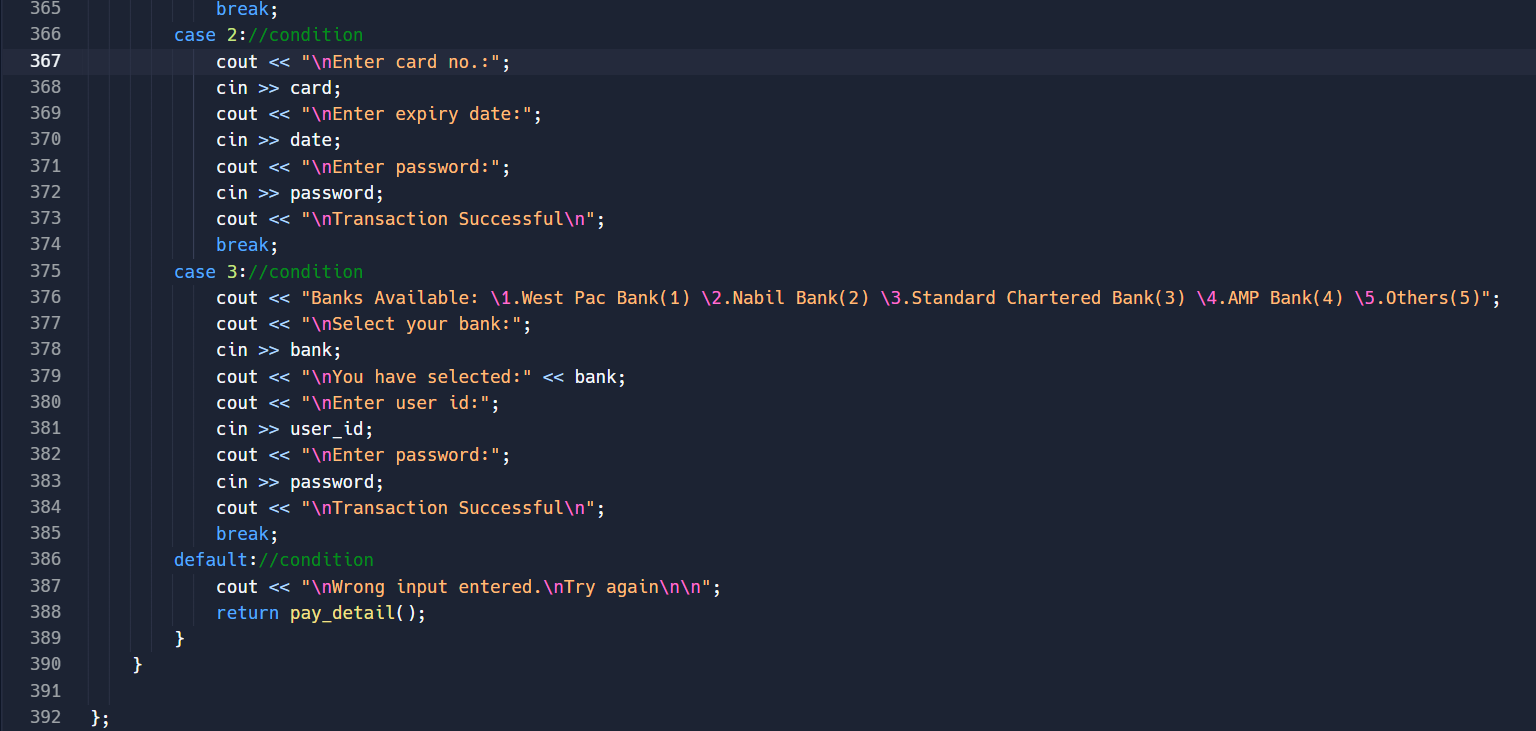
****

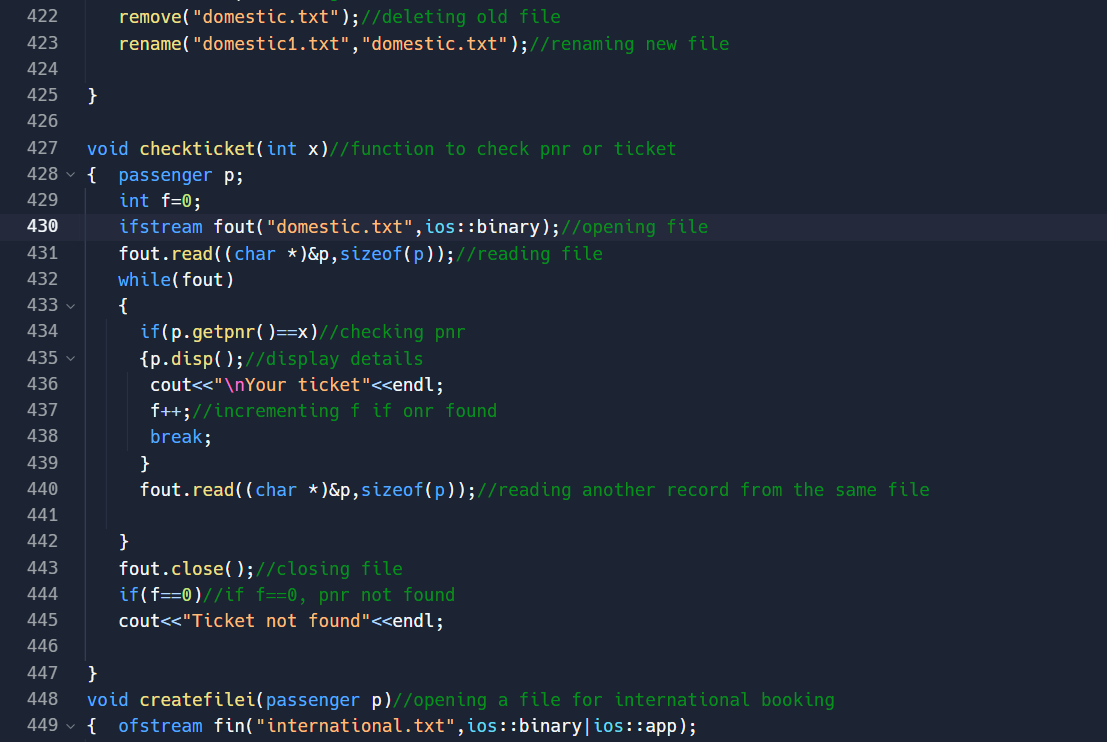
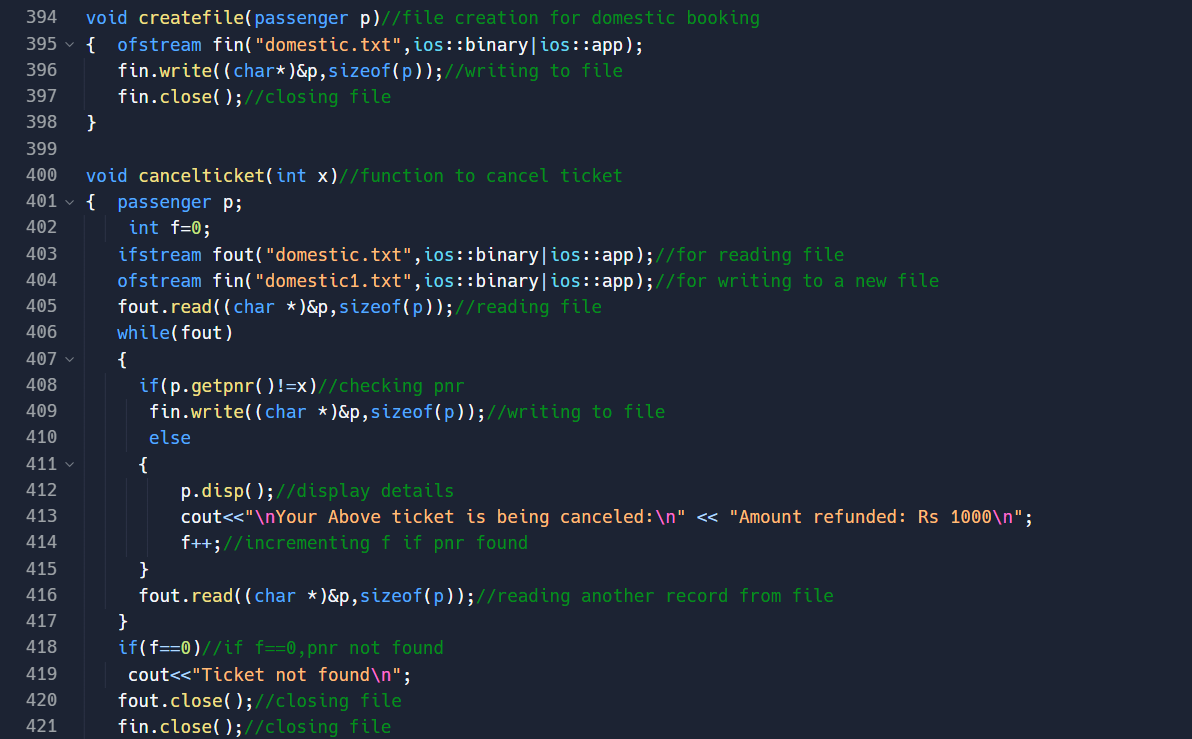
****

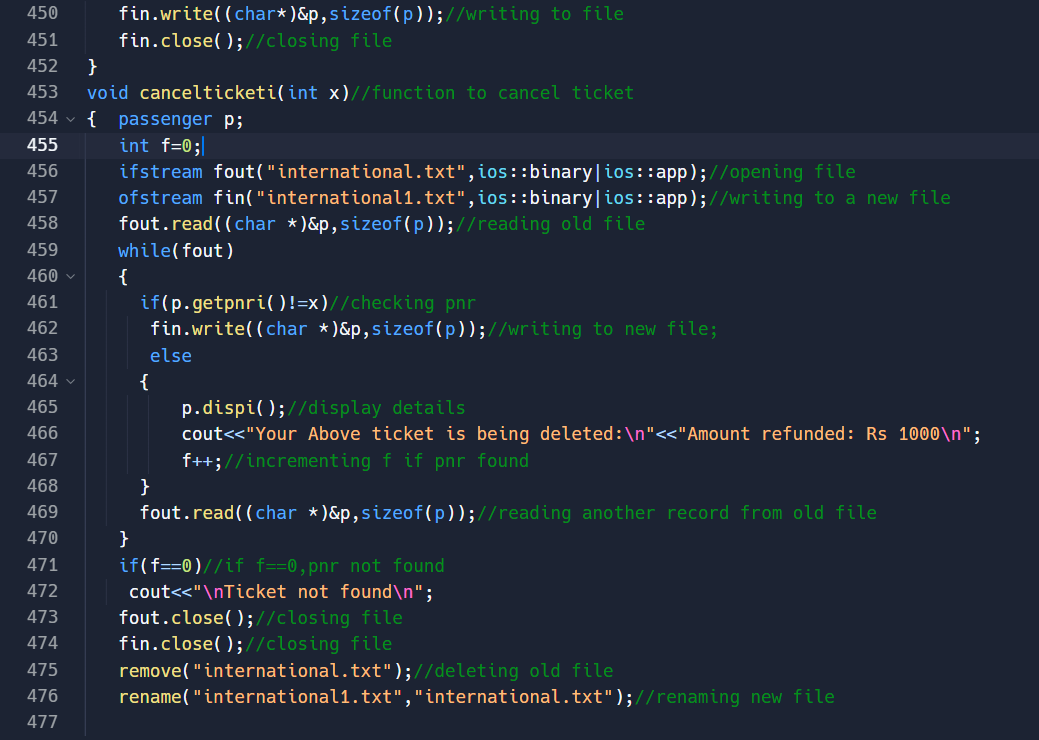
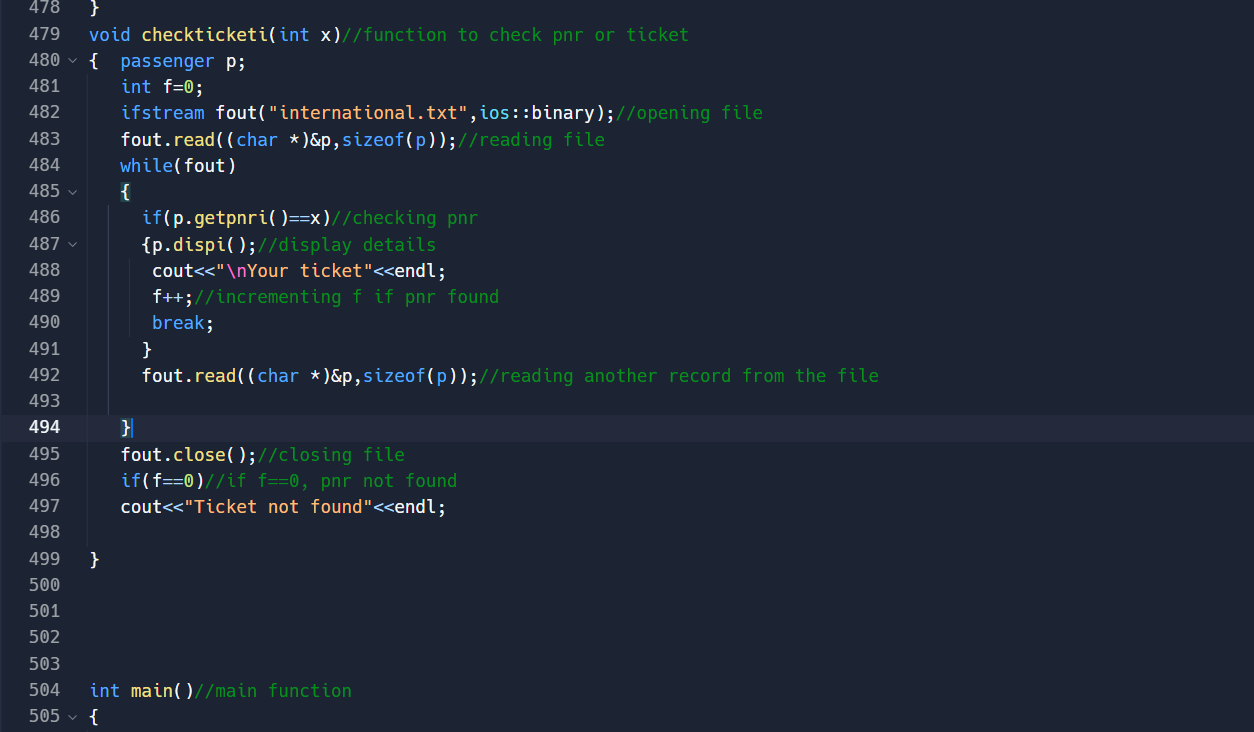
****

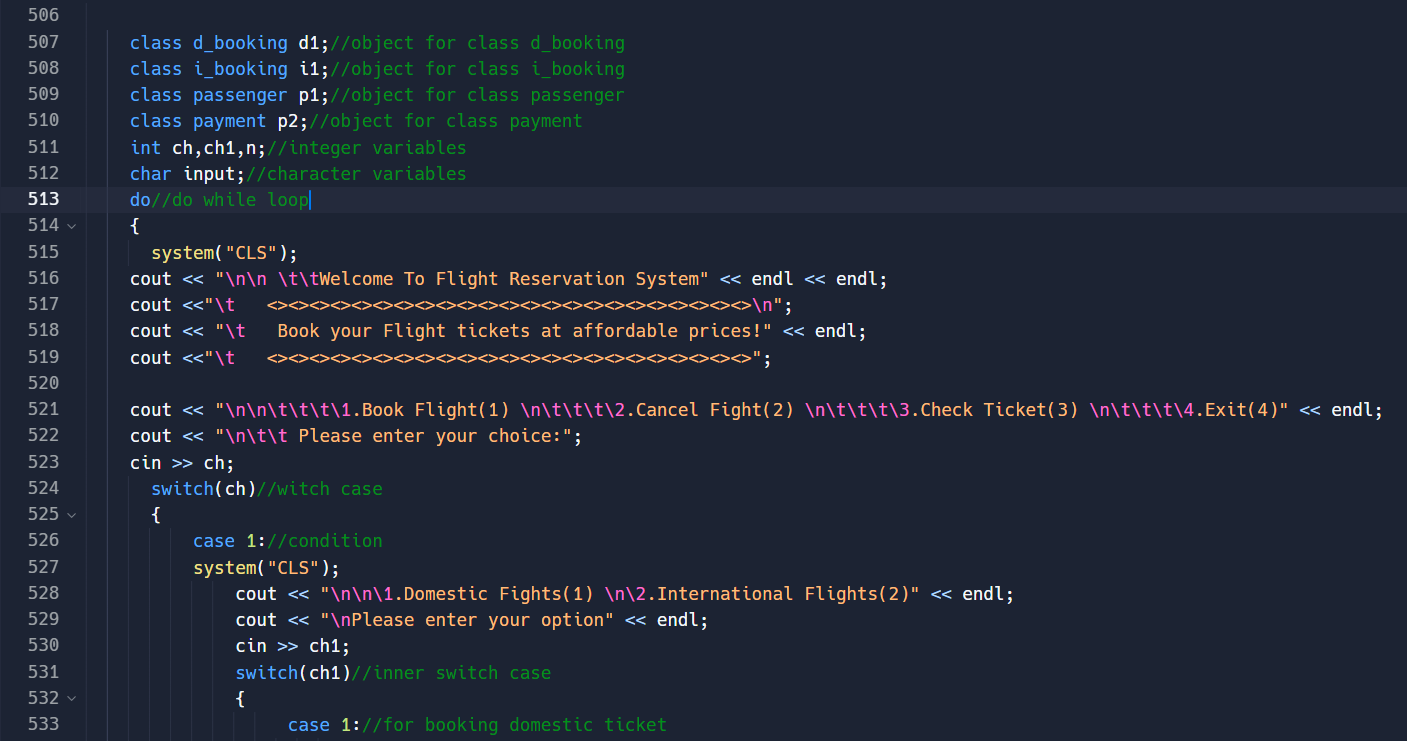
****

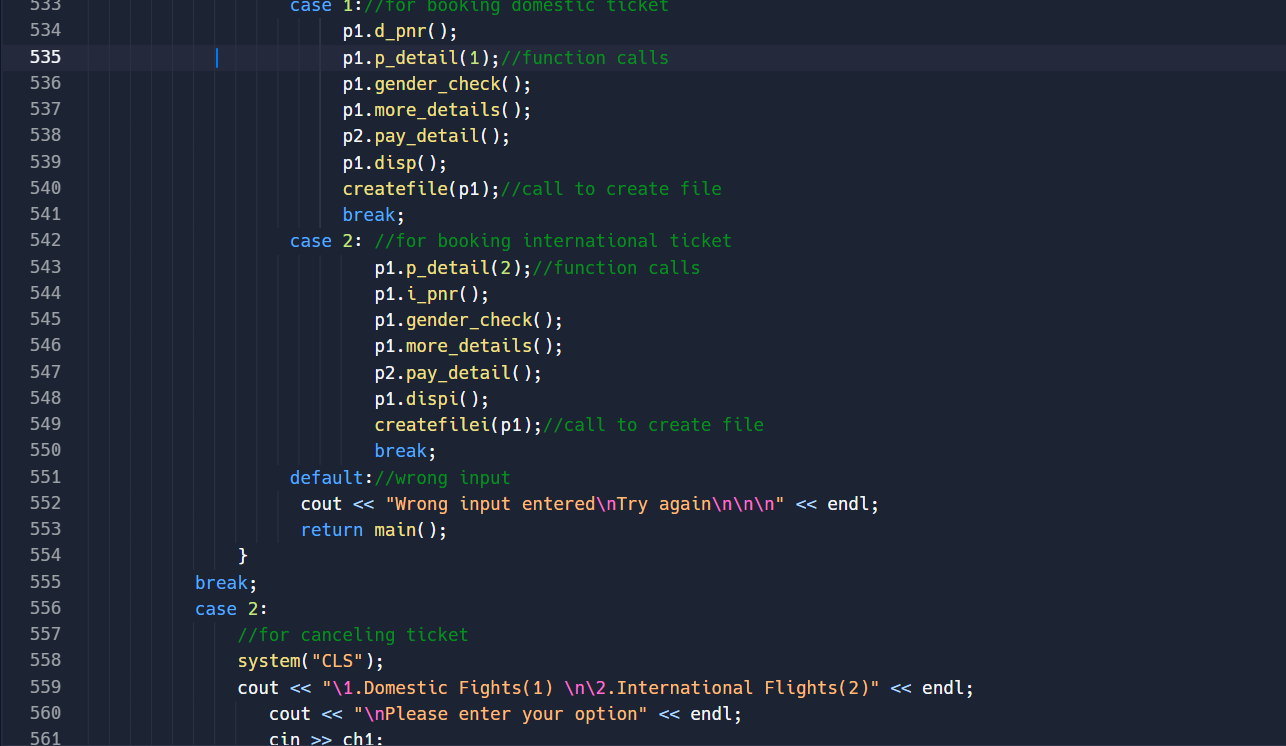
****

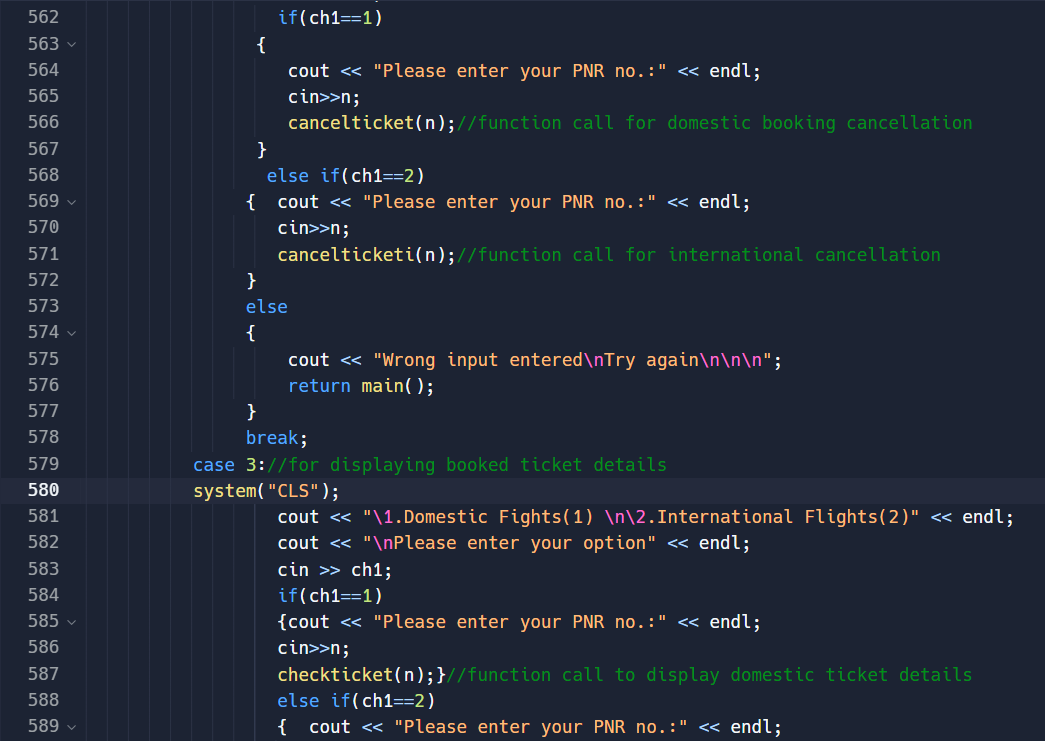
****

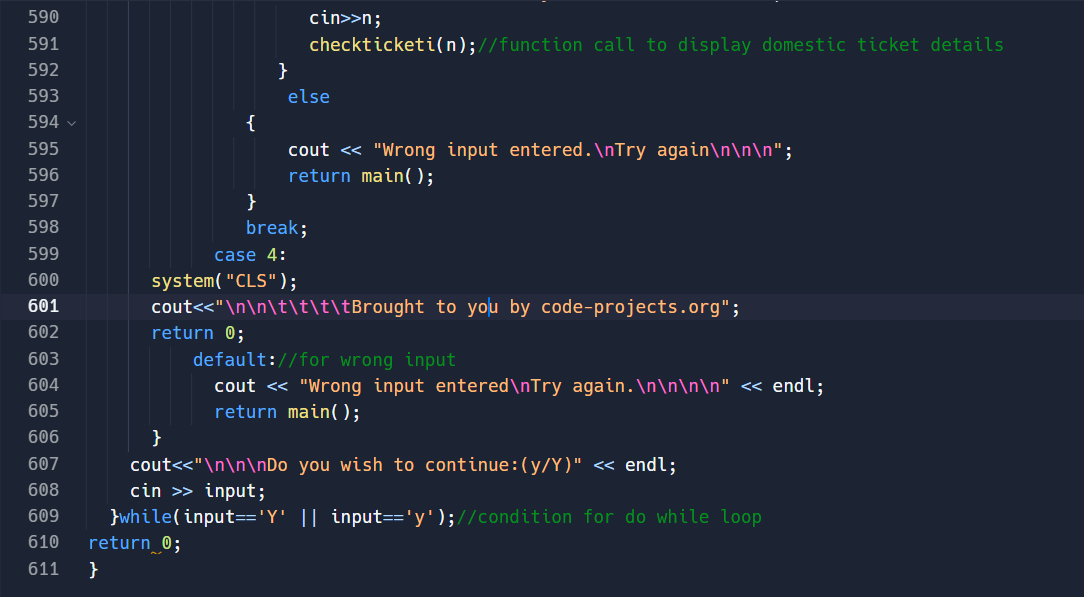
****

****

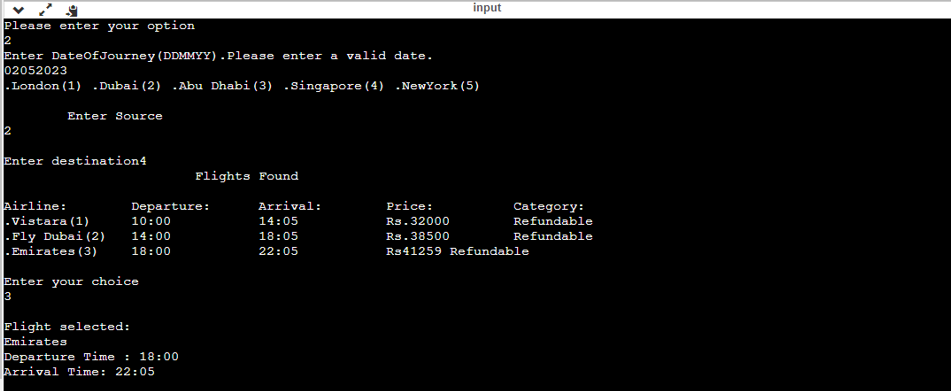
****

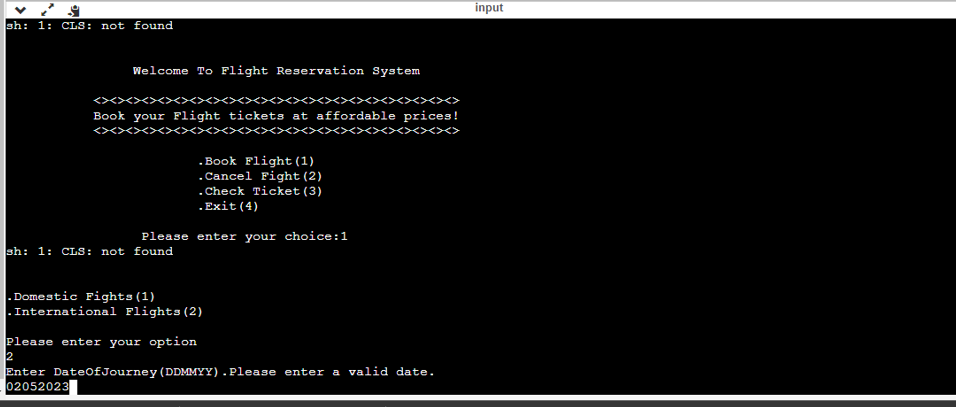
****

****

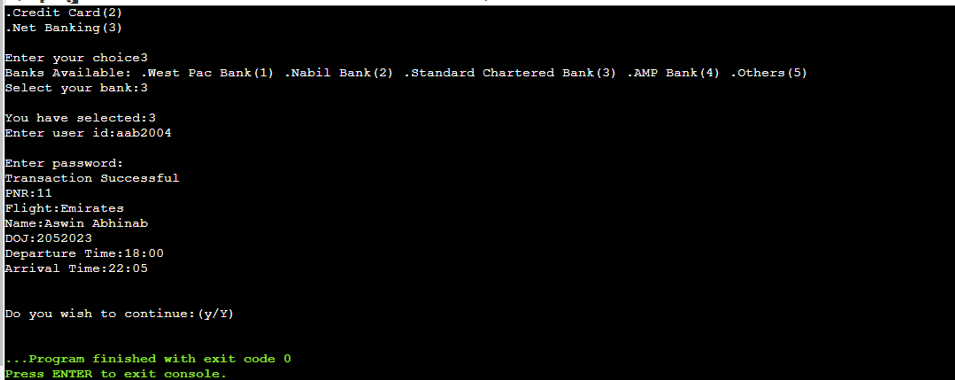
****

**Output:**









**Conclusion and Results:**

A well-designed and implemented airline reservation system should provide a reliable and efficient platform for customers to book flights, manage their reservations, and obtain necessary information about their flights. The system should be user-friendly, secure, and able to handle a large volume of users and transactions simultaneously. It should also be able to integrate with other systems such as payment gateways, flight management systems, and airport information systems to provide a seamless and convenient experience for customers.

A successful airline reservation system should enable airlines to improve their revenue and operational efficiency by reducing manual processes and improving customer satisfaction. The system should also have robust reporting capabilities to enable airlines to track their revenue, customer data, and other key metrics.

In conclusion, the success of an airline reservation system project depends on how well it meets the needs of its intended users, its ability to handle unexpected events, and its scalability and extensibility. The project should prioritize the key features and functionality to be developed and ensure that the system is user-friendly, secure, and able to handle a large volume of users and transactions simultaneously.

**Reference:**

The following references have been used by me, during all the phases of the MSE project:

1. <http://www.w3schools.com/>

2. [www.msdn.microsoft.com](http://www.msdn.microsoft.com)

3. Apache J Meter - <http://jakarta.apache.org/jmeter/>

4. http://mse.cis.ksu.edu/ - For MSE Project Portfolio.

5. SLOC Metrics Tool for .NET framework 1.1 <http://www.softpedia.com/get/Programming/Other-ProgrammingFiles/SLOC-Metrics.shtml>

6. http://www.devarticles.com/c/b/SQL-Server/ - SQL server 2000 help

7. SQL Server 2000 download - <http://www.microsoft.com/downloads/>

8. http://www.c-sharpcorner.com/