

PROJECT REPORT

Submitted By

AKHIL A - REG. NO – RA2211033010005
RAHUL N - REG. NO – RA2211033010019
DHARAN SS - REG. NO – RA2211033010021

Under the guidance of

DR.P.SARAVANAN

Assistant Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in SWE



COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603203

MAY 2023



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203**

BONAFIDE CERTIFICATE

Certified that this Project Report titled **"AIRLINE MANAGEMENT SYSTEM"** is the bonafide work done By RAHUL N (Reg. no-RA2211033010019), AKHIL A(Reg. no-RA2211033010005), DHARAN SS(Reg. no-RA2211033010021) who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

A handwritten signature in black ink, appearing to read 'Dr. P. Saravanan'.

SIGNATURE

Dr.P.SARAVANAN

ODDP – Course Faculty

Assistant Professor

Department of Computational Intelligence

SRMIST

A handwritten signature in blue ink, appearing to read 'Dr. R. Annie Uthra'.

SIGNATURE

Dr.R.ANNIE UTHRA

Professor & Head

**Department of Computational
Intelligence**

School of Computing

SRMIST

TABLE OF CONTENTS

S.No	CONTENT	PAGE NO
1.	Problem Statement	4
2.	Modules of Project	5
3.	Diagrams	6-20
	a. Use case Diagram	6
	b. Class Diagram	7-8
	c. Sequence Diagram	9
	d. Collaboration Diagram	10-11
	e. State Chart Diagram	12-13
	f. Activity Diagram	14
	g. Component Diagram	15-16
	h. Deployment Diagram	17-18
	i. Package Diagram	19-20
4.	Code/Output Screenshots	21-35
5.	Conclusion and Results	36
6.	References	37

PROBLEM STATEMENT

This is an airline reservation system written in C++20. This project is a simple

idea with a sophisticated execution.

Users that hop onto the platform may log in using existing credentials or create a new account with minimal effort. All credentials are stored and maintained on the platform's database for authentication.

Once logged in, the user is greeted by the main profile window from where they may

- Check existing bookings, made using the platform.
- Initiate a new booking using searching for flights given fixed criteria such as arrival and departure times, flight class, meal preferences, etc.
- Logout, and thereby exiting the platform

Checking existing bookings:

Data fetched from the platform's bookings database is displayed in a tabular form containing all essential details regarding the flight. The user may also cancel an existing active booking.

Searching for flights:

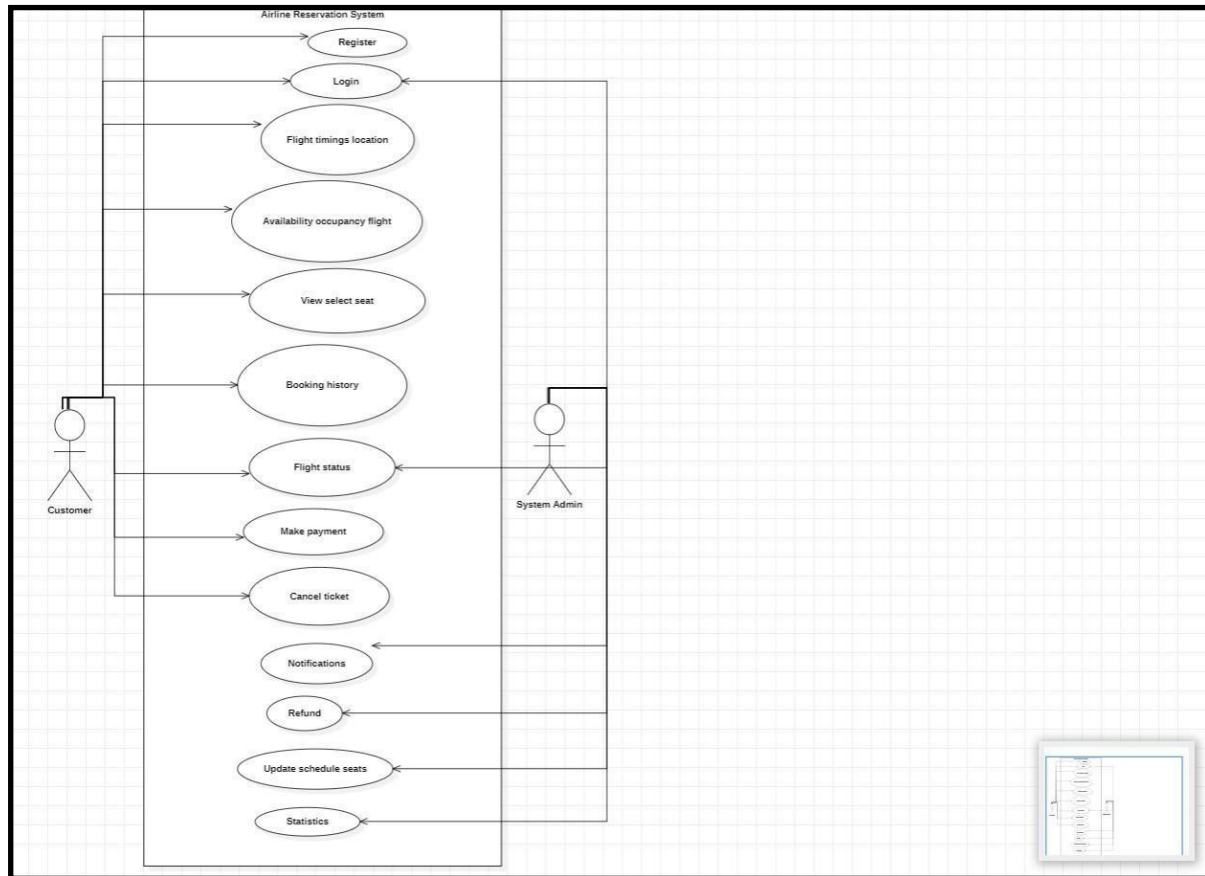
Criteria filled in by the user are processed and cross-referenced with the airlines' flight databases to yield accurate results for onward and return trip flights. The user may choose from the list of flights according to their needs and proceed to checkout. If the user has opted for online payment, the bank transaction window shows up holding an array of payment options such as credit, debit, upi or wallet. They may then complete the transaction and reserve the seat, failing which the reservation is withdrawn and the checkout cancelled, bringing the user back to their main profile window.

MODULES OF PROJECT

Systems known as airline reservation systems (ARS) enable airlines to market their inventory (seats). It includes data on prices and schedules as well as a database of issued tickets and reservations (or passenger name records) (if applicable). ARSs are a component of passenger service systems (PSSs), which are software programmes that facilitate direct passenger communication.

The computer reservations system finally replaced ARS (CRS). A specific airline's reservations are processed by a computer reservation system that connects to a global distribution system (GDS), which enables travel agencies and other distribution channels to process reservations for the majority of major airlines through a single system.

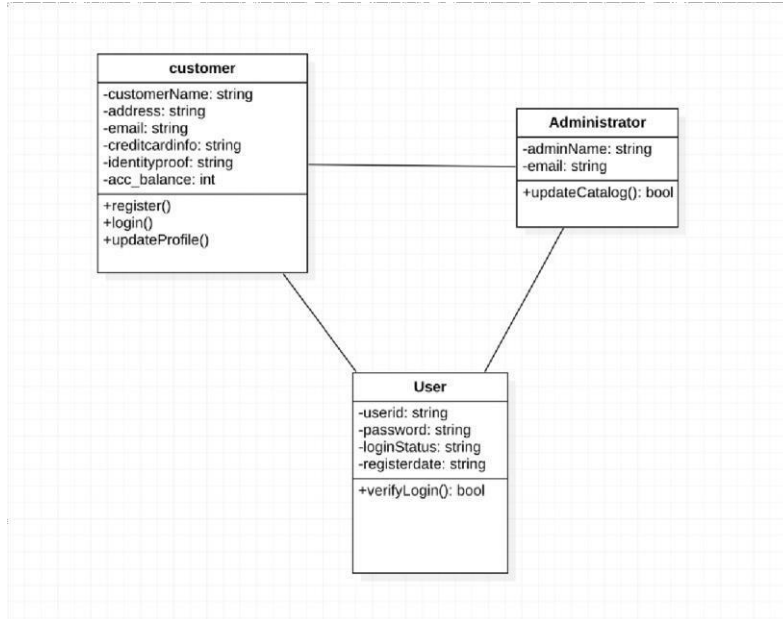
Use case diagram with explanation



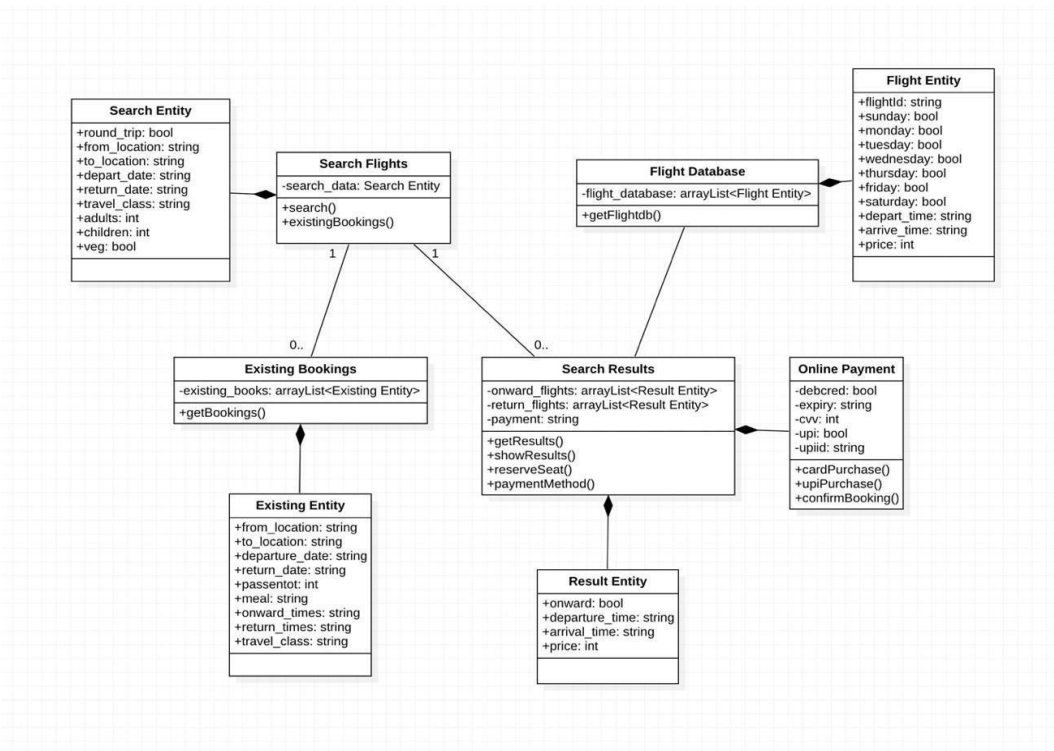
The use cases in the diagram represents the main processes in AIRLINE MANAGEMENT SYSTEM. Then they will be broken down into more specific use cases depending on the included processes of the main use case. Each of these use cases explains how the system handles the actions or scenarios requested by the user.

Class diagram with explanation

Login:



Airline Reservation:



Class Diagrams

This class diagram consists of 3 classes.

Association is used in the diagram.

The customer class is responsible for credential input and profile updating,

the User class is responsible for verifying credentials entered and the Administrator class is responsible for updating the user catalogue.

Airline Reservation:

The class diagram consists of 5 principal classes and 4 subsidiary classes.

Association and Aggregation are used in the diagram. The Search Flights class may take in criteria and initiate a search as well as go to the existing bookings page.

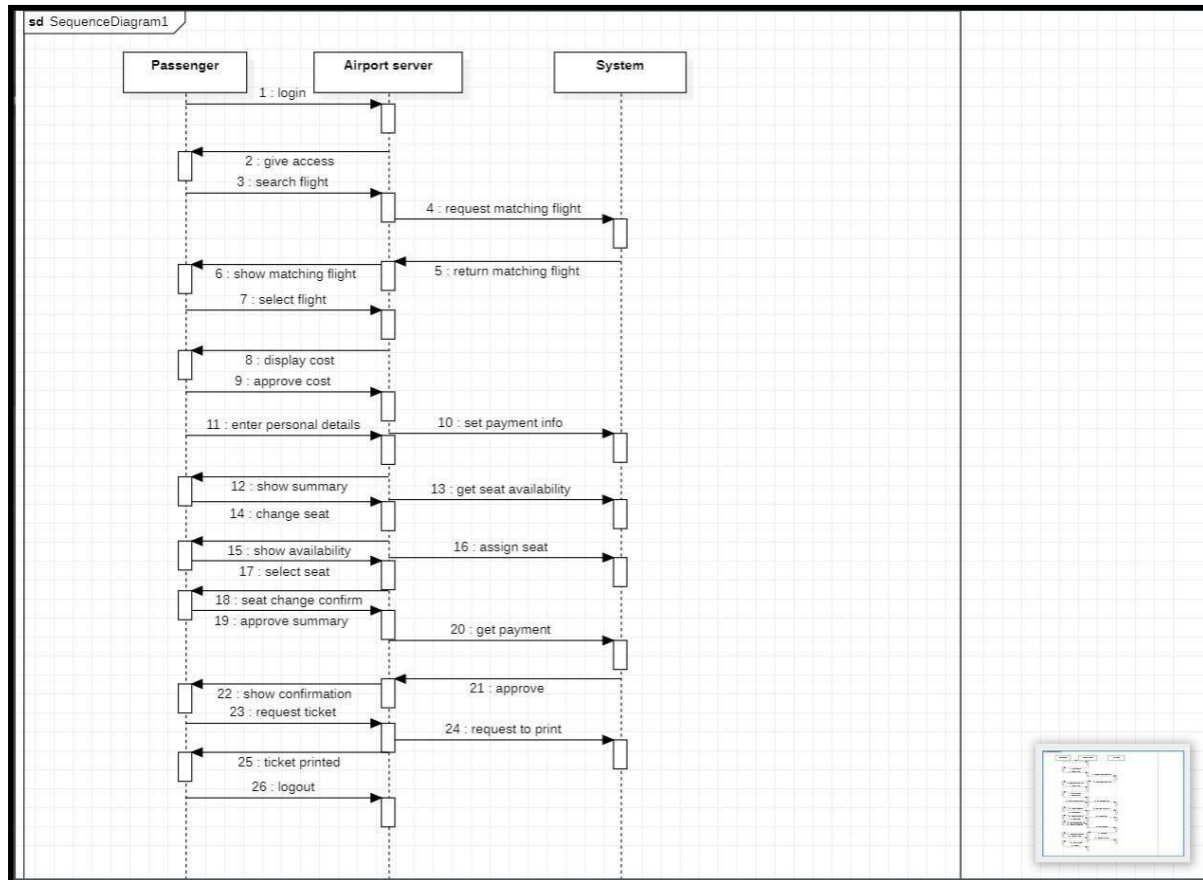
The Existing Bookings class is used to display all bookings made by the user on the platform, both active and inactive ones.

The Search Results class displays all available flights according to the user's criteria and the user may proceed to checkout.

The Online Payment Class is responsible for handling all payment operations pertaining to the booking being made securely.

The Flight Database class is responsible for storing, updating and maintaining all user data such as profile information and bookings made.

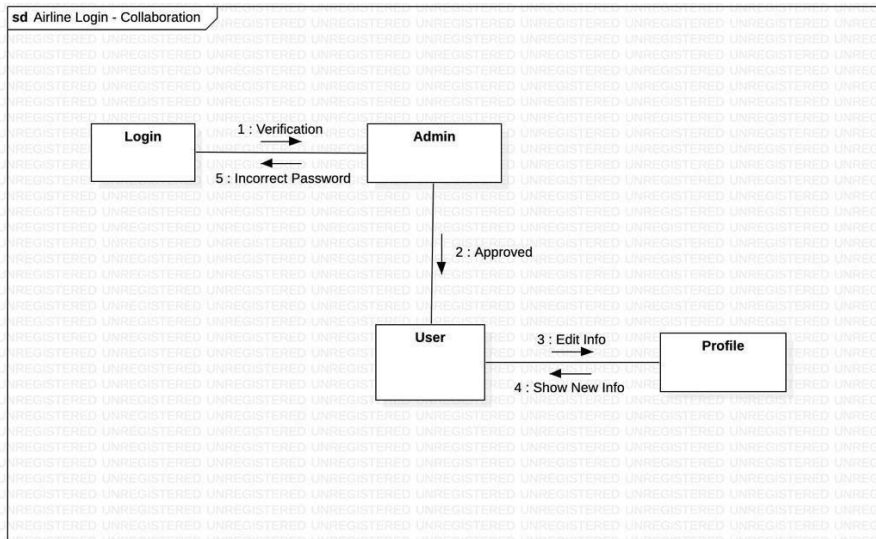
Sequence diagram with explanation



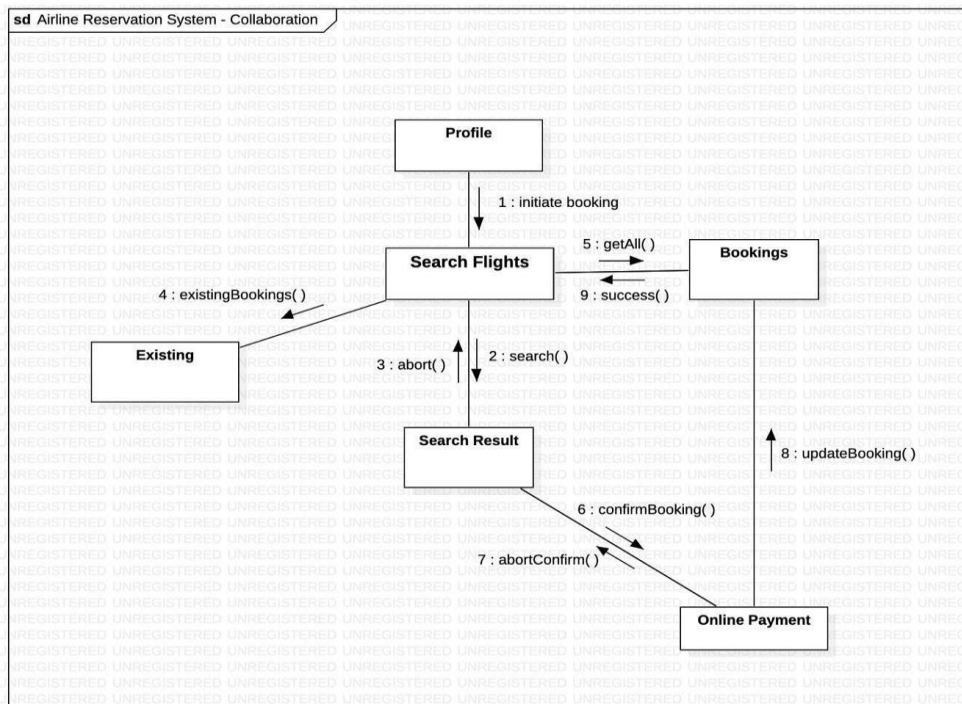
A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

Collaboration diagram with explanation

Login:



Airline reservation system:



Login:

This collaboration diagram consists of 4 lifelines. The Login lifeline is responsible for credential input and updating profile and passwords, the Admin lifeline is responsible for verifying credentials entered and the User lifeline is invoked on a successful login.

Airline Reservation:

The collaboration diagram consists of 6 lifelines.

The Search Flights lifeline may take in criteria and initiate a search as well as go to the existing bookings page.

The Existing Bookings lifeline is used to display all bookings made by the user on the platform, both active and inactive ones.

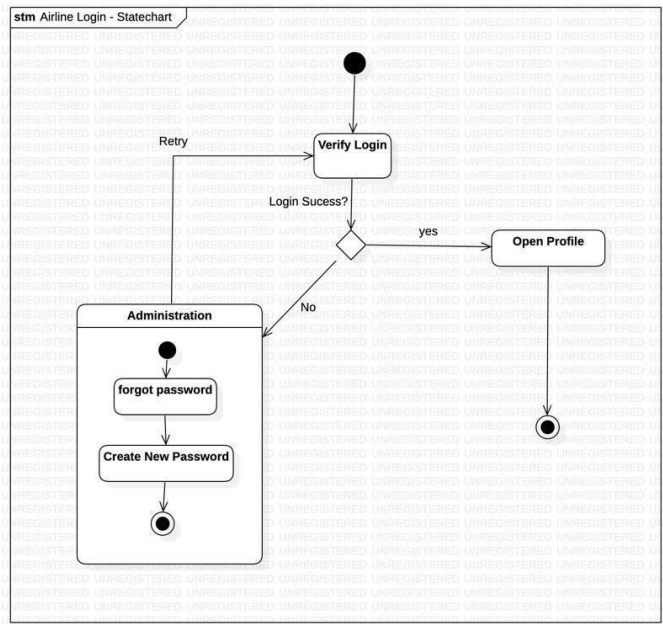
The Search Results lifeline displays all available flights according to the user's criteria and the user may proceed to checkout.

The Online Payment lifeline is responsible for handling all payment operations pertaining to the booking being made securely.

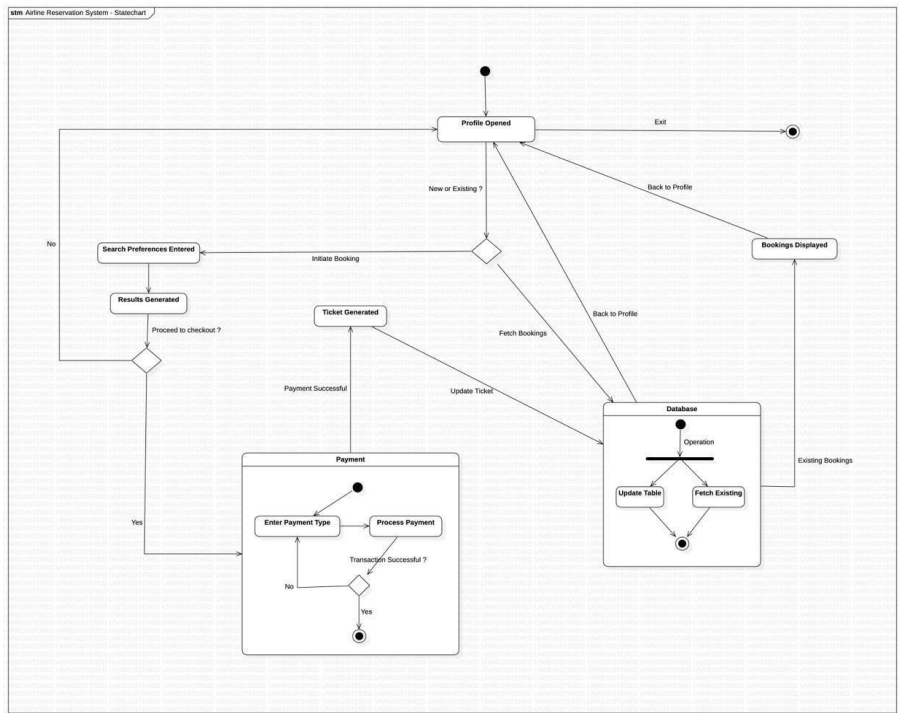
The Bookings lifeline is responsible for storing, updating and maintaining all user data such as profile information and bookings made.

State chart diagram with explanation

Login:



Airline Reservation System:



The statechart diagram consists of 2 states and 1 composite state.

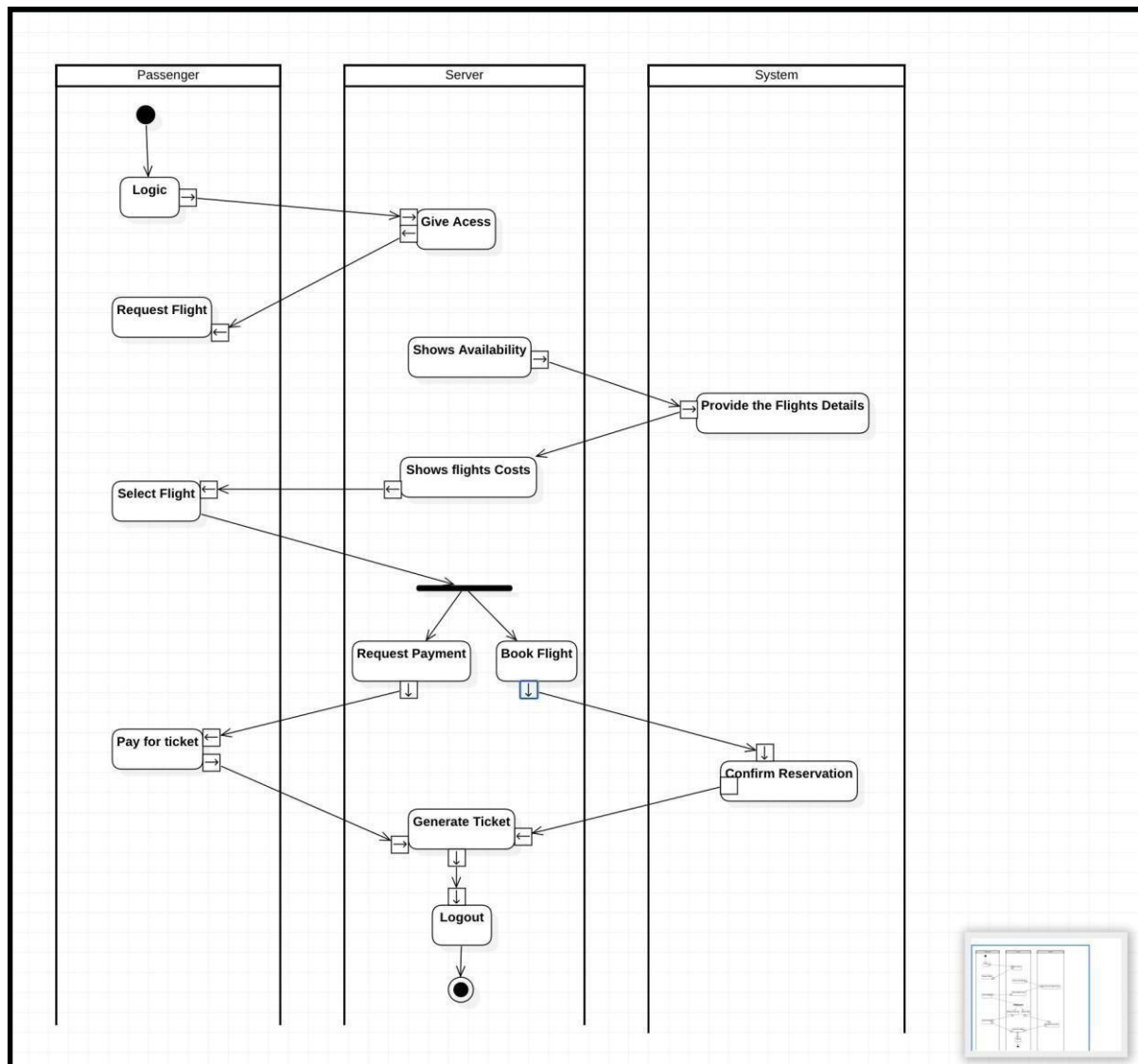
- 1 . User may enter password
- 2 . If password is correct, user is authenticated and may access their profile.
- 3 . If the password is wrong, user may opt to change their password and proceed to authentication.
- 4 . Once authenticated, user may access their profile.

Airline Reservation:

The statechart diagram consists of 5 states and 2 composite states.

- 1 . From the main page, if the user choses to make a new booking
 - a . If booking is initiated, search criteria is entered by the user.
 - b . The user may select from the flights provided.
 - c . If the user decides to abort the booking, they are redirected back to the homepage, but if the user proceeds to check-out, they are redirected to the payment portal.
 - d . If transaction is successful, the booking is updated in the flight database. Else, the user is either prompted to retry or redirected to the homepage.
- 2 . If the user choses to view existing bookings
 - a . All bookings details made on the platform are fetched from the database and displayed to the user.

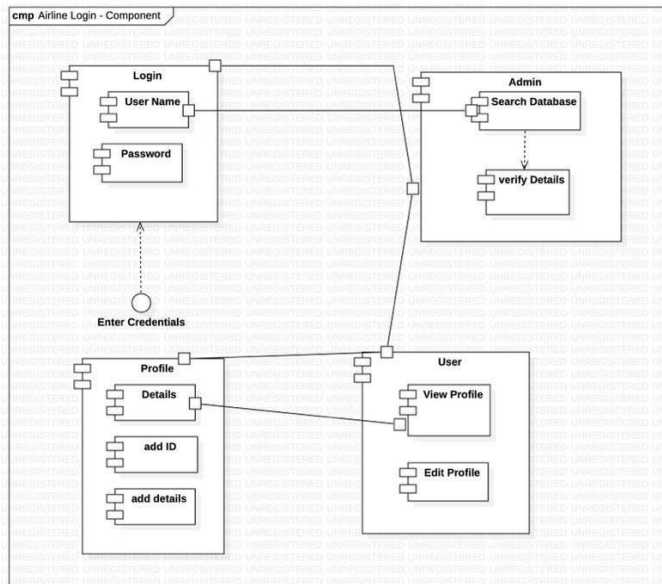
Activity diagram with explanation



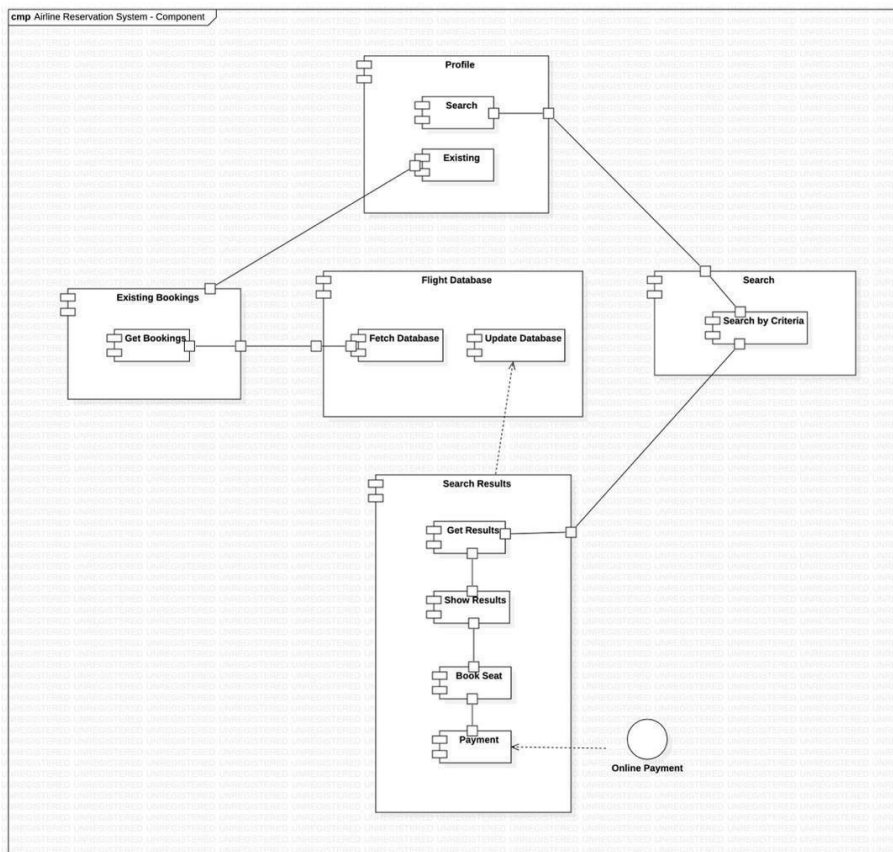
Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. In this case it is an activity diagram with swim lane. This diagram shows the flow between passenger, server and system.

Component Diagram with explanation

Login:



Airline reservation:



II Explanation – Component Diagrams

Login:

The component diagram consists of 4 base components and 9 sub components.

The diagram starts with interface where the user enters their credentials in login page which is verified by the admin database. After verification the user is redirected to user page where they can view or update their profile.

Airline Reservation:

The component diagram consists of 5 base components and 10 sub components.

The user may choose to view existing bookings or make a new booking from the Profile component. The Search component may take in criteria and initiate a search.

The Search Results component displays all available flights according to the user's criteria and the user may proceed to checkout.

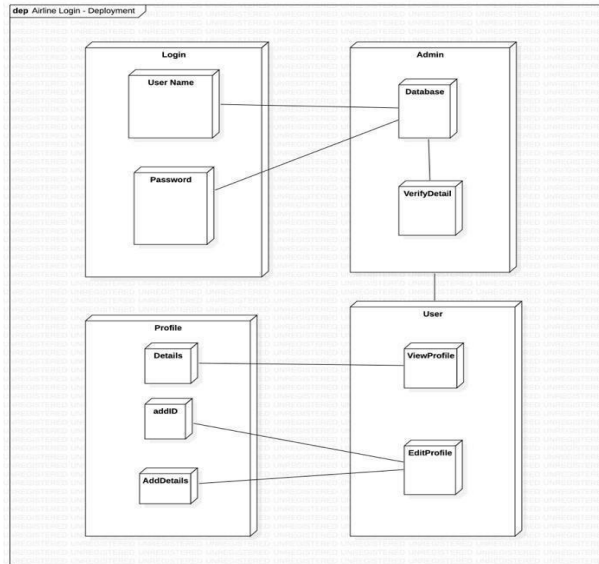
The Online Payment component is responsible for handling all payment operations pertaining to the booking being made securely.

The Existing Bookings component is used to display all bookings made by the user on the platform, both active and inactive ones.

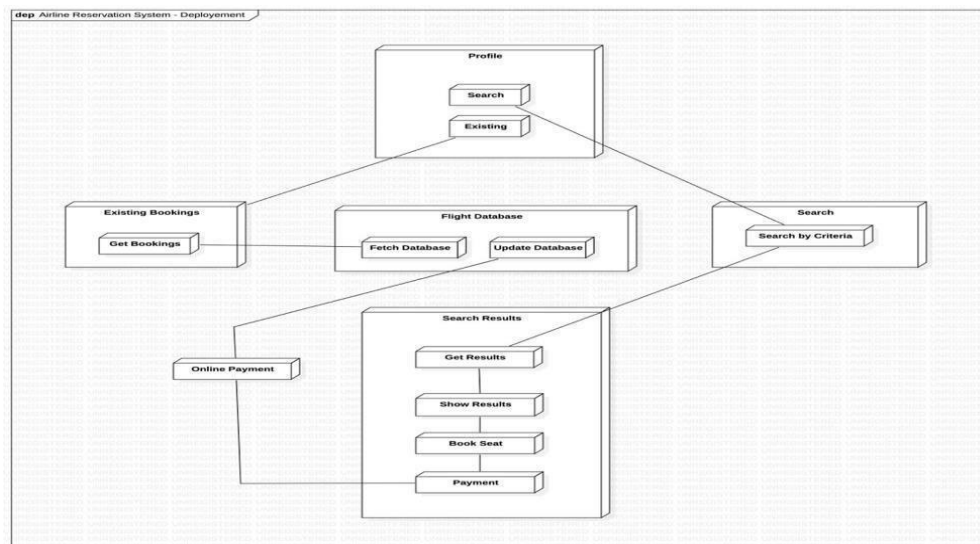
The Flight Database component is responsible for storing, updating and maintaining all user data such as profile information and bookings made.

Deployment diagram with explanation

Login:



Airline reservation:



III Explanation – Deployment Diagrams

Login:

The deployment diagram consists of 4 base nodes and sub nodes.

The diagram starts with interface where the user enters their credentials in login page which is verified by the admin database. After verification the user is redirected to user page where they can view or update their profile.

Airline Reservation:

The component diagram consists of 5 base nodes and sub nodes.

The user may choose to view existing bookings or make a new booking from the Profile node.

The Search node may take in criteria and initiate a search.

The Search Results node displays all available flights according to the user's criteria and the user may proceed to checkout.

The Online Payment node is responsible for handling all payment operations pertaining to the booking being made securely.

The Existing Bookings node is used to display all bookings made by the user on the platform, both active and inactive ones.

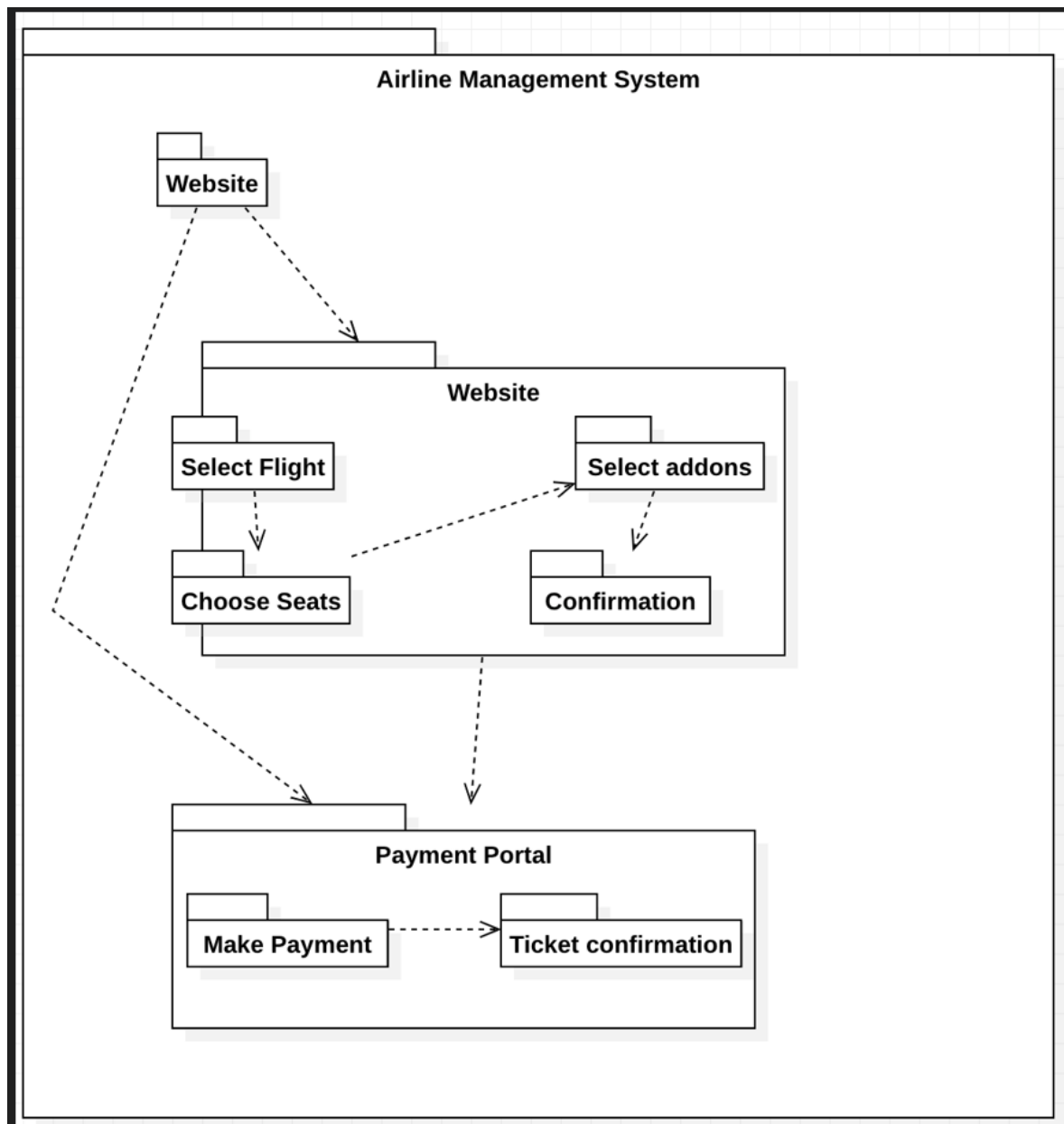
The Flight Database node is responsible for storing, updating and maintaining all user data such as profile information and bookings made.

IV Explanation –Package Diagram

A package is a grouping of model elements. Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. Packages themselves may contain other packages. A package may contain both subordinate packages and ordinary model elements. All UML model elements and diagrams can be organized into packages.

A package is represented as a folder, shown as a large rectangle with a tab attached to its upper left corner. If contents of the package are not shown, then the name of the package is placed within the large rectangle. If the contents of the package are shown, then the name of the package may be placed on the tab. The contents of the package are shown within the large rectangle.

Packages can be used to designate not only logical and physical groupings but also use-case groups. A use-case group, as the name suggests, is a package of use cases.



C++ Code

```
#include <iostream>
#include <fstream>
#include <string.h>
using namespace std;
int glob=0;
int global=10;
//=====
//=====
//=====
//=====
class d_booking
{
protected:
    int pnr;
    char f_d[10],toja[7],toj_d[7];
    long int doj;
    int choice,src,dest;
public:
    void d_pnr()
    {
        glob++;
        pnr=glob;
    }
    int j_detail() // function declaration and definition for domestic
journey
    {
        cout << "\n===== " << endl
;
        cout << "\nEnter Date Of Journey (DDMMYY)\n>>> ";
        cin >> doj;
        cout << "\n===== " << endl ;
        cout << "\n1.Delhi(1) \t2.Mumbai(2) \t3.Bangalore(3)
\t4.Chennai(4)" << endl << endl;
        cout << "Enter Source\n>>> ";
        cin >> src;
        cout << "\nEnter destination\n>>> ";
        cin >> dest;
        cout << "\n===== " << endl ;
        if((src==1 && dest==2) || (src==2 && dest==1))
        {
            cout << "Flights Found" << endl << endl;
            cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
            cout << "\1.Air
```

```

India(1)\t08:00\t\t11:05\t\tRs.5000\t\tRefundable\n";
    cout <<
"\2.Spicejet(2)\t14:00\t\t17:05\t\tRs.5500\t\tRefundable\n";
cout << "\3.Jet
Airways(3)\t19:00\t\t22:05\t\tRs.6000\t\tRefundable\n";
}
else if((src==1 && dest==3) || (src==3 && dest==1))
{
    cout << "Flights Found" << endl << endl;
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
cout << "\1.Air
India(1)\t08:00\t\t11:05\t\tRs.5000\t\tRefundable\n";
    cout <<
"\2.Spicejet(2)\t14:00\t\t17:05\t\tRs.5500\t\tRefundable\n";
cout << "\3.Jet
Airways(3)\t19:00\t\t22:05\t\tRs.6000\t\tRefundable\n";
}
else if((src==1 && dest==4) || (src==4 && dest==1))
{
    cout << "Flights Found" << endl << endl;
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
cout << "\1.Air
India(1)\t08:00\t\t11:05\t\tRs.4000\t\tRefundable\n";
    cout <<
"\2.Spicejet(2)\t14:00\t\t17:05\t\tRs.4250\t\tRefundable\n";
cout << "\3.Jet
Airways(3)\t19:00\t\t22:05\t\tRs.6100\t\tRefundable\n";
}
else if((src==2 && dest==3) || (src==3 && dest==2))
{
    cout << "Flights Found" << endl << endl;
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
cout << "\1.Air
India(1)\t08:00\t\t11:05\t\tRs.5400\t\tRefundable\n";
    cout <<
"\2.Spicejet(2)\t14:00\t\t17:05\t\tRs.2500\t\tRefundable\n";
cout << "\3.Jet
Airways(3)\t19:00\t\t22:05\t\tRs.2890\t\tRefundable\n";
}
else if((src==2 && dest==4) || (src==4 && dest==2))
{
    cout << "Flights Found" << endl << endl;
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
cout << "\1.Air
India(1)\t08:00\t\t11:05\t\tRs.5000\t\tRefundable\n";
    cout <<
"\2.Spicejet(2)\t14:00\t\t17:05\t\tRs.4500\t\tRefundable\n";

```

```

    cout << "\3.Jet
Airways(3)\t19:00\t\t22:05\t\tRs.6000\t\tRefundable\n";
}
else if((src==3 && dest==4) || (src==4 && dest==3))
{
    cout << "Flights Found" << endl << endl;
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
    cout << "\1.Air
India(1)\t08:00\t\t11:05\t\tRs.5800\t\tRefundable\n";
    cout <<
"\2.Spicejet(2)\t14:00\t\t17:05\t\tRs.5508\t\tRefundable\n";
    cout << "\3.Jet
Airways(3)\t19:00\t\t22:05\t\tRs.6050\t\tRefundable\n";
}
else if(src==dest)
{
    cout << "\nSource and destination can't be same.\nTry
again\n\n\n" << endl;
    return j_detail();
}
else
{
    cout << "\nWrong input entered\nTry again\n\n\n" << endl;
    return j_detail();
}
}

int select_flight() //function declaration and definition for selecting
flight
{
    cout << "\nEnter your choice\n>>> " << endl;
    cin >> choice;
    switch(choice)
    {
    case 1:
        cout << "\nFlight selected:"<<endl;
        cout << "Air India"<<endl;
        strcpy(f_d,"Air India");//copy to string
        cout << "Departure Time : 08:00"<<endl;
        cout<<"Arrival Time: 11:05"<<endl;
        strcpy(tojd,"8:00"); //copy to string
        strcpy(toja,"11:05");// copy to string
        break;
    case 2:
        cout << "\nFlight selected:"<<endl;
        cout << "Spicejet"<<endl;
        strcpy(f_d,"Spicejet");//copy to string
        cout << "Departure Time : 14:00"<<endl;

```

```

cout<<"Arrival Time: 17:05"<<endl;
strcpy(tojd,"14:00");//copy to string
strcpy(toja,"17:05");//copy to string
break;
case 3:
cout << "\nFlight selected:" << endl;
cout << "Jet Airways" << endl;
strcpy(f_d,"Jet Airways");//copy to string
cout << "Departure Time : 19:00" << endl;
cout << "Arrival Time: 22:05" << endl;
strcpy(tojd,"19:00");//copy to string
strcpy(toja,"22:05");//copy to string
break;
default:
cout << "Wrong input entered.\nTry again" << endl;
return select_flight();
}
}
};
//=====
//=====
class i_booking
{
protected:
int pnri;
char f_i[10],tojai[7],tojdi[7];
long int doji;
int srci,desti,choicei;
public:
void i_pnr()
{
global++;
pnri=global;
}
int j_detaili()// function declaration and definition for journey details
{
cout << "\n===== " << endl
;
cout << "\nEnter Date Of Journey (DDMMYY)\n>>> ";
cin >> doji;
cout << "\n===== " << endl ;
cout << "\1.London(1) \2.Dubai(2) \3.Abu Dhabi(3) \4.Singapore(4)
\5.NewYork(5) " << endl << endl;
cout << "\tEnter Source\n>>> ";
cin >> srci;
cout << "\nEnter destination\n>>> ";
cin >> desti;

```



```

cout << "\n===== " << endl ;
cout << "Flights Found" << endl << endl;
if((srci==1 && desti==3) || (srci==3 && desti==1))
{
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
    cout <<
"\1.Vistara(1)\t10:00\t\t14:05\t\tRs.25000\tRefundable\n";
    cout << "\2.Fly
Dubai(2)\t14:00\t\t18:05\t\tRs.21500\tRefundable\n";
    cout <<
"\3.Emirates(3)\t18:00\t\t22:05\t\tRs.24000\tRefundable\n";
}
else if((srci==1 && desti==4) || (srci==4 && desti==1))
{
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
    cout <<
"\1.Vistara(1)\t10:00\t\t14:05\t\tRs.25500\tRefundable\n";
    cout << "\2.Fly
Dubai(2)\t14:00\t\t18:05\t\tRs.21300\tRefundable\n";
    cout <<
"\3.Emirates(3)\t18:00\t\t22:05\t\tRs.24650\t\tRefundable\n";
}
else if((srci==1 && desti==5) || (srci==5 || desti==1))
{
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
    cout <<
"\1.Vistara(1)\t10:00\t\t14:05\t\tRs.52500\tRefundable\n";
    cout << "\2.Fly
Dubai(2)\t14:00\t\t18:05\t\tRs.59420\tRefundable\n";
    cout <<
"\3.Emirates(3)\t18:00\t\t22:05\t\tRs.64892\tRefundable\n";
}
else if((srci==2 && desti==3) || (srci==3 && desti==2))
{
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
    cout <<
"\1.Vistara(1)\t10:00\t\t14:05\t\tRs.17800\tRefundable\n";
    cout << "\2.Fly
Dubai(2)\t14:00\t\t18:05\t\tRs.14900\tRefundable\n";
    cout <<
"\3.Emirates(3)\t18:00\t\t22:05\t\tRs.18700\tRefundable\n";
}
else if((srci==2 && desti==4) || (srci==4 && desti==2))
{
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
    cout <<
"\1.Vistara(1)\t10:00\t\t14:05\t\tRs.32000\tRefundable\n";

```

```

    cout << "\2.Fly
Dubai(2)\t14:00\t\t18:05\t\tRs.38500\tRefundable\n";
cout <<
"\3.Emirates(3)\t18:00\t\t22:05\t\tRs41259\tRefundable\n";
}
else if(srci==2 && desti==5 || (srci==5 && desti==2))
{
    cout << "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\n";
cout <<
"\1.Vistara(1)\t10:00\t\t14:05\t\tRs.82500\tRefundable\n";
    cout << "\2.Fly
Dubai(2)\t14:00\t\t18:05\t\tRs.87550\tRefundable\n";
cout <<
"\3.Emirates(3)\t18:00\t\t22:05\t\tRs81478\tRefundable\n";
}
else if(srci==desti)
{
    cout << "wrong input entered.\nTry again\n\n\n" << endl;
    return j_detaili();
}
else
{
    cout << "Wrong input entered.\nTry again\n\n\n";
    return j_detaili();
}
}

int select_flighti()//function declaration and definition for selecting
flight
{
    cout << "\nEnter your choice\n>>> " << endl;
    cin >> choicei;
    switch(choicei)//switch case
    {
        case 1:
            cout << "\nFlight selected:" << endl;
            cout << "Vistara" << endl;
            strcpy(f_i,"Vistara");//copy to string
            cout << "Departure Time: 10:00" << endl;
            cout << "Arrival Time: 14:05" << endl;
            strcpy(tojdi,"10:00");//copy to string
            strcpy(tojai,"14:05");//copy to string
            break;
        case 2:
            cout << "\nFlight selected:" << endl;
            cout << "Fly Dubai" << endl;
            strcpy(f_i,"Fly Dubai");//copy to string
            cout << "Departure Time: 14:00" << endl;

```

```

cout << "Arrival Time: 18:05" << endl;
strcpy(tojdi,"14:00");//copy to string
strcpy(tojai,"18:05");//copy to string
break;
case 3:
cout << "\nFlight selected:" << endl;
cout << "Emirates" << endl;
strcpy(f_i,"Emirates");//copy to string
cout << "Departure Time : 18:00" << endl;
cout << "Arrival Time: 22:05" << endl;
strcpy(tojdi,"18:00");//copy to string
strcpy(tojai,"22:05");//copy to string
break;
default:
cout << "Wrong input entered" << endl;
return select_flighti();
}
}
};
//=====
//=====
class passenger: public d_booking,public i_booking//class passenger publicly
inherited from class d_booking and i_booking
{
protected:
char f_name[20],l_name[20],email[50];
int age,gender;
long int c_no;
public:
void p_detail(int x)
{ if(x==1)
{
j_detail();
select_flight();
}
else
{
j_detaili();
select_flighti();
}
cout << "\n===== " << endl
;
cout << "\n\nEnter passenger details\n";
cout << "\nFirst Name: ";
cin >> f_name;
cout << "Last Name: ";
cin >> l_name;

```

```

}
int gender_check()//to check gender input as valid
{
cout << "\n===== " << endl ;
cout << "\nGender:\n.Male (1)\n.Female (2)\n>>> ";
cin >> gender;
if(gender>2)
{
cout << "\n\nWrong input entered.\nTry again\n\n" << endl;
return gender_check();
}
}
void more_details()//to take more details of the passenger
{
cout << "\n===== " << endl ;
cout << "Age: ";
cin >> age;
cout << "Email Id: ";
cin >> email;
cout << "Contact no.: ";
cin >> c_no;
cout << "\n===== " << endl ;
cout << "\n===== " << endl ;
cout << "\n\nDetails Entered:\n";
cout << "Name:" << f_name << " " << l_name << endl;
cout << "Gender:" << gender << endl;
cout << "Age:" << age << endl;
cout << "Email id:" << email << endl;
cout << "Contact No.:" << c_no << endl;
cout << "\n===== " << endl ;
cout << "\n===== " << endl ;
}
int getpnr()//function to get pnr for domestic booking
{
return pnr;
}
int getpnri()//function to get pnr for international booking
{
return pnri;
}
void disp()//function to display details for domestic booking
{
cout << "\n===== " << endl ;
cout << "\n===== " << endl ;
cout<<"PNR:" << pnr << endl;
cout<<"Flight:" << f_d << endl;
cout<<"Name:" << f_name << " " << l_name << endl;

```

```

cout<<"DOJ:" << doj << endl;
cout<<"Departure Time:" << tojd << endl;
cout<<"Arrival Time:" << toja;
cout << "\n=====" << endl ;
cout << "\n=====" << endl ;
}
void dispi()//function to display details for international booking
{
cout << "\n=====" << endl ;
cout << "\n=====" << endl ;
cout<<"PNR:" << pnri << endl;
cout<<"Flight:" << f_i << endl;
cout<<"Name:" << f_name << " " << l_name << endl;
cout<<"DOJ:" << doji << endl;
cout<<"Departure Time:" << tojdi << endl;
cout<<"Arrival Time:" << tojai;
cout << "\n=====" << endl ;
cout << "\n=====" << endl ;
}
};
//=====
//=====
//=====
//=====
class payment//class for payment
{
protected:
long
int choice1,bank,card,date,card,cv,cv,user_id;
char password[10];
public:
void pay_detail()//function declaration and definition for payment method
{
cout << "\n=====" << endl
;
cout << "\n\n\nHow would you like to pay?:\n";
cout << "\n\n1.Debit Card(1) \n\n2.Credit Card(2) \n\n3.Net Banking(3)";
cout << "\n\nEnter your choice";
cin >> choice1;
switch(choice1)
{
case 1:
cout << "\nEnter card no.:";
cin >> card;
cout << "\nEnter expiry date:";
cin >> date;
cout << "\nEnter CVV no.:";

```

```

cin >> cvv;
cout << "\nTransaction Successful\n";
break;
case 2:
cout << "\nEnter card no.:";
cin >> card;
cout << "\nEnter expiry date:";
cin >> date;
cout << "\nEnter password:";
cin >> password;
cout << "\nTransaction Successful\n";
break;
case 3:
cout << "Banks Available: \1.HDFC(1) \2.Axis Bank(2) \3.Standard
Chartered Bank(3) \4.ICICI(4) \5.Others(5)";
cout << "\nSelect your bank:";
cin >> bank;
cout << "\nYou have selected:" << bank;
cout << "\nEnter user id:";
cin >> user_id;
cout << "\nEnter password:";
cin >> password;
cout << "\nTransaction Successful\n";
break;
default:
cout << "\nWrong input entered.\nTry again\n\n";
return pay_detail();
}
}
};
void createfile(passenger p)//file creation for domestic booking
{
ofstream fin("domestic.txt",ios::binary|ios::app);
fin.write((char*)&p,sizeof(p));
fin.close();
}
void cancelticket(int x)//function to cancel ticket
{
passenger p;
int f=0;
ifstream fout("domestic.txt",ios::binary|ios::app);
ofstream fin("domestic1.txt",ios::binary|ios::app);
fout.read((char *)&p,sizeof(p));
while(fout)
{
if(p.getpnr()!=x)
fin.write((char *)&p,sizeof(p));
}
}

```

```

else
{
p.disp();
cout<<"\nYour Above ticket is being canceled:\n" << "Amount
refunded: Rs 1000\n";
f++; //incrementing f if pnr found
}
fout.read((char *)&p,sizeof(p)); //reading another record from file
}
if(f==0) //if f==0, pnr not found
cout<<"Ticket not found\n";
fout.close();
fin.close();
remove("domestic.txt");
rename("domestic1.txt","domestic.txt");
}
void checkticket(int x) //function to check pnr or ticket
{
passenger p;
int f=0;
ifstream fout("domestic.txt",ios::binary);
fout.read((char *)&p,sizeof(p));
while(fout)
{
if(p.getpnr()==x)
{
p.disp();
cout<<"\nYour ticket"<<endl;
f++; //incrementing f if onr found
break;
}
fout.read((char *)&p,sizeof(p));
}
fout.close();
if(f==0) //if f==0, pnr not found
cout<<"Ticket not found"<<endl;
}
void createfilei(passenger p) //opening a file for international booking
{
ofstream fin("international.txt",ios::binary|ios::app);
fin.write((char*)&p,sizeof(p));
fin.close();
}
void cancelticketi(int x) //function to cancel ticket
{
passenger p;
int f=0;

```

```

ifstream fout("international.txt",ios::binary|ios::app);
ofstream fin("international1.txt",ios::binary|ios::app);
fout.read((char *)&p,sizeof(p));
while(fout)
{
    if(p.getpnri()!=x)
    fin.write((char *)&p,sizeof(p));
    else
    {
        p.dispi();
        cout<<"Your Above ticket is being deleted:\n"<<"Amount refunded:
Rs 1000\n";
        f++;//incrementing f if pnr found
    }
    fout.read((char *)&p,sizeof(p));
}
if(f==0)//if f==0,pnr not found
cout<<"\nTicket not found\n";
fout.close();
fin.close();
remove("international.txt");
rename("international1.txt","international.txt");
}
void checkticketi(int x)//function to check pnr or ticket
{ passenger p;
  int f=0;
  ifstream fout("international.txt",ios::binary);//opening file
  fout.read((char *)&p,sizeof(p));//reading file
  while(fout)
  {
      if(p.getpnri()==x)//checking pnr
      {p.dispi();//display details
      cout<<"\nYour ticket"<<endl;
      f++;//incrementing f if pnr found
      break;
      }
      fout.read((char *)&p,sizeof(p));//reading another record from the file
  }
  fout.close();//closing file
  if(f==0)//if f==0, pnr not found
  cout<<"Ticket not found"<<endl;
}
//=====
//=====
//=====
//=====
int main()

```



```

{
class d_booking d1;
class i_booking i1;
class passenger p1;
class payment p2;
int ch,ch1,n;
char input;
do
{
//system("CLS");
cout << "\n\n \t\tWelcome To Airline Reservation System" << endl <<
endl;
cout << "\t <><><><><><><><><><><><><><><><><><><><><>\n";

cout << "\n\n\t\t\t1.Book Flight (1) \n\t\t\t2.Cancel Fight (2)
\n\t\t\t3.Check Ticket (3) \n\t\t\t4.Exit (4)" << endl;
cout << "\n\t\tPlease enter your choice\n\t\t>>> ";
cin >> ch;
switch(ch)
{
case 1:
//system("CLS");
cout << "\n\n1.Domestic Fights (1) \n2.International
Flights (2)" << endl;
cout << "\nPlease enter your option\n>>> ";
cin >> ch1;
switch(ch1)
{
case 1://for booking domestic ticket
p1.d_pnr();
p1.p_detail(1);
p1.gender_check();
p1.more_details();
p2.pay_detail();
p1.disp();
createfile(p1);//call to create file
break;
case 2: //for booking international ticket
p1.p_detail(2);
p1.i_pnr();
p1.gender_check();
p1.more_details();
p2.pay_detail();
p1.dispi();
createfilei(p1);//call to create file
break;
default://wrong input

```

```
    cout << "Wrong input entered\nTry again\n\n\n" <<
endl;
    return main();
}
break;
case 2:
    //for canceling ticket
    //system("CLS");
    cout << "\1.Domestic Fights(1) \n\2.International Flights(2)"
<< endl;
    cout << "\nPlea
```

Input Output

```

Welcome To Airline Reservation System

<><><><><><><><><><><><><><><>

        .Book Flight (1)
        .Cancel Fight (2)
        .Check Ticket (3)
        .Exit (4)

Please enter your choice
>>> 3
.Domestic Flights(1)
.International Flights(2)

Please enter your option
1
Please enter your PNR no.:
1
=====
=====
PNR:1
Flight:Spicejet
Name:Johan Joseph
DOJ:211122
Departure Time:14:00
Arrival Time:17:05
=====
=====

Your ticket

Do you wish to continue:(y/Y)
n
```

Conclusion

The Airline Management system is a c++ project which will reduce the scope of manual error and conveniently maintain any modifications, cancellations in the reservations. In airline reservation system, the streamlining of the process of reservation without human interaction is highly needed so as to perform well in the highly competitive market. It not only provides flight details but also but also creates a platform to book tickets, cancels or modifies ticket timings or dates .The system in this place lacks a login feature. Users may simply check tickets, cancel reservations, and book flights. For the reservations, it has many payment alternatives. The features in this project are fewer, but they are vital. This project does not contain all the features but more features such as connecting flight routes, layover, document checking such as ID proofs can be added. Inform about the number of people on board, availability of tickets can be added to make this more efficient and user friendly .

References

1. https://moqups.com/uml-diagram-tool/?gspk=a3Jpc2huYXJ1bmd0YQ&gsxid=vm3DXxYISoVz&partnerstack_group=GoldPartner
2. <https://staruml.io/>
3. <https://creately.com/lp/uml-diagram-tool/?gspk=a3Jpc2huYXJ1bmd0YQ&gsxid=mFoPjxWa6Khb>