Indira Gandhi Delhi Technical University for Women

SOFTWARE ENGINEERING

PROJECT FILE [2022]
ON
AIRLINE RESERVATION SYSTEM

Submitted to:

Dr. NISHA RATHI

Submitted by:

NIDHI MITRA	**03304092021**
DIMPLE KUMARI	**07304092021**
SURBHI YADAV	**08004092021**

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our Software Engineering professor, **Mrs. Nisha Rathee Ma'am**, for her invaluable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism without which this project would not have been able to sail through. We deeply feel honored and blessed for getting this opportunity of being guided by her. We express our sincere thanks to her for being there constantly for help and supporting, encouraging and guiding us to successfully complete and present this project on the topic " **ONLINE AIRLINE RESEVATION SYSTEM ."**

Name of students:

Nidhi Mitra (03304092021)

Dimple Kumari (07304092021)

Surbhi Yadav (08004092021)

CERTIFICATE

This is to certify that Nidhi Mitra (03304092021), Dimple Kumari (07304092021), Surbhi Yadav(08004092021) have successfully carried out the completion of the project entitled “ONLINE AIRLINE RESERVATION SYSTEM” under my supervision. The project has been submitted as per the requirement of Lab based on Software Engineering in the second semester of MCA.

Project_in_charge:

Mrs. Nisha Rathee

TABLE OF CONTENT

1. PROBLEM STATEMENT

> The purpose of this document is to build an online system to manage flights and passengers to ease the flight management.

> Users of the system should be able to retrieve flight information between two given cities with the given date/time of travel from the database. The system will support two types of user privileges, Customer, and Employee.

> Customers will have access to customer functions, and the employees will have access to both customer and flight management functions. The customer should be able to make a flight reservation for either one-way trip or a round trip on a particular date. He will be asked for the confirmation before making his reservation. Customer will also be able to cancel the existing reservation and can also view the details of already reserved flights.

> On the other hand, Employee can access details of all the customers as well as of the flights, view flight schedule and calculate sales for a given flight. The administrator access will allow an employee to add, delete and update the information. He will be able to add/delete a flight, update fare for flights and update arrival/departure time of flights. Each flight has a limited number of available seats. There are a number of flights which depart from or arrive at different cities on different dates and time.

PROCESS MODEL:

For the development process of this system, we have chosen the *Waterfall model.* We decided to go with this model because:

> Requirements are very well documented, clear and fixed at the start of the software making.

> There are no ambiguous requirements.

> Technology used in making this software is not new and well understood by the developers.

> Since it is not a real project to be used in the market, we can use Waterfall mode

2. SRS DOCUMENT

1. INTRODUCTION

1.1 PURPOSE

The goal of this article is to create an online platform for handling flights as well as travellers in order to make flight management easie r.

1.2 DOCUMENT CONVENTIONS

The mentioned conventions are used in this document:

- ARS: Airline Reservation System
- ERD: Entity Relationship Diagram
- DBMS: Database Management System

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This whole project is a model for a flight management system that is confined to the university premises. This is expected to be beneficial for both the flight managing crew as well as the passengers.

1.4 PROJECT SCOPE

The goal of said online ARS is to make flight management much easier and also to build a comfortable and user-friendly interface for customers attempting to purchase air tickets. Including its flight management and booking functionalities, the platform is constructed on a relational database. We will have a database system that would accommodate hundreds of major locations worldwide along with thousands of flights from numerous airlines. And thus, we want to ensure a delightful customer experience as well as an efficient and fast system for the management.

1.5 REFERENCES

Project Report: Contains the different views of the system using Sequence Diagrams, Activity Diagrams etc. and the details of the requirement elicitation for the development.

1.6 OVERVIEW

Section 2 of this document describes overview of system in terms of general characteristics of the system, information about possible users of the system, possible constraints on the system, operating environment of the system etc. The Section 3 describes in detail, the different interfaces and their requirements. The Section 4 describes the system features and their relationships in brief. Finally, the Section 5 discusses about the non-functional requirements like safety, performance, etc requirements.

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The following records will be maintained within a distributed airline DBMS:

- **Flight Details**
  This comprises the source and destination flight terminals, the schedule, total number of seats reserved or vacant for each class, and so on.
- **Customer/Passenger Details**
  This comprises the information like customer id, name, address, phone number and so on which may be used to retain the customer's records in case of an emergency or for any other purpose.

- **Reservation Details**
  This contains the passenger information, a PNR numbers, flight numbers, the date of reservation, the date of travel, billing information, and so forth.
- **Employee Details**
  This contains the details of the flight management crew like name, ids, phone numbers, etc.

2.2 PRODUCT FEATURES

The ARS supports following functions:

- Functions by which customers can book or cancel an online ticket for a particular date, place, class and check the ticket status.
- Functions by which employees get the status of all customers and flights.
- Functions by which employee can add/delete a flight.
- Functions by which employee can update a flight's information/status like updating the departure/arrival time of a flight, etc.

The access to these different functions by different users is restricted.

2.3 USER CHARACTERISTICS

Users of the system will be permitted to get flight info connecting two specified locations on the specified date and time of travel. Customer and Employee user privileges will be supported by the platform. Customers will be able to use customer services, while staff will be able to access both customer and flight management capabilities

The Customer should be able to do the following functions:

- Make a new reservation based on availability
  - Flexible Date/time and other requirements
  - Payment and Confirmation
- Cancel an existing reservation
- View his itinerary

The Employee should have following management functionalities:

- Customer Functions:

- Get all the passengers and their details, who have seats reserved on a given flight.
- Get all flights and their details for a given airport and time.

- Administrative Functions:
  - Add and Delete a flight
  - Update fare for flights.
  - Update departure/arrival.

As a result, each trip will have a fixed number of available seats in various classes, and there are a number of flights that depart or arrive at various places on various days and times.

2.4 OPERATING ENVIRONMENT

Operating environment for the airline management system is as listed below.

Distributed database
Operating system: Windows. Database: SQL+
Platform: To be decided

2.5 CONSTRAINTS

User is required to remember the login ID and password for his account. If lost, user will not be able to access his account again. Customers will be allowed to check the ticket status and book ticket only through internet. Payment for the tickets can only be made through net banking.

2.6 ASSUMPTION AND DEPENDENCIES

As of now, there are no assumptions. This article will be updated in future editions.

3. EXTERNAL INTERFACE REQUIREMENTS

3.1 USER INTERFACES

The ARS project has two sorts of users. The first is the customer, while the second is the employee/administrator. Both Customer and Employee user interfaces would be graphical. The following software is being used for functionalities:

- Front-end software is not yet decided
- Back-end software we are using is SQL+

3.2 HARDWARE INTERFACES

The machines should be running Windows and also have a browser that supports CGI, HTML, and JavaScript. For interactivity, the computer must communicate with a conventional output device, a keyboard, and a mouse.

3.3 SOFTWARE INTERFACES

The programme should be compatible with the Windows operating system. SQL+ is

required because the programme requires a database to hold all of the client information, airline information, and reservation information.

The decision for the coding language to implement the coding of the software is yet to be made. It will be either C++/Python/PHP.

Accordingly, we will decide the IDE to work on.

4. SYSTEM FEATURES

4.1 FUNCTIONAL REQUIREMENTS

Function Name: Login

Description: This function describes how a user logs into the system.

Basic flow: User wishes to log in to the system and has to enter valid credentials necessary to log in to the system

Alternate flow: Credentials entered by the User do not follow the format prescribed for the credentials.

Function Name: Verify Password

Description: Credentials entered are verified against the database

Basic flow: Entered credentials match that of the ones present in the database User is logged in to the system in their respective account type.

Alternate flow: If the entered credentials are not present in the database, an error message is generated.

Function Name: Check Availability

Description: Customers can check for tickets based on their requirements

Basic flow: Customers can search for tickets based on the following parameters:

- Date and Time
- Location
- Class

Alternate flow: None

Function Name: Book Ticket and Payment

Description: Provides the main payment functionality that allows Customers to book and pay for the tickets.

Basic flow: Customer selects the available tickets and initiates the payment functionality after which verification takes place.

Alternate flow: If the payment is unable to be completed due to any reason, a transactional error message is generated.

Function Name: Cancellation

Description: Customers can initiate the Cancel functionality if they wish.

Basic flow: The Customer is given the option to confirm cancellation after which they are refunded and the system records are updated.

Alternate flow: None

Function Name: View Itinerary

Description: Customer can view their flight status, details, etc.

Basic flow: Customers can now view their flight itinerary which includes the departure and arrival airports, dates and times of the flights, flight numbers, etc.

Alternate flow: None

Function Name: View Customer/Flight Information

Description: Allows Employees to access customer and flight information from the database.

Basic flow: Employees can search for information using customer id, flight id, etc depending on their needs.

Alternate flow: None

Function Name: Add Flight

Description: Allows Employees to add new flights and their respective information

Basic flow: The employee can add a new flight along with their respective information like departure and arrival times, fares, seats, etc.

Alternate flow: None

Function Name: Update Flight Information

Description: Allows employees to update information of existing flights.

Basic flow: Employees can search for the flight and perform the following:

- Update Arrival/Departure
- Update Fares
- Delete Flight

Alternate flow: If a flight does not exist in the database, an error message is generated.

---

5. NON-FUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

Queries filed by users should be responded quickly by the ARS. When a user looks for a flight departing from one airport to another airport, the ARS should quickly return the results. As the ARS application is not that big, it should show six-seven results at a time on each page to the user, when he/she searches for some data. Time taken by the application to answer the requests of users should be less than two and a half seconds.

5.2 SAFETY REQUIREMENTS

If a large chunk of the database suffers significant harm due to accidental failure, for example a storage device crash, a recovery technique should be present that recovers a former copy of the database that was backed up to backed-up storage and recreates a more present state by reapplying or repeating the operations of committed transactions from the backed-up log, up until the moment of failure.

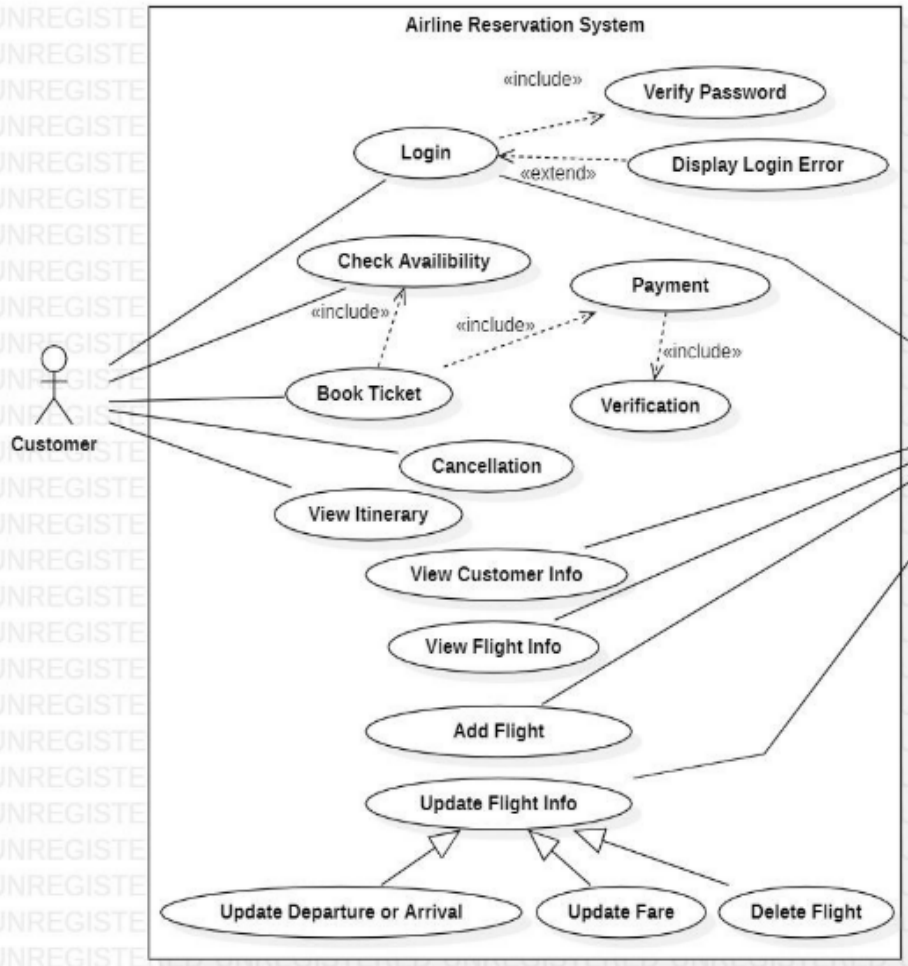
5.3 SECURITY REQUIREMENTS

Security systems, like so many other systems, require storage in the form of database. But, due to the unique needs of the security sector, suppliers must hand pick their database partner.

5.4 SOFTWARE QUALITY ATTRIBUTES

- **AVAILABILITY of system:** Users should be able to access application on any time of the day for booking or cancelling the tickets.
- **CORRECTNESS of system:** The flights available on application should depart from the correct terminal and also arrive at the proper destination.
- **MAINTAINABILITY of system:** Correct schedules of flights should be maintained by the administrators and flight-in chargers.
- **USABILITY of system:** Flight itineraries should be designed to meet the requirements of as many consumers as possible.

3. USE CASE DIAGRAM



USE CASE DESCRIPTION

Function Name: Login

Brief	This use case describes how a user logs into the system
Actors	Customer/Employee
Precondition	Actors should have already had an existing account of their respective type
Basic flow	Actor wishes to log in to the system and has to enter valid credentials necessary to log in to the system
Alternate flow	Credentials entered by the Actor do not follow the format prescribed for the credentials. Actor is returned to the beginning of the Use case.
Post condition	Verification of credentials

Function Name: Verify Password

Brief	Credentials entered are verified against the database
Actors	None
Precondition	Valid credentials should be present in the database
Basic flow	Entered credentials match that of the ones present in the database User is logged in to the system in their respective account type
Alternate flow	If the entered credentials are not present in the database, an error message is generated.
Post condition	Customer/Employee is allowed access to his account

Function Name: Check Availability

Brief	Customers can check for tickets based on their requirements
Actors	Customer
Precondition	The Customer must be logged in.
Basic flow	Customers can search for tickets based on the following parameters: • Date and Time • Location • Class
Alternate flow	None.
Post condition	Customers can now book the tickets if they want.

Function Name: Book Ticket and Payment

Brief	Provides the main payment functionality that allows Customers to book and pay for the tickets.
Actors	Customer
Precondition	The Customer must be logged in and must choose the appropriate available tickets based on their need.
Basic flow	Customer selects the available tickets and initiates the payment functionality after which verification takes place.
Alternate flow	If the payment is unable to be completed due to any reason, a transactional error message is generated.
Post condition	Customer receives the booking confirmation and can log out of the system.

Function Name: Cancellation

Brief	Customers can initiate the Cancel functionality if they wish
Actors	Customer
Precondition	Customer has already booked and paid for tickets.
Basic flow	The Customer is given the option to confirm cancellation after which they are refunded and the system records are updated.
Alternate flow	None
Post condition	User receives the confirmation and can log out of the system.

Function Name: View Itinerary

Brief	Customer can view their flight status, details, etc.
Actors	Customer
Precondition	Customer must be logged in and the payment for the tickets is completed.
Basic flow	Customers can now view their flight itinerary which includes the departure and arrival airports, dates and times of the flights, flight numbers, etc.
Alternate flow	None
Post condition	None

Function Name: View Customer/Flight Information

Brief	Allows Employees to access customer and flight information from the database.
Actors	Employee
Precondition	Employee must be logged in.
Basic flow	Employees can search for information using customer id, flight id, etc depending on their needs.
Alternate flow	None
Post condition	If the entity exists, the employee can now access it.

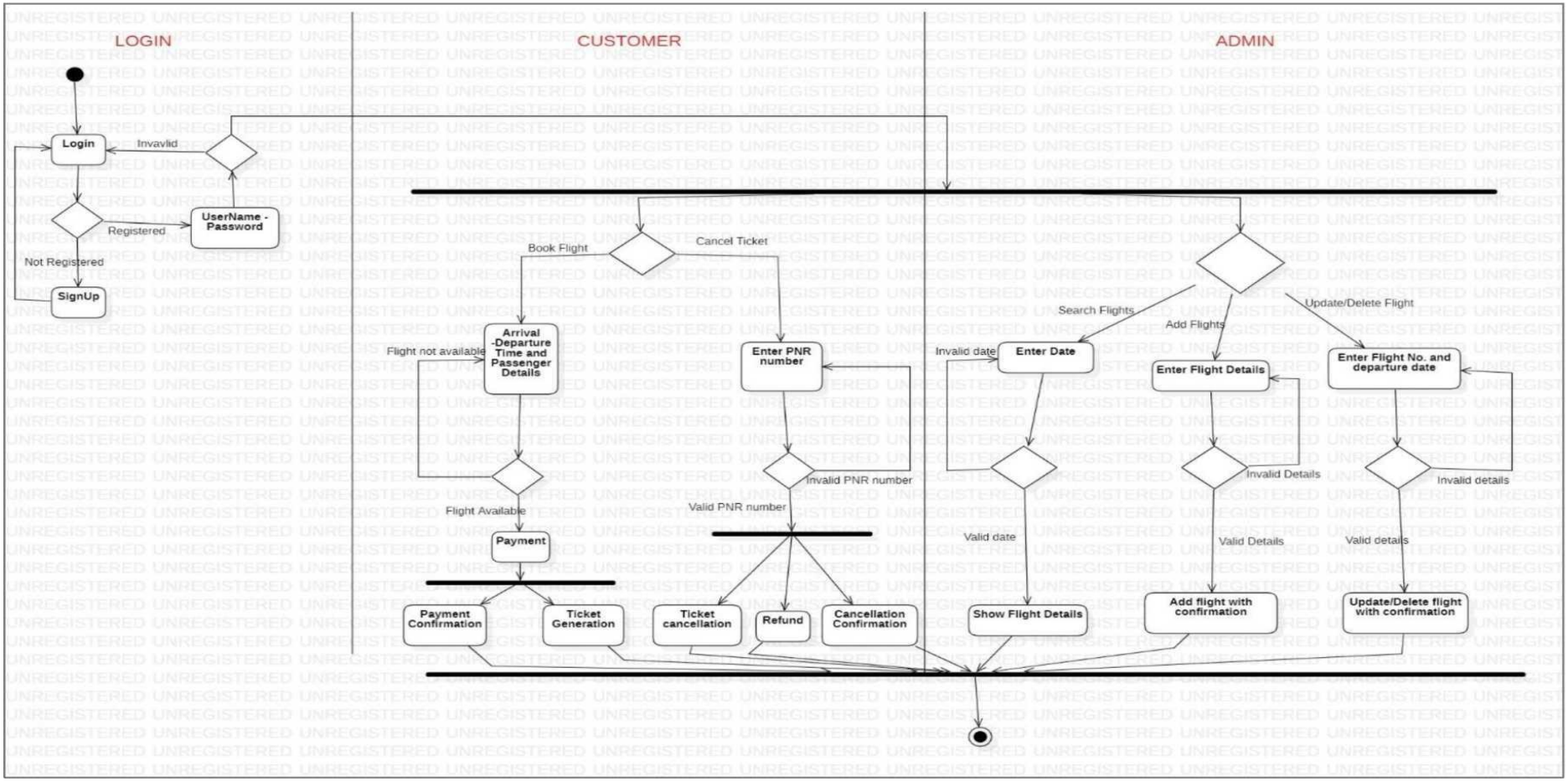
Function Name: Add Flight

Brief	Allows Employees to add new flights and their respective information.
Actors	Employee
Precondition	Employee must be logged in.
Basic flow	The employee can add a new flight along with their respective information like departure and arrival times, fares, seats, etc.
Alternate flow	None.
Post condition	Database is updated according to the changes.

Function Name: Update Flight Information

Brief	Allows employees to update information of existing flights.
Actors	Employee
Precondition	Employee must be logged in.
Basic flow	Employees can search for the flight and perform the following: • Update Arrival/Departure • Update Fares • Delete Flight
Alternate flow	If a flight does not exist in the database, an error message is generated.
Post condition	Database is updated according to the changes.

ACTIVITY DIAGRAMS

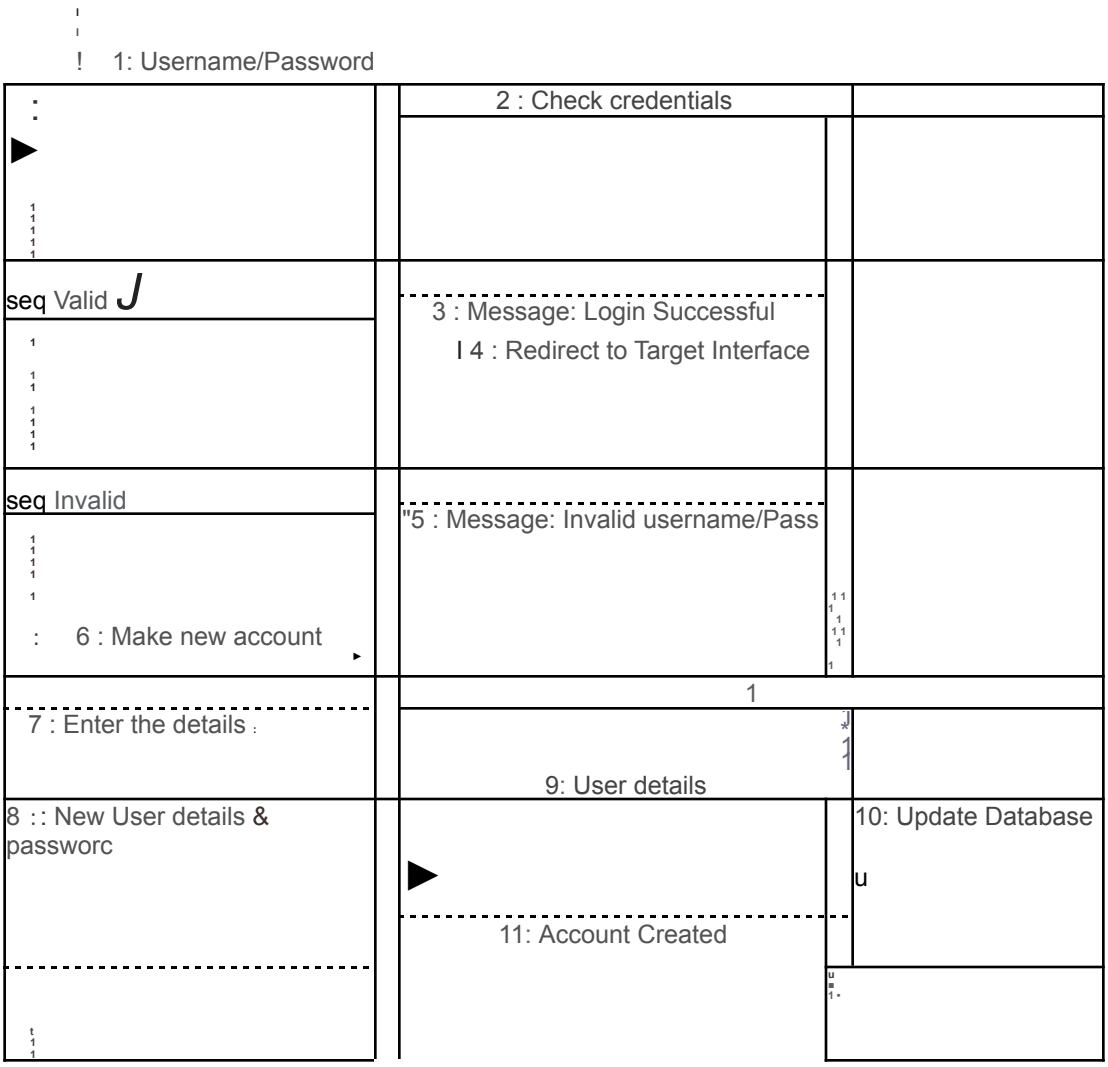


5. SEQUENCE DIAGRAMS

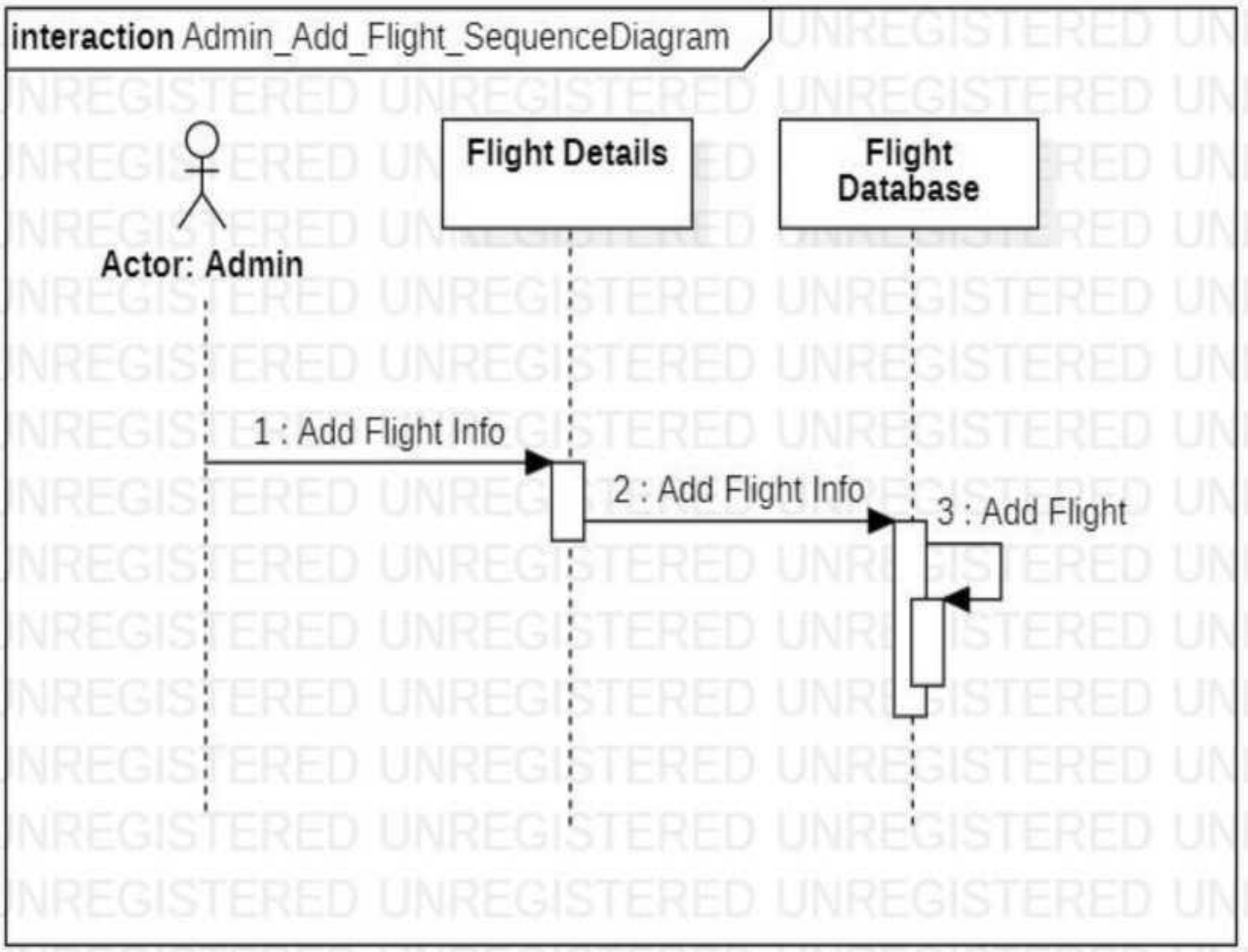
All the sequence diagrams of our project are given below:

1. Login (Login and Sign Up)

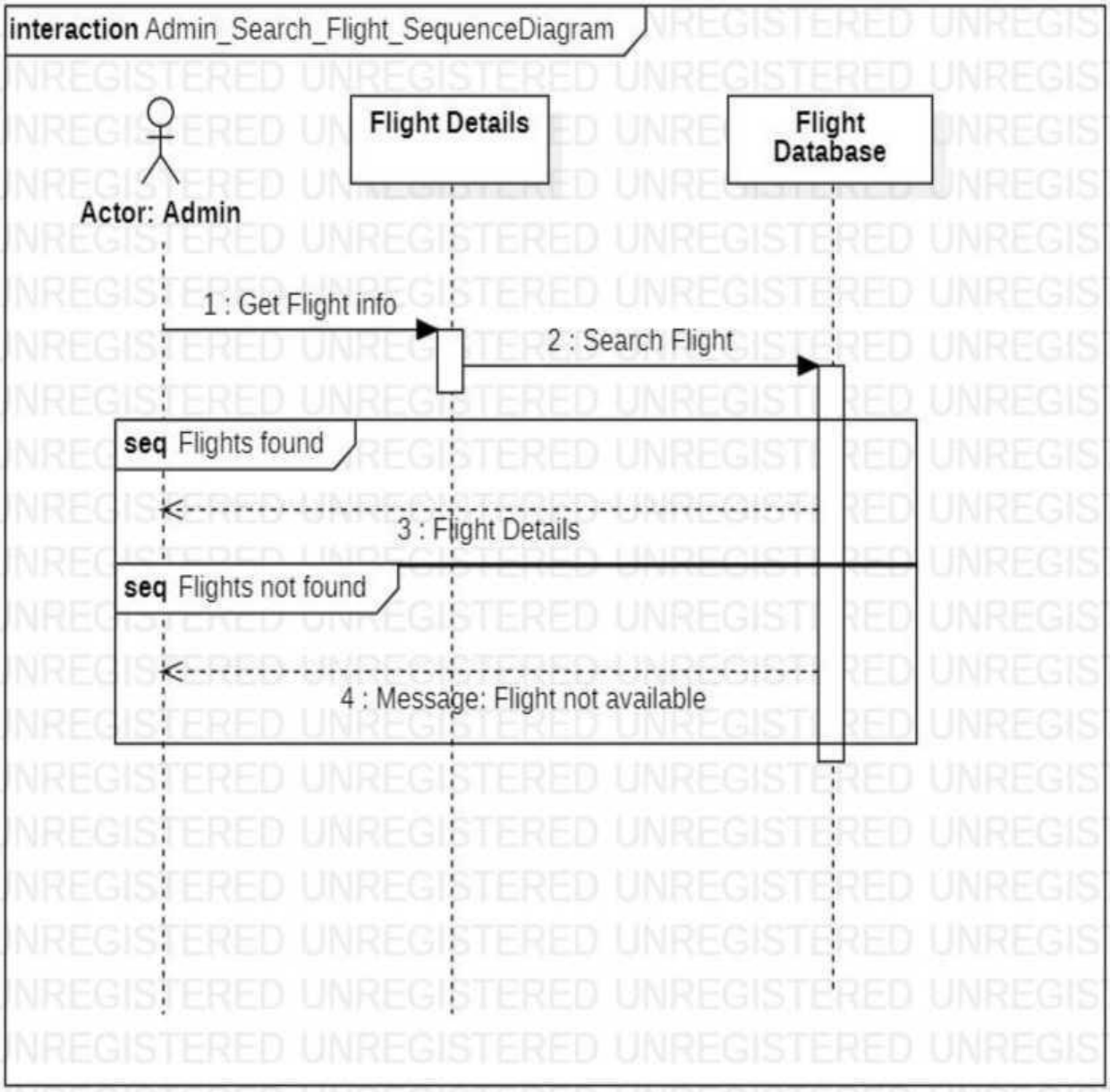
interaction Login_SequenceDiagram
Account Database
Login/SignUp
Actor: User



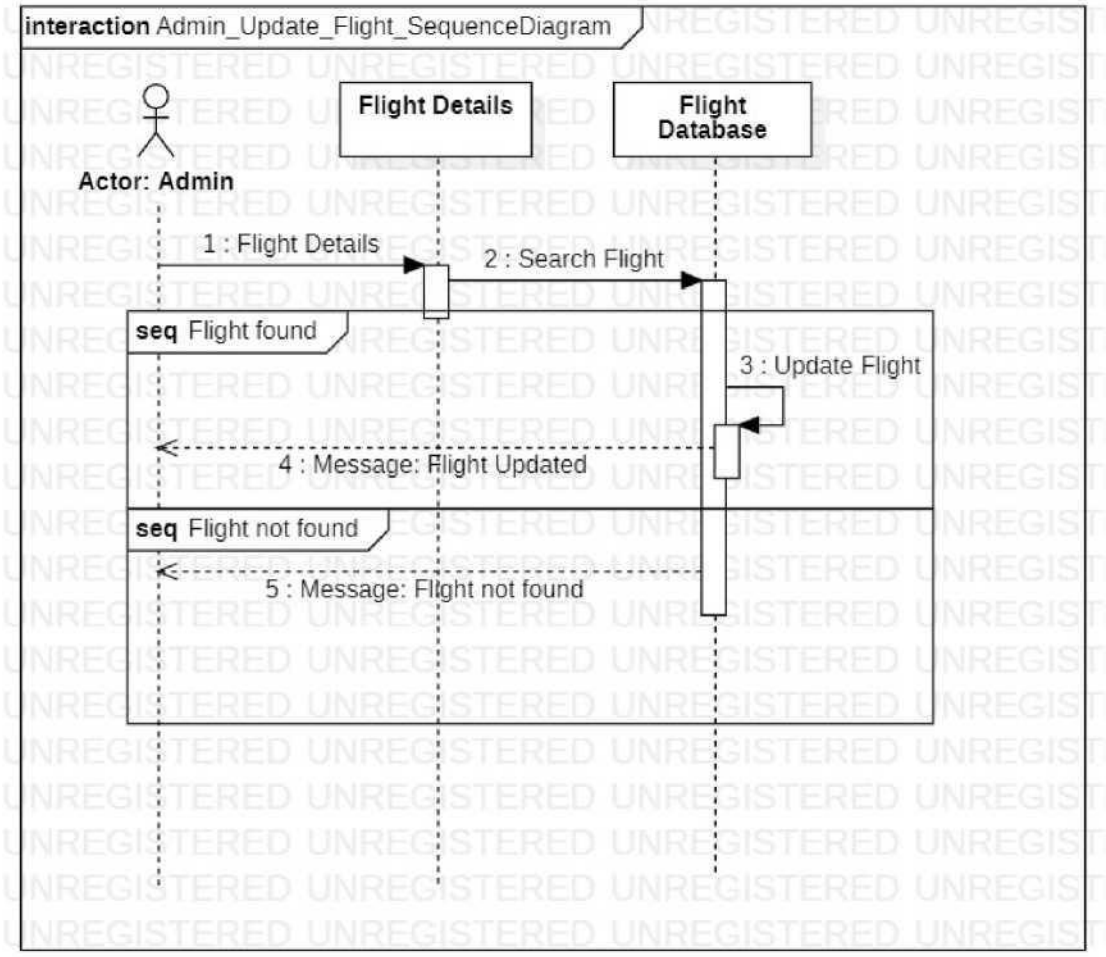
2. Admin's Add Flight Function



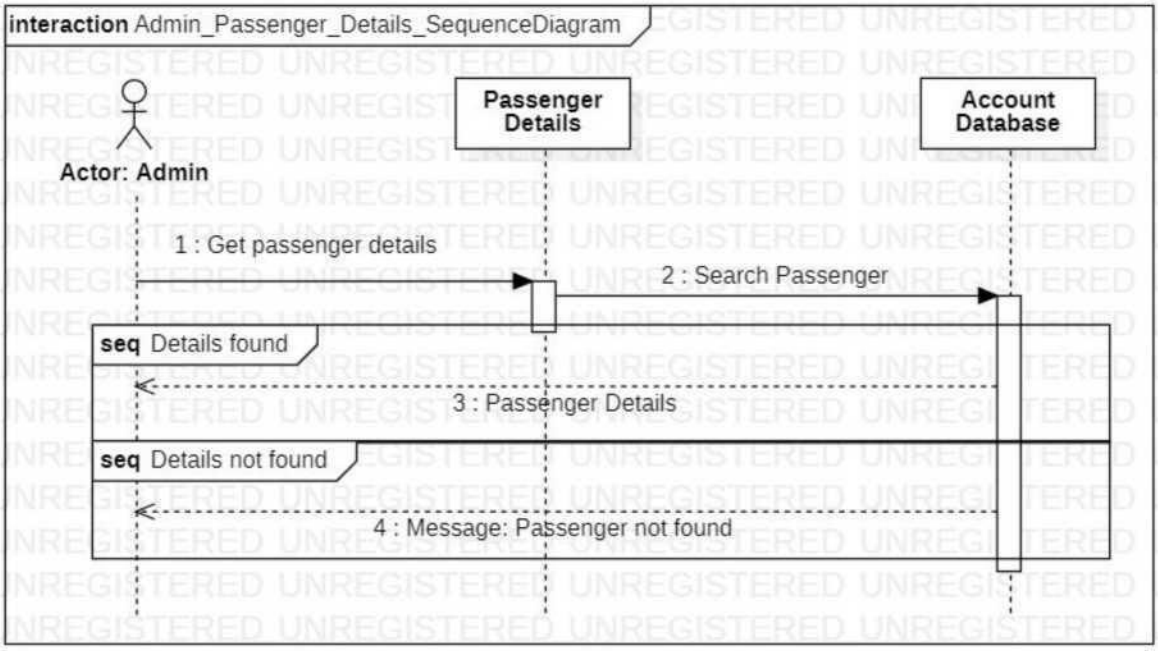
3. Admin's Search Flight Function


interaction Admin_Search_Flight_SequenceDiagram
Flight Details
Flight Database
Actor: Admin
1 : Get Flight info
2 : Search Flight
seq Flights found
3 : Flight Details
seq Flights not found
4 : Message: Flight not available

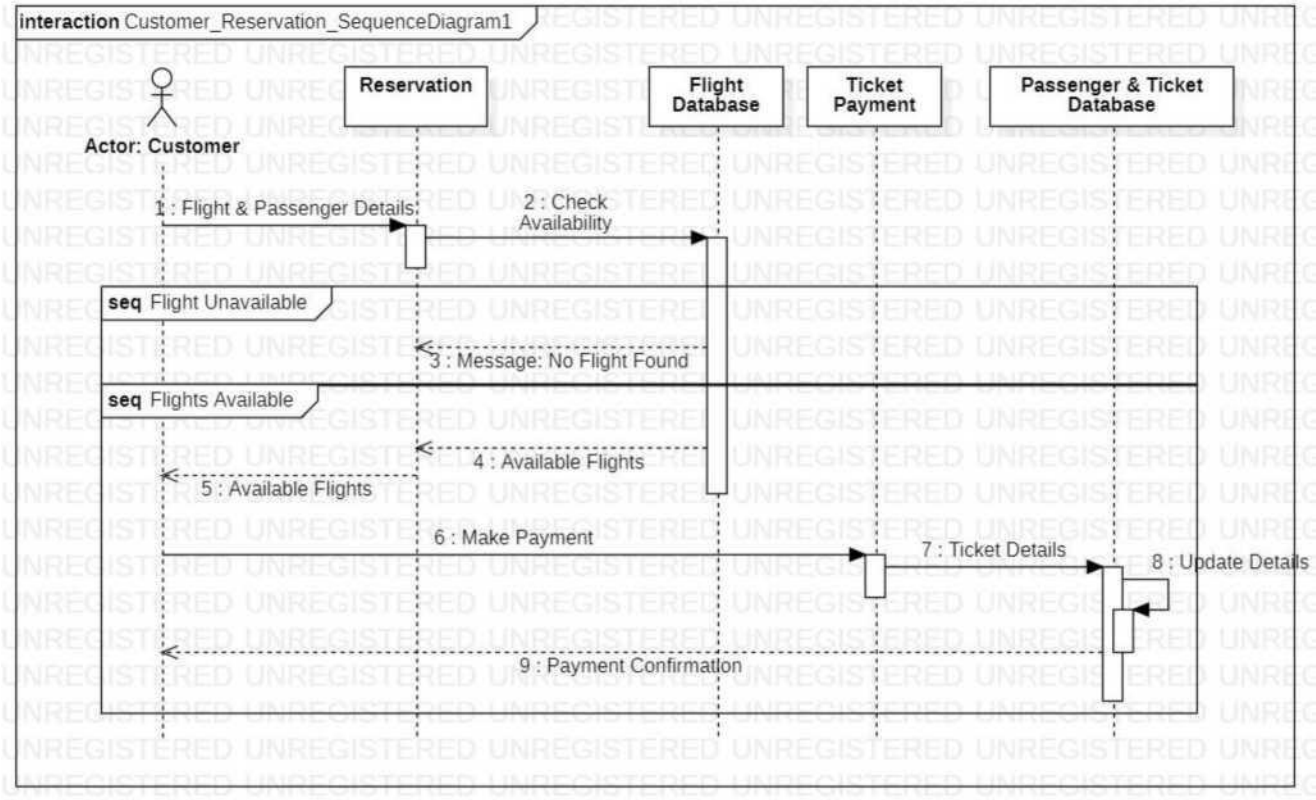
4. Admin's Update Flight Function



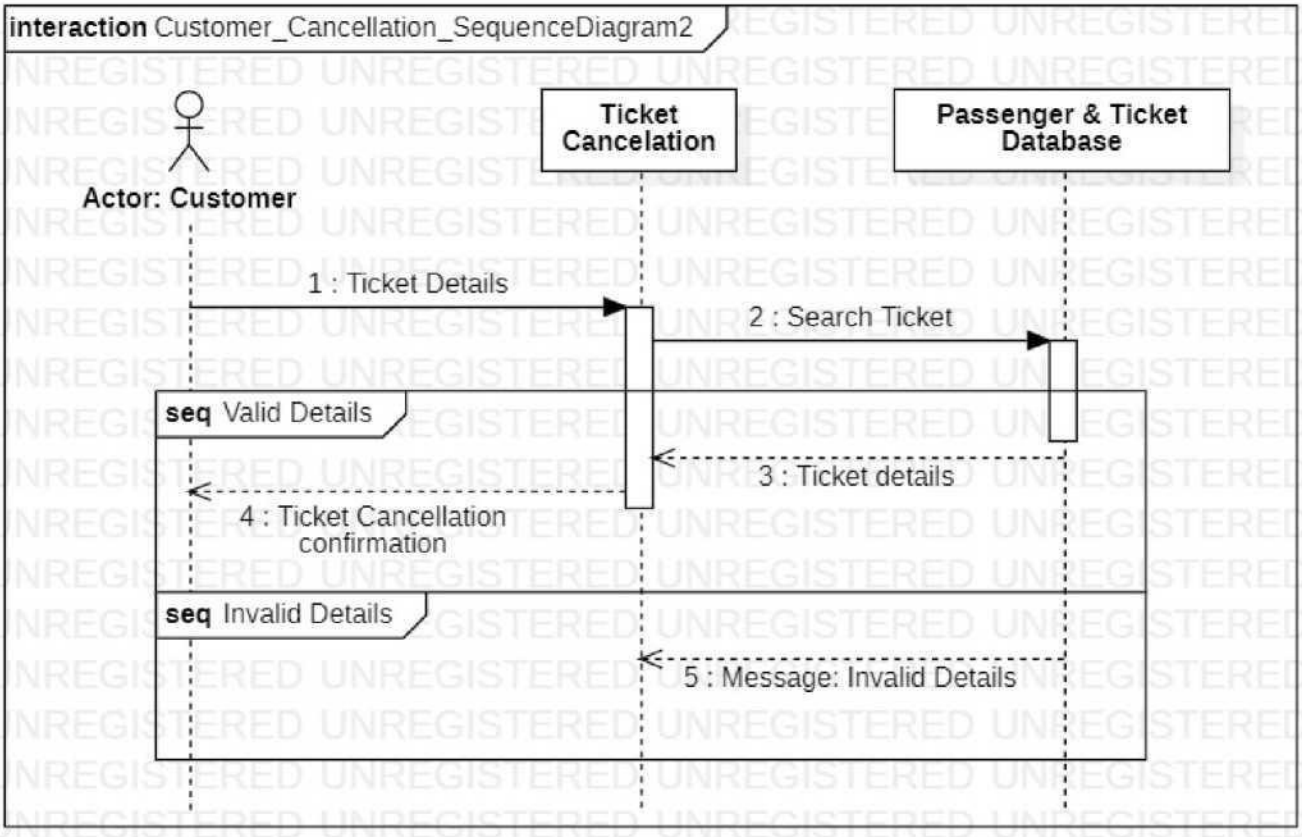
5. Admin's Get Passenger Details Function


interaction Admin_Passenger_Details_SequenceDiagram
Actor: Admin
Passenger Details
Account Database
1 : Get passenger details
2 : Search Passenger
seq Details found
3 : Passenger Details
seq Details not found
4 : Message: Passenger not found

6. Customer's Reservation Function

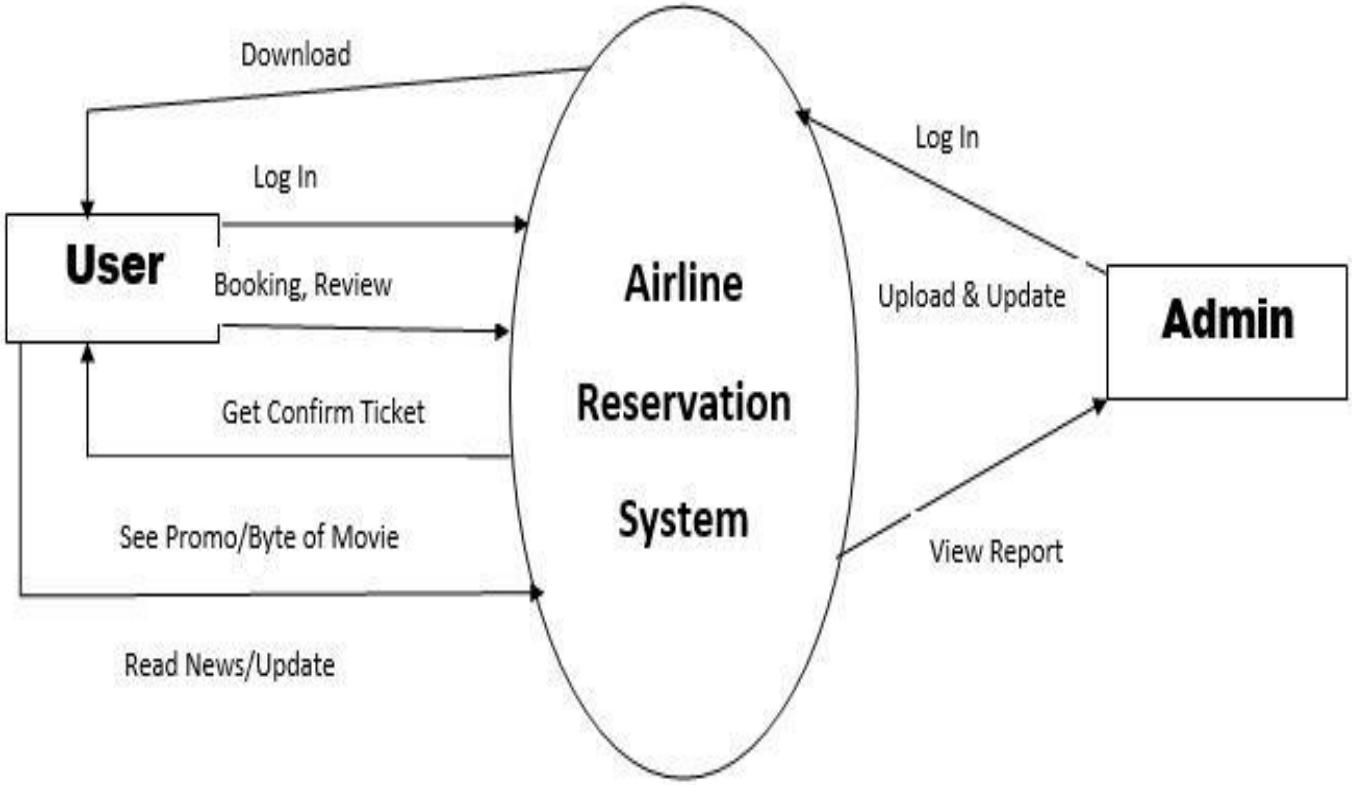


7. Customer's Ticket Cancellation Function

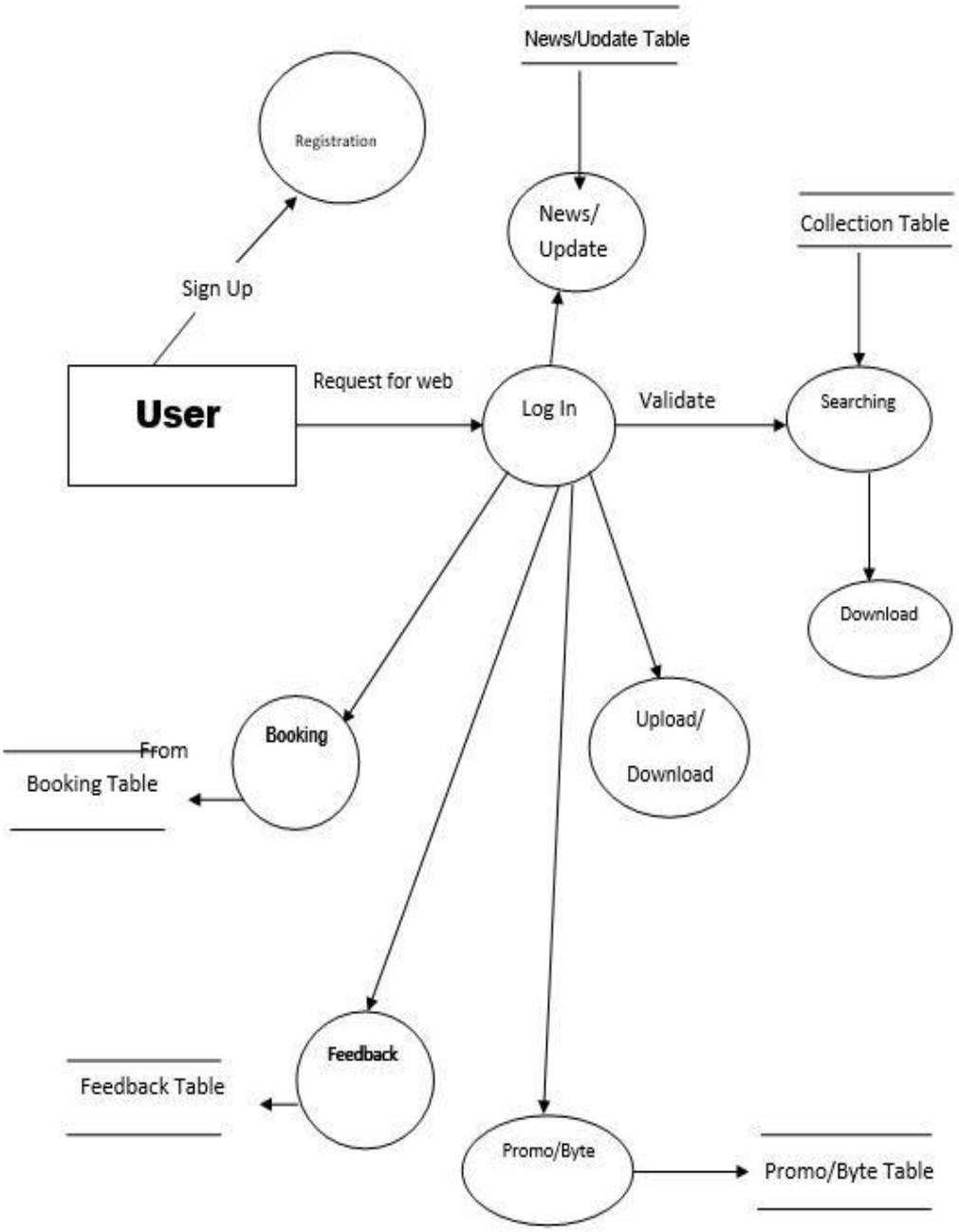

interaction Customer_Cancellation_SequenceDiagram2
Actor: Customer
Ticket Cancelation
Passenger & Ticket Database
1 : Ticket Details
2 : Search Ticket
seq Valid Details
3 : Ticket details
4 : Ticket Cancellation confirmation
seq Invalid Details
5 : Message: Invalid Details

6. DFD DIAGRAM

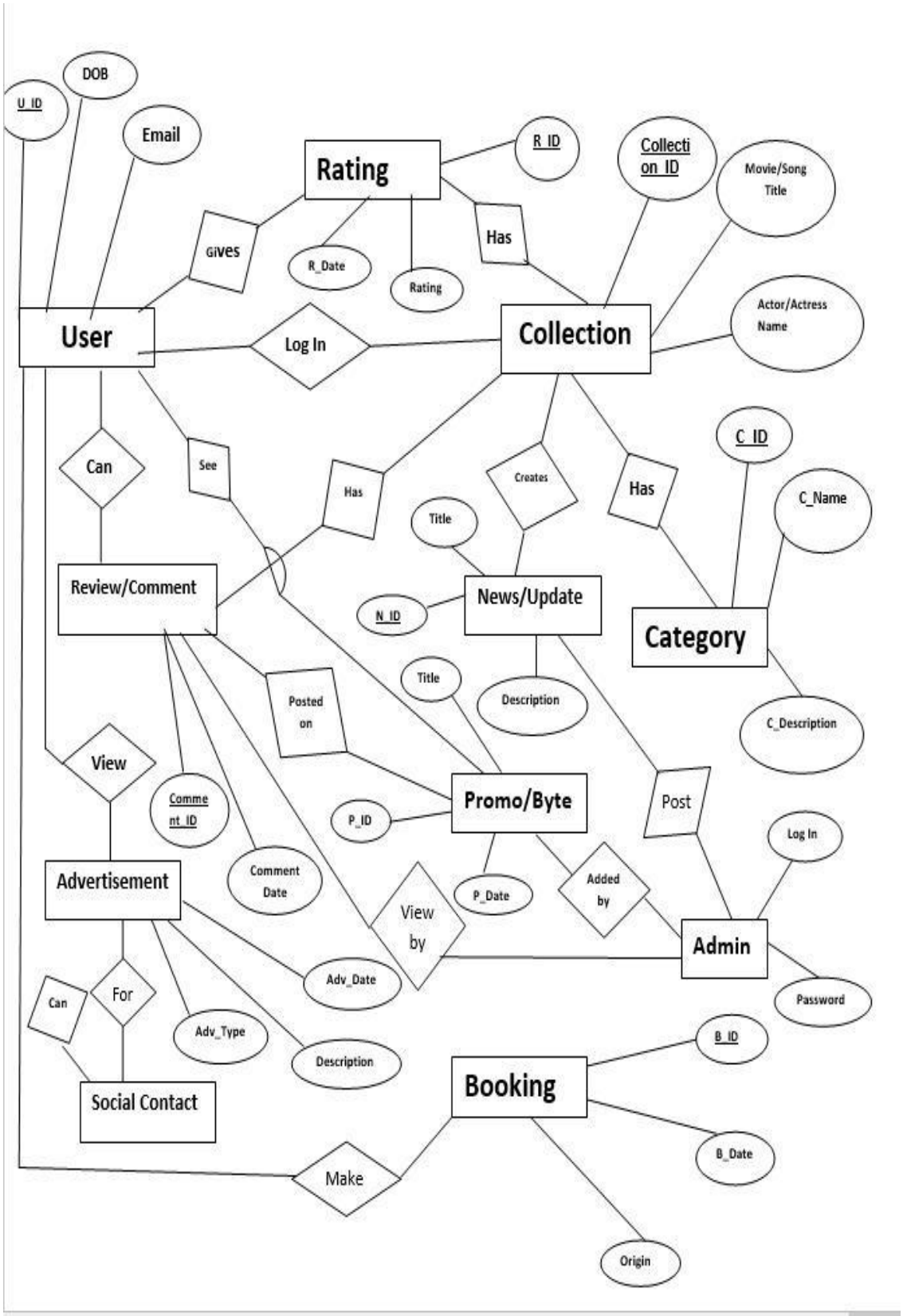
0 Level DFD:



User 1 Level DFD:



7. ER-DIAGRAM


DOB
U_ID
Email
Rating
R_ID
Collecti on_ID
Movie/Song Title
Gives
R_Date
Rating
Has
User
Log In
Collection
Actor/Actress Name
Can
See
Has
Creates
Has
C_ID
C_Name
Title
Review/Comment
N_ID
News/Update
Category
Title
Description
Posted on
C_Description
View
Comme nt_ID
P_ID
Promo/Byte
Post
Log In
Advertisement
Comment Date
P_Date
Added by
View by
Admin
Adv_Date
Can
For
Password
Adv_Type
Description
B_ID
Social Contact
Booking
B_Date
Make
Origin

8.TEST CASE

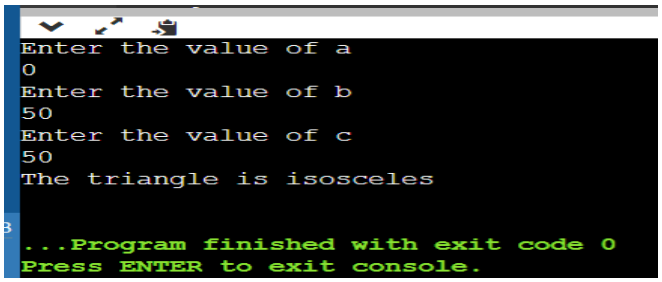
Ques: Consider a simple program to classify a triangle. Its inputs is a triple of positive integers (say a, b, c) and the date type for input parameters ensures that these will be integers greater than 0 and less than or equal to 100. The program output may be one of the following words:
[Scalene; Isosceles; Equilateral]
Code :

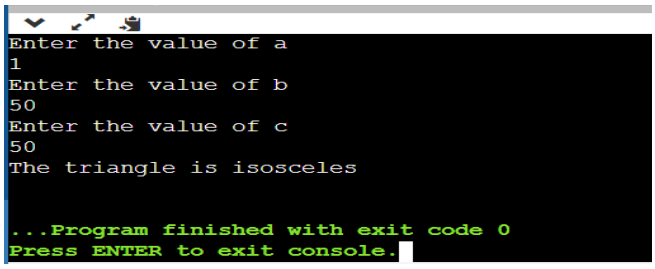
```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;
    cout<<"Enter the value of a \n";
    cin >> a;
    cout<<"Enter the value of b \n";
    cin >> b;
    cout<<"Enter the value of c \n";
    cin >> c;
    if(a<100 && b<100 && c<100 && a>0 && b>0 && c>0) {
        if(a==b && a==c){
        cout << "The triangle is equilateral \n";

        }
        else if(a!=b && b!=c && a!=c){
            cout << "The triangle is scalene \n";

        }
        else{
            cout << "The triangle is isosceles \n";
        }
    }
    else{
        cout << "Invalid input \n";
        return 0;
    }

    return 0;
}
```

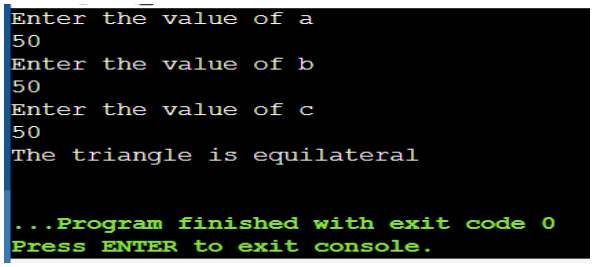
Boundary value analysis :
Total no of test case : 4n+1
n = 3
no of test cases : 13

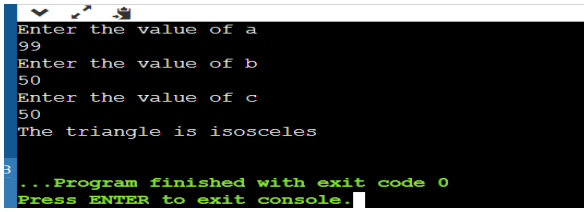
S.No	a	b	c	Actual output
1	0	50	50	Isosceles
2	1	50	50	Isosceles
3	50	50	50	Equilateral
4	99	50	50	Isosceles
5	100	50	50	Isosceles
6	50	0	50	Isosceles
7	50	1	50	Isosceles
8	50	99	50	Isosceles
9	50	100	50	Isosceles
10	50	50	0	Isosceles
11	50	50	1	Isosceles
12	50	50	99	Isosceles
13	50	50	100	Isosceles

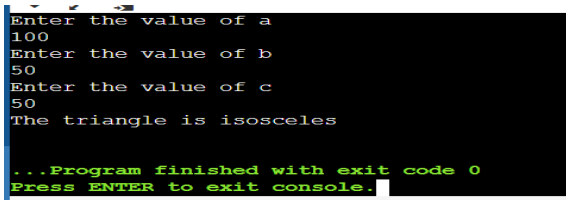
Output of the code :

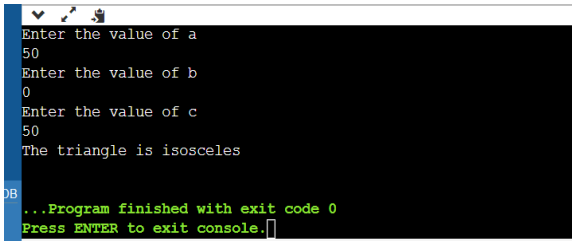


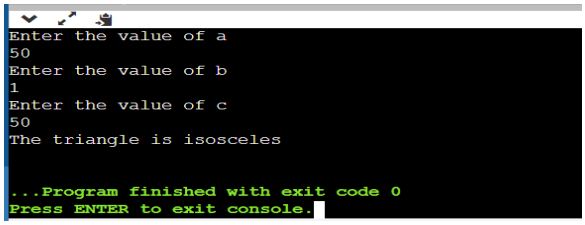


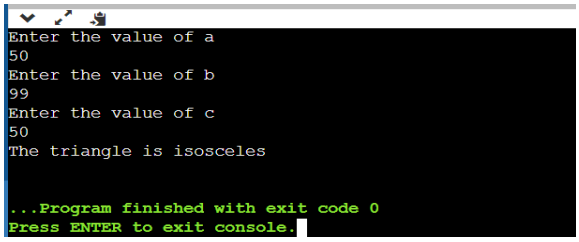


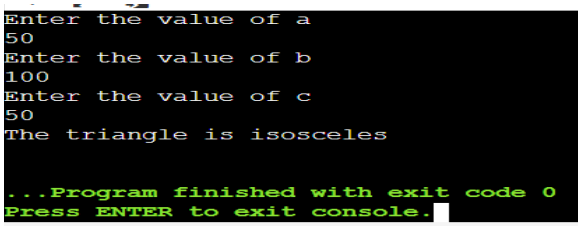

Enter the value of a
99
Enter the value of b
50
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.

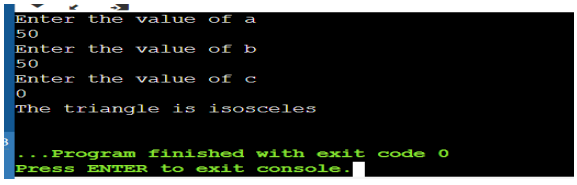

Enter the value of a
100
Enter the value of b
50
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.

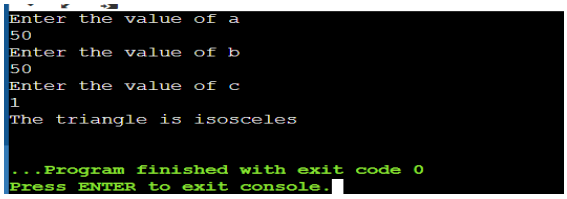

Enter the value of a
50
Enter the value of b
0
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.

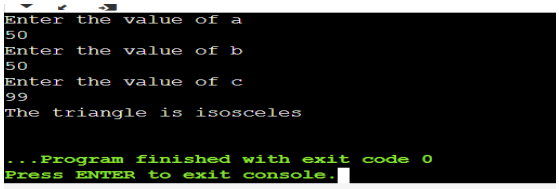

Enter the value of a
50
Enter the value of b
1
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.

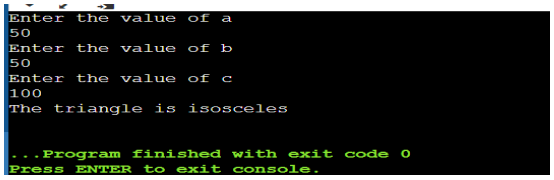

Enter the value of a
50
Enter the value of b
99
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
50
Enter the value of b
100
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
50
Enter the value of b
50
Enter the value of c
0
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.



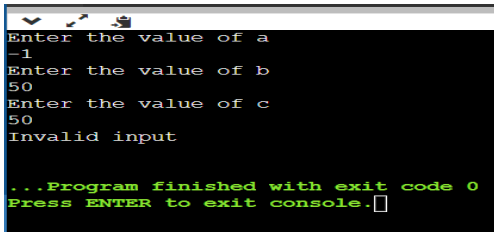


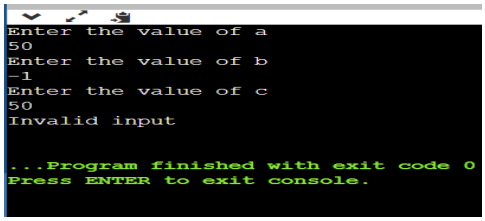


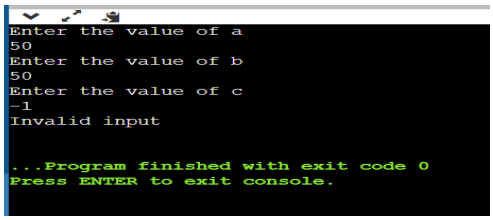
Equivalence class testing technique :
Ques : Consider a simple program to classify a triangle. Its inputs is a triple of positive integers (say a, b, c) and the date type for input parameters ensures that these will be integers greater than 0 and less than or equal to 100. The program output may be one of the following words:
[Scalene; Isosceles; Equilateral]

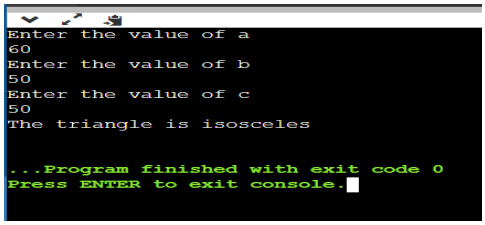
equivalence class test cases:
O1 : {0>a>100 : Invalid Input}
O2 : {0>b>100 : Invalid Input}
O3 : {0>c>100 : Invalid Input}
O4 : {0<a<100 : Valid Input}
O5 : {0<b<100 : Valid Input}
O6 : {0<c<100 : Valid Input}

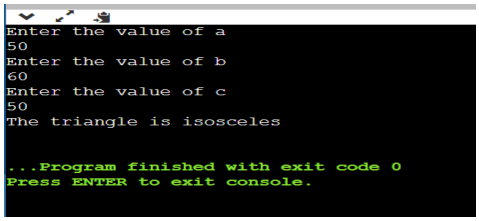
SNo	a	b	c	Actual output
1	-1	50	50	Invalid input
2	50	-1	50	Invalid input
3	50	50	-1	Invalid input
4	60	50	50	Isosceles
5	50	60	50	Isosceles
6	50	50	60	Isosceles

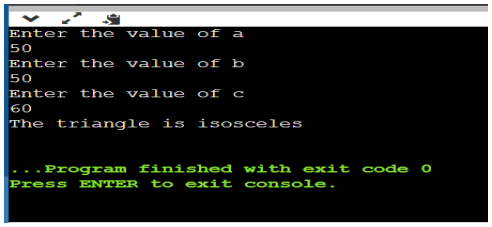

Enter the value of a
-1
Enter the value of b
50
Enter the value of c
50
Invalid input
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
50
Enter the value of b
-1
Enter the value of c
50
Invalid input
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
50
Enter the value of b
50
Enter the value of c
-1
Invalid input
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
60
Enter the value of b
50
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
50
Enter the value of b
60
Enter the value of c
50
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.


Enter the value of a
50
Enter the value of b
50
Enter the value of c
60
The triangle is isosceles
...Program finished with exit code 0
Press ENTER to exit console.