Software Engineering

Airline Ticketing System

Final Project Report

Supervised by: Dr. Youssry Taha

Eng.Hashim Hossam

Name	**ID**
Mayar Ahmed Ramadan	6055
Aisha Hatem Nashed	6372
Merna Mourad William	6377

Table of contents:

1- Purpose :

The airline ticketing system provides a user-friendly interface that is supported by a well-designed database which contains all available air flight information that is integrated together and can be accessed easily through the website so this system helps in speedy information retrieval.

The system is designed in such a way that the booking list is available on time before the scheduled departure time, Moreover the system supports cancellation so that another customer can book it thus exploiting the system efficiently in order to do that the booking list should reflect the latest status at any given time. The system enables customers to look at the airline tickets' cost and accessibilities.

System stakeholders:

• Customers

• Administrative

2- Requirements :

2.1 User Requirements :

2.1.1 Functional User Requirements:

1. The Airline Ticketing System should allow customers to log in if they already have an account.
2. The Airline Ticketing System should allow new customers to register an account in the system. If the registration is not correct or email already exists, alert and allow them to register again.
3. The Airline Ticketing System should provide authentication to differentiate between customers and admin.
4. Any user can view available flights from the home page without necessarily being registered or logged in.
5. Customers who log into the system have the permission to book flights.
6. Admins who log into the system have the permission to be able to add, edit, delete customers.
7. Admins who log into the system have the permission to be able to add,edit,delete flights.

8. The Airline Ticketing System supports searching option for customers with different categories
9. End all sessions when logged out of the system.

2.1.2 Non-Functional User Requirements:

1. The Airline Ticketing System should be free of errors
2. The Airline Ticketing System is open to changes, modifiable so that it reaches maintainability.
3. The Airline Ticketing System should not show passwords to enhance security.

2.2 System Requirements :

2.2.1 Functional System Requirements:

1. If a user tries logging into the system, check information from the database and compare with database existing data to check if it is valid or not.
2. The Airline Ticketing System has 2 pages one for login and other for registration.
3. The Airline Ticketing System customer is able to search for flights on a specific airline.
4. The Airline Ticketing System customer is able to view all flights and book the desired one.
5. The Airline Ticketing System customer is able to search with for flights with dates and determine one-way or 2 way
6. The Airline Ticketing System customer is able to view tickets and cancel any one.
7. The Airline Ticketing System admin is able to view all system reports.
8. The Airline Ticketing System admin is able to add, edit, delete customers.
9. The Airline Ticketing System admin is able to add, edit, delete flights.
10. The Airline Ticketing System admin is able to add, edit, delete airlines.
11. The Airline Ticketing System admin is able to add airport and update booked
12. The Airline Ticketing System admin is able to check income.

2.2.2 Non-Functional System Requirements:

1. Passwords must be secured
2. Easy to use system and to understand the web pages (user-friendly design)
3. The system shouldn't allow time lag
4. The system mustn't fail while deleting, updating or adding to the database.

3- Software Process :

3.1. Suggested Type of Software Process:

We used agile [incremental], We used agile method as we care about working software more than comprehensive documentation, responding to change over following a plan, giving the instructor the ability to track the development. Specification, design, implementation and testing are interleaved during the development process.

3.2 Phases Division:

1. Login and registration
2. Customer:

   2.1 Home screen: airlines logos (to search by airline), search by dates, see flight list

   2.2 Flight Screen: show flight details for available flights if any.

   2.3 Book: choose number of persons and enter packages and confirm booking

   2.4 Arrival Flights: this is shown in case of 2 way flights and shows details of flights that can be booked with the number of people chosen.

   2.5 View tickets: to show tickets just reserved

   2.6 Your Tickets: show all tickets of customer with cancel option
3. Admin:

   3.1 Home screen: displays welcome with admin name and dashboard to navigate through pages.

   3.2 Users: displays all customers details and ability to edit, delete, save.

   3.3 Flights: displays all flight details and ability to edit, delete, save.

   3.4 Booked: displays all booked details and ability to update.

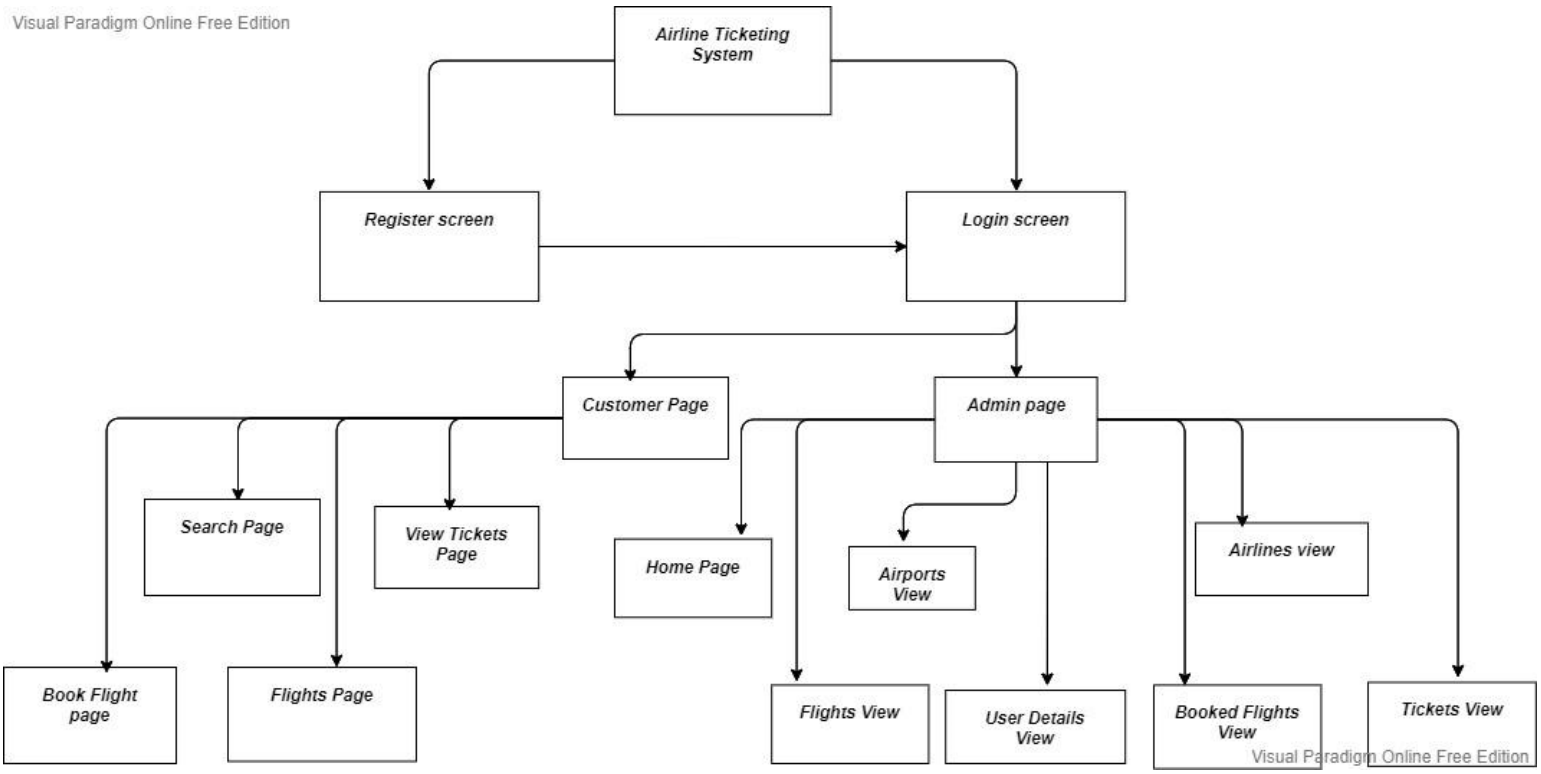
   3.5 Airlines: displays all airline details and ability to edit, save and delete.

3.6 Airports: displays all airport details and ability to save.

3.7 Tickets: displays all ticket details and ability to search for income by certain date and retrieve total income.

4- Architectural Design:

4.1 Suggested System Architecture:



4.2 Suggested Application Architecture:

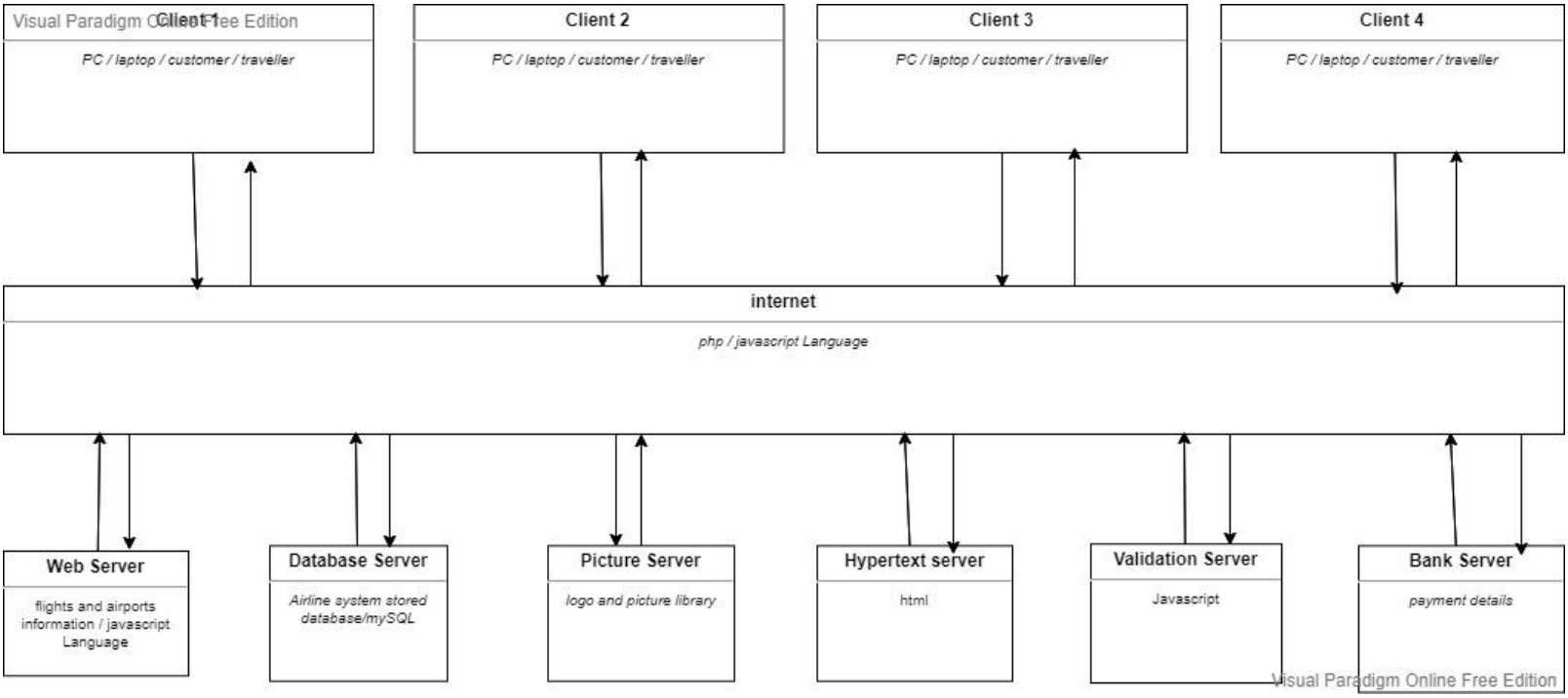
In our airline ticketing system, Client-Server Architecture was used.

This is clear in the implementation of the system due to the fact that the system is organized into services. These services include providing flight lists of available flights in every airline, booking one way flights, booking two way flights and searching. Each service is delivered from a separate server. Clients are users of the servers, and access servers to make use of them.

We used the client-server architecture, because the airline ticketing system can be used from various locations all over the world. Also servers may be replicated especially when the load on a system is variable.

In a client-server architecture, general functionality can be available to all clients and does not have to be implemented by all services. The main advantage of this model is that servers can be distributed across a network.

However, generally speaking, in client-server architecture, there is a single point of failure, therefore it is susceptible to denial of service attacks or failure of server .Moreover, performance may be unpredictable because it depends on the network as well as the system and management problems may be faced if servers are owned by different organizations.



5- Backlog :

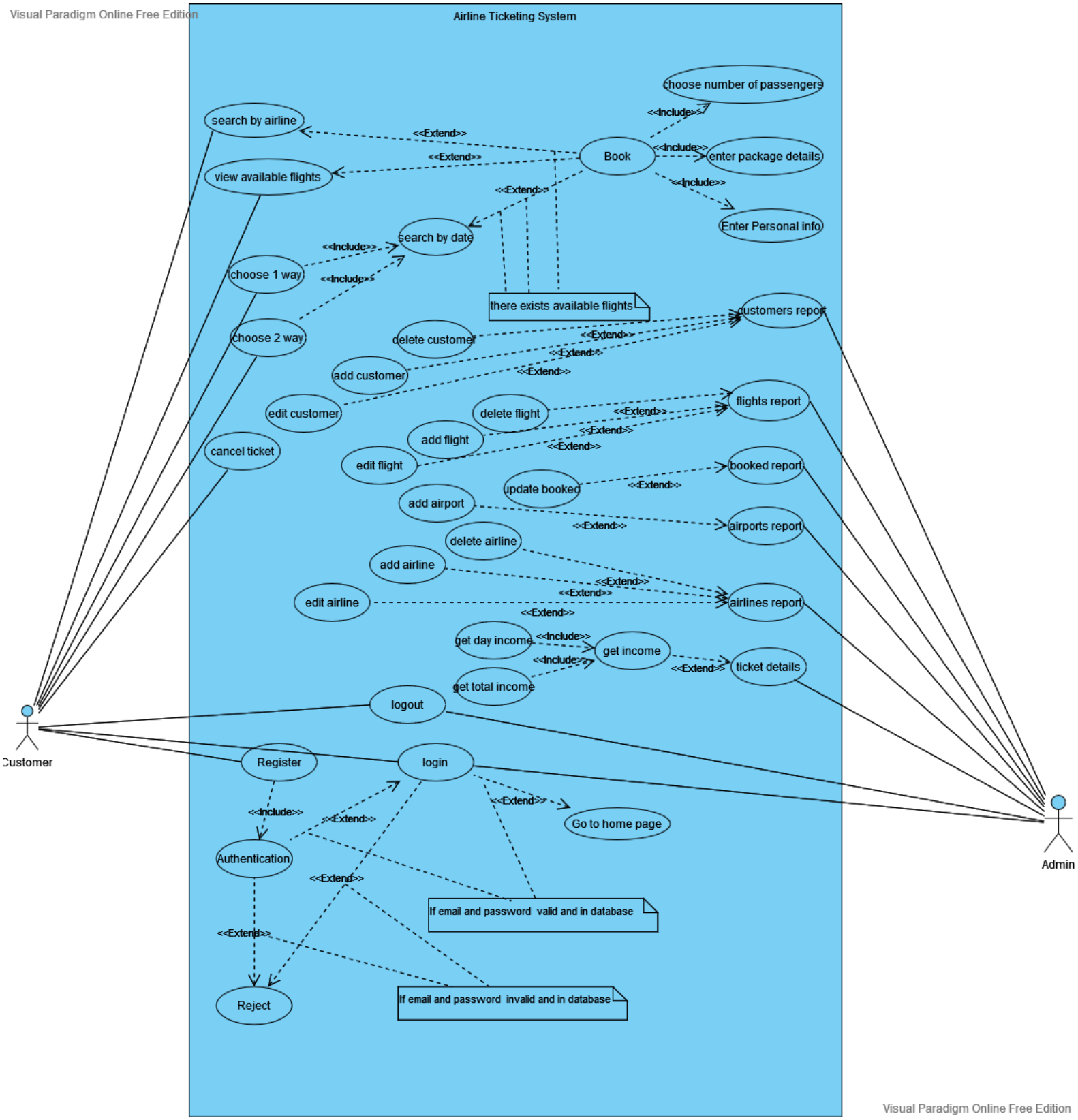
priority	Scenario	Time Estimation
1	Login & Registration to be able to go through the rest of	1

	the system.	
2	Packages comparison with saved ones in the database to complete booking.	3
3	Customers can be able to complete booking .	3
4	Customers can cancel to complete booking .	1
5	Customers can be able to search with airlines to see flights .	2
6	Customers can be able to search with dates to see flights and choose one way/2 way.	3
7	Customers can be able to see all flights	1
8	Admins be able to view all reports	1
9	Admins are able to display all customers details and ability to edit,delete,save.	1
10	Admins are able to display all ticket details and ability to search for income by certain date and retrieve total income.	1
11		1
12	Admins are able to display all airline details and ability to edit,save and delete.	1

13	Admins are able to display all booked details and ability to update.	1
14	Admins are able to display all airport details and ability to add.	1
15	Ability to logout of system	1

6- Diagrams :

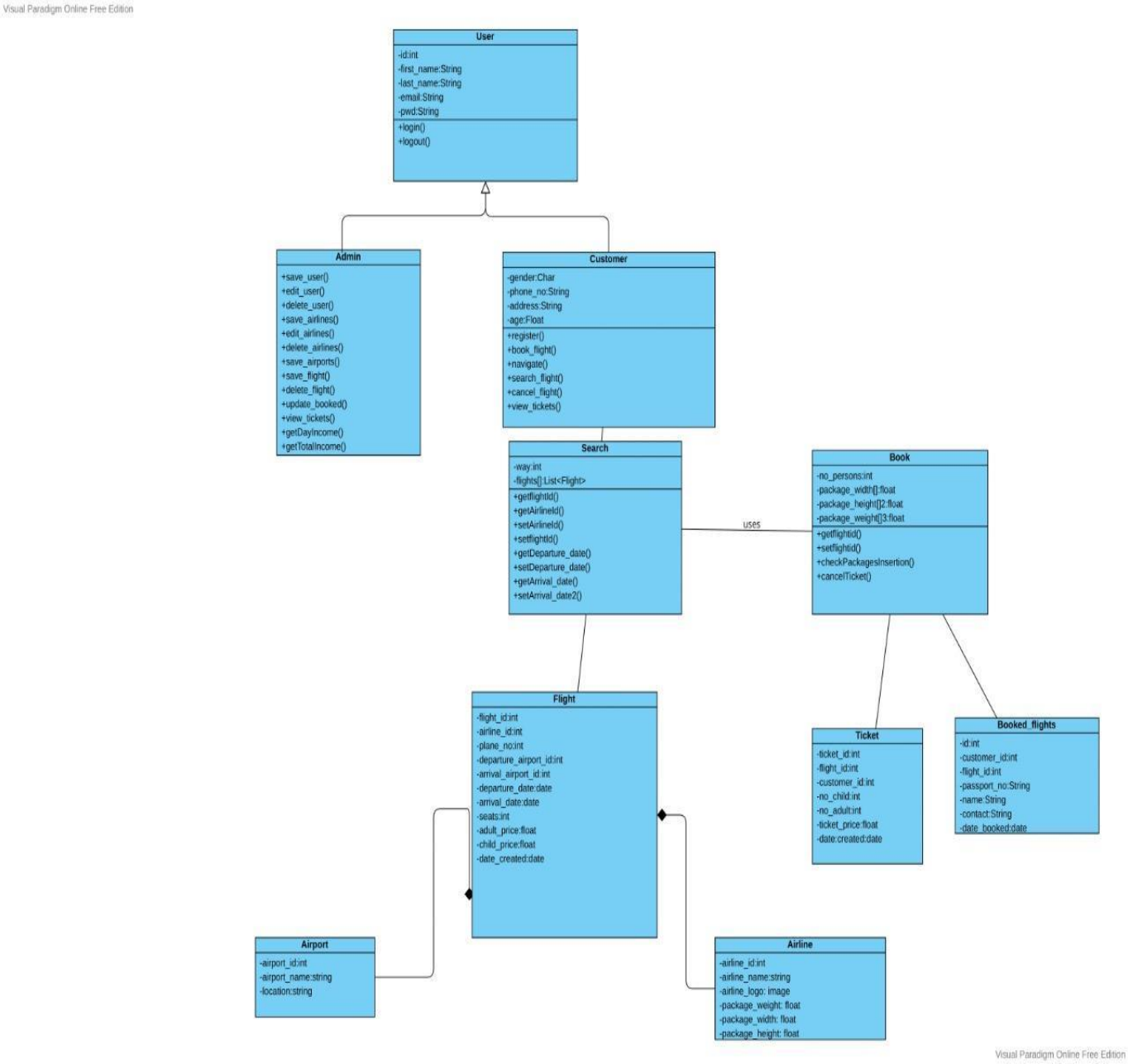
1-Use Case Diagram:



It is a behavioral diagram which shows the interactions between system and its environment at which the customer can search for a flight, book a flight, see all tickets, cancel ticket booking and the admin can manage the system by having the ability to edit,delete,save: Flights :displays all flight details and ability to edit,delete,save.

Booked :displays all booked details and ability to update. Airlines :displays all airline details and ability to edit,save and delete. Airports: displays all airport details and ability to save. Tickets: displays all ticket details and ability to search for income by a certain date and retrieve total income.

2-ClassDiagram:

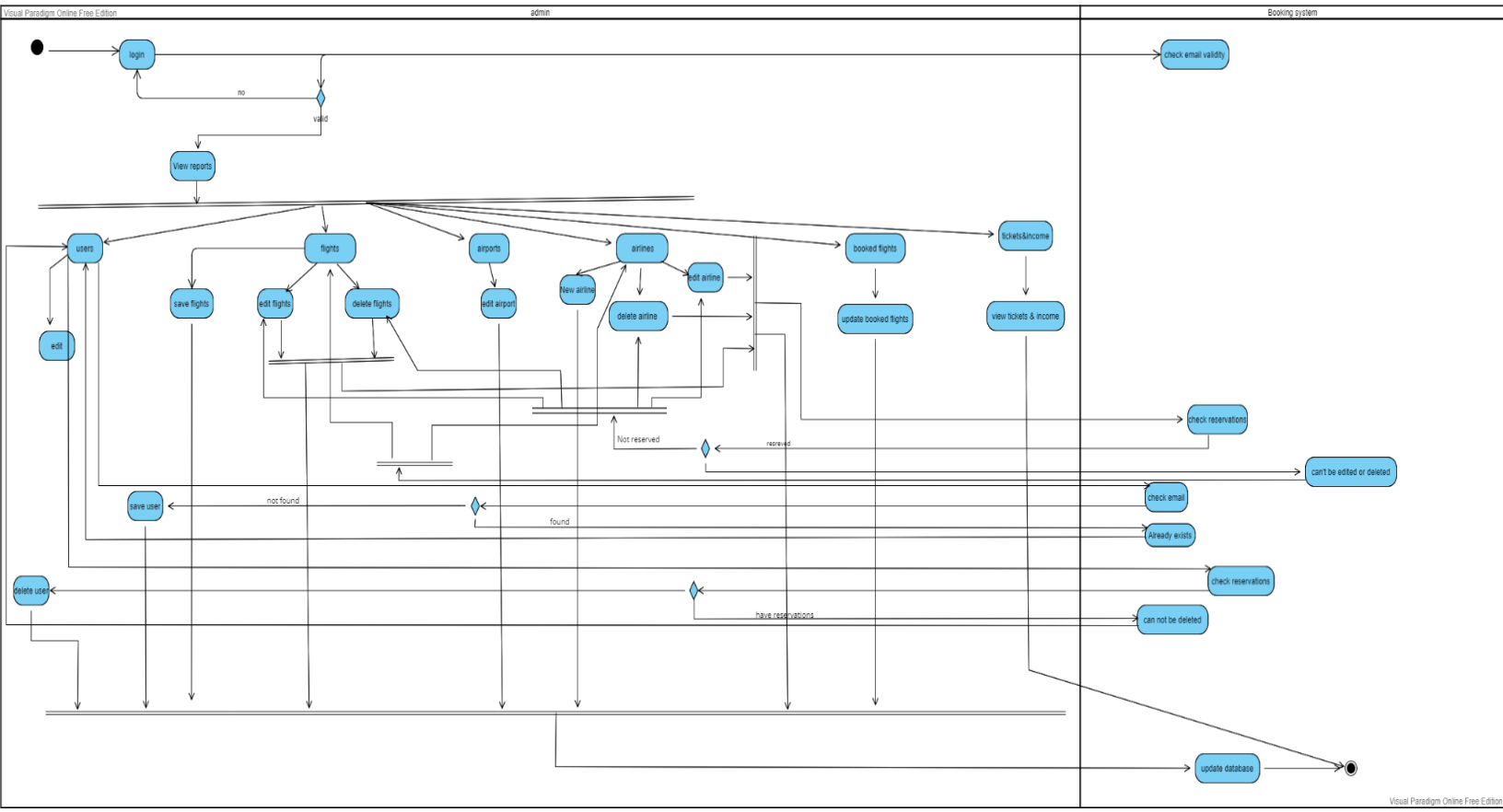


Class diagram is a structure diagram which shows the object classes in the system and the associations between these classes. We have customer and admin classes that extend user class, search class with different criteria using flight, airport, airline classes proceeding with book class

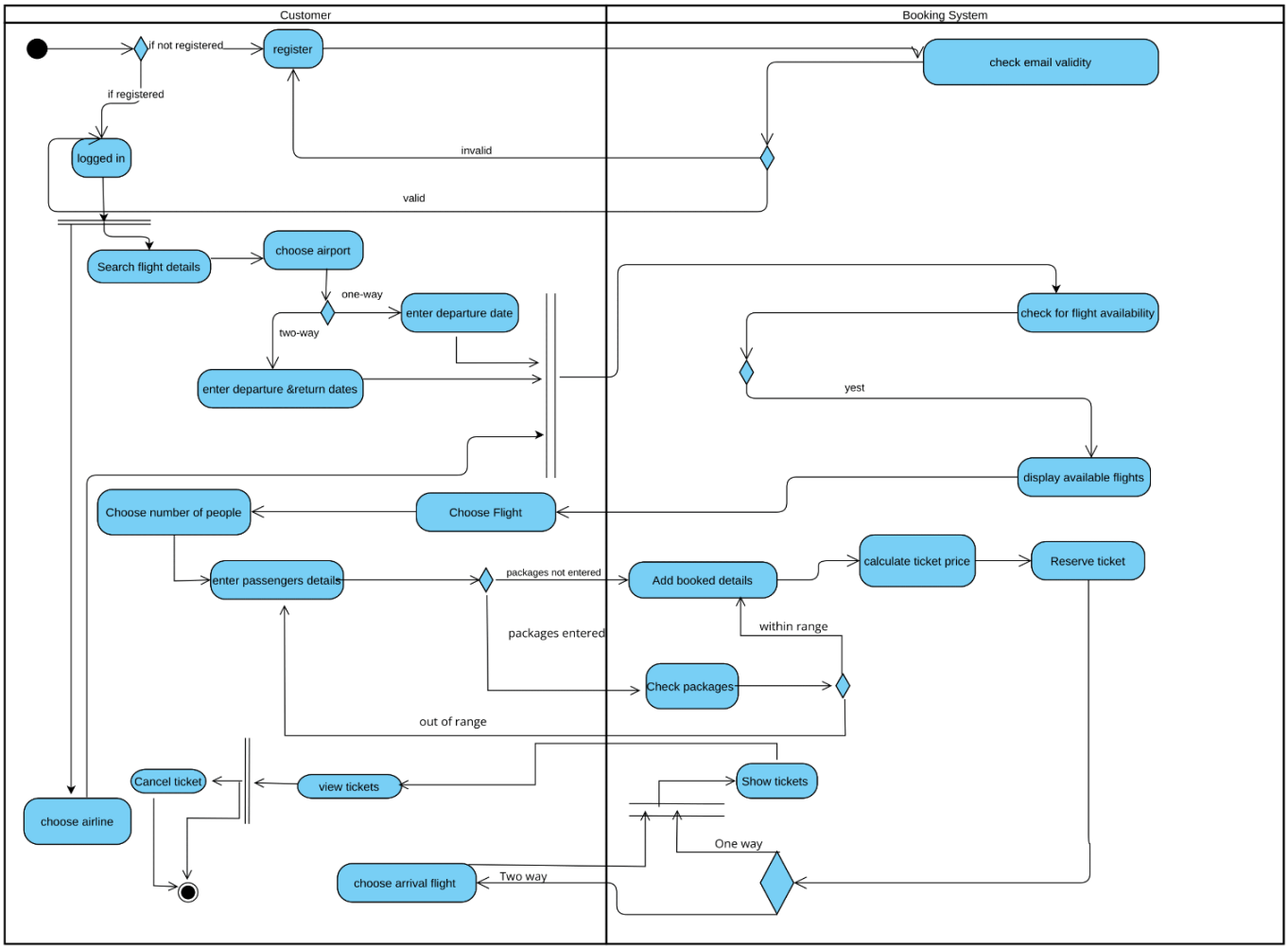
by inserting data into booked flight class and ticket class. Admin can manage the system by adding, editing, deleting other classes.

3-Activity Diagram:

Admin:



Customer:

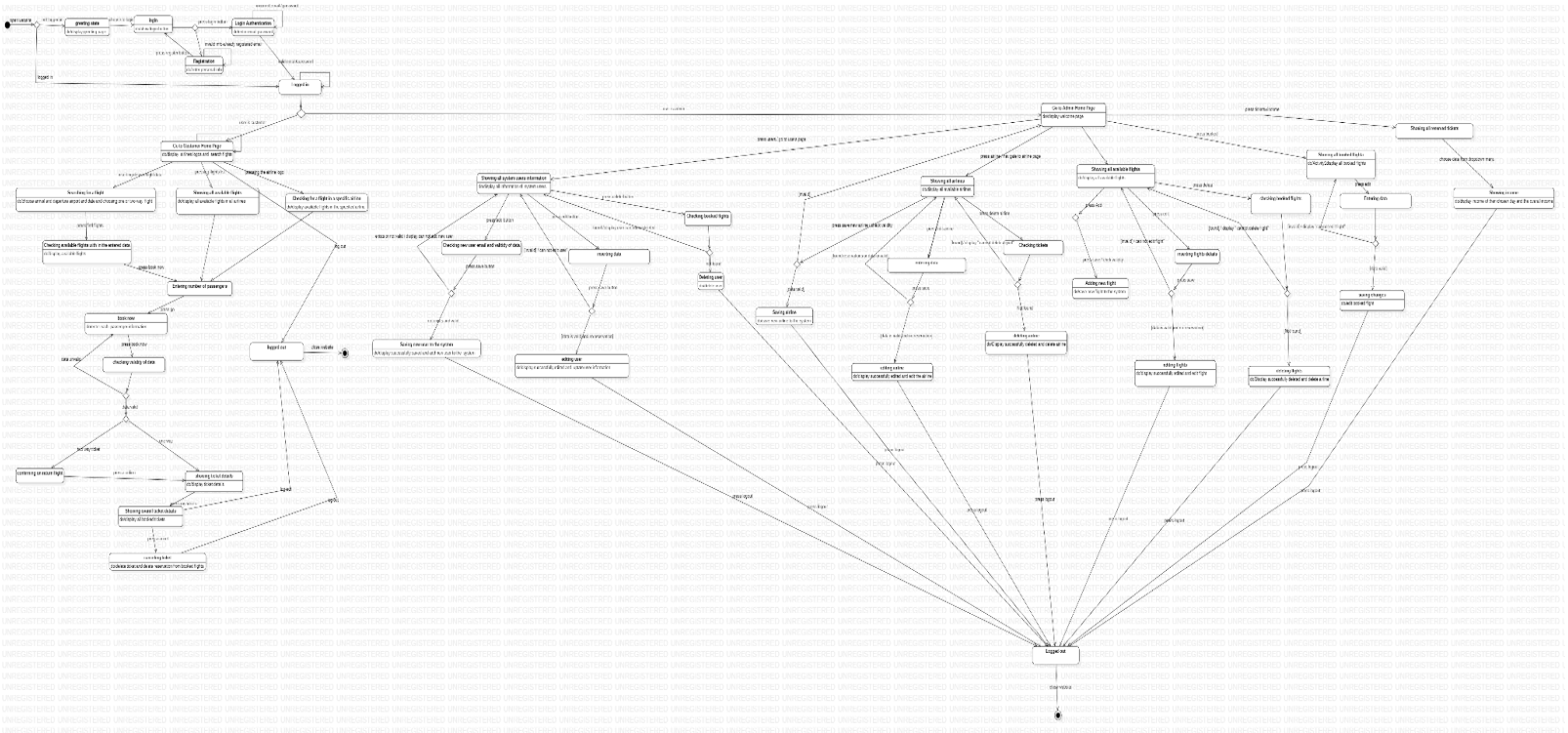


Activity diagram is a behavioral diagram which shows the activities involved in a process or in data processing. It describes the admin, customer activities for admin,

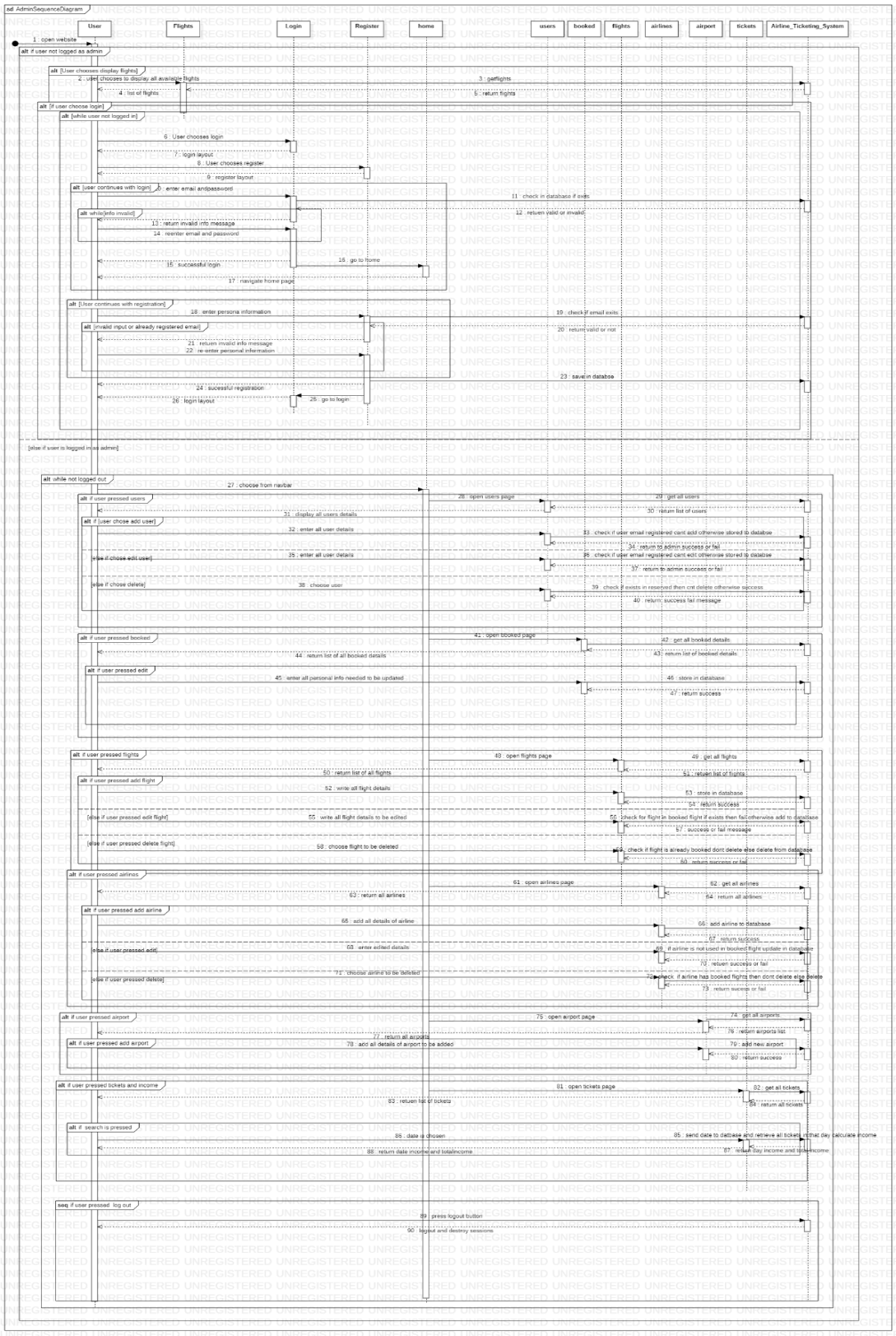
The admin views all details for all available data and can modify it. For the customer, the customer can log into the system, if logged in ;search for a flight, book flight, view tickets and cancel booking.

4-State Diagram:

State diagram is a behavioral diagram which shows how the system reacts to internal and external events. The system is divided into two halves admin and customer each with the possible states of his own to wall through the whole system with this diagram. Our system ends with the logout.

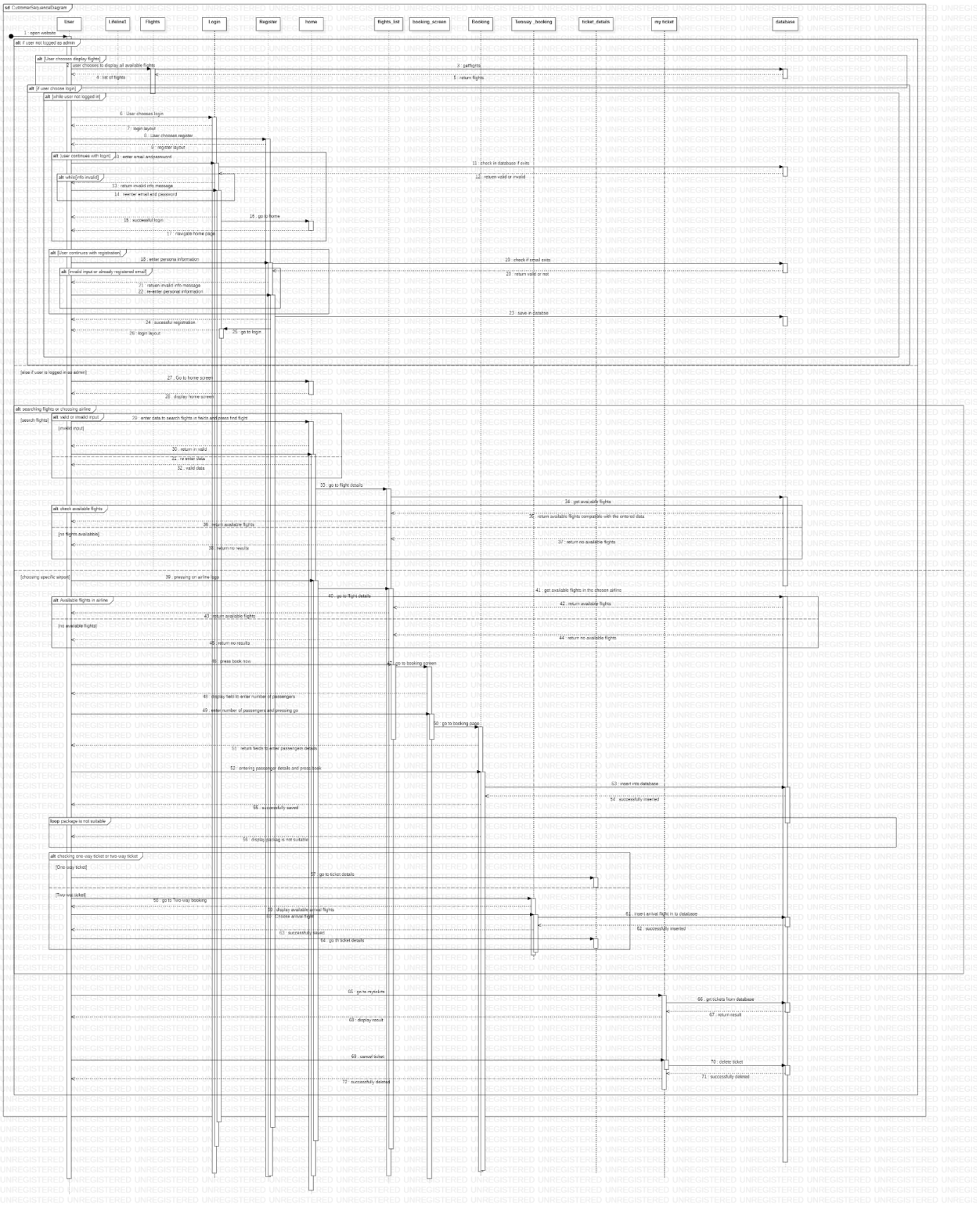


5-


sd AdminSequenceDiagram
User
Flights
Login
Register
home
users
booked
flights
airlines
airport
tickets
Airline_Ticketing_System
1 : open website
alt if user not logged as admin
alt [User chooses display flights]
2 : user chooses to display all available flights
3 : getflights
4 : list of flights
5 : return flights
alt [if user choose login]
alt [while user not logged in]
6 : User chooses login
7 : login layout
8 : User chooses register
9 : register layout
alt [user continues with login]
10 : enter email andpassword
11 : check in database if exits
12 : reteurn valid or invalid
alt while[info invalid]
13 : return invalid info message
14 : reenter email and password
15 : successful login
16 : go to home
17 : navigate home page
alt [User continues with registration]
18 : enter persona information
19 : check if email exits
20 : return valid or not
alt [invalid input or already registered email]
21 : return invalid info message
22 : re-enter personal information
23 : save in database
24 : sucessful registration
25 : go to login
26 : login layout
[else if user is logged in as admin]
alt while not logged out
27 : choose from navbar
alt if user pressed users
28 : open users page
29 : get all users
30 : return list of users
31 : display all users details
alt if [user chose add user]
32 : enter all user details
33 : check if user email registered cant add otherwise stored to database
34 : return to admin success or fail
else if chose edit user]
35 : enter all user details
36 : check if user email registered cant edit otherwise stored to database
37 : return to admin success or fail
else if chose delete]
38 : choose user
39 : check if exists in reserved then cnt delete otherwise success
40 : return success fail message
alt if user pressed booked
41 : open booked page
42 : get all booked details
43 : return list of booked details
44 : return list of all booked details
alt if user pressed edit
45 : enter all personal info needed to be updated
46 : store in database
47 : return success
alt if user pressed flights
48 : open flights page
49 : get all flights
50 : return list of all flights
51 : return list of flights
alt if user pressed add flight
52 : write all flight details
53 : store in database
54 : return success
[else if user pressed edit flight]
55 : write all flight details to be edited
56 : check for flight in booked flight if exists then fail otherwise add to database
57 : success or fail message
[else if user pressed delete flight]
58 : choose flight to be deleted
59 : check if flight is already booked dont delete else delete from database
60 : return success or fail
alt if user pressed airlines
61 : open airlines page
62 : get all airlines
63 : return all airlines
64 : return all airlines
alt if user pressed add airlina
65 : add all details of airline
66 : add airline to database
67 : return success
[else if user pressed edit]
68 : enter edited details
69 : if airline is not used in booked flight update in database
70 : return success or fail
[else if user pressed delete]
71 : choose airline to be deleted
72 : check if airline has booked flights then dont delete else delete
73 : return success or fail
alt if user pressed airport
74 : get all airports
75 : open airport page
76 : return airports list
77 : return all airports
alt if user pressed add airport
78 : add all details of airport to be added
79 : add new airport
80 : return success
alt if user pressed tickets and income
81 : open tickets page
82 : get all tickets
83 : return list of tickets
84 : return all tickets
alt if search is pressed
85 : date is chosen
86 : send date to database and retrieve all tickets, in that day calculate income
87 : return day income and totalincome
88 : return date income and totalincome
seq if user pressed log out
89 : press logout button
90 : logout and destroy sessions

Sequence Diagram:admin:

Customer:



Sequence diagram is behavior interaction diagram. First, if user is not logged in as admin or customer then he will be in a greeting page then from it can navigate the login tab and register

tab from it once user is logged in he has the accessibility to the functions of either an admin or customer according to the authentication.

7- Implementation and Modifications:

Implementation:

- Development and coding using:PHP,mySQL,javascript

Modification:

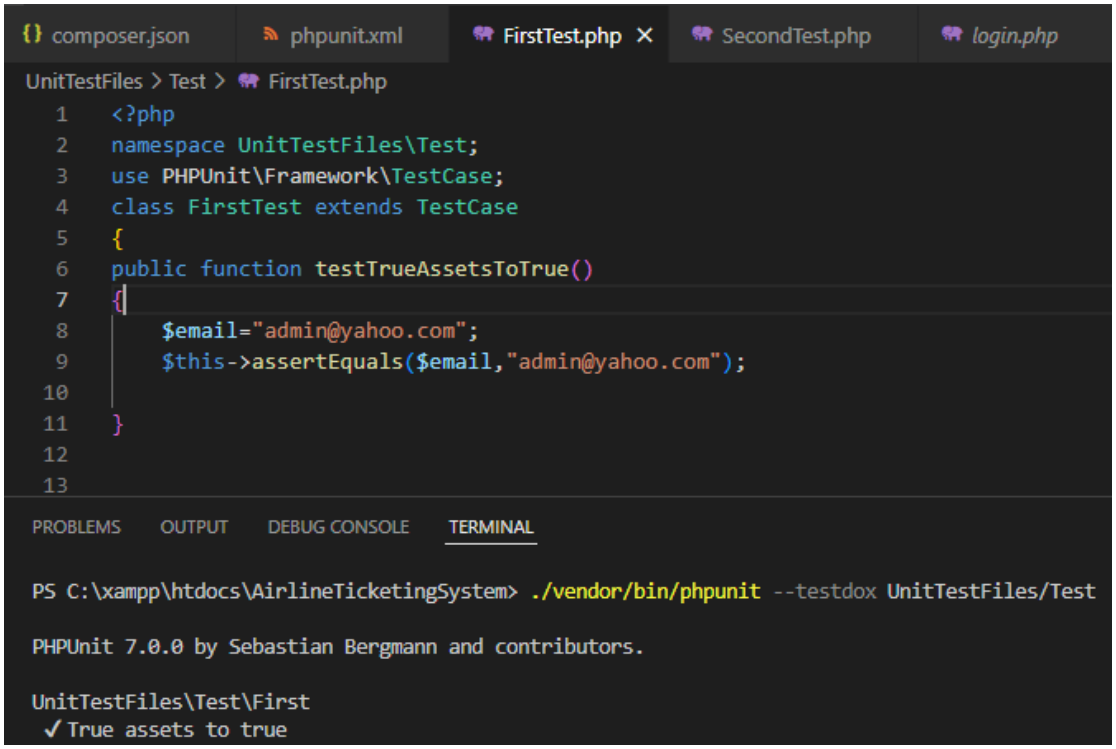
We saw that giving the customer the opportunity to choose the flights to be one way or 2 ways to ease the booking process and making our website more accessible and user friendly.

8-Testing:

8.1- Login page testing:

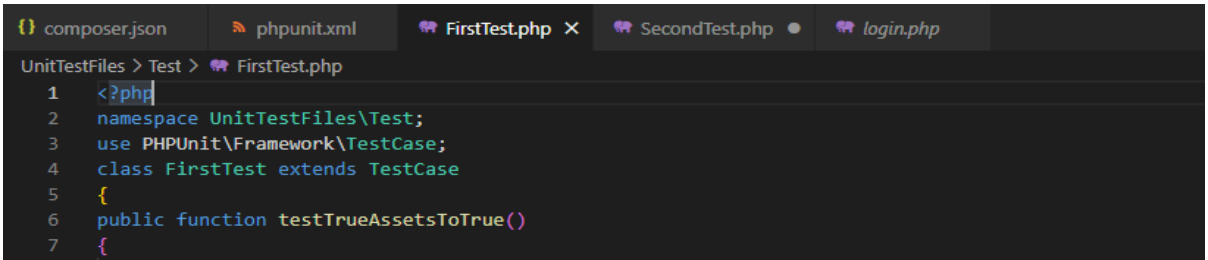
8.1.1 Correct email

- Expected : ”No errors”



8.1.2 wrong email

- Expected :”error Incorrect email ”



-
- 8.1.3 Correct password
- Expected : ”No errors”

8.1.4 Correct password

- Expected : ”No error: correct password ”

```
13
14    public function testTrueAssetsToTrue2()
15    {
16
17        $pass="1234";
18        $this->assertEquals($pass,"1234");
19
20    }
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

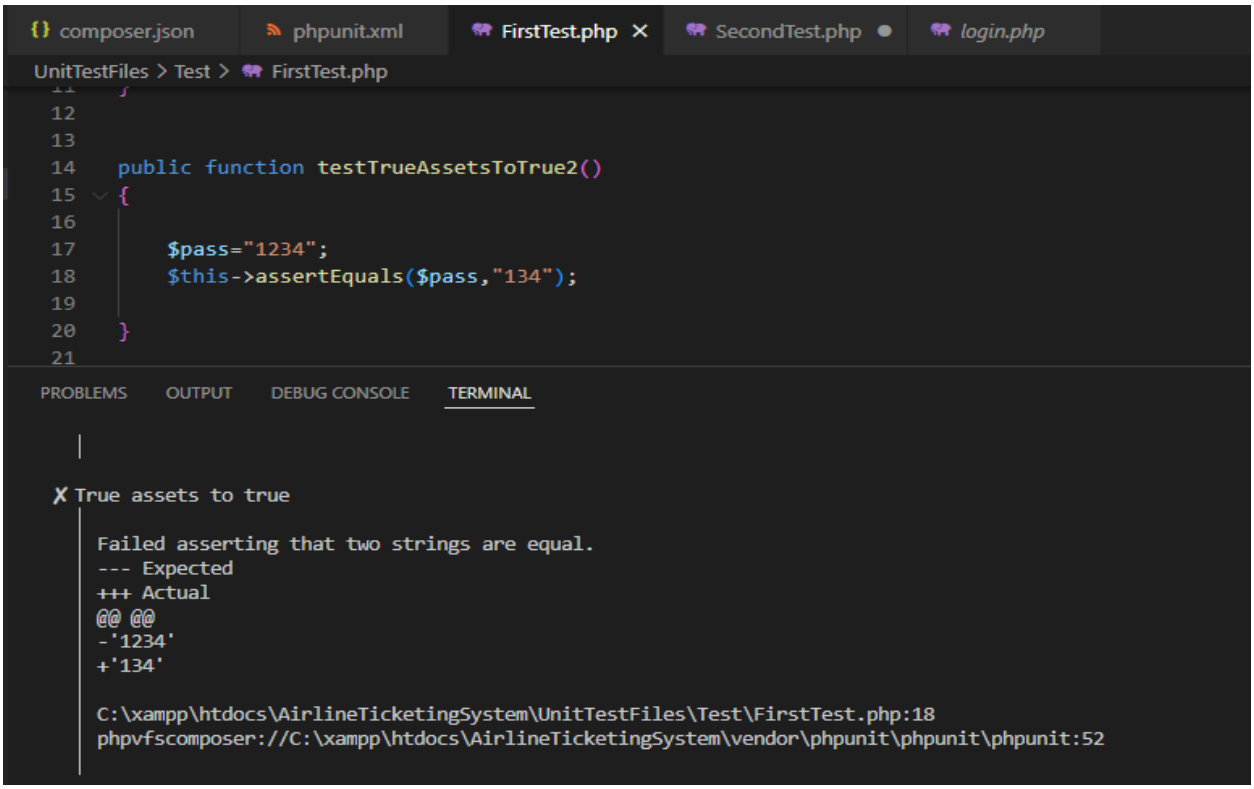
```
PS C:\xampp\htdocs\AirlineTicketingSystem> ./vendor/bin/phpunit --testdox UnitTestFiles/Test

PHPUnit 7.0.0 by Sebastian Bergmann and contributors.

UnitTestFiles\Test\First
 ✓ True assets to true
 ✓ True assets to true
```

8.1.4 wrong password

- Expected : ”Error: Incorrect password ”



8.1.5 Correct email & password

- Expected: ”No errors ”

```
21
22  public function testTrueAssetsToTrue3()
23  {
24
25      $pass="1234";
26      $email="admin@yahoo.com";
27      $this->assertEquals($pass,"1234");
28      $this->assertEquals($email,"admin@yahoo.com");
29
30  }
31
32
33  }
34  ?>
35
```

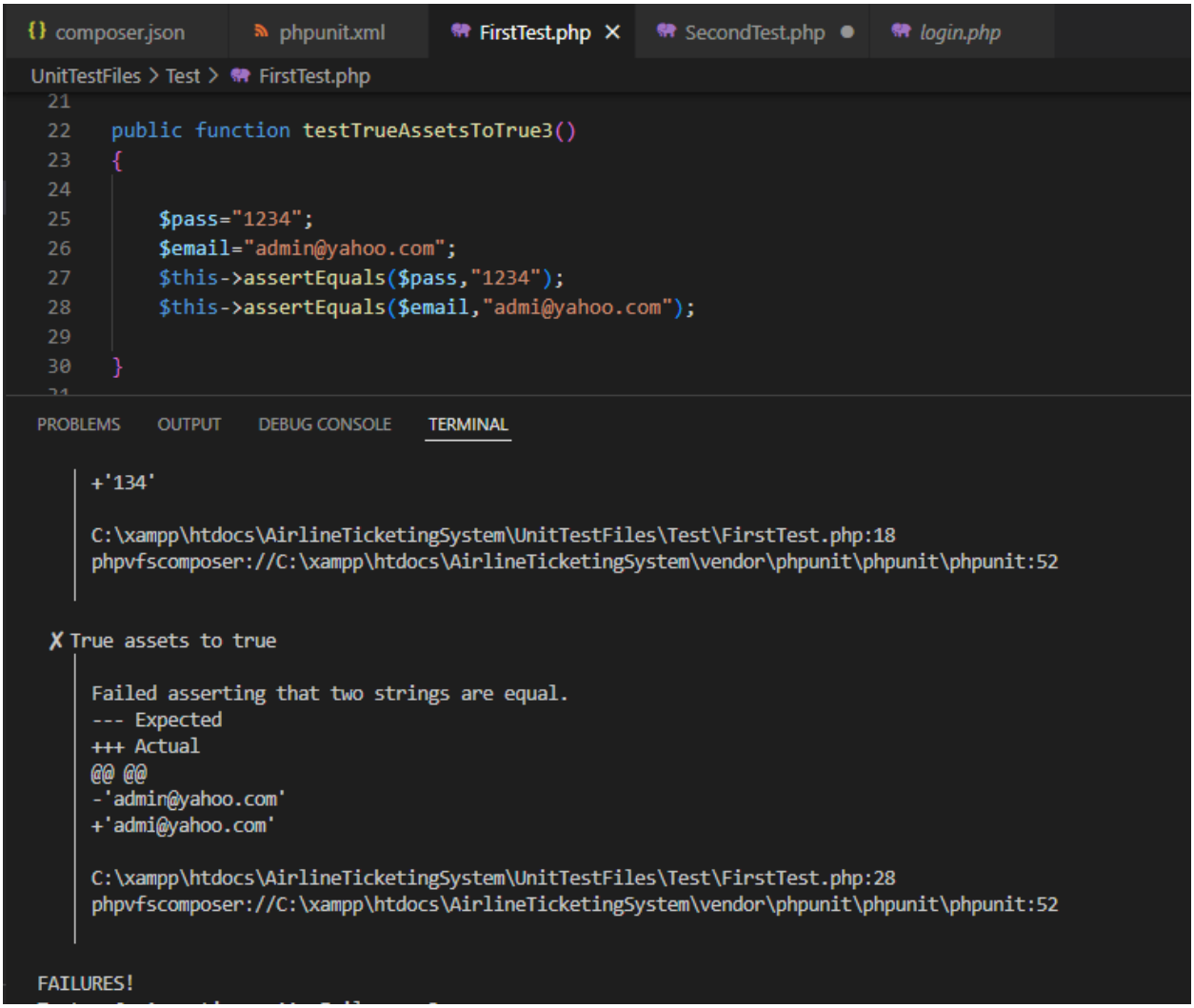
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PHPUnit 7.0.0 by Sebastian Bergmann and contributors.

UnitTestFiles\Test\First
 ✓ True assets to true
 ✓ True assets to true
 ✓ True assets to true
```

8.1.6 Incorrect email & correct password

- Expected: ”Error : incorrect email”



The login is one of the most prioritized functions as it is the basis of our website .In login after validating that email is correct and password as well this will move the user to admin or customer according to his email .

8.2- booking testing:

8.2.1 Entered packages weight, height, width is a number

- Expected: ”No error”

composer.json | phpunit.xml | FirstTest.php | SecondTest.php

UnitTestFiles > Test > SecondTest.php

```php
<?php
namespace UnitTestFiles\Test;
use PHPUnit\Framework\TestCase;
class SecondTest extends TestCase
{
public function testTrueAssetsToTrue_booking1()
{

    $package_weight= 46;
    $package_width=50;
    $package_hight=60;
    $this->assertTrue(is_int($package_weight));
    $this->assertTrue(is_int($package_width));
   $this->assertTrue(is_int($package_hight));

}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
UnitTestFiles\Test\First
 ✓ True assets to true
 ✓ True assets to true
 ✓ True assets to true

UnitTestFiles\Test\Second
 ✓ TrueAssetsToTrue booking1
```

8.2.2 Entered packages weight is a string, height, width is a number

- Expected: ”Error : package weight have to be a number”

composer.json | phpunit.xml | FirstTest.php | SecondTest.php | login.php

UnitTestFiles > Test > SecondTest.php

```php
<?php
namespace UnitTestFiles\Test;
use PHPUnit\Framework\TestCase;
class SecondTest extends TestCase
{
public function testTrueAssetsToTrue_booking1()
{

    $package_weight="46";
    $package_width=50;
    $package_hight=60;
    $this->assertTrue(is_int($package_weight));
    $this->assertTrue(is_int($package_width));
   $this->assertTrue(is_int($package_hight));

}
```

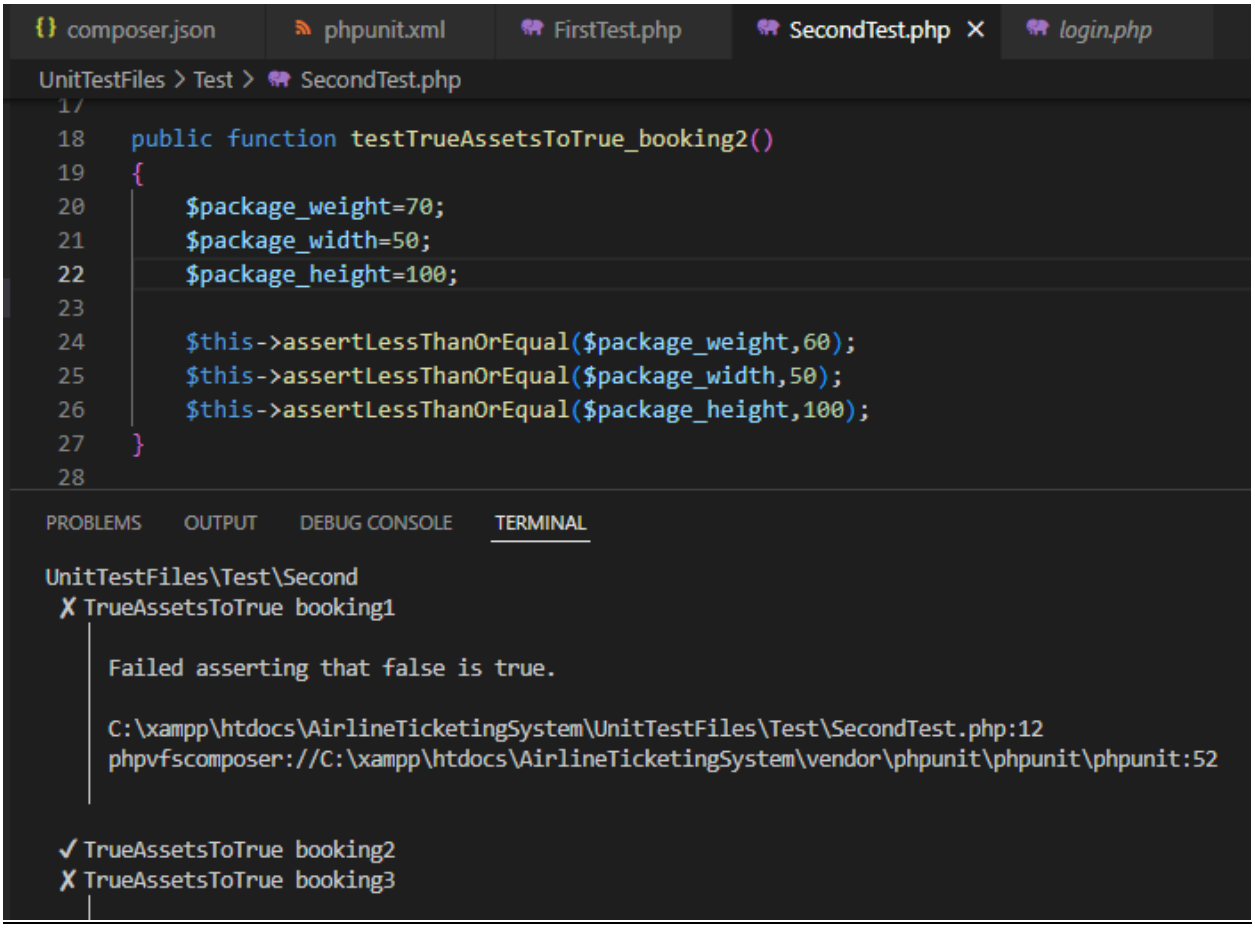
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
UnitTestFiles\Test\Second
 ✗ TrueAssetsToTrue booking1
   Failed asserting that false is true.

   C:\xampp\htdocs\AirlineTicketingSystem\UnitTestFiles\Test\SecondTest.php:12
   phpvfscomposer://C:\xampp\htdocs\AirlineTicketingSystem\vendor\phpunit\phpunit\phpunit:52
```

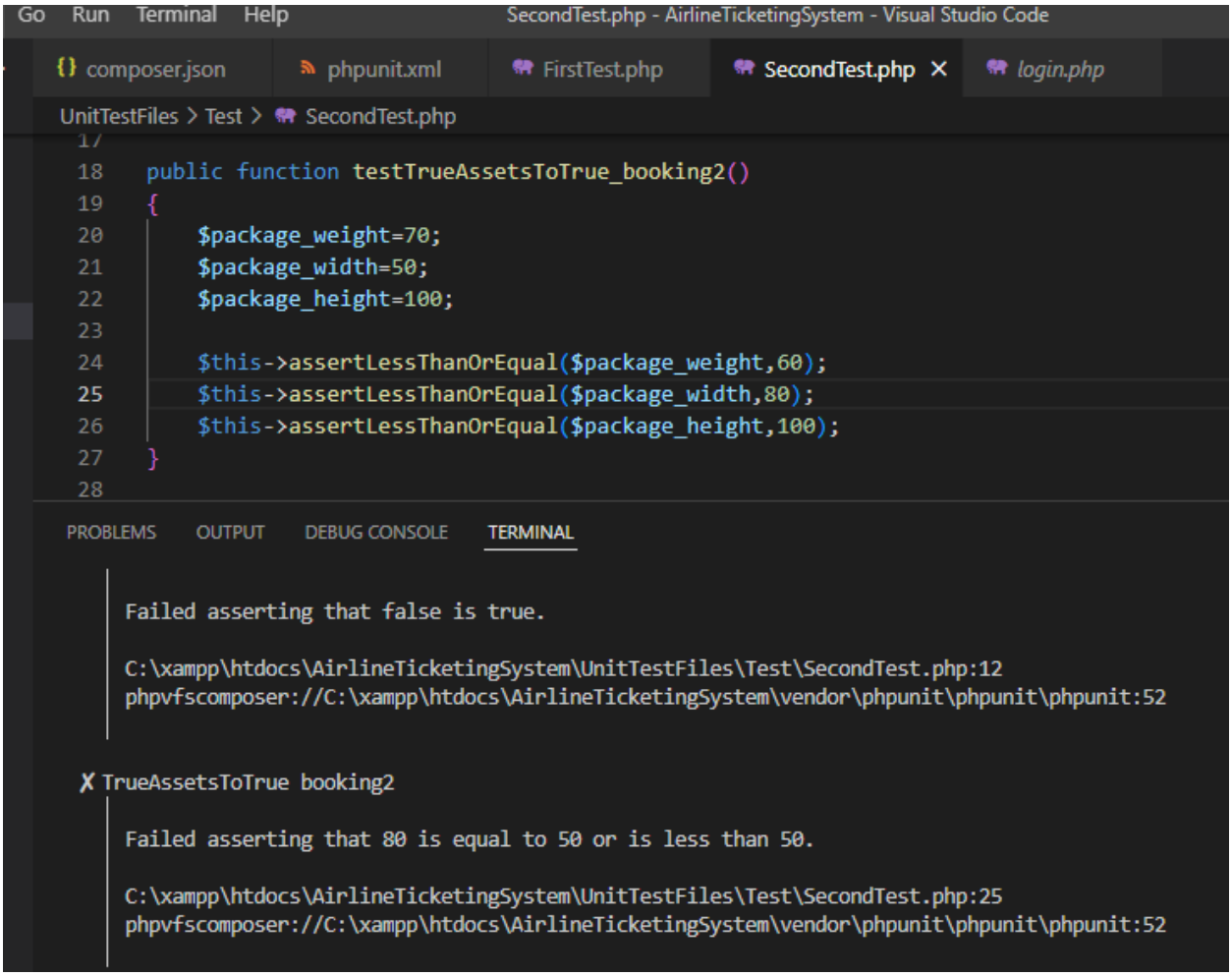
8.2.3. Entered package weight, width and height is less than or equal the airline specified package weight and dimensions

- Expected: ”No error”



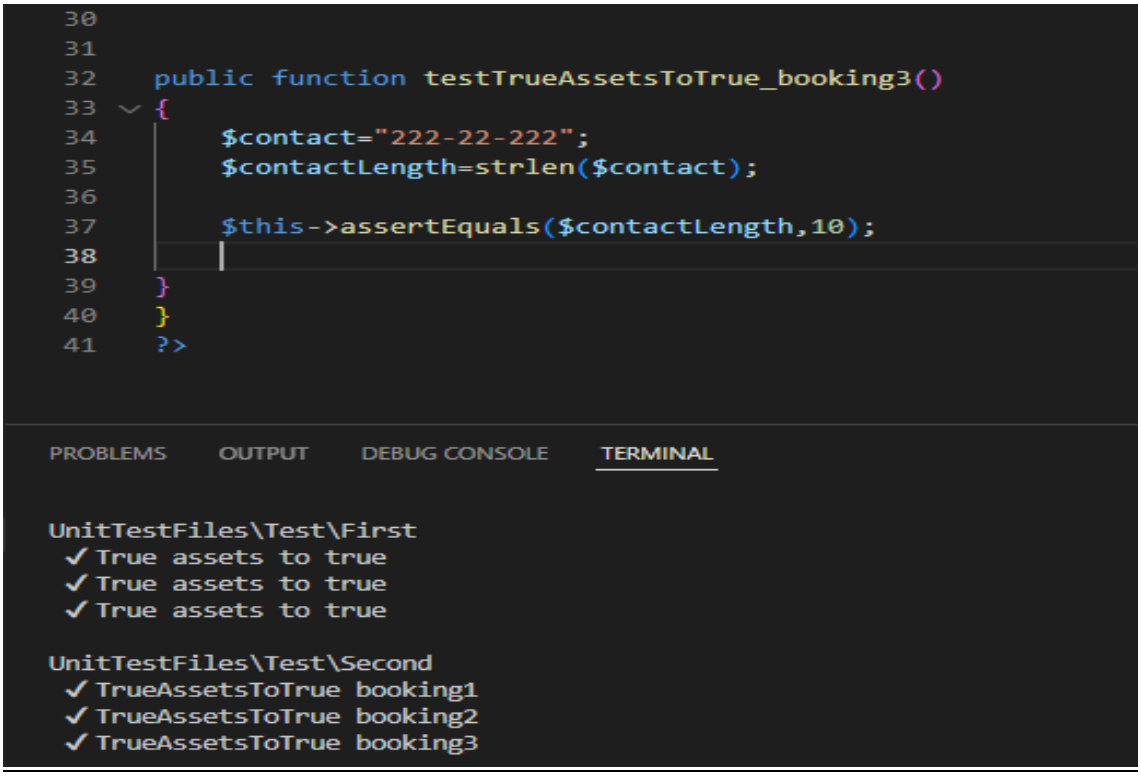
8.2.4. Entered package weight, width and height is greater than the airline specified package weight and dimensions

- Expected: ”Error : greater than the allowed weight / dimensions ”



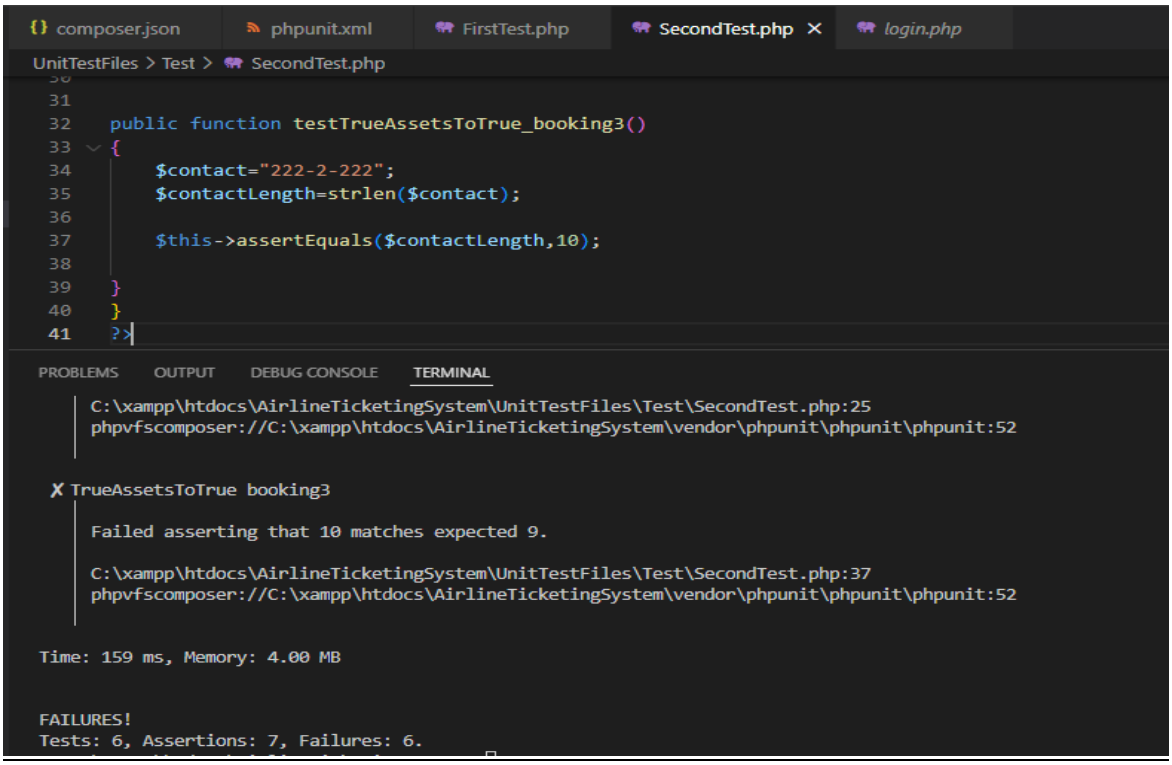
8.2.5. Entered contact number is valid

- Expected: ”No Error ”



8.2.6. Entered contact number is not valid

- Expected: ”Error : contact number length is not suitable”



Validation of booking is also of significance because according to whether the data is valid or not the data of tickets and booking is inserted to the database; if package height, width and weight are within specified range and valid and also contact is valid the data is inserted to the database and ticket is successfully booked.

Release:

Test cases are to be tested afterward on our code, the Code must be convenient with the automated test driven and gives no errors indicating successful implementation and design of system.