Group 19

Software Engineering

Dr.Manal Ismael

Billing Management System

Esraa Ahmrd Taha Ali	Sec1
Ziad Ahmed Hossam ElDin	Sec2
Omar Ahmed Kadry	Sec3
Doaa Shabaan Mahmoud	Sec2

Faculty of Engineering
3RD COMPUTER DEPARTMENT

Participations & Team Work

Diagrams:

SRS (Zaid Ahmed)
ER Diagram (Esraa Ahmed-Omar Ahmed)
DFD (Esraa Ahmed-Doaa Shabaan)
Use case (Ziad Ahmed)
Class Diagram (Esraa Ahmed)
Sequence Diagram (Omar Ahmed)

Test Black Box

Ziad Ahmed (Add buyer- Update Buyer)
Omar Ahmed (Add Product-Delete Product)
Esraa Ahmed (Billing-Login-Update Product)
Doaa Shabaan (Delete Buyer)

Test white Box CFG

Doaa Shabaan (Login-Buyer Details-Product Details)
Ziad Ahmed (Add Buyer-Delete Buyer-Update Buyer)
Omar Ahmed (Add product-Delete Product-Update Product)
Esraa Ahmed (Billing) & reviewed The CFG Diagrams

Junit

All team make it (Add buyer- Delete buyer- Update buyer)
Test report has been collected by Omar

Billing Management System

1. Introduction

a. Introduction

The purpose of this document is to present a detailed description of the Billing System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do and the constrains under which it will operate.

This document is intended for both the customers and the developers of the system. This SRS will allow for a complete understanding of what is to be expected of the BMS to be constructed. The clear understanding of the BMS and its functionality will allow for the correct software to be developed for the end user and will be used for the development of the future stages of the project.

b. Problem definition

This system of exclusively needed in many places that needs to have a billing system (Cashier), a list of the products exist and the details of the customers.

In so many stores you can see that without the functionality of our system you will find that there disorder, random and poor management of the store billing criteria.

c. Scope of the system

This Billing System facilitates to any store that has billing system. Such as: Grocery, Market, Bootshop etc... The main goal of introducing this system is to benefit the customers so that all thebills could be paid with ease.

d. Similar systems

The BMS is a new system which is different from the existing systems as it merges various billing systems as one single unit and provides a reliable service to the customers with all the required functionalities. This product is totally self-contained.

e. Glossary

BMS- Billing Management System.

EBS- Electricity Billing System.

SP- Service Provider.

2. User requirements

a) Functional Requirements

i. This software could be used either by an organization to carry out the billing transactions or by any user who can avail online services and can make online payment.

ii. The user can register and he will be provided with a log-in id. Thus he can manage his account through a secure password.

iii. Based on the total usage, the system shall calculate the amount payable and thus produce the bill.

iv. The calculation method for the total bill is embedded in the software. The system also gives the last date for payment as set by the standards. It later adds on the fine amount per day for delayed payment.

v. The system provides access only to the administrator to view databases, update unit costs, delete accounts, etc.

b) Non-functional Requirements

i. The load time for user interface screens shall take no longer than two seconds.
ii. The log in information shall be verified within ten seconds.
iii. The Billing System shall be a stand-alone system running in a Windows environment
iv. The system must store several databases like user information, usernames, customer feedback, and so on

c) General constraints

i. The user doesn't need to have any experience of using the BS software. Little technical expertise is sufficient to operate the software.
ii. It is assumed that the user will be a registered user and will be provided with a username and password.
iii. The system must store several data like user information, usernames and so on

.

3. System users

-Developer: the one who make changes on code if an error is found.
-Administrator: the man who has the right to delete, view data, update data of a customer.

4. System interfaces

- User Interfaces

The user interface would include the following features:

1) Log In : ask for Username and Password

2) Bill payable: Shows the buyer details, product details and the amount to be paid by the customer.

3) Buyer details: show the list of buyer that are stored in system's data base

4) Delete buyer: allows to delete a specific buyer from the system's data base.

5) Delete product: allows to delete a specific product from the system's data base.

6) New Buyer: used for adding a new customer to system's data base.

7)New Product: used for adding a new product to system's data base.

8)Product Details : show the list of products that are stored in system's data base.

8) Update Buyer: used for updating a customer info.

9)Update Product: used for updating a product info.

- Hardware Interfaces

  The system can run on any Windows based system.

- Software Interfaces

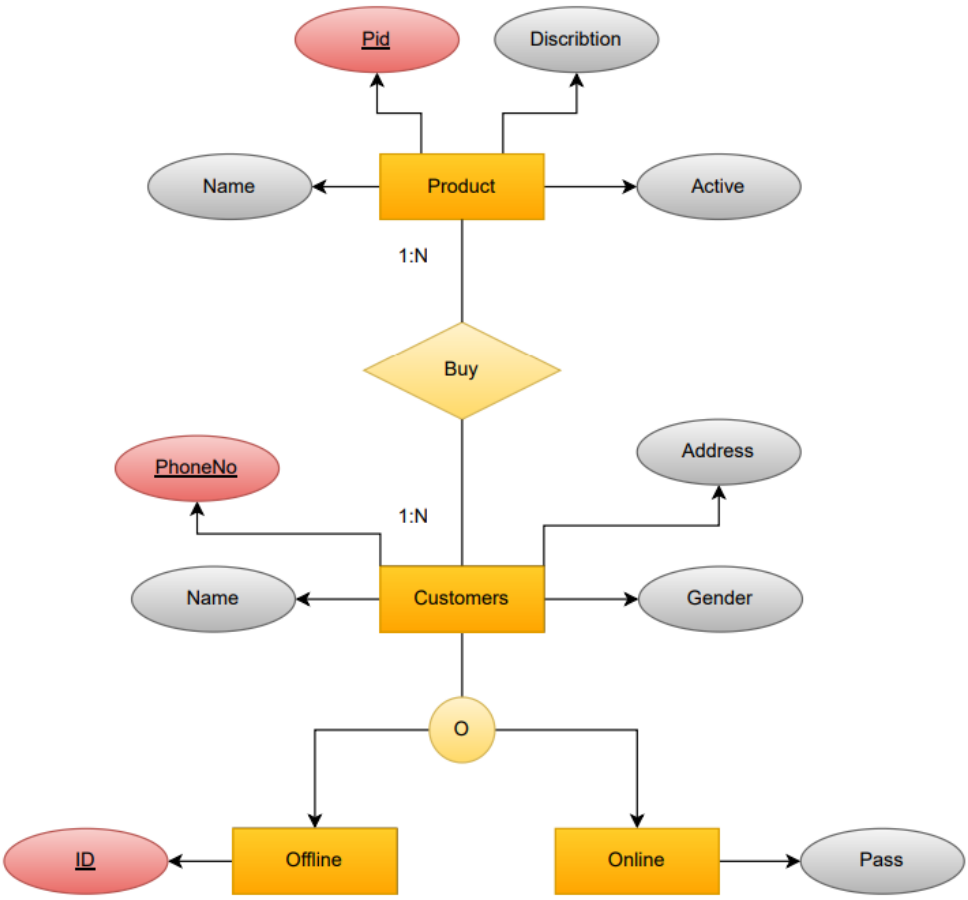
  The system shall interface with a database.

- Communication Interfaces

  This system is a stand-alone product and does not interface with any other communication device.

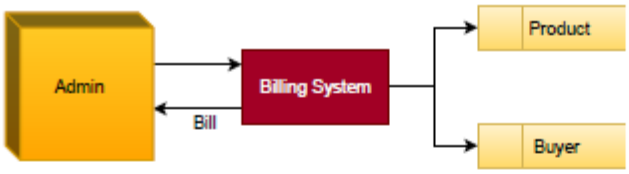
5. Software requirements and specifications

ERD Diagram:

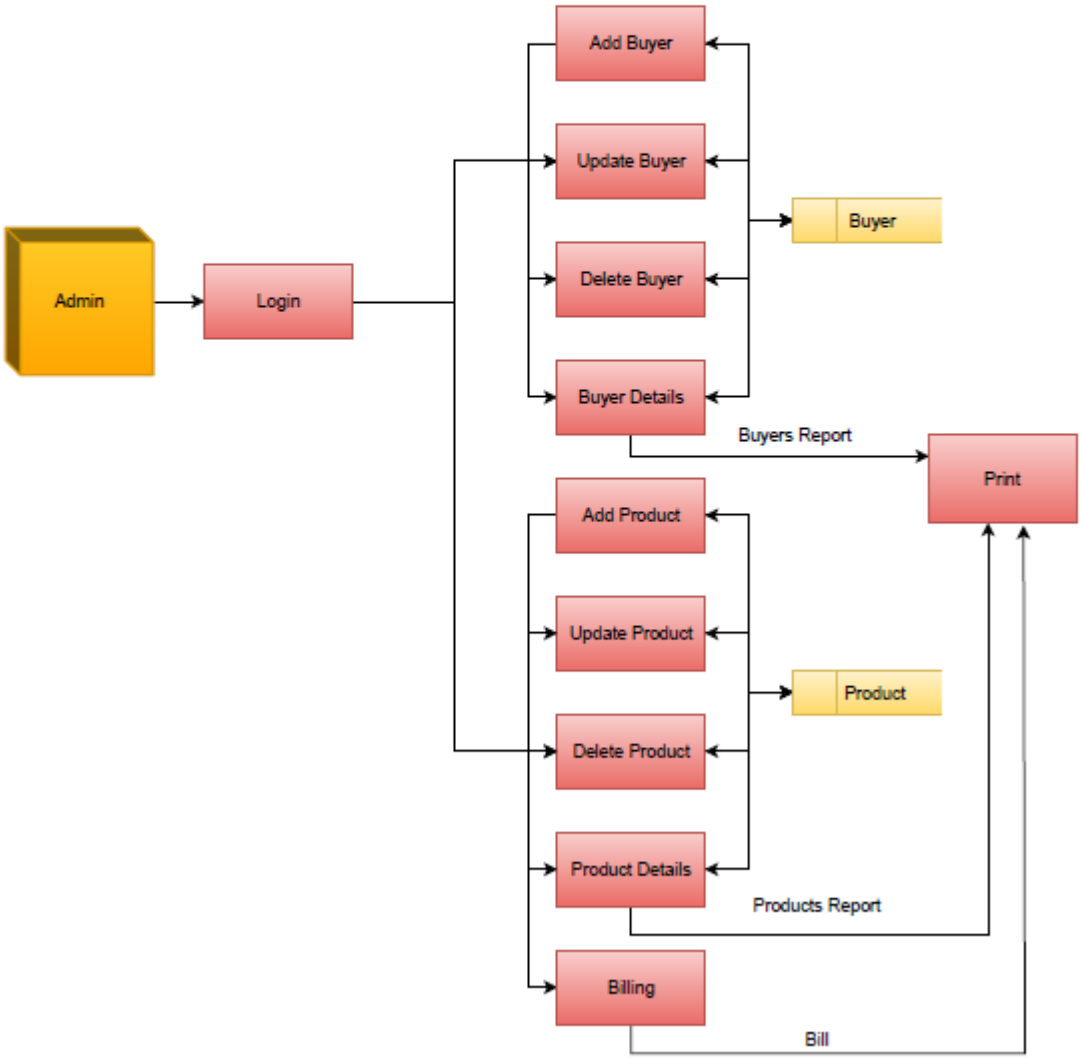


DFD Diagram:

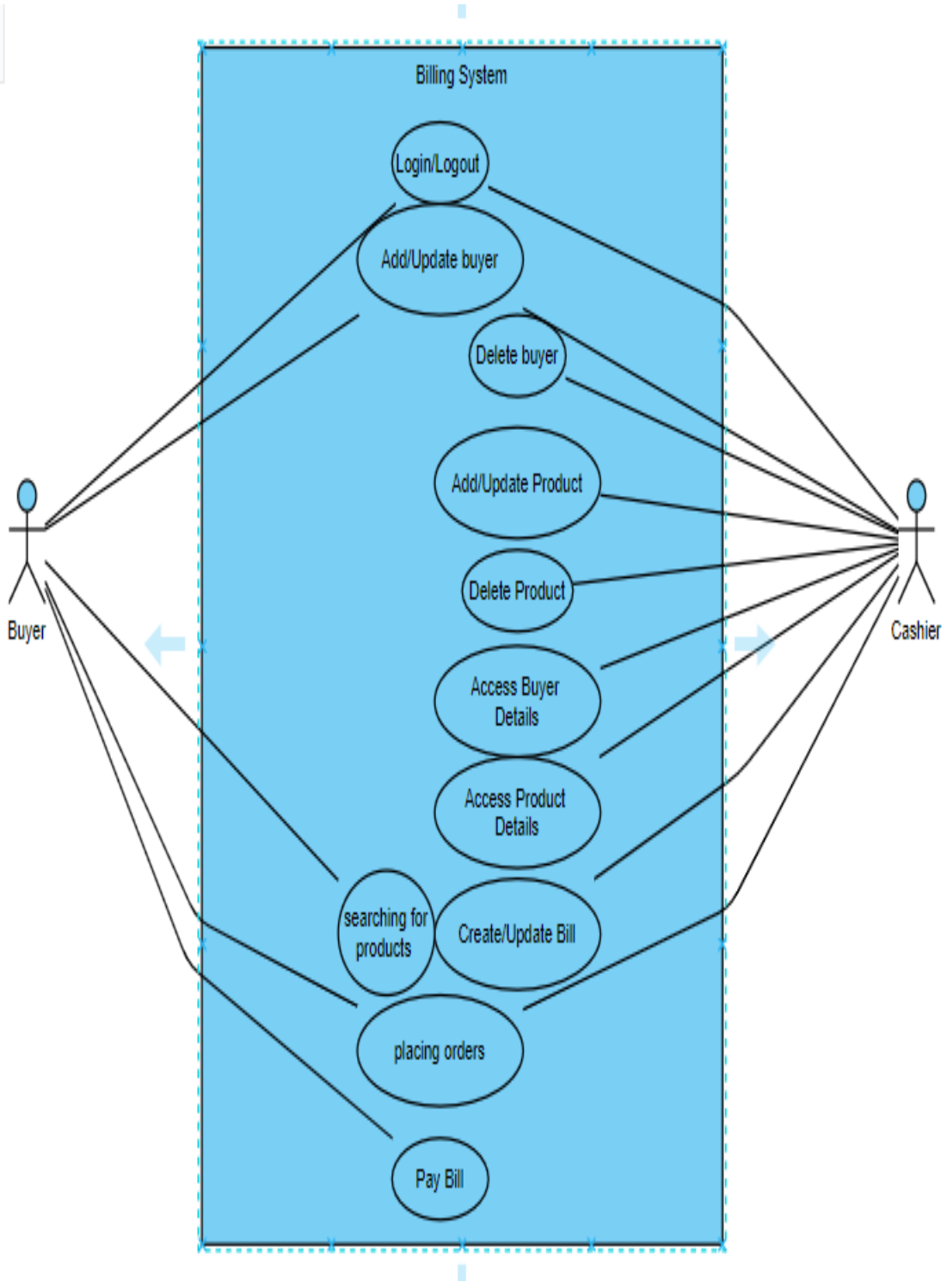
Level (0):



Level (1):

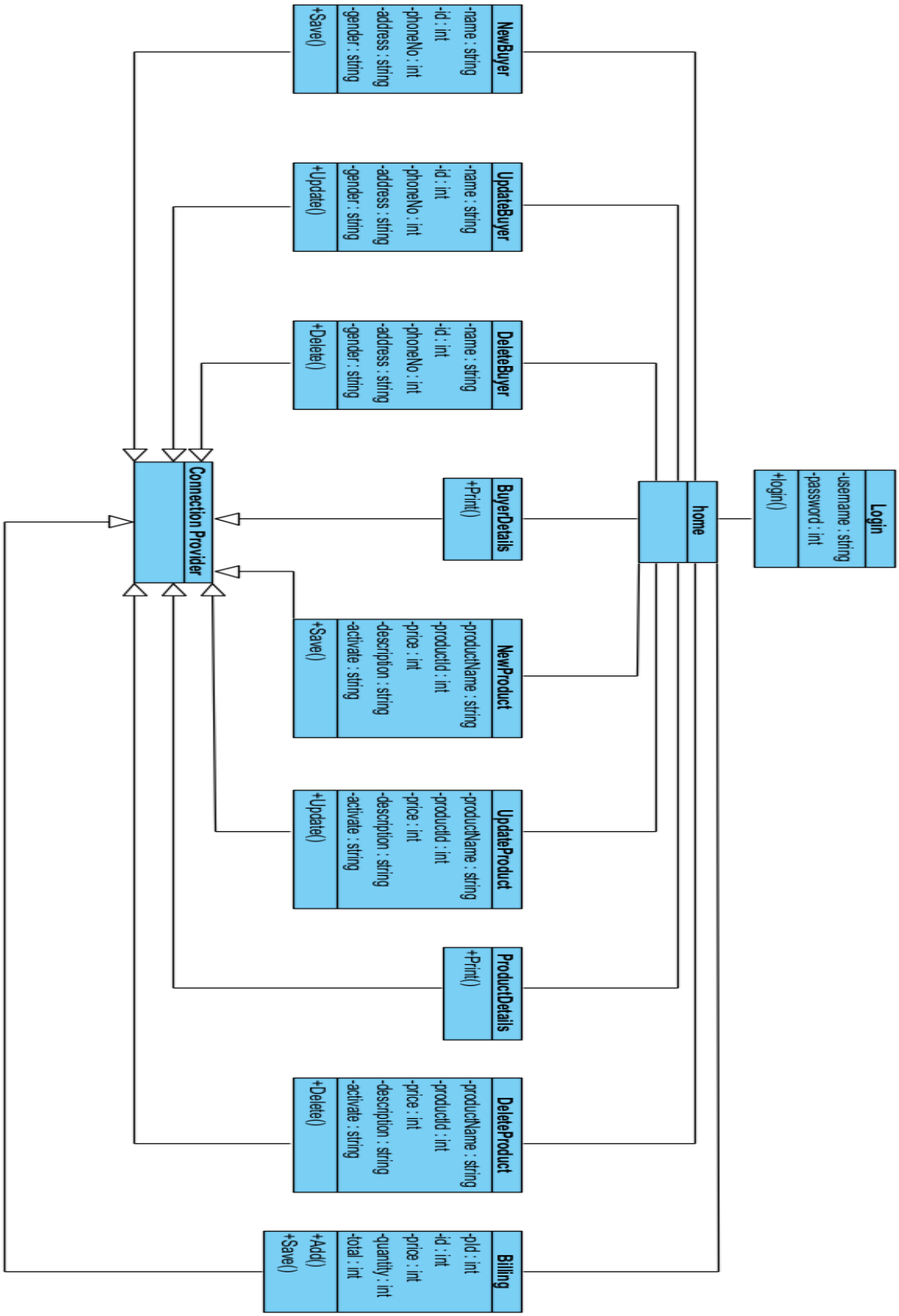


Use Case:

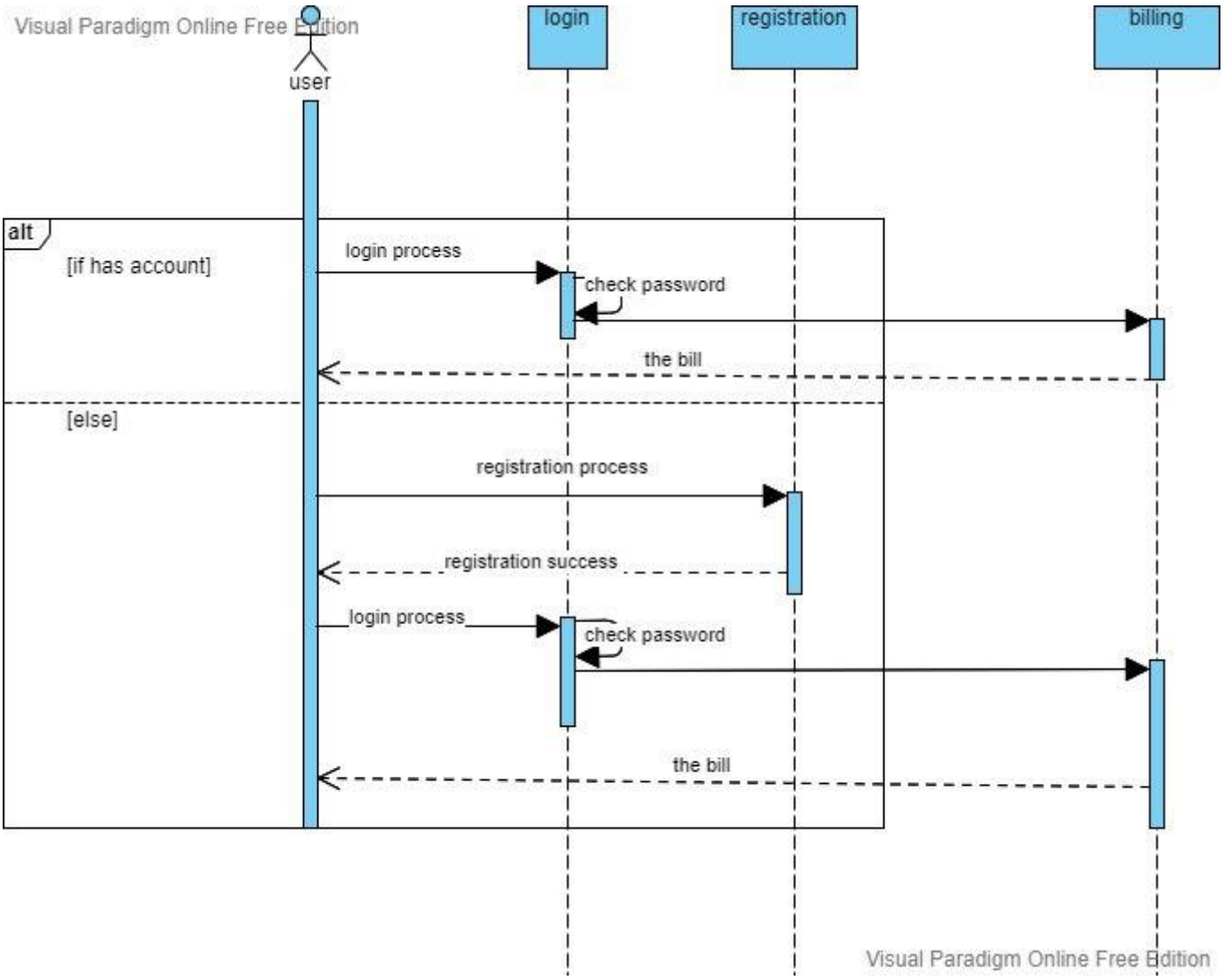

Billing System
Login/Logout
Add/Update buyer
Delete buyer
Add/Update Product
Delete Product
Access Buyer Details
Access Product Details
searching for products
Create/Update Bill
placing orders
Pay Bill
Buyer
Cashier

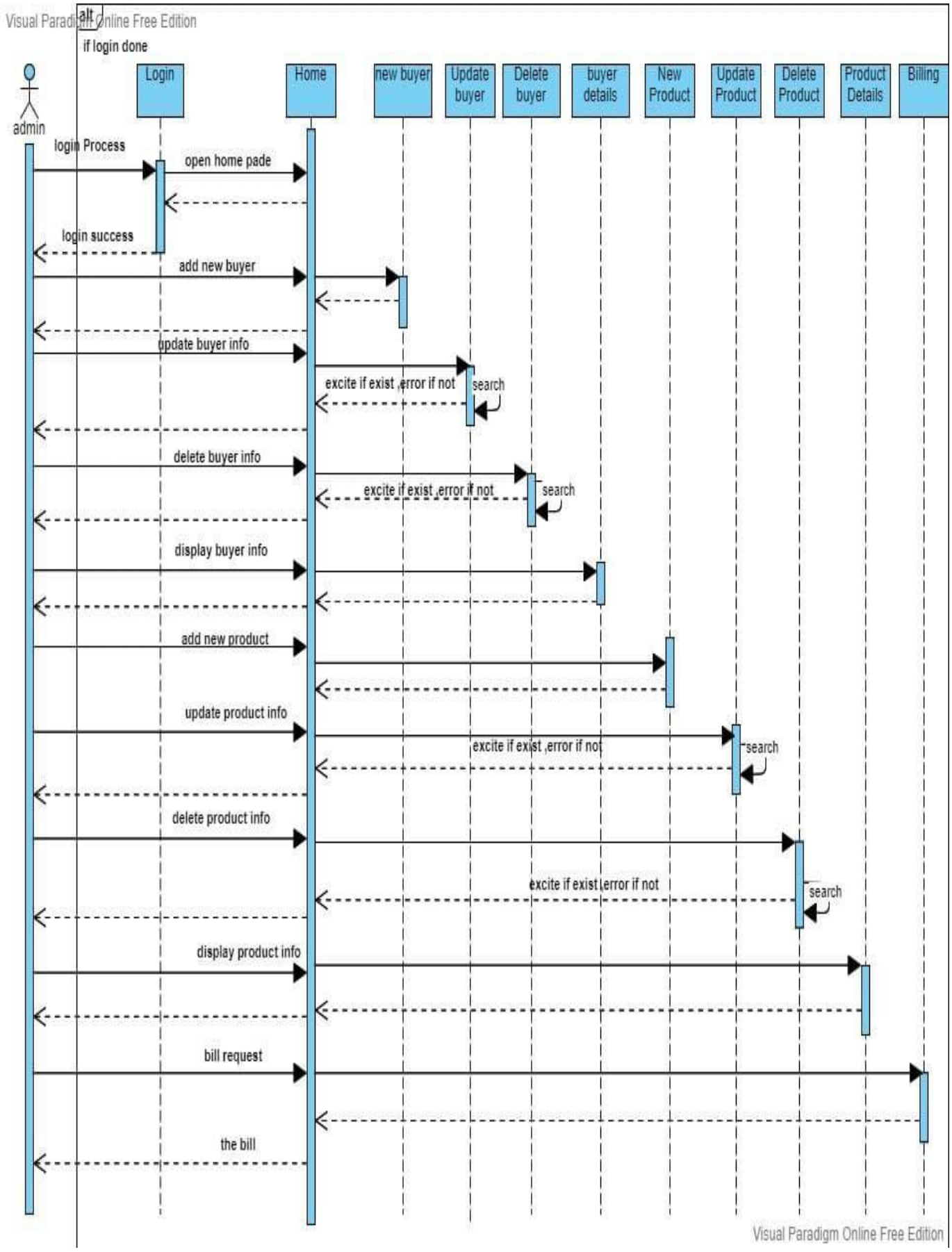
6. System Modeling

Class Diagram:



Sequence Diagram:




Visual Paradigm Online Free Edition
alt
if login done
admin
Login
Home
new buyer
Update buyer
Delete buyer
buyer details
New Product
Update Product
Delete Product
Product Details
Billing
login Process
open home pade
login success
add new buyer
update buyer info
excite if exist ,error if not
search
delete buyer info
excite if exist ,error if not
search
display buyer info
add new product
update product info
excite if exist ,error if not
search
delete product info
excite if exist ,error if not
search
display product info
bill request
the bill
Visual Paradigm Online Free Edition

7. Testing Report

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it's performing well and as per the specifications. While determining the test coverage the test cases should be designed well with maximum possibilities of finding the errors or bugs. The test cases should be very effective. This objective can be measured by the number of defects reported per test cases. Higher the number of the defects reported the more effective are the test cases.

Tests are classified into 2 categories 1) Black Box and 2) White Box.

In each test case:

- We are defining the number of test cases needed.
- Test the validity of the value of input of the function.
- Test if the data is (deleted/added/updated).
- Test of the function followed the Query or not.

1) Black Box Testing

- IT depends on inputs and outputs, with no knowledge of its implementation.
- It is used to ensure that the program does what it is supposed to do.

Test case ID	Test Case Description	Expected	Acc	Pass/Fail
TCL1	No Input	Not Login	Same as Expected	Pass
TCL2	Invalid pass Invalid User	Not Login	Same as Expected	Pass
TCL3	valid pass Invalid User	Not Login	Same as Expected	Pass
TCL4	Invalid pass valid User	Not Login	Same as Expected	Pass
TCL5	valid pass valid User	Login	Same as Expected	Pass
TCAC1	No Input	Not ADD	Same as Expected	Pass
TCAC2	All Input without ID	Not ADD	Same as Expected	Pass
TCAC3	All Input	ADD	Same as Expected	Pass
TCUC1	No Input	No result	Same as Expected	Pass
TCUC2	Update all input without ID	Same ID with new Data	Same as Expected	Pass
TCUC3	Update all input	Error	Same as Expected	Pass
TCDC1	No Input	No result	Same as Expected	Pass
TCDC2	Delete user with ID	Deleted	Same as Expected	Pass

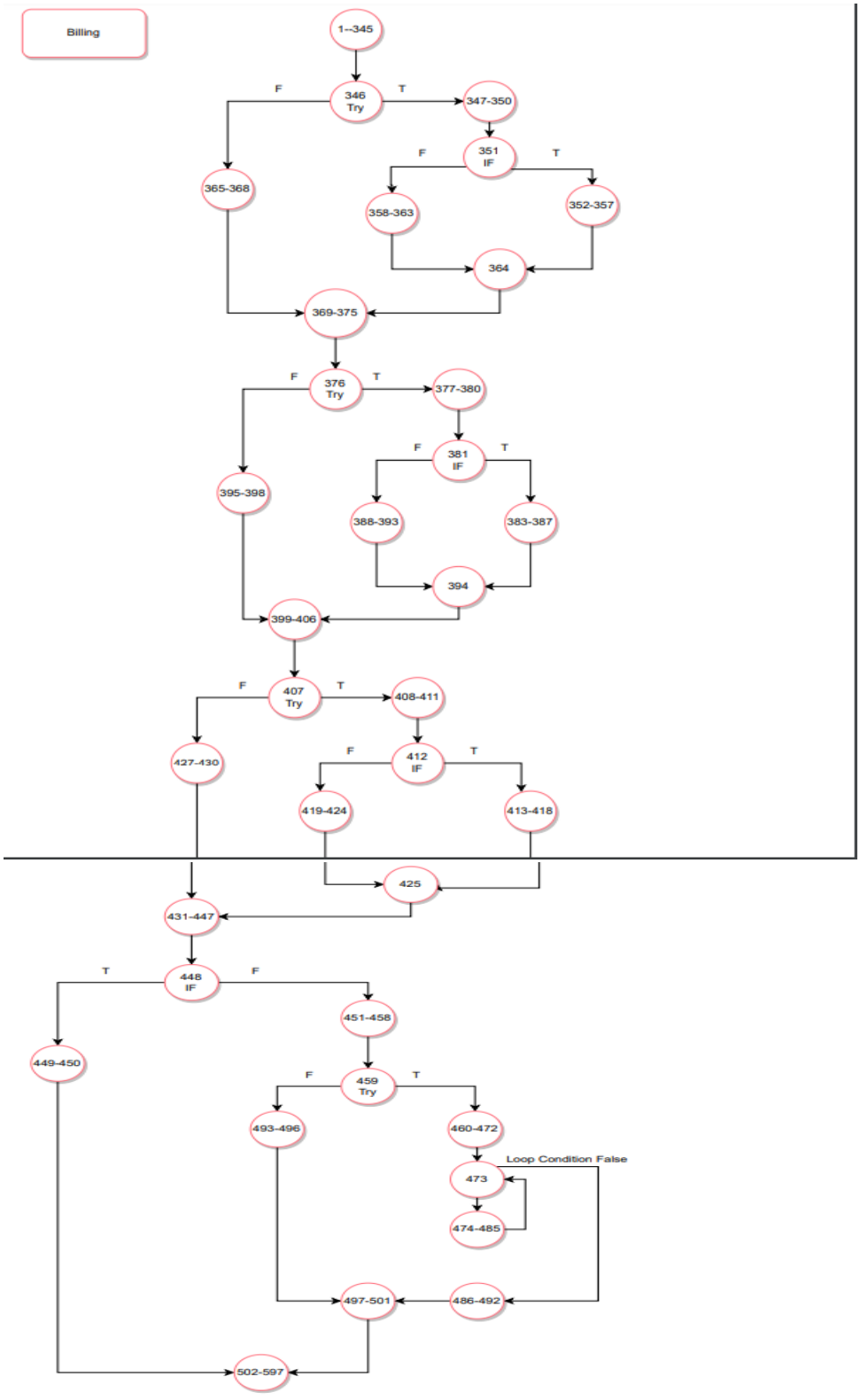
TCAP1	No Input	Not ADD	Same as Expected	Pass
TCAP2	All Input	ADD	Same as Expected	Pass
TCUP1	No Input	No result	Same as Expected	Pass
TCUP2	Update all input without ID	Same ID with new Data	Same as Expected	Pass
TCUP3	Update all input	Error	Same as Expected	Pass
TCDP1	No Input	No result	Same as Expected	Pass
TCDP2	Delete Product with ID	Deleted	Same as Expected	Pass

2) White Box Testing

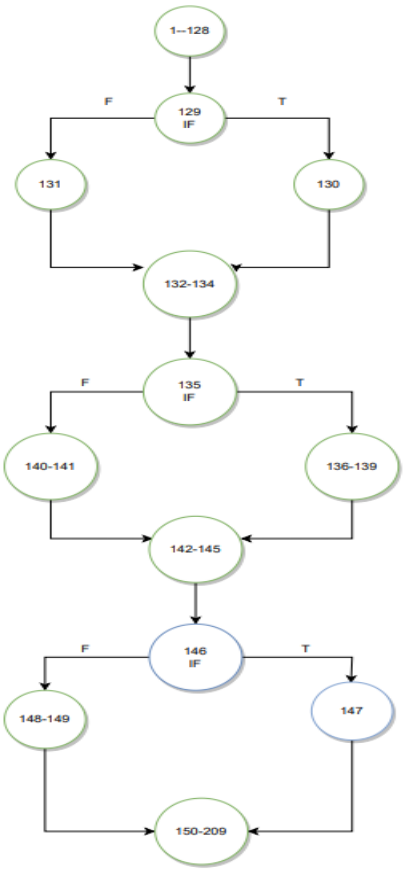
- It derives test cases based on the program's logic.
- White box testing goal is to ensure that all statements and conditions have been executed at least once.
- Its goal is to determine whether all components of the software do what they are designed for and error-free.
- It is primarily used during unit testing and usually performed by the engineer who wrote the code.

CFG:

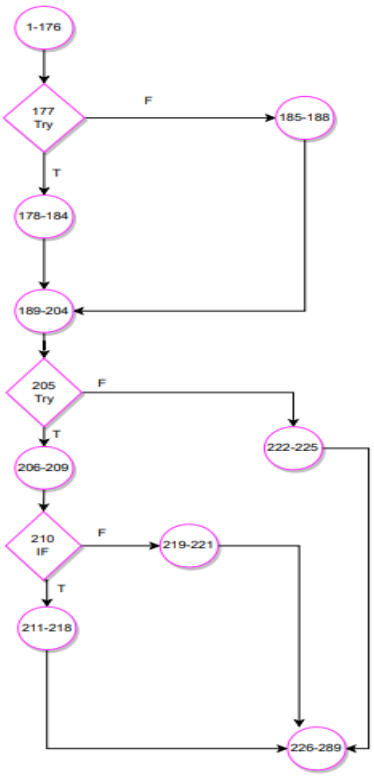
- A representation, using graph notation, of all paths that might be traversed through a program during its execution.
- It's useful when computing different coverage criteria of test cases

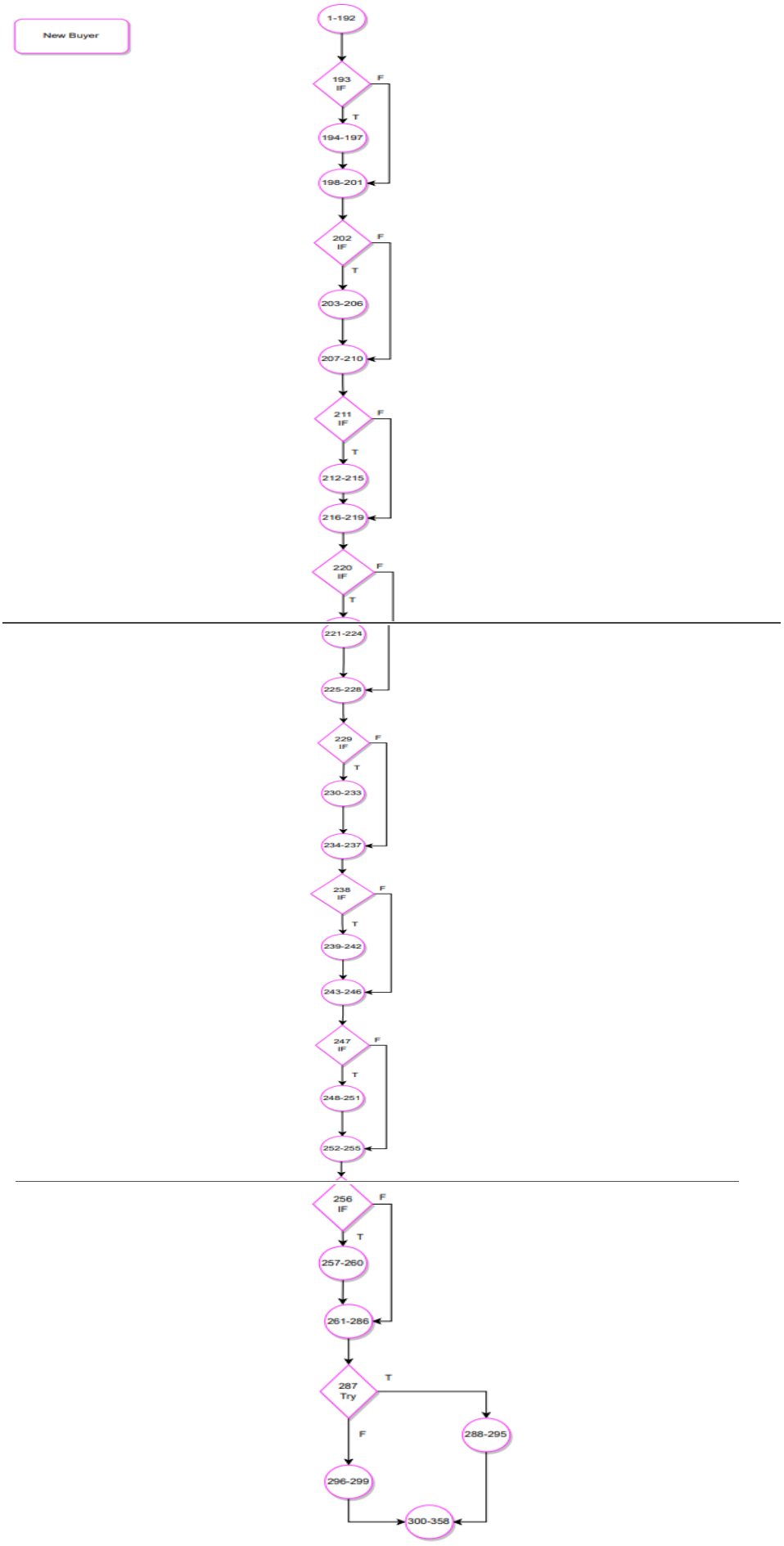

Billing
1--345
346
Try
F
T
347-350
351
IF
F
T
365-368
358-363
352-357
364
369-375
376
Try
F
T
377-380
381
IF
F
T
395-398
388-393
383-387
394
399-406
407
Try
F
T
408-411
427-430
412
IF
F
T
419-424
413-418
425
431-447
448
IF
T
F
451-458
449-450
459
Try
F
T
493-496
460-472
Loop Condition False
473
474-485
497-501
486-492
502-597

Log in

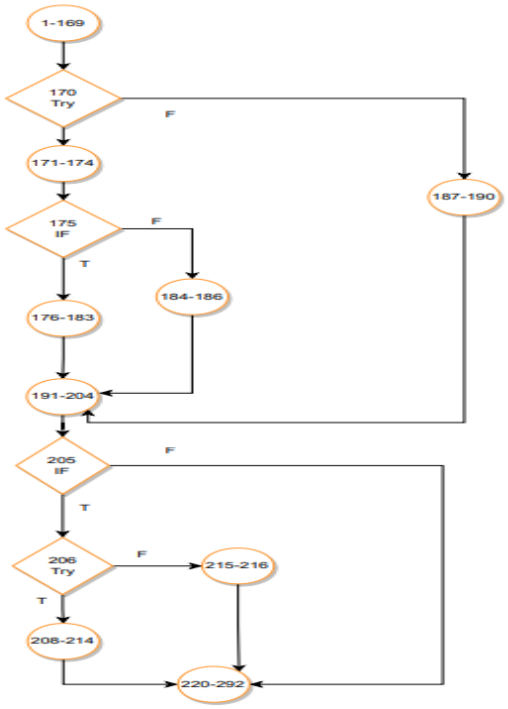


Update Buyer

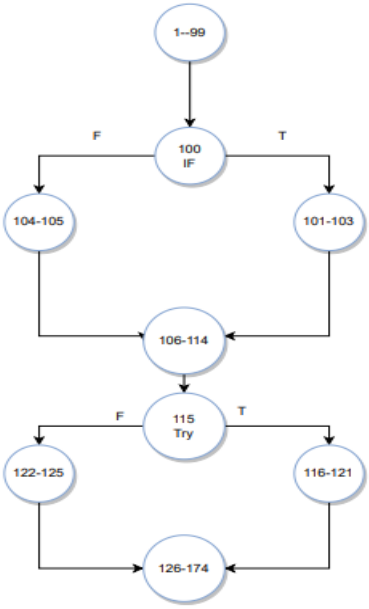


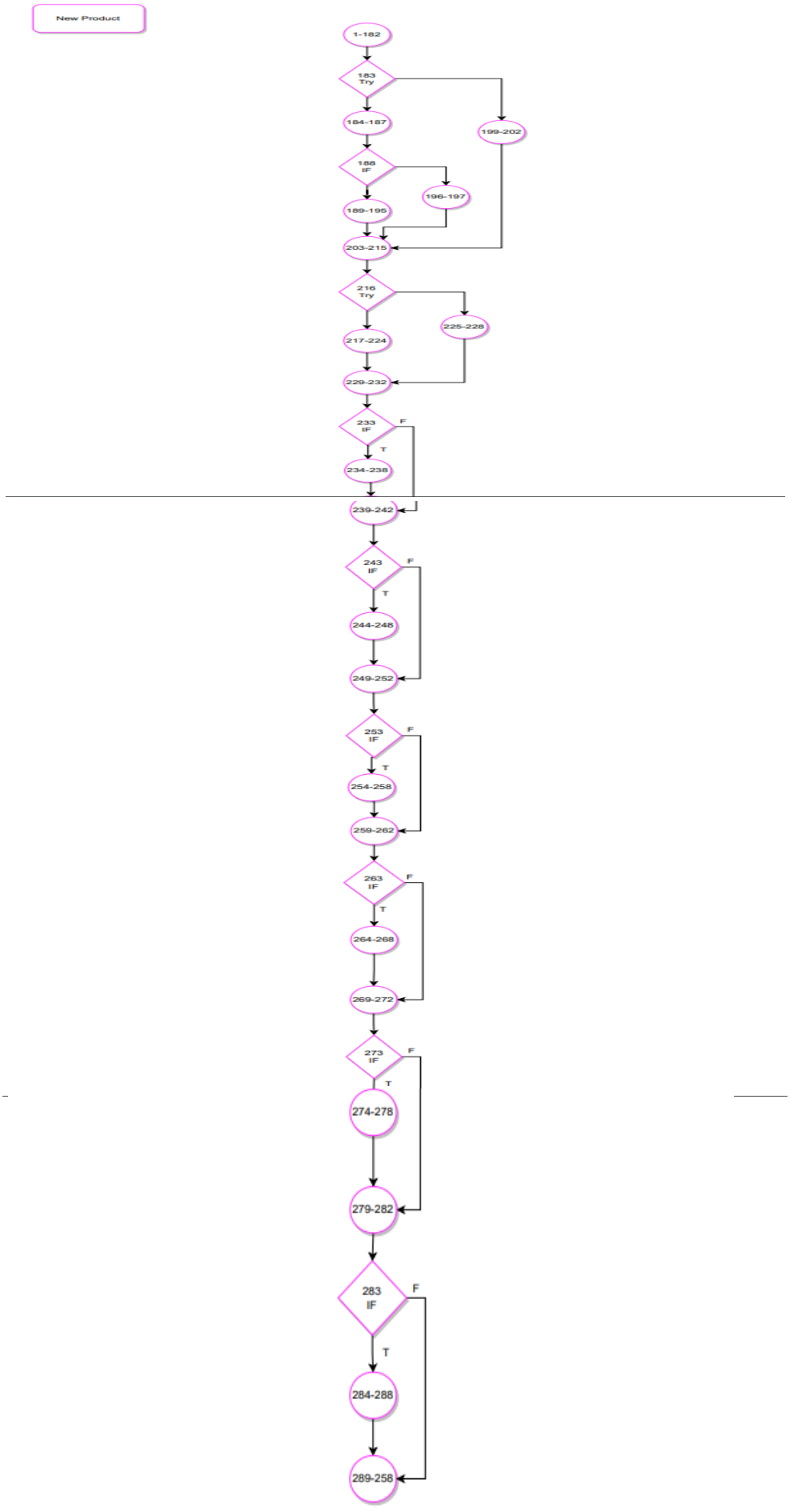

New Buyer
1-192
193 IF
F
T
194-197
198-201
202 IF
F
T
203-206
207-210
211 IF
F
T
212-215
216-219
220 IF
F
T
221-224
225-228
229 IF
F
T
230-233
234-237
238 IF
F
T
239-242
243-246
247 IF
F
T
248-251
252-255
256 IF
F
T
257-260
261-286
287 Try
T
F
288-295
296-299
300-358

Delete Buyer

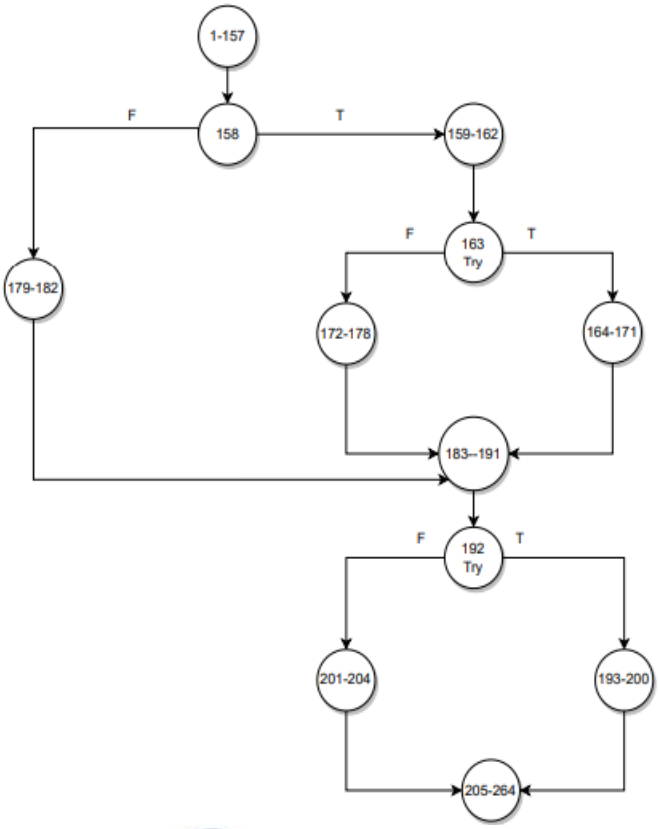


Buyer Details

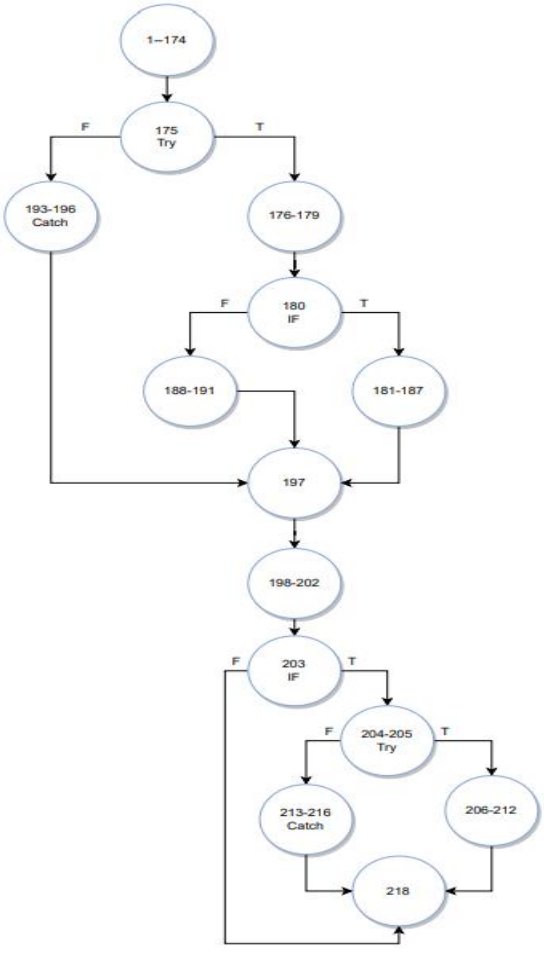


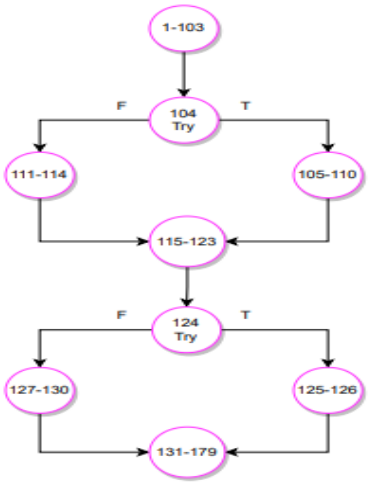

New Product
1-182
183
Try
184-187
199-202
188
IF
196-197
189-195
203-215
216
Try
225-228
217-224
229-232
233
IF
F
T
234-238
239-242
243
IF
F
T
244-248
249-252
253
IF
F
T
254-258
259-262
263
IF
F
T
264-268
269-272
273
IF
F
T
274-278
279-282
283
IF
F
T
284-288
289-258

Update Product



Delete Product





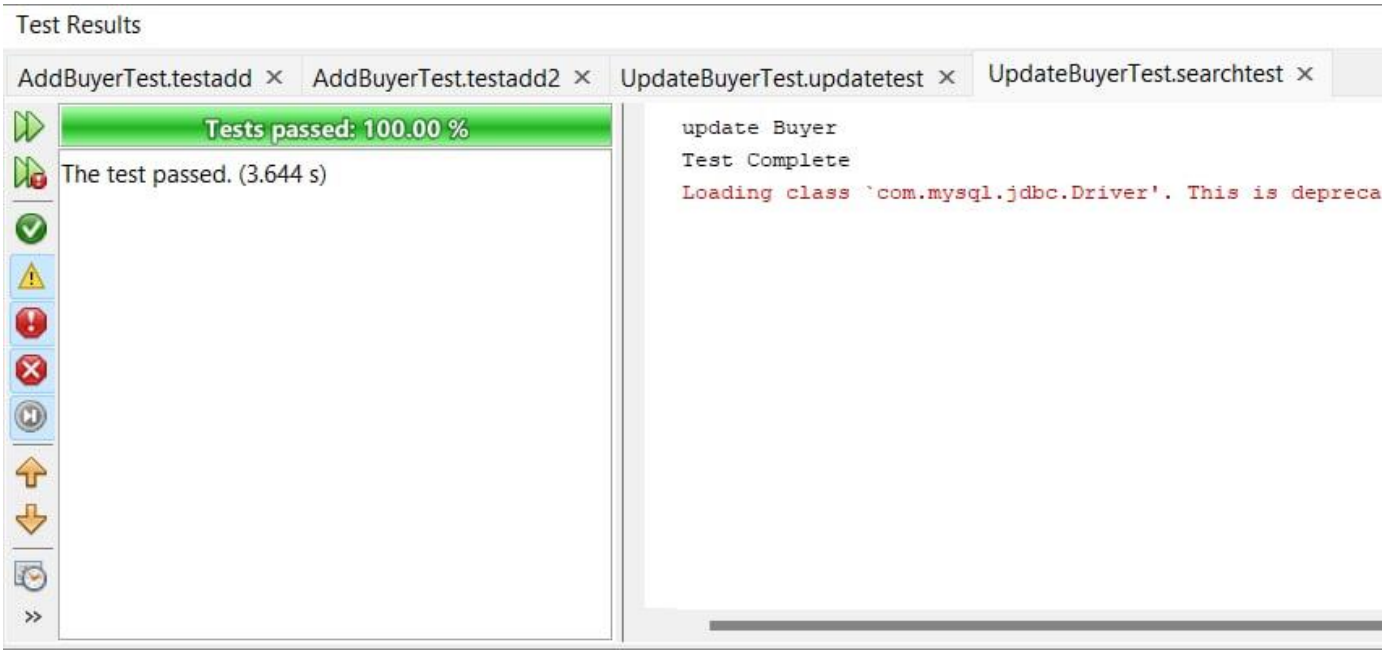
8.Junit:

We are using a search function on which our test functions depend.

Search function:

```
////////////////////////////////////////////////////////////////////////////////////////////////
///////To test the Search button
public int searchtest (int id){
    String bid = Integer.toString(id);
    try
    {
        Connection con=ConnectionProvider.getcon();
        Statement st=con.createStatement();
        ResultSet rs=st.executeQuery("select * from buyer where contactNo='"+bid+"'");
        if(rs.next())
        {
            String name = rs.getString(1);
            String contactNo = rs.getString(2);
            String phoneNo = rs.getString(3);
            String address = rs.getString(4);
            String gender =rs.getString(5);
            JOptionPane.showMessageDialog(null, "This Buyer does Exist");
            return 1;
        }
        else
        {JOptionPane.showMessageDialog(null, "This Buyer does not Exist"); return -1;}
    }
    catch(Exception e)
    {
        JOptionPane.showMessageDialog(null, e);
        return 0 ;
    }
}
```

The output of search function test:



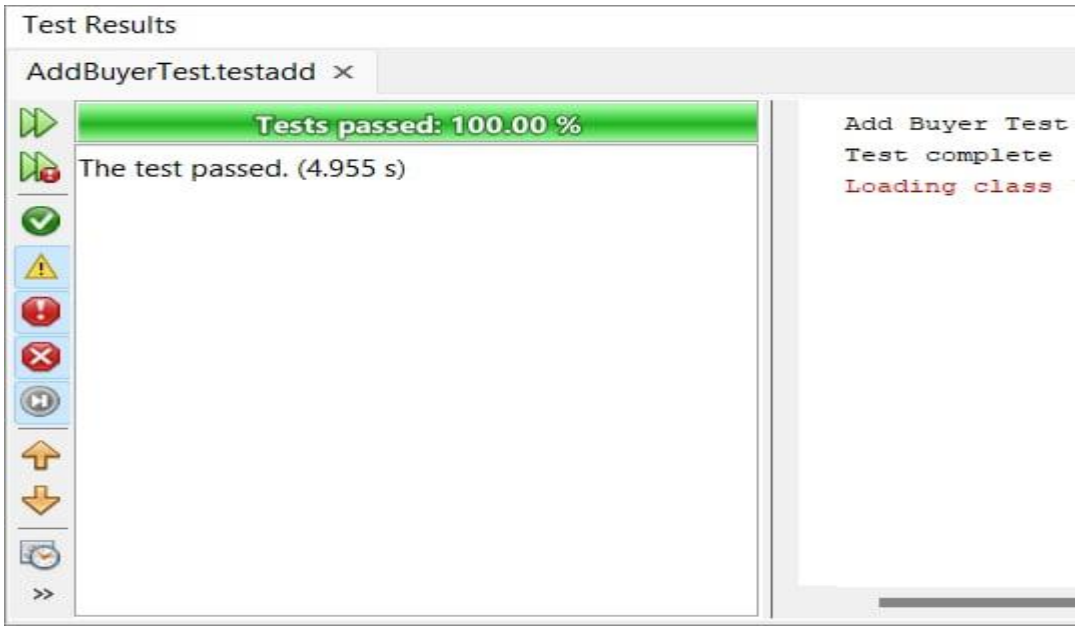
Add buyer function:

```
////////////////////////////////////////////////////////////////////////////////////////////
///////To test the ADD button
public int addtest(String n,int id,int phoneno,String a,String g){
    String name = n;
    String contactNo = String.valueOf(id);
    String phoneNo = String.valueOf(phoneno);
    String address = a;
    String gender = g;
    try
    {
        Connection con = ConnectionProvider.getcon();
        Statement st = con.createStatement();
        st.executeUpdate("Insert into buyer values ('"+name+"', '"+contactNo+"', '"+phoneNo+"', '"+address+"', '"+gender+"' )");
        JOptionPane.showMessageDialog(null, "successfully Added");
        return 1;
    }
    catch (Exception e)
    {
        JOptionPane.showMessageDialog(null, "This Buyer is already exists");
        return 0;
    }
}
////////////////////////////////////////////////////////////////////////////////////////////
```

Add buyer Test functions:

```
@BeforeClass
public static void setUpClass() {
    System.out.println("Add Buyer Test");
}

@AfterClass
public static void tearDownClass() {
    System.out.println("Test complete");
}

@Before
public void setUp() {
}

@After
public void tearDown() {
}

@Test
public void testadd ()
{
    newBuyer obj = new newBuyer();
    int res = obj.addtest("esraa",1010,55889,"haram","female");
    int exp = 1;
    assertEquals(res, exp);
}

@Test
public void testadd2 ()
{
    newBuyer obj = new newBuyer();
    int res = obj.addtest("esraa", 1000, 55889, "haram", "female");
    int exp = 0;
    assertEquals(res, exp);
}
```

The output:



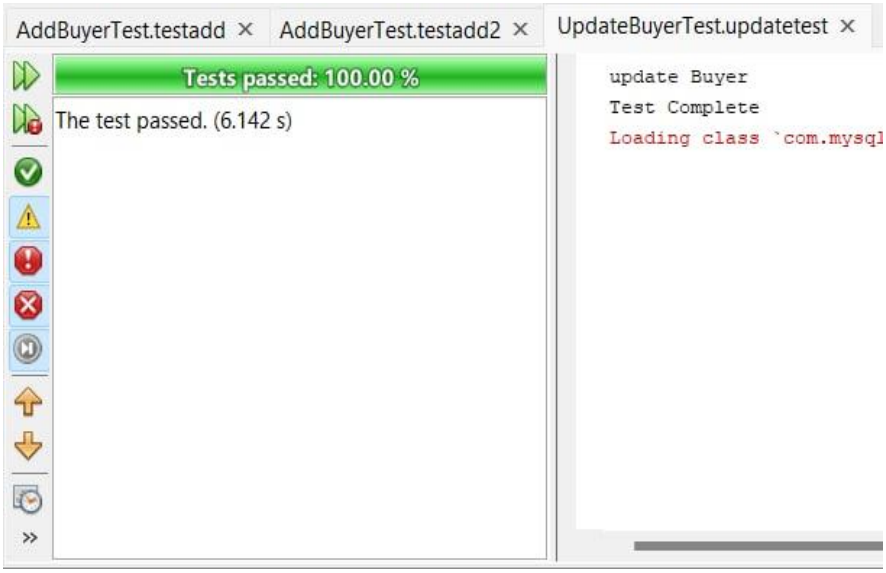
Update buyer function:

```
///////To test the Update button
public int updatetest(int x,String name){
    String id=Integer.toString(x);
    String n = name;
    if(searchtest(x)==1)
    {
    try{
        Connection con=ConnectionProvider.getcon();
        Statement st=con.createStatement();
        st.executeUpdate("update buyer set name ='"+n+"' where contactNo='"+id+"'");
        JOptionPane.showMessageDialog(null, "Successfully Updated");
        return 1;
       }
    catch(Exception e)
        {
            return 0;
        }
    }
    else
    {return -1;}
}
//////////////////////////////////////////////////////////////////////////////////////////
```

Update buyer Test functions:

```
@Test
public void updatetest()
{
   updateBuyer obj = new updateBuyer();
   int res = obj.updatetest(1010, "omar");
   int exp = 1;
   assertEquals(exp, res);
}

@Test
public void updatetest1()
{
   updateBuyer obj = new updateBuyer();
   int res = obj.updatetest(10, "omar");
   int exp = -1;
   assertEquals(exp, res);
}

@Test
public void searchtest()
{
   updateBuyer obj = new updateBuyer();
   int res = obj.searchtest(6);
   int exp = 1;
   assertEquals(exp, res);
}

@Test
public void searchtest1()
{
   updateBuyer obj = new updateBuyer();
   int res = obj.searchtest(10);
   int exp = -1;
   assertEquals(exp, res);
}
```

The output:



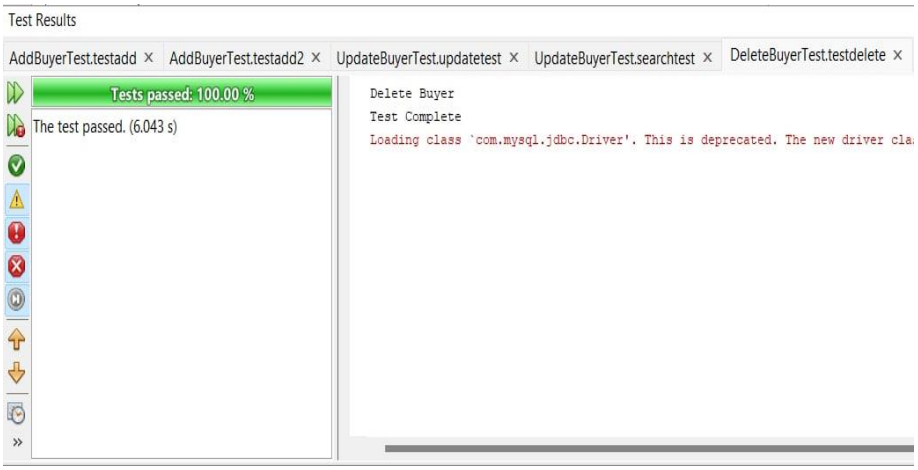
Delete buyer Test functions:

```
///////To test the Delete button
public int deletetest (int id){
    String contactNo=Integer.toString(id);
    if(searchtest(id)==1){
        try
        {
            Connection con=ConnectionProvider.getcon();
            Statement at=con.createStatement();
            at.executeUpdate("delete from buyer where contactNo ='"+contactNo+"'");
            JOptionPane.showMessageDialog(null, "success");
            return 1;
        }
        catch(Exception e)
            {return 0;}
    }
    else
    {return -1;}
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Update buyer Test functions:

```
@Test
public void testdelete(){
    DeleteBuyer obj = new DeleteBuyer();
    int res = obj.deletetest(5);
    int exp = 1 ;
    assertEquals(exp ,res);
}

@Test
public void testdelete2(){
    DeleteBuyer obj = new DeleteBuyer();
    int res = obj.deletetest(10);
    int exp = -1 ;
    assertEquals(exp ,res);
}
```

The output:



9. Future Work

What is your future plans for this system? How will you improve it?

Future plans is already added to Sequence diagram and ER diagram where we made it available for an online buyer to access our application to place an order to buy products. But first we need to

Future plans for this system is to make it available on many different operating systems like Linux and IOS.