REPORT

ON

JAVA AWT BASED
AUTOMATED FOOD DELIVERY SYSTEM
[E.g. ZOMATO]

(SQL CONNECTIVITY DATABASE MANAGEMNET)

A

Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

B.VIVEK <1602-18-737-310>

Department of Information Technology

Vasavi College of Engineering (Autonomous)

Ibrahimbagh, Hyderabad-31

2020

Vasavi College of Engineering (Autonomous)

Ibrahimbagh, Hyderabad-31

BONAFIED CERTIFICATE

This is to certify that the project report entitled "**AUTOMATED FOOD DELIVERY SYSTEM [E.g. ZOMATO]**" has been carried out by MR.B.VIVEK of bearing ROLL NO: **1602-18-737-310** who carried out this project under my suspension in the IV semester during the academic year 2019-2020.

We also certify that this project is sufficiently warranted for the submission in the partially fulfilment of requirements for award of the degree of **"BACHELOR OF ENGINEERING"** IN **INFORMATION TECHNOILOGY.**

Signature of the Examiner

B.LEELAVATHY

Lecturer

Department of Information Technology.

ABSTRACT

A project on “AUTOMATED FOOD DELIVERY SYSTEM” like example: ZOMATO.

A good restaurant means a restaurant that provides a good services, delicious food as well as promising comfort and a hygienic place to have a meal. AQ waiter plays an important role in order to satisfy the customer with a good services. Waiter usually have the flaw of tend to make some mistakes when taking the customer’s order. This will effects the restaurant’s reputation and customer’s satisfaction. Hence, with the existence of smart waiters system, this problem can be avoided as the customers can make their order from their own seats via touch screen LCD’s which are available on each table in the restaurant. As we are living in the era of high-tech devices, ordering food from a restaurant should also be brought to a whole new level. Going through the menu and ordering food from an LCD screen will be something common among restaurants and acceptable to the society. The server will store transaction_details, customer, food and other information in database. Automated Food Delivery System also can view the most high rated food in the system and automatically update it daily.

INTRODUCTION

➢ REQUIREMENTS FOR AUTOMATED FOOD DELIVERY SYSTEM [E.G. ZOMATO] :

List of tables :

- Customer
- Orders
- Address
- Food
- Delivery
- Vehicle

List of attributes with their domain types:

ENTITY	ATTRIBUTES	DOMAIN
Customer	1. C_name 2. C_id 3. Phone number 4. Email-ID	Varchar2(20) Number(5) Number(10) Varchar2(20)
Orders	1. O_id 2. Amount 3. C_id	Number(5) Number(10) Number(5)
Address	1. Add_id 2. C_id 3. Door_no 4. Place 5. Postal_code	Number(5) Number(5) Number(5) Varchar(10) Number(5)
Food	1. O_id 2. F_id 3. Costs 4. F_name	Number(5) Number(5) Number(5) Varchar(10)
Delivery	1. O_id 2. D_id 3. Add_id 4. Emp_name	Number(5) Number(5) Varchar(10) Varchar(10)

Vehicle	1. Add_id 2. V_id 3. V_type 4. V_no	Number(5) Number(5) Varchar(10) Number(5)

➢ SPECIFIC GOAL OF THE PROJECT:

The main goal of the Automated Food Delivery System is to manage the details of item category, Food, Delivery Address, Order. It manages all the information about item category, Customer, Item category. The purpose of the project is to build an application program to reduce the manual work for managing the Item category, Food, Customer, Delivery Address. It track all the details about the Delivery Address, Orders .

SQL particular Automated Food Delivery System stores the details of Address Details, Order details, Food details can be executed.

➢ Architecture and technology used:

SQL Plus is the most basic Oracle Database utility with a basic command-line interface, commonly used by users, administrators and programmers.

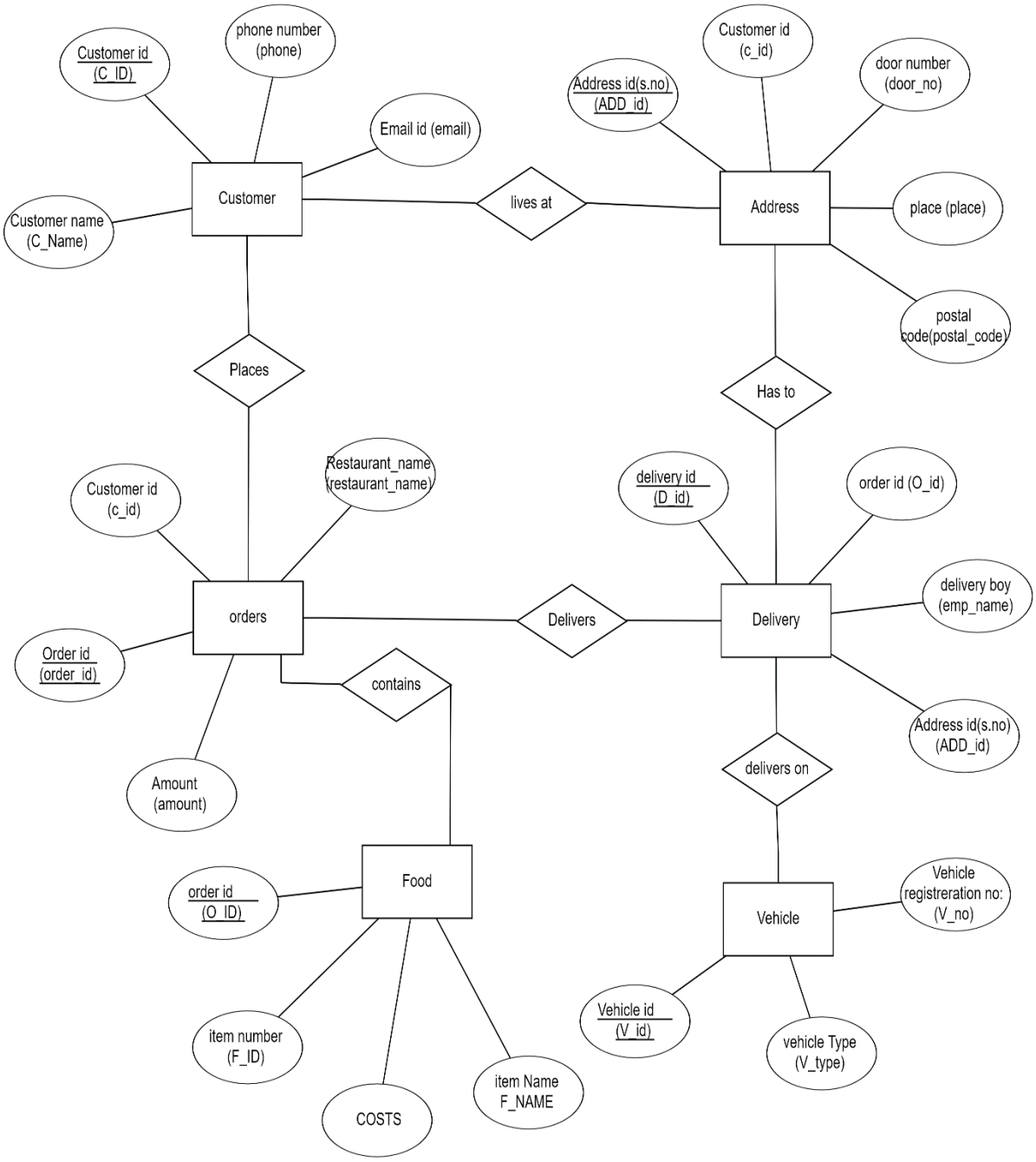
The interface of SQL Plus is used for creating the database. DDL and DML commands are implemented for operations being executed. The details of various Online MOOC's provider, courses, student, assignments, and results are stored in the form of tables in the database.

Eclipse is an integrated development environment(IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Erlang, JavaScripts etc.

The front end application code is written in "**Java**" using Eclipse. The portal for front end application is designed through Eclipse, runs and has the capacity to connect with the database which has data inserted using SQL.

➢ DESIGN:

i) ER DIAGRAM:



B.VIVEK.
1602-18-737-310

MAPPING CARDINALITIES AND PARTICIPATION CONSTRAINTS:

- Customers can order from any restaurant and any food items. So, it is a one to many mapping. As it is not necessary should only order one food item.
- Customer can orders food,it is many to many mapping as any number of customers can order any number of items from a restaurant.
- Customer pays bill payment, It is one to one customer should pay bills as per the order of the food and the restaurant.
- Customers receives the delivery if he/she pays the bill, it is one to one mapping as customer gets their orders.

DDL Commands:

SQL> create table customer(c_name varchar(20),c_id number(20) primary key,phone number(10),email varchar(20));

Table created.

SQL> create table orders(o_id number(20) primary key,amount number(10),c_id number(20));

Table created.

SQL> alter table orders add foreign key (c_id) references customer;

Table altered.

SQL> alter table orders add(restaurant_name varchar(40));

Table altered.

SQL> create table food(o_id number(20),f_id number(20),costs number(10),f_name varchar(20),foreign key(o_id) references orders);

Table created.

SQL> alter table food add primary key(f_id);

Table altered.

SQL> create table address(c_id number(20),door_no number(10),place varchar(100),postal_code number(10),foreign key(c_id) references customer);

Table created.

SQL> alter table address add(add_id number(10));

Table altered.

SQL> alter table address add primary key(add_id);

Table altered.

SQL> create table delivery(o_id number(20),d_id number(20) primary key,add_id number(10),emp_name varchar(20),foreign key(add_id) references address);

Table created.

SQL> alter table delivery add foreign key(o_id) references orders;

Table altered.

SQL> create table vehicle(add_id number(10),v_id number(10) primary key,v_type varchar(10),v_no number(20),foreign key(add_id)references address);

Table created.

SQL> select * from tab;

```
TNAME                          TABTYPE  CLUSTERID

------------------------------ ------- ----------
```

```
ADDRESS                    TABLE
CUSTOMER                   TABLE
DELIVERY                   TABLE
FOOD                       TABLE
ORDERS                     TABLE
VEHICLE                    TABLE

6 rows selected.


SQL> desc customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_NAME                                             VARCHAR2(20)
 C_ID                                      NOT NULL NUMBER(20)
 PHONE                                              NUMBER(10)
 EMAIL                                              VARCHAR2(20)

SQL> desc address;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                               NUMBER(20)
```

```
DOOR_NO                                   NUMBER(10)
PLACE                                     VARCHAR2(100)
POSTAL_CODE                               NUMBER(10)
ADD_ID                                    NOT NULL NUMBER(10)
```

```
SQL> desc orders;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 O_ID                                      NOT NULL NUMBER(20)
 AMOUNT                                    NUMBER(10)
 C_ID                                      NUMBER(20)
 RESTAURANT_NAME                           VARCHAR2(40)
```

```
SQL> desc delivery;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 O_ID                                      NUMBER(20)
 D_ID                                      NOT NULL NUMBER(20)
 ADD_ID                                    NUMBER(10)
 EMP_NAME                                  VARCHAR2(20)
```

```
SQL> desc food;
```

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
O_ID                                                NUMBER(20)
F_ID                                      NOT NULL NUMBER(20)
COSTS                                               NUMBER(10)
F_NAME                                              VARCHAR2(20)
```

```
SQL> desc vehicle;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ADD_ID                                              NUMBER(10)
 V_ID                                      NOT NULL NUMBER(10)
 V_TYPE                                              VARCHAR2(10)
 V_NO                                                NUMBER(20)
```

<u>DML COMMANDS</u>:

```
SQL> insert into customer
values('&c_name',&c_id,&phone,'&email');
```

```
Enter value for c_name: vivek

Enter value for c_id: 01

Enter value for phone: 6305314935

Enter value for email: vivek.basa@gmail.com

old   1: insert into customer
values('&c_name',&c_id,&phone,'&email')

new   1: insert into customer
values('vivek',01,6305314935,'vivek.basa@gmail.com')


1 row created.



SQL> /

Enter value for c_name: rohith

Enter value for c_id: 02

Enter value for phone: 7995702445

Enter value for email: sairohith@gmail.com

old   1: insert into customer
values('&c_name',&c_id,&phone,'&email')

new   1: insert into customer
values('rohith',02,7995702445,'sairohith@gmail.com')


1 row created.
```

```
SQL> /

SQL> insert into customer
values('&c_name',&c_id,&phone,'&email');

Enter value for c_name: ram

Enter value for c_id: 3

Enter value for phone: 8977652535

Enter value for email: ram1234@gmail.com

old   1: insert into customer
values('&c_name',&c_id,&phone,'&email')

new   1: insert into customer
values('ram',3,8977652535,'ram1234@gmail.com')


1 row created.


SQL> select * from customer;


C_NAME                 C_ID      PHONE EMAIL

------------------- ---------- ---------- --------------------

vivek                       1 6305314935 vivek.basa@gmail.com

rohith                      2 7995702445 sairohith@gmail.com

ram                         3 8977652535 ram1234@gmail.com
```

```
SQL> insert into address
values(&c_id,&door_no,'&place',&postal_code,&add_id);

Enter value for c_id: 01

Enter value for door_no: 8-3-72

Enter value for place: karmanghat,l.b.nagar

Enter value for postal_code: 500097

Enter value for add_id: 01

old   1: insert into address
values(&c_id,&door_no,'&place',&postal_code,&add_id)

new   1: insert into address values(01,8-3-
72,'karmanghat,l.b.nagar',500097,01)


1 row created.

SQL> /

Enter value for c_id: 2

Enter value for door_no: 7-6-123

Enter value for place: mehdipatnam, near pillar no: 19

Enter value for postal_code: 500079

Enter value for add_id: 2

old   1: insert into address
values(&c_id,&door_no,'&place',&postal_code,&add_id)

new   1: insert into address values(2,7-6-123,'mehdipatnam,
near pillar no: 19',500079,2)
```

1 row created.

SQL> /

Enter value for c_id: 3

Enter value for door_no: 7-9-5

Enter value for place: nallakunta

Enter value for postal_code: 500098

Enter value for add_id: 3

old 1: insert into address values(&c_id,&door_no,'&place',&postal_code,&add_id)

new 1: insert into address values(3,7-9-5,'nallakunta',500098,3)

1 row created.

SQL> select * from address;

C_ID DOOR_NO

---------- ----------

PLACE

POSTAL_CODE ADD_ID

--- ---

1 -67

karmanghat,l.b.nagar

500097 1

2 -122

mehdipatnam, near pillar no: 19

500079 2

C_ID DOOR_NO

--- ---

PLACE

POSTAL_CODE ADD_ID

--- ---

3 -7

nallakunta

500098 3

```
SQL> select * from address;

     C_ID    DOOR_NO
---------- ----------
PLACE
--------------------------------------------------------------------------------
POSTAL_CODE     ADD_ID
----------- ----------
        1        -67
karmanghat,l.b.nagar
    500097         1

        2       -122
mehdipatnam, near pillar no: 19
    500079         2

     C_ID    DOOR_NO
---------- ----------
PLACE
--------------------------------------------------------------------------------
POSTAL_CODE     ADD_ID
----------- ----------
```

```
        3        -7
nallakunta
    500098         3


SQL> insert into orders
values(&o_id,&amount,&c_id,'&restaurant_name');
Enter value for o_id: 1
Enter value for amount: 500
Enter value for c_id: 1
Enter value for restaurant_name: bawarchi
old   1: insert into orders
values(&o_id,&amount,&c_id,'&restaurant_name')
new   1: insert into orders values(1,500,1,'bawarchi')


1 row created.


SQL> /
Enter value for o_id: 2
Enter value for amount: 850
Enter value for c_id: 2
Enter value for restaurant_name: shah ghouse
```

```
old   1: insert into orders
values(&o_id,&amount,&c_id,'&restaurant_name')
new   1: insert into orders values(2,850,2,'shah ghouse')


1 row created.


SQL> /
Enter value for o_id: 3
Enter value for amount: 600
Enter value for c_id: 3
Enter value for restaurant_name: mehfil
old   1: insert into orders
values(&o_id,&amount,&c_id,'&restaurant_name')
new   1: insert into orders values(3,600,3,'mehfil')


1 row created.


SQL> select * from orders;


     O_ID     AMOUNT       C_ID RESTAURANT_NAME
---------- ---------- ---------- ----------------------------------------
         1        500          1 bawarchi
```

2	850	2 shah ghouse
3	600	3 mehfil

```
SQL> insert into food values(&o_id,&f_id,&costs,'&f_item');
Enter value for o_id: 1
Enter value for f_id: 3.4
Enter value for costs: 450
Enter value for f_item: biryani
old   1: insert into food values(&o_id,&f_id,&costs,'&f_item')
new   1: insert into food values(1,3.4,450,'biryani')

1 row created.

SQL> /
Enter value for o_id: 2
Enter value for f_id: 5.4
Enter value for costs: 60
Enter value for f_item: idli,dosa
old   1: insert into food values(&o_id,&f_id,&costs,'&f_item')
new   1: insert into food values(2,5.4,60,'idli,dosa')
```

```
1 row created.

SQL> /
Enter value for o_id: 3
Enter value for f_id: 1.6
Enter value for costs: 200
Enter value for f_item: french fries
old   1: insert into food values(&o_id,&f_id,&costs,'&f_item')
new   1: insert into food values(3,1.6,200,'french fries')

1 row created.

SQL> select * from food;

     O_ID       F_ID      COSTS F_NAME
---------- ---------- ---------- --------------------
         1          3        450 biryani
         2          5         60 idli,dosa
         3          2        200 french fries
SQL> insert into delivery
values(&o_id,&d_id,&add_id,'&emp_name');
Enter value for o_id: 1
```

```
Enter value for d_id: 1

Enter value for add_id: 1

Enter value for emp_name: sai

old   1: insert into delivery
values(&o_id,&d_id,&add_id,'&emp_name')

new   1: insert into delivery values(1,1,1,'sai')


1 row created.


SQL> /

Enter value for o_id: 2

Enter value for d_id: 4

Enter value for add_id: 3

Enter value for emp_name: ram

old   1: insert into delivery
values(&o_id,&d_id,&add_id,'&emp_name')

new   1: insert into delivery values(2,4,3,'ram')


1 row created.

SQL>/

Enter value for o_id: 3

Enter value for d_id: 3
```

```
Enter value for add_id: 3

Enter value for emp_name: dj

old   1: insert into delivery
values(&o_id,&d_id,&add_id,'&emp_name')

new   1: insert into delivery values(3,3,3,'dj')


1 row created.

SQL> select * from delivery;


     O_ID       D_ID     ADD_ID EMP_NAME

---------- ---------- ---------- --------------------

         1          1          1 sai

         2          4          3 ram

         3          3          3 dj


SQL>  insert into vehicle
values(&add_id,&v_id,'&v_type',&v_no);

Enter value for add_id: 1

Enter value for v_id: 2

Enter value for v_type: hereo

Enter value for v_no: 5311
```

```
old   1:  insert into vehicle
values(&add_id,&v_id,'&v_type',&v_no)
new   1:  insert into vehicle values(1,2,'hereo',5311)

1 row created.

SQL> /
Enter value for add_id: 2
Enter value for v_id: 1
Enter value for v_type: yamaha
Enter value for v_no: 0013
old   1:  insert into vehicle
values(&add_id,&v_id,'&v_type',&v_no)
new   1:  insert into vehicle values(2,1,'yamaha',0013)

1 row created.

SQL> /
Enter value for add_id: 3
Enter value for v_id: 3
Enter value for v_type: fz
Enter value for v_no: 5463
```

```
old   1:  insert into vehicle
values(&add_id,&v_id,'&v_type',&v_no)

new   1:  insert into vehicle values(3,3,'fz',5463)


1 row created.


SQL> select * from vehicle;


   ADD_ID      V_ID V_TYPE         V_NO
---------- ---------- ---------- ----------
         1          2 hereo          5311
         2          1 yamaha           13
         3          3 fz             5463
```

Implementation

- Front end programs:

FrontPage program:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
class FrontPage extends JFrame implements ActionListener
{
    /**
    *
    */
    private static final long serialVersionUID = 1L;
    String msg = "";
     Label ll;
     CardLayout cardLO;

     //Create Panels for each of the menu items, welcome screen panel and home
screen panel with CardLayout
     AddCustomer add;
     UpdateCustomer ups;
     DeleteCustomer dels;
     AddOrders addP;
     UpdateOrders upP;
     DeleteOrders delP;
     AddAddress addR;
     DeleteAddress delR;
     UpdateAddress upR;
     AddDelivery addS;
```

```
UpdateDelivery upS;
DeleteDelivery delS;
AddFood addf;
DeleteFood delf;
UpdateFood upf;
AddVehicle addv;
DeleteVehicle delv;
UpdateVehicle upv;
Panel home,welcome;

public FrontPage()
{
        cardLO = new CardLayout();

        //Create an empty home panel and set its layout to card layout
        home = new Panel();
        home.setLayout(cardLO);



        ll = new Label();
        ll.setAlignment(Label.CENTER);
        ll.setText("Welcome to AUTOMATED FOOD DELIVERY System
database");

        //Create welcome panel and add the label to it
        welcome = new Panel();
        welcome.add(ll);

        //create panels for each of our menu items and build them with
respective components
```

```
add=new AddCustomer();add.buildGUI();
addP = new AddOrders();addP.buildGUI();
ups = new UpdateCustomer();  ups.buildGUI();
dels = new DeleteCustomer();          dels.buildGUI();
upP = new UpdateOrders();upP.buildGUI();
delP=new DeleteOrders();delP.buildGUI();
addR=new AddAddress();addR.buildGUI();
delR=new DeleteAddress();delR.buildGUI();
upR=new UpdateAddress();upR.buildGUI();
addS=new AddDelivery();addS.buildGUI();
upS=new UpdateDelivery();upS.buildGUI();
delS=new DeleteDelivery();delS.buildGUI();
addf=new AddFood();addf.buildGUI();
delf=new DeleteFood();delf.buildGUI();
upf=new UpdateFood();upf.buildGUI();
addv=new AddVehicle();addv.buildGUI();
delv=new DeleteVehicle();delv.buildGUI();
upv=new UpdateVehicle();upv.buildGUI();
//add all the panels to the home panel which has a cardlayout
home.add(welcome, "Welcome");
home.add(add, "Add Customer");
home.add(ups, "Update Customer");
home.add(dels, "Delete Customer");
home.add(addP, "Add Orders");
home.add(upP,"Update Orders");
home.add(delP,"Delete Orders");
home.add(addR,"Add Address");
home.add(delR,"Delete Address");
home.add(upR,"Update Address");
```

```
home.add(addS,"Add Delivery");
home.add(upS,"Update Delivery");
home.add(delS,"Delete Delivery");
home.add(addf,"Add Food");
home.add(delf,"Delete Food");
home.add(upf,"Update Food");
home.add(addv,"Add Vehicle");
home.add(delv,"Delete Vehicle");
home.add(upv,"Update Vehicle");
// add home panel to main frame
add(home);

// create menu bar and add it to frame
MenuBar mbar = new MenuBar();
setMenuBar(mbar);

// create the menu items and add it to Menu
Menu Customer = new Menu("Customer ");
MenuItem item1, item2, item3;
Customer.add(item1 = new MenuItem("Add Customer"));
Customer.add(item2 = new MenuItem("View Customer"));
Customer.add(item3 = new MenuItem("Delete Customer"));
mbar.add(Customer);

Menu orders = new Menu("orders");
MenuItem item4, item5, item6;
orders.add(item4 = new MenuItem("Add Orders"));
orders.add(item5 = new MenuItem("View Orders"));
orders.add(item6 = new MenuItem("Delete Orders"));
```

```
mbar.add(orders);

Menu address = new Menu("address");
MenuItem item7, item8, item9;
address.add(item7 = new MenuItem("Add Address"));
address.add(item8 = new MenuItem("View Address"));
address.add(item9 = new MenuItem("Delete Address"));
mbar.add(address);

Menu food = new Menu("food");
MenuItem item10, item11, item12;
food.add(item10 = new MenuItem("Add Food"));
food.add(item11 = new MenuItem("View Food"));
food.add(item12 = new MenuItem("Delete Food"));
mbar.add(food);

Menu Delivery = new Menu("Delivery");
MenuItem item13, item14, item15;
Delivery.add(item13 = new MenuItem("Add Delivery"));
Delivery.add(item14 = new MenuItem("View Delivery"));
Delivery.add(item15 = new MenuItem("Delete Delivery"));
mbar.add(Delivery);

Menu vehicle = new Menu("vehicle");
MenuItem item16, item17, item18;
vehicle.add(item16 = new MenuItem("Add Vehicle"));
vehicle.add(item17 = new MenuItem("View Vehicle"));
vehicle.add(item18 = new MenuItem("Delete Vehicle"));
mbar.add(vehicle);
```

```
// register listeners
item1.addActionListener(this);
item2.addActionListener(this);
item3.addActionListener(this);
item4.addActionListener(this);
item5.addActionListener(this);
item6.addActionListener(this);
item7.addActionListener(this);
item8.addActionListener(this);
item9.addActionListener(this);
item10.addActionListener(this);
item11.addActionListener(this);
item12.addActionListener(this);
item13.addActionListener(this);
item14.addActionListener(this);
item15.addActionListener(this);
item16.addActionListener(this);
item17.addActionListener(this);
item18.addActionListener(this);
 // Anonymous inner class which extends WindowAdaptor to handle
the Window event: windowClosing
addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we)
        {
                quitApp();
        }
});
```

```
                //Frame properties
                setTitle("AUTOMATED FOOD DELIVERY System");
                setSize(500, 600);
                setVisible(true);

        }

        public void actionPerformed(ActionEvent ae)
        {
                String arg = ae.getActionCommand();
                if(arg.equals("Add Customer"))
                {
                        cardLO.show(home, "Add Customer");
                }

                else if(arg.equals("View Customer"))
                {
                        cardLO.show(home, "Update Customer");
                        ups.loadCustomer();
                }

                else if(arg.equals("Delete Customer"))
                {
                        cardLO.show(home, "Delete Customer");
                        dels.loadCustomer();
                }
                else if(arg.equals("Add Orders"))
                {
```

```
            cardLO.show(home, "Add Orders");
        }
        else if(arg.equals("View Orders"))
        {
            cardLO.show(home, "Update Orders");
            upP.loadOrders();
        }
        else if(arg.equals("Delete Orders"))
        {
            cardLO.show(home, "Delete Orders");
            delP.loadOrders();
        }
        else if(arg.equals("Add Address"))
        {
            cardLO.show(home, "Add Address");
        }
        else if(arg.equals("Delete Address"))
        {
            cardLO.show(home, "Delete Address");
            delR.loadAddress();
        }
        else if(arg.equals("View Address"))
        {
            cardLO.show(home, "Update Address");
            upR.loadAddress();
        }
        else if(arg.equals("Add Delivery"))
        {
            cardLO.show(home, "Add Delivery");
```

```
}
else if(arg.equals("View Delivery"))
{
        cardLO.show(home, "Update Delivery");
        upS.loadDelivery();
}
else if(arg.equals("Delete Delivery"))
{
        cardLO.show(home, "Delete Delivery");
        delS.loadDelivery();

}
else if(arg.equals("Add Food"))
{
        cardLO.show(home,"Add Food");
}
else if(arg.equals("Delete Food"))
{
        cardLO.show(home, "Delete Food");
        delf.loadFood();
}
else if(arg.equals("View Food"))
{
        cardLO.show(home, "Update Food");
        upf.loadFood();
}
else if(arg.equals("Add Vehicle"))
{
        cardLO.show(home,"Add Vehicle");
```

```
            }
            else if(arg.equals("Delete Vehicle"))
            {
                cardLO.show(home, "Delete Vehicle");
                delv.loadVehicle();
            }
            else if(arg.equals("View Vehicle"))
            {
                cardLO.show(home, "Update Vehicle");
                upv.loadVehicle();
            }
        }
        private void quitApp () {

                try {
                    //Show a Confirmation Dialog.
                    int reply = JOptionPane.showConfirmDialog (this,
                                "Are you really want to exit\nFrom
AUTOMATED FOOD DELIVERY System?",
                                "FOOD DELIVERY SYSTEm - Exit",
JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);
                    //Check the User Selection.
                    if (reply == JOptionPane.YES_OPTION) {
                        setVisible (false);     //Hide the Frame.
                        dispose();              //Free the System Resources.
                        System.out.println ("Thanks for Using AUTOMATED
FOOD DELIVERY System\nAuthor - vivek");
                        System.exit (0);    //Close the Application.
                    }
                    else if (reply == JOptionPane.NO_OPTION) {
```

```
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                    }
            }

            catch (Exception e) {}

        }
 public static void main(String ... args)
 {
            new FrontPage();
 }


}
```

Insert a Customer:

```
package Customer;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class AddCustomer extends Panel{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    Button AddCustomerButton;
    TextField C_ID,C_NAME,PHONE,EMAIL;
    TextArea errorText;
    Connection connection;
```

```
    Statement statement;
    public AddCustomer()
    {
            try
            {
                    Class.forName("oracle.jdbc.driver.OracleDriver");
            }
            catch (Exception e)
            {
                    System.err.println("Unable to find and load driver");
                    System.exit(1);
            }
            connectToDB();

    }

    public void connectToDB()
   {
            try
            {
             connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","vivek","vivek12
3");
             statement = connection.createStatement();
             statement.executeUpdate("commit");

            }
            catch (SQLException connectException)
            {
             System.out.println(connectException.getMessage());
```

```
                System.out.println(connectException.getSQLState());
                System.out.println(connectException.getErrorCode());
                System.exit(1);
            }
    }
    public void buildGUI()
    {

            AddCustomerButton = new Button("Add Customer");
            AddCustomerButton.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                      //String query = "INSERT INTO
CUSTOMER(ID,NAME,PHONE,EMAIL) VALUES
(2,'vivek','9645843635',vivek.basa1217@gmail.com)";
                      String query= "INSERT INTO customer VALUES('" +
C_NAME.getText() + "', "  + C_ID.getText() + "," + PHONE.getText() + ",'" +
EMAIL.getText() + "')";
                      int i = statement.executeUpdate(query);
                      statement.executeUpdate("commit");
                      errorText.append("\nInserted " + i + " rows successfully");
                    }
                    catch (SQLException insertException)
                    {
                      displaySQLErrors(insertException);
                    }
                }
```

```
        });
        C_ID=new TextField(20);
        C_NAME = new TextField(20);
        PHONE = new TextField(10);
        EMAIL = new TextField(20);



        errorText = new TextArea(10, 40);
        errorText.setEditable(false);

        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
        first.add(new Label("Name:"));
        first.add(C_NAME);
        first.add(new Label("Customer ID:"));
        first.add(C_ID);
        first.add(new Label("PHONE"));
        first.add(PHONE);
        first.add(new Label("EMAIL:"));
        first.add(EMAIL);
        first.setBounds(125,90,200,100);

        Panel second = new Panel(new GridLayout(4, 1));
        second.add(AddCustomerButton);
    second.setBounds(125,220,150,100);

        Panel third = new Panel();
        third.add(errorText);
        third.setBounds(125,320,300,200);
```

```
            setLayout(null);
            add(first);
            add(second);
            add(third);
            setSize(500, 600);
            setVisible(true);
            System.out.println("hello");
    }
    private void displaySQLErrors(SQLException e)
    {
            errorText.append("\nSQLException: " + e.getMessage() + "\n");
            errorText.append("SQLState:     " + e.getSQLState() + "\n");
            errorText.append("VendorError:  " + e.getErrorCode() + "\n");
    }

}
```

Delete a Customer:

```
package Customer;
import java.awt.*;
import java.sql.*;
public class DeleteCustomer extends Panel
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    Button deleteCustomerButton;
    List customerIDList=null;
```

```
TextField C_ID,C_NAME,PHONE,EMAIL;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;


public DeleteCustomer()

{

        try

        {

                Class.forName("oracle.jdbc.driver.OracleDriver");

        }

        catch (Exception e)

        {

                System.err.println("Unable to find and load driver");

                System.exit(1);

        }

        connectToDB();

}


public void connectToDB()

 {

        try

        {

         connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","vivek","vivek12
3");

         statement = connection.createStatement();


        }
```

```
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
}

  public void loadCustomer()
  {
            try
            {

              customerIDList.removeAll();
              rs = statement.executeQuery("SELECT * FROM customer");
              while (rs.next())
              {
                        customerIDList.add(rs.getString("C_ID"));
              }
            }
            catch (SQLException e)
            {
              displaySQLErrors(e);
            }
  }

  public void buildGUI()
  {
```

```
customerIDList = new List(10);
        loadCustomer();
        add(customerIDList);

        //When a list item is selected populate the text fields
        customerIDList.addItemListener(new ItemListener()
        {
                public void itemStateChanged(ItemEvent e)
                {
                        try
                        {
                                rs = statement.executeQuery("SELECT * FROM
customer");
                                while (rs.next())
                                {
                                        if
(rs.getString("C_ID").equals(customerIDList.getSelectedItem()))
                                        break;
                                }
                                if (!rs.isAfterLast())
                                {
                                        C_NAME.setText(rs.getString("C_NAME"));
                                        C_ID.setText(rs.getString("C_ID"));
                                        PHONE.setText(rs.getString("PHONE"));
                                        EMAIL.setText(rs.getString("EMAIL"));
                                }
                        }
                        catch (SQLException selectException)
                        {
                                displaySQLErrors(selectException);
```

```
                }
            }
        });


        deleteCustomerButton = new Button("Delete Customer");
        deleteCustomerButton.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                try
                {
                    Statement statement = connection.createStatement();
                    int i = statement.executeUpdate("DELETE FROM
customer WHERE C_ID = "
                            + customerIDList.getSelectedItem());
                    errorText.append("\nDeleted " + i + " rows
successfully");
                    C_NAME.setText(null);
                    C_ID.setText(null);
                    PHONE.setText(null);
                    EMAIL.setText(null);
                    statement.executeUpdate("commit");
                    loadCustomer();
                }
                catch (SQLException insertException)
                {
                    displaySQLErrors(insertException);
                }
            }
```

```
});
C_NAME = new TextField(15);
C_ID = new TextField(15);
PHONE = new TextField(15);
EMAIL = new TextField(15);

errorText = new TextArea(10, 40);
errorText.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Name:"));
first.add(C_NAME);
first.add(new Label("Customer ID:"));
first.add(C_ID);
first.add(new Label("PHONE:"));
first.add(PHONE);
first.add(new Label("EMAIL:"));
first.add(EMAIL);

Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteCustomerButton);

Panel third = new Panel();
third.add(errorText);

add(first);
add(second);
add(third);
```

```
            setSize(450, 600);
            setLayout(new FlowLayout());
            setVisible(true);

    }




    private void displaySQLErrors(SQLException e)
    {
            errorText.append("\nSQLException: " + e.getMessage() + "\n");
            errorText.append("SQLState:     " + e.getSQLState() + "\n");
            errorText.append("VendorError:  " + e.getErrorCode() + "\n");
    }
}
```

Update a Customer:

```
package Customer;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateCustomer extends Panel
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    Button updateCustomerButton;
    List customerIDList;
```

```
TextField C_ID,C_NAME,PHONE,EMAIL;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;


public UpdateCustomer()

{

        try

        {

                Class.forName("oracle.jdbc.driver.OracleDriver");

        }

        catch (Exception e)

        {

                System.err.println("Unable to find and load driver");

                System.exit(1);

        }

        connectToDB();

}


public void connectToDB()

 {

        try

        {

         connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","vivek","vivek12
3");

         statement = connection.createStatement();


        }
```

```
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
}

  public void loadCustomer()
  {
            try
            {
                        customerIDList.removeAll();
              rs = statement.executeQuery("SELECT C_ID FROM customer");
              while (rs.next())
              {
                        customerIDList.add(rs.getString("C_ID"));
              }
            }
            catch (SQLException e)
            {
              displaySQLErrors(e);
            }
  }

  public void buildGUI()
  {
     customerIDList = new List(10);
```

```
loadCustomer();
add(customerIDList);

customerIDList.addItemListener(new ItemListener()
{
	public void itemStateChanged(ItemEvent e)
	{
		try
		{
			rs = statement.executeQuery("SELECT * FROM
customer where C_ID ="+customerIDList.getSelectedItem());
			rs.next();
			C_NAME.setText(rs.getString("C_NAME"));
			C_ID.setText(rs.getString("C_ID"));
			PHONE.setText(rs.getString("PHONE"));
			EMAIL.setText(rs.getString("EMAIL"));
		}
		catch (SQLException selectException)
		{
			displaySQLErrors(selectException);
		}
	}
});


//Handle Update Sailor Button
updateCustomerButton = new Button("Update Customer");
updateCustomerButton.addActionListener(new ActionListener()
{
```

```
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
                                Statement statement = connection.createStatement();
                                int i = statement.executeUpdate("UPDATE customer "
                                + "SET C_NAME='" + C_NAME.getText() + "', "
                                + "PHONE=" + PHONE.getText() + ", "
                                + "EMAIL='"+ EMAIL.getText() + "' WHERE C_ID = "
                                + customerIDList.getSelectedItem());
                                errorText.append("\nUpdated " + i + " rows
successfully");
                                i = statement.executeUpdate("commit");

                                loadCustomer();
                        }
                        catch (SQLException insertException)
                        {
                                displaySQLErrors(insertException);
                        }
                }
        });
        C_NAME = new TextField(15);
        C_ID = new TextField(15);
        C_ID.setEditable(false);
        PHONE = new TextField(15);
        EMAIL = new TextField(15);

        errorText = new TextArea(10, 40);
```

```
        errorText.setEditable(false);

        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
        first.add(new Label("C_NAME:"));
        first.add(C_NAME);
        first.add(new Label("C_ID:"));
        first.add(C_ID);
        first.add(new Label("PHONE:"));
        first.add(PHONE);
        first.add(new Label("EMAIL:"));
        first.add(EMAIL);

        Panel second = new Panel(new GridLayout(4, 1));
        second.add(updateCustomerButton);

        Panel third = new Panel();
        third.add(errorText);

        add(first);
        add(second);
        add(third);

        setSize(500, 600);
        setLayout(new FlowLayout());
        setVisible(true);

    }
```

```
        private void displaySQLErrors(SQLException e)
        {
                errorText.append("\nSQLException: " + e.getMessage() + "\n");
                errorText.append("SQLState:     " + e.getSQLState() + "\n");
                errorText.append("VendorError:  " + e.getErrorCode() + "\n");
        }



    }
```

Connectivity with the Database:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

<u>Block of code for JAVA- SQL connectivity with JDBC:</u>

```
public void connectToDB()
        {
                try
                {

        connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:
1521:orcl","hemanth","oracle");
                        statement=connection.createStatement();
                }
                catch(SQLException connectException)
```

```
        {
                System.out.println(connectException.getMessage());
                System.out.println(connectException.getSQLState());
                System.out.println(connectException.getErrorCode());
                System.exit(1);
        }
    }
```

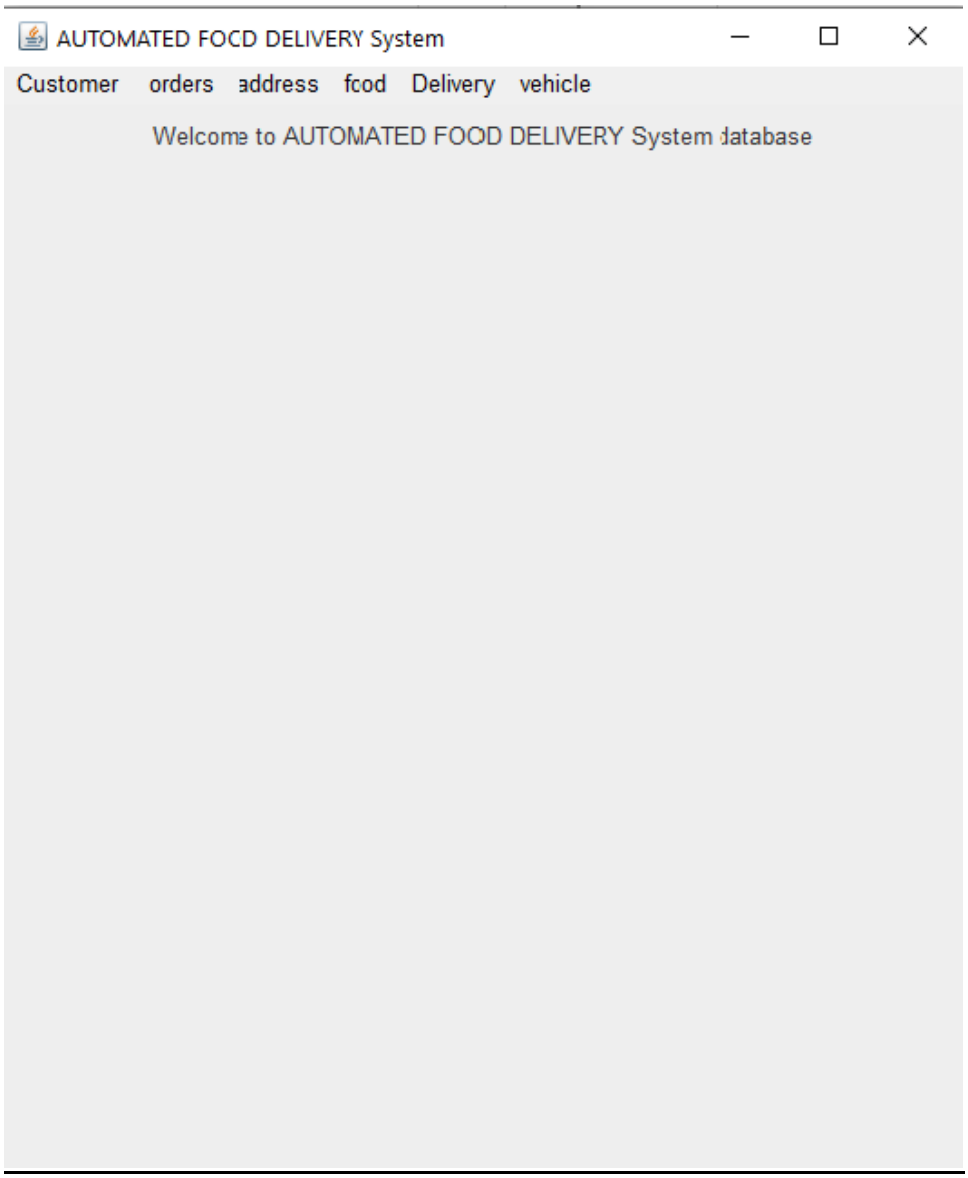
GITHUB LINK:

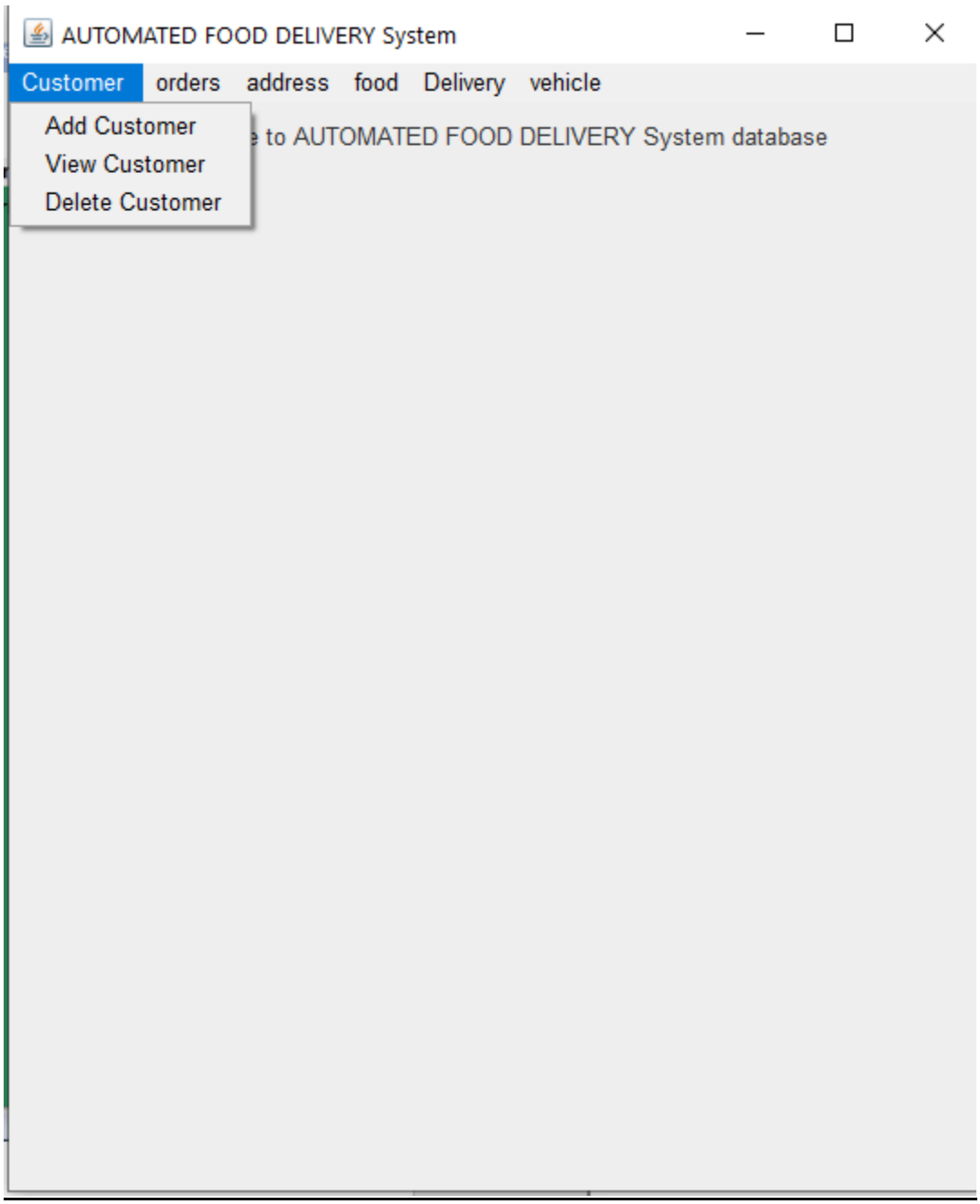
https://github.com/vivek-vk19/DBMS-project.git

TESTING

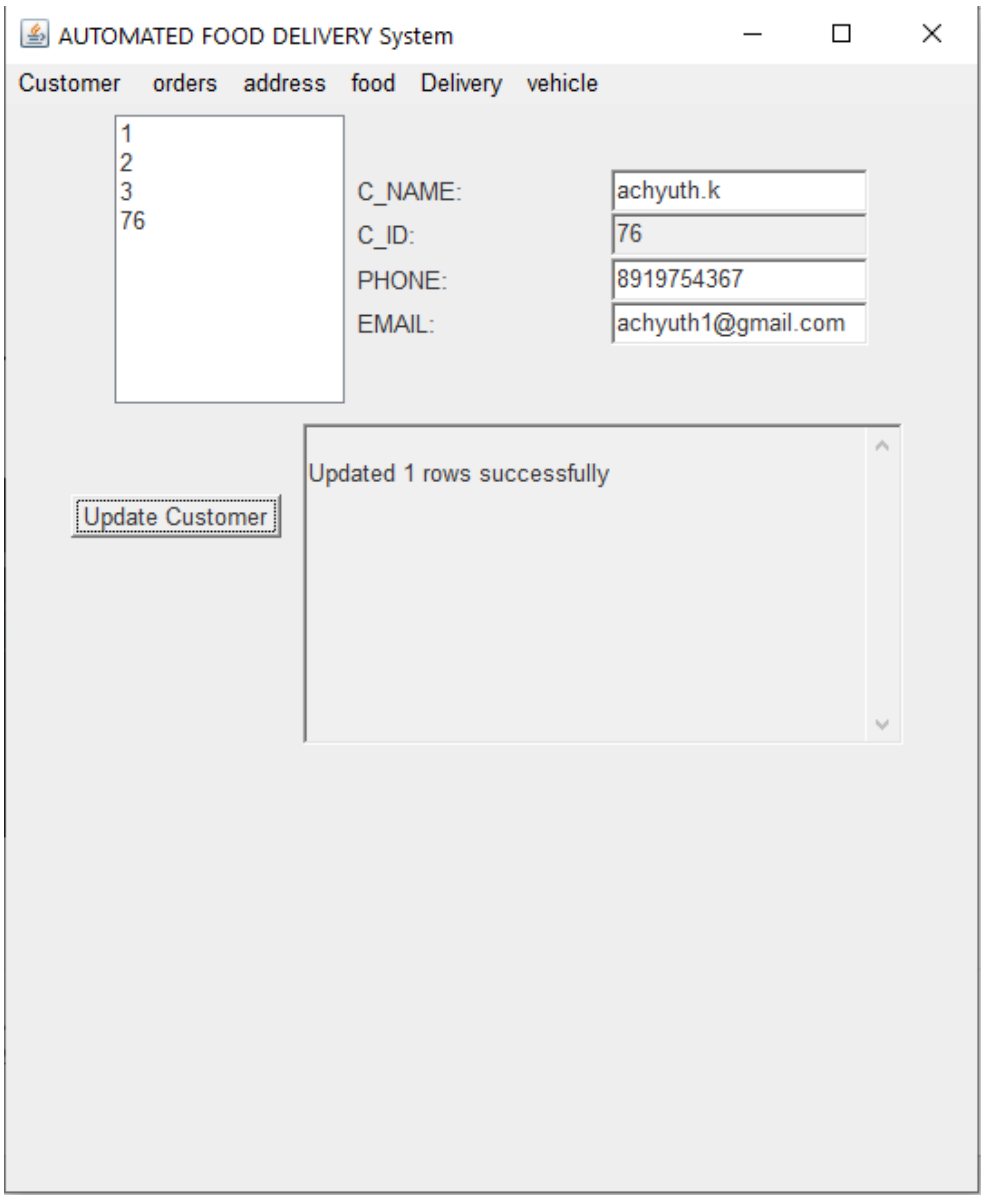
The program runs for execution of three basic operations of insertion, update and delete on 5 different table. Along with this, it also has a output column which gives the information about how many rows have been edited. Errors, syntactical or exceptional will be shown if occurred.

HOMEPAGE:

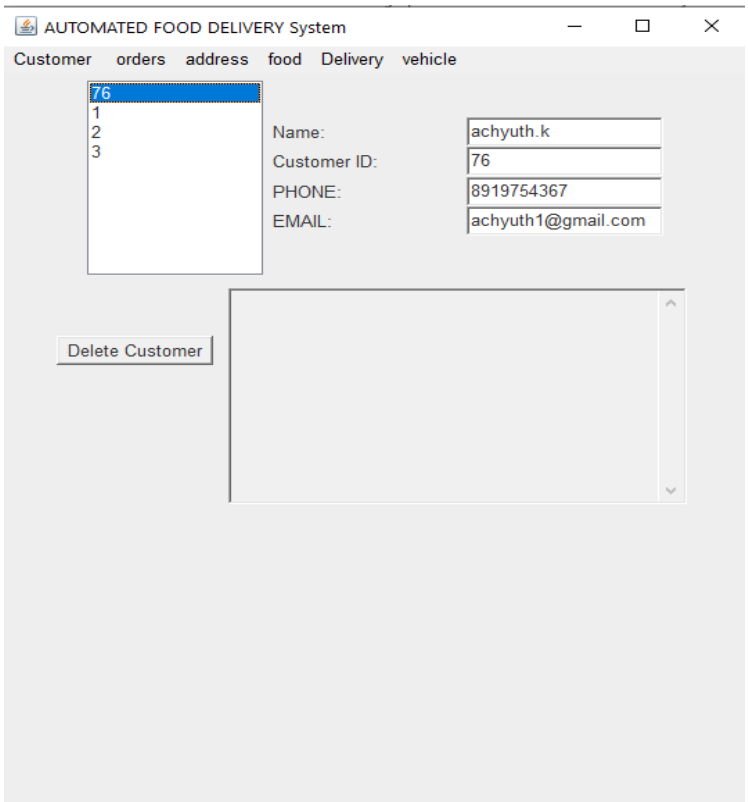

AUTOMATED FOCD DELIVERY System
Customer
orders
address
fcod
Delivery
vehicle
Welcome to AUTOMATED FOOD DELIVERY System database

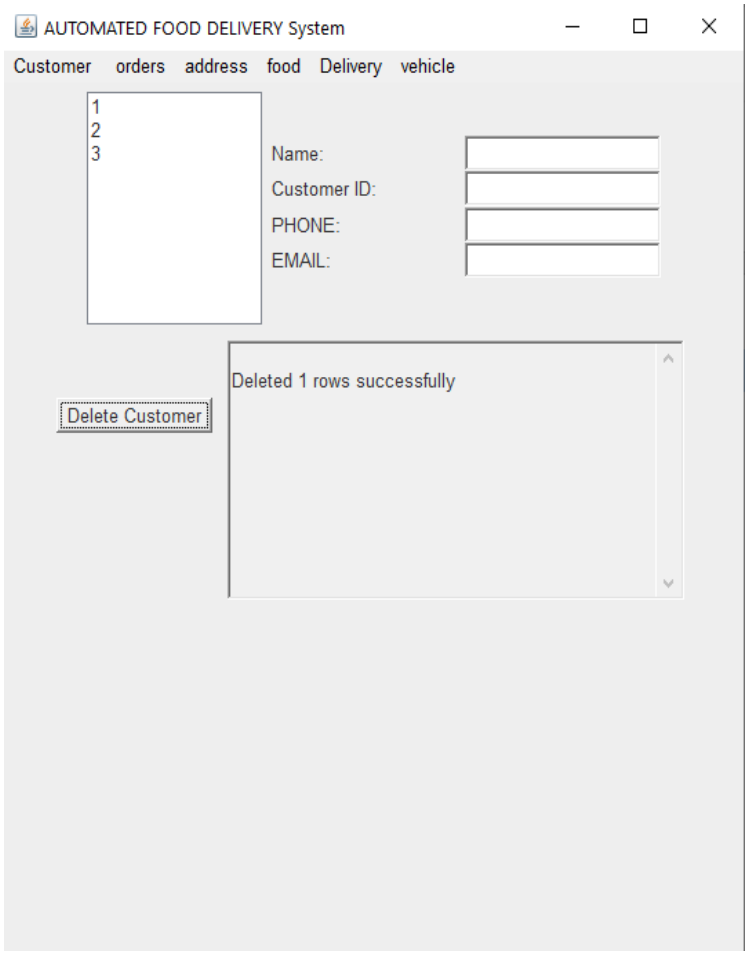

AUTOMATED FOOD DELIVERY System
Customer
orders
address
food
Delivery
vehicle
Add Customer
View Customer
Delete Customer
e to AUTOMATED FOOD DELIVERY System database


```
SQL> select  * from customer;

C_NAME                    C_ID      PHONE EMAIL
-------------------- ---------- ---------- --------------------
achyuth                      76 8919754367 achyuth@gmail.com
vivek                         1 6305314935 vivek.basa@gmail.com
rohith                        2 7995702445 sairohith@gmail.com
ram                           3 8977652534 ram1234@gmail.com

SQL>
```




```
SQL> select * from customer;

C_NAME                     C_ID      PHONE EMAIL
-------------------- ---------- ---------- --------------------
achyuth.k                    76 8919754367 achyuth1@gmail.com
vivek                         1 6305314935 vivek.basa@gmail.com
rohith                        2 7995702445 sairohith@gmail.com
ram                           3 8977652534 ram1234@gmail.com

SQL>
```


AUTOMATED FOOD DELIVERY System
Customer
orders
address
food
Delivery
vehicle
76
1
2
3
Name:
achyuth.k
Customer ID:
76
PHONE:
8919754367
EMAIL:
achyuth1@gmail.com
Delete Customer


AUTOMATED FOOD DELIVERY System
Customer
orders
address
food
Delivery
vehicle
1
2
3
Name:
Customer ID:
PHONE:
EMAIL:
Deleted 1 rows successfully
Delete Customer

```
SQL> select * from customer;

C_NAME                     C_ID      PHONE EMAIL
-------------------- ---------- ---------- --------------------
achyuth.k                    76 8919754367 achyuth1@gmail.com
vivek                         1 6305314935 vivek.basa@gmail.com
rohith                        2 7995702445 sairohith@gmail.com
ram                           3 8977652534 ram1234@gmail.com

SQL> select * from customer;

C_NAME                     C_ID      PHONE EMAIL
-------------------- ---------- ---------- --------------------
vivek                         1 6305314935 vivek.basa@gmail.com
rohith                        2 7995702445 sairohith@gmail.com
ram                           3 8977652534 ram1234@gmail.com

SQL>
```

RESULTS

The DML commands,Insert,Update and delete for one of the tables in given below:

For customer table(in java as per the application):

Insert:
```
"INSERT INTO customer VALUES('" + C_NAME.getText() + "', "  +
C_ID.getText() + "," + PHONE.getText() + ",'" + EMAIL.getText() + "')";
```

Update:
```
"UPDATE customer "
+ "SET C_NAME='" + C_NAME.getText() + "', "
+ "PHONE=" + PHONE.getText() + ", "
+ "EMAIL='"+ EMAIL.getText() + "' WHERE C_ID = "                                +
customerIDList.getSelectedItem());
```

Delete:
```
"DELETE FROM customer WHERE C_ID = "+
customerIDList.getSelectedItem());
```

REFERENCES

1. https://en.wikipedia.org/wiki/Online_food_ordering
2. https ://en.wikipedia.org/wiki/Online_food_ordering
3. https://github.com/vivek-vk19/DBMS-project.git(ASSIGNMENT 1)