**BAKERY**

**MANAGEMENT SYSTEM**

Name: P Sai Saketh

Roll no: 1602-19-737-097

Section: IT B

**Abstract**

The main objective of any management system is to information and maintain a right order of the information. And to provide the information at required time efficiently.

Bakery Management System is designed to deal with daily operation taking place in a bakery. It helps to record all essential details. It can manage products, customers details, employee details and sales details. It keeps track of the item quality and rating. This helps to improve the menu list as per the data. The total sails in a particular day is stored. The purpose of this project is to computerize the management of Bakery and develop a software which is user friendly and works efficiently.

**REQUIREMENT ANALYSIS:**

**List of tables:**

1. Items\_list
2. Order\_list
3. Customer
4. Sales
5. Employee
6. Payment

**List of attributes with their domain type:**

Items\_list

* item\_ID varchar2(50)
* item\_name varchar2(50)
* cost number(20)
* available\_quantity number(20)

order\_list

* order\_ID varchar2(50)
* item\_ID varchar2(50)
* ordered\_quantity number(20)

customer

* customer\_ID varchar2(50)
* customer\_name varchar2(50)
* customer\_PHNO number(20)
* token number(20)
* order\_ID varchar2(50)

sales

* item\_ID varchar2(50)
* Quantity\_sold number(20)
* Total\_amount number(20)

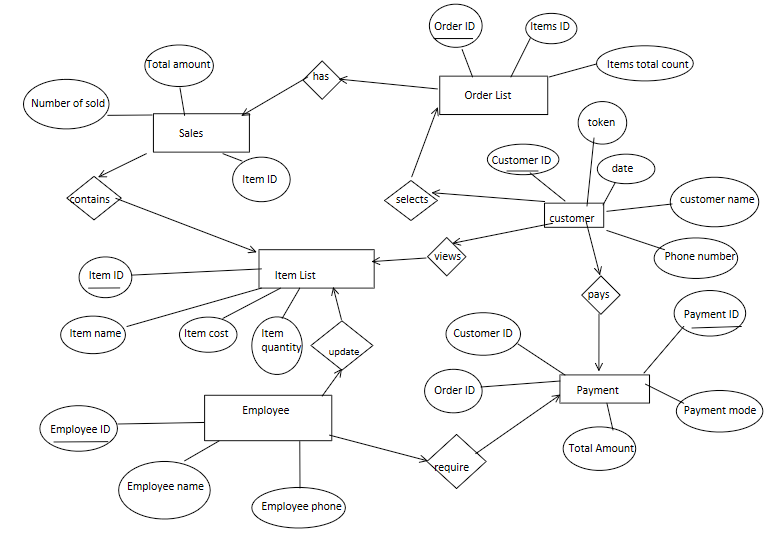
employee\_list

* employee\_ID varchar2(50)
* employee\_name varchar2(50)
* employee\_PHNO number(20)

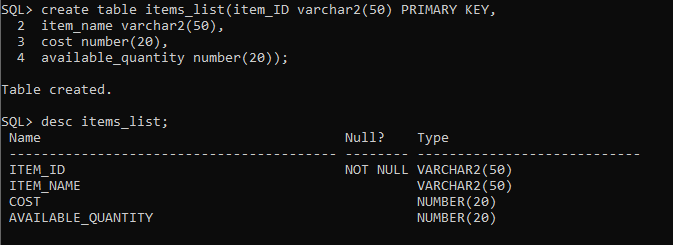
payment

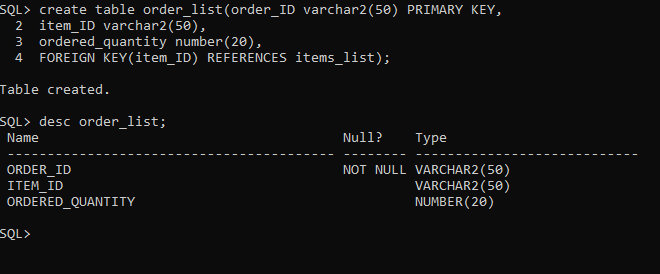
* customer\_ID varchar2(50)
* order\_ID varchar2(50)
* pay\_mode varchar2(50)
* pay\_ID varchar2(50)
* total number(20)

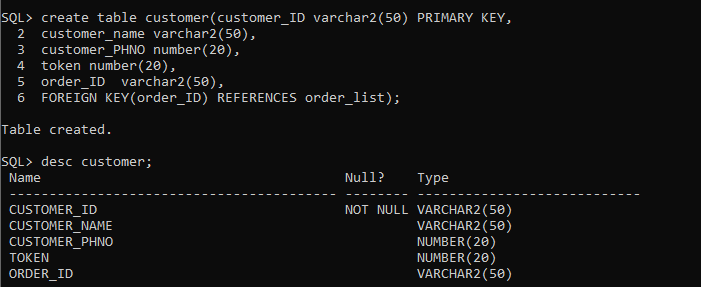
**ER DIAGRAM:**

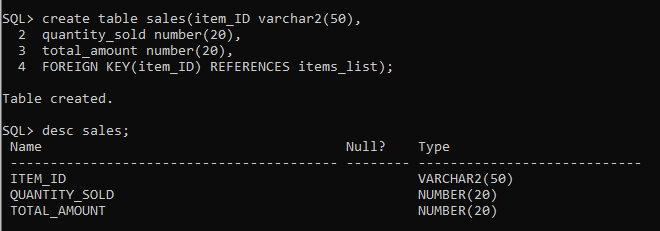


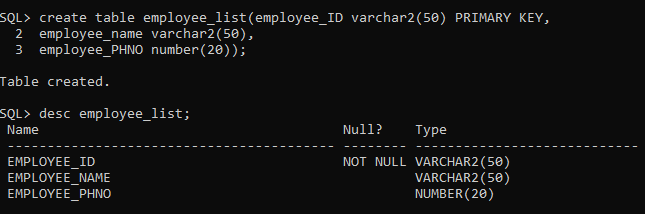
**DDL COMMANDS:**

****

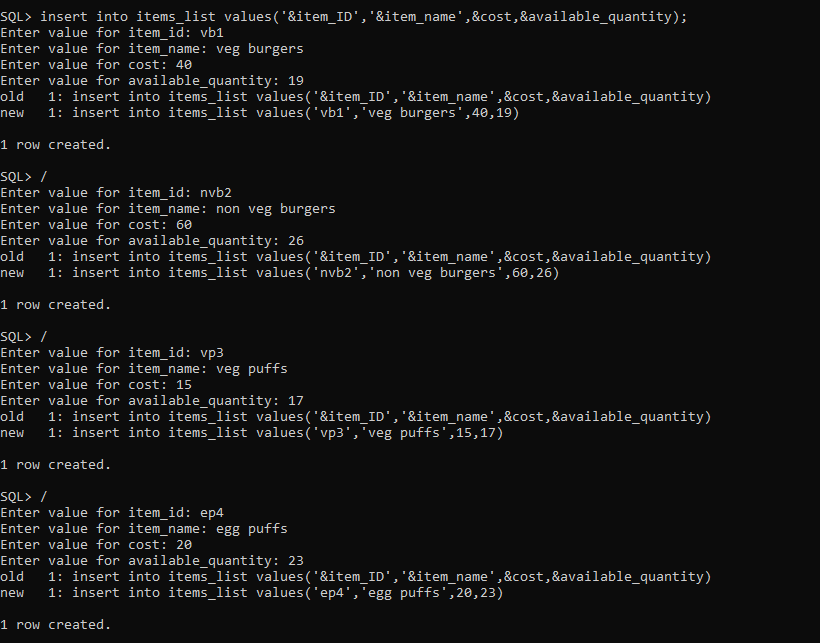
****

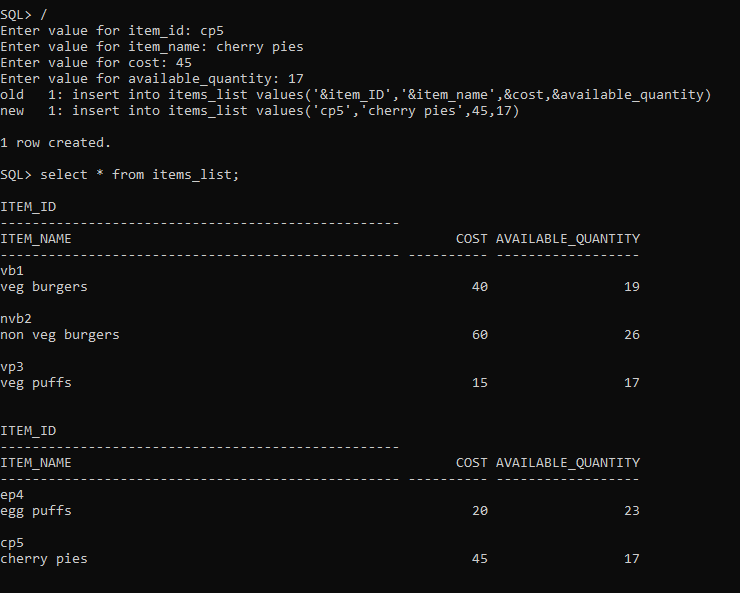
****

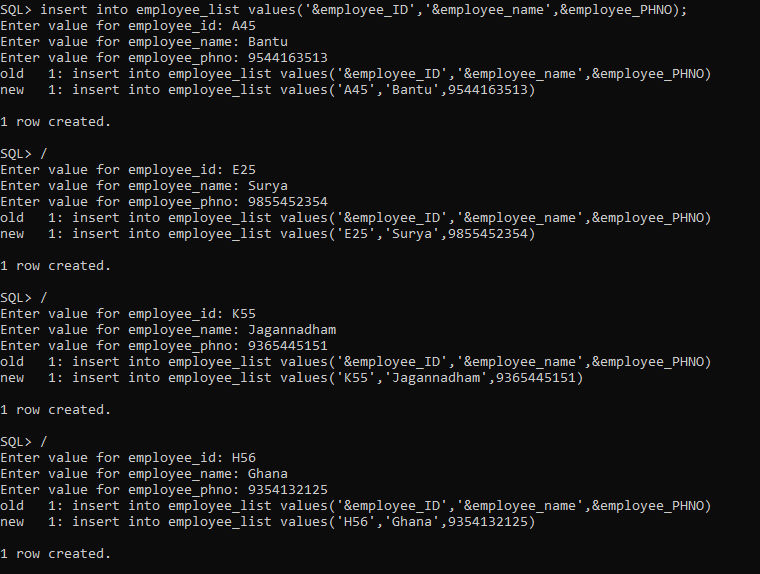
****

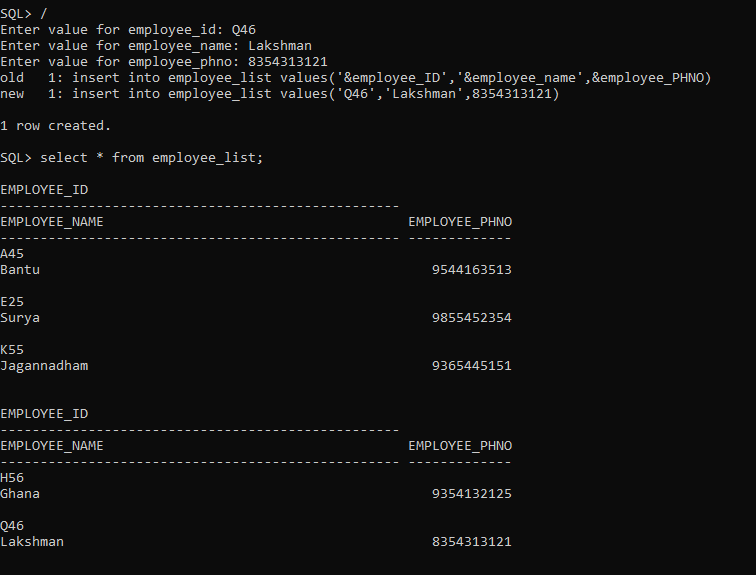
****

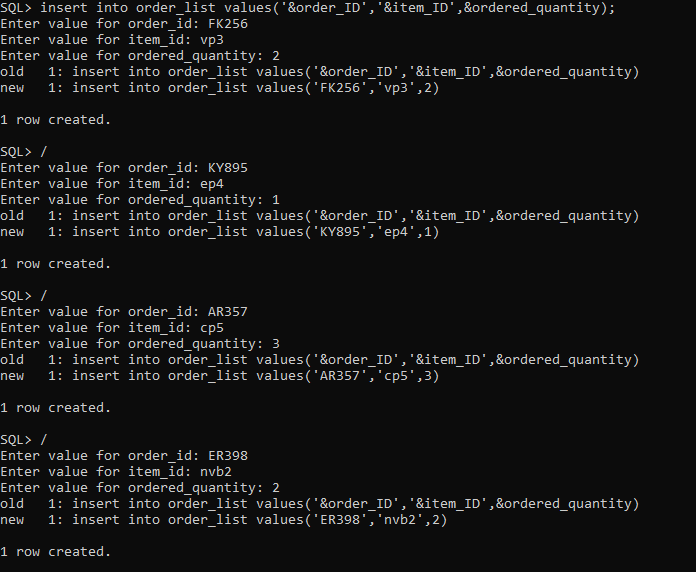
**DML COMMANDS:**

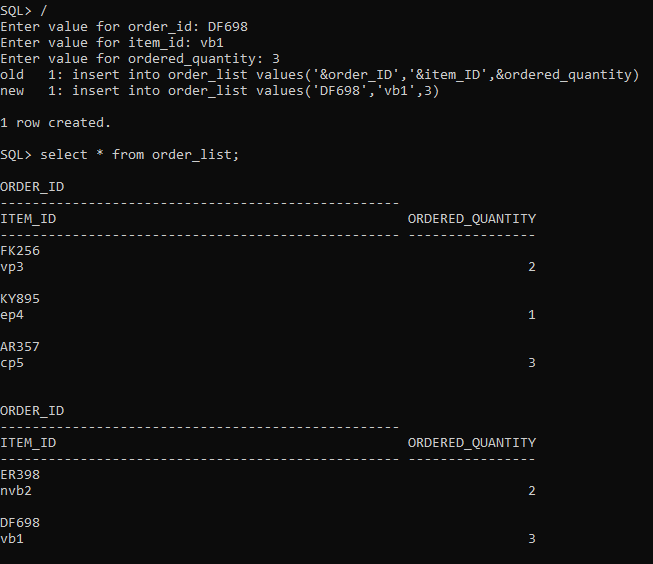
****

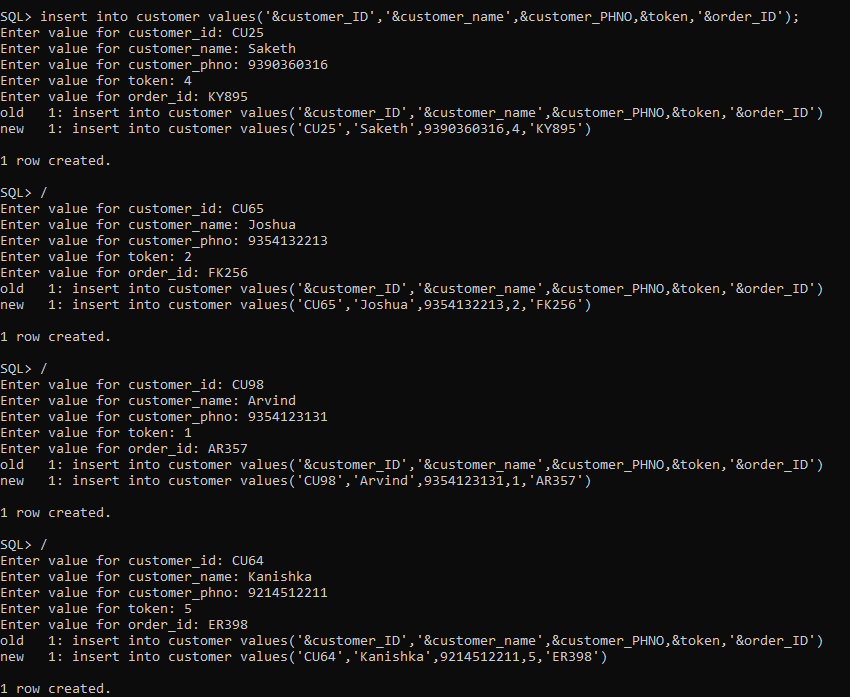
****

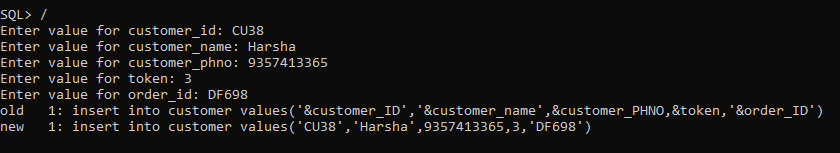
****

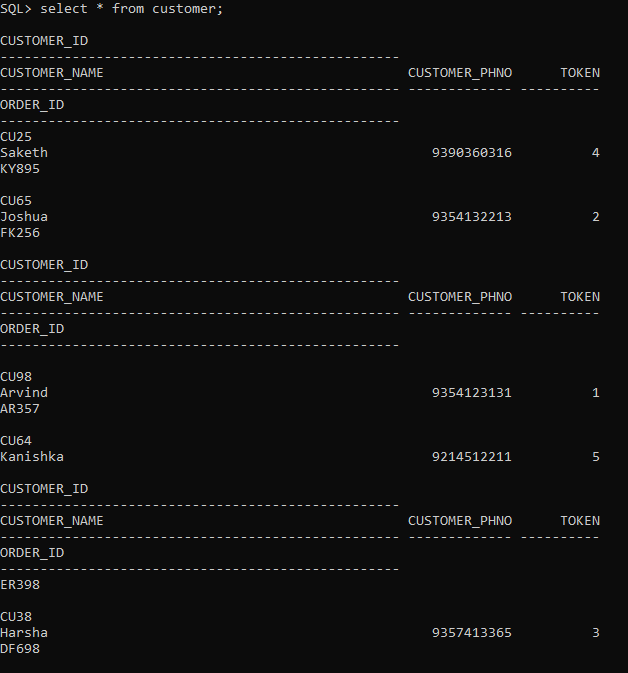
****

****

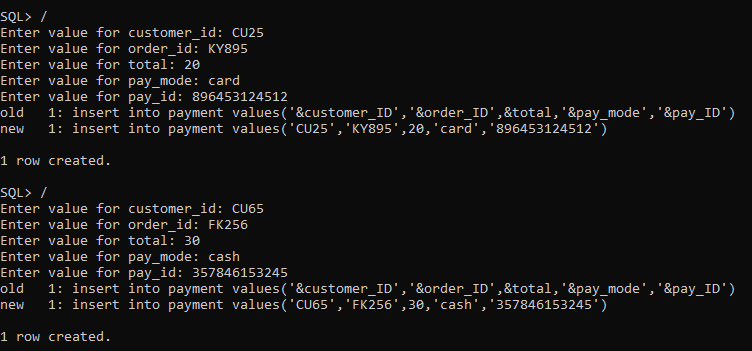
****

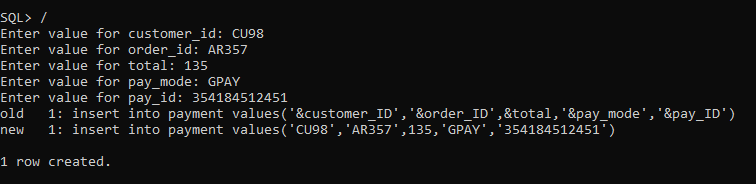
****

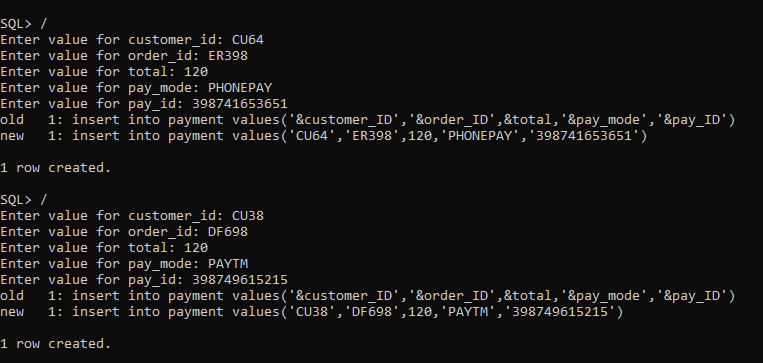
****

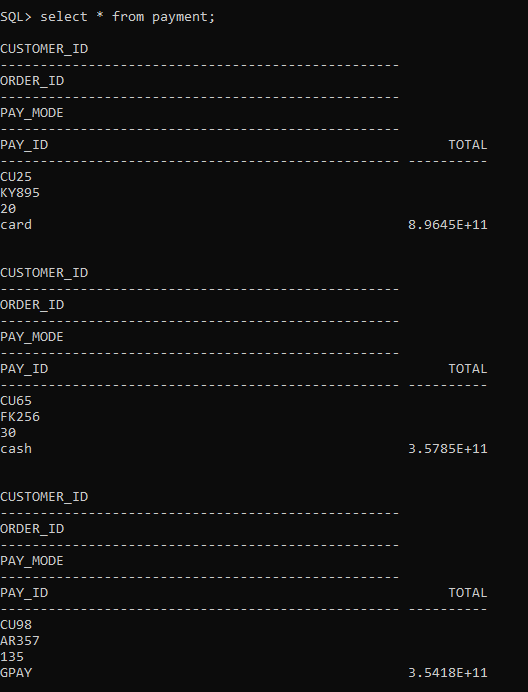
****

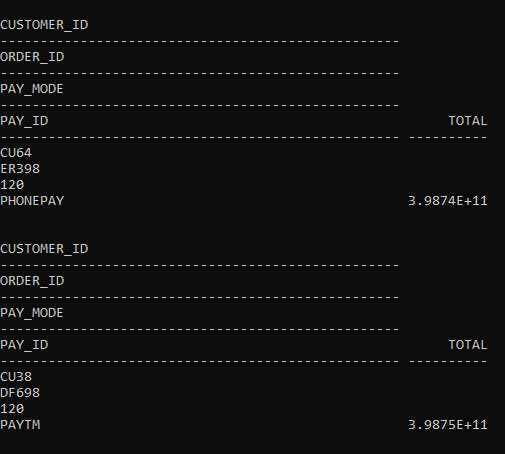
Insert into payment(‘&customer\_ID’,’&order\_ID’,&total,’&pay\_mode’,’&pay\_ID’);

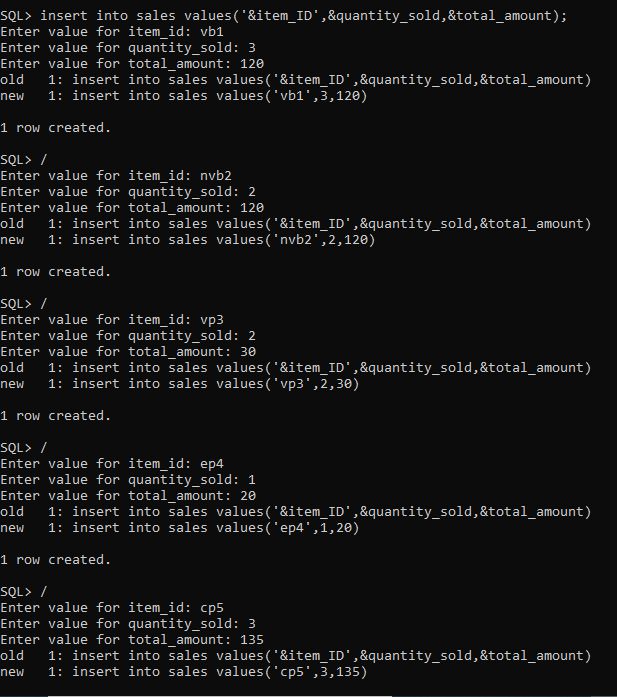
****

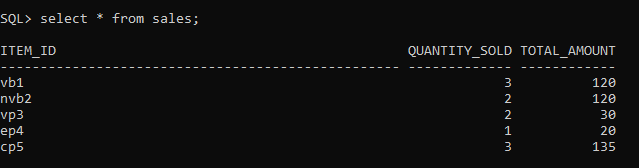
****

****

****

****

****

****

## **IMPLEMENTATION:**

**Front end programs and its connectivity**

**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.

The connection to the database can be performed using Java programming (JDBC API) as:

**public** **void** connectToDB()

{

**try** {

Connection con=DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:xe","Saketh","vasavi");

statement=con.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);

}

}

Thus, the connection from Java to Oracle database is performed and therefore, can be used for updating tables in the database directly.

**CODE:**

package DBMS;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

//1602-19-737-097 P Sai Saketh

@SuppressWarnings("serial")

public class InsertTables extends Frame implements ActionListener

{

private static Dialog d;

String Password="Ss@8096571377";

String Password2="Q9W8E7R6T5";

MenuBar mb;

MenuItem m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,m17,m18,m19,m20,m21,m22,m23,m24;

Menu menu,Customer,orders,sales,employee,payment;

TextField item\_IDText, item\_nameText, costText, available\_quantityText ;

TextField customer\_IDText, customer\_nameText, customer\_PHNOText, tokenText, order\_id1Text;

TextField order\_idText, item\_idText, ordered\_quantityText ;

TextField item\_ID1Text, quantity\_soldText, total\_amountText;

TextField employee\_IDText,employee\_nameText,employee\_PHNOText;

TextField customer\_ID1Text,order\_id2Text,pay\_modeText,pay\_IDText,totalText;

TextField avail1;

Choice menuChoice,ordersChoice,CustomerChoice;

TextArea errorText;

Connection connection;

Statement statement;

StringBuffer avail\_quan;

StringBuffer Cst;

StringBuffer order\_quan;

StringBuffer I\_ID;

StringBuffer s;

//For updates

Button updateButton;

Button insertButton;

List menuList,ordersList,CustomerList,salesList,employeeList,payList;

ResultSet rs;

//TextField item\_IDText, item\_nameText, costText, available\_quantityText ;

//For delete

Button deleteRowButton;

public InsertTables()

{

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:xe","saketh","vasavi");

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 buildFrame()

{

//Basic Frame Properties

setTitle("Bakery Management System");

setSize(500, 600);

setVisible(true);

//menubar

mb = new MenuBar();

setMenuBar(mb);

setSize(550,500);

setLayout(null);

setVisible(true);

//Menu

menu=new Menu("Menu");

m1=new MenuItem("Insert Menu");

m2=new MenuItem("Update Menu");

m3=new MenuItem("Delete Menu");

m4=new MenuItem("View Menu");

menu.add(m1);

menu.add(m2);

menu.add(m3);

menu.add(m4);

// mb.add(menu);

//Customer

Customer=new Menu("Customer");

m5=new MenuItem("Insert Customer");

m6=new MenuItem("Update Customer");

m7=new MenuItem("Delete Customer");

m8=new MenuItem("View Customer");

Customer.add(m5);

Customer.add(m6);

Customer.add(m7);

Customer.add(m8);

//Orders

orders=new Menu("Orders");

m9=new MenuItem("Insert Orders");

m10=new MenuItem("Update Orders");

m11=new MenuItem("Delete Orders");

m12=new MenuItem("View Orders");

orders.add(m9);

orders.add(m10);

orders.add(m11);

orders.add(m12);

// mb.add(orders);

//mb.add(Customer);

//sales

sales=new Menu("Sales");

m13=new MenuItem("Insert Sales");

m14=new MenuItem("Update Sales");

m15=new MenuItem("Delete Sales");

m16=new MenuItem("View Sales");

sales.add(m13);

sales.add(m14);

sales.add(m15);

sales.add(m16);

// mb.add(sales);

employee=new Menu("Employee");

m17=new MenuItem("Insert Employee");

m18=new MenuItem("Update Employee");

m19=new MenuItem("Delete Employee");

m20=new MenuItem("View Employee");

employee.add(m17);

employee.add(m18);

employee.add(m19);

employee.add(m20);

// mb.add(employee);

payment=new Menu("Payment");

m21=new MenuItem("Insert Payment");

m22=new MenuItem("Update Payment");

m23=new MenuItem("Delete Payment");

m24=new MenuItem("View Payment");

payment.add(m21);

payment.add(m22);

payment.add(m23);

payment.add(m24);

// mb.add(payment);

mb.add(menu);

mb.add(orders);

mb.add(Customer);

mb.add(payment);

mb.add(sales);

mb.add(employee);

//Registering action Listeners

m1.addActionListener(this);

m2.addActionListener(this);

m3.addActionListener(this);

m4.addActionListener(this);

m5.addActionListener(this);

m6.addActionListener(this);

m7.addActionListener(this);

m8.addActionListener(this);

m9.addActionListener(this);

m10.addActionListener(this);

m11.addActionListener(this);

m12.addActionListener(this);

m13.addActionListener(this);

m14.addActionListener(this);

m15.addActionListener(this);

m16.addActionListener(this);

m17.addActionListener(this);

m18.addActionListener(this);

m19.addActionListener(this);

m20.addActionListener(this);

m21.addActionListener(this);

m22.addActionListener(this);

m23.addActionListener(this);

m24.addActionListener(this);

// setBackground(Color.blue);

repaint();

}

public void actionPerformed(ActionEvent ae)

{

String arg = ae.getActionCommand();

if(arg.equals("Insert Menu"))

this.buildGUIMenu();

if(arg.equals("Update Menu"))

this.updateMenuGUI();

if(arg.equals("Delete Menu"))

this.deleteGUIMenu();

if(arg.equals("View Menu"))

this.viewMenuGUI();

if(arg.equals("Insert Customer"))

this.buildGUICustomer();

if(arg.equals("Update Customer"))

this.updateCustomerGUI();

if(arg.equals("Delete Customer"))

this.deleteGUICustomer();

if(arg.equals("View Customer"))

this.viewCustomerGUI();

if(arg.equals("Insert Orders"))

this.buildGUIOrders();

if(arg.equals("Update Orders"))

this.updateOrdersGUI();

if(arg.equals("Delete Orders"))

this.deleteGUIOrders();

if(arg.equals("View Orders"))

this.viewOrdersGUI();

if(arg.equals("Insert Sales"))

this.buildGUISales();

if(arg.equals("Update Sales"))

this.updateGUISales();

if(arg.equals("Delete Sales"))

this.deleteGUISales();

if(arg.equals("View Sales"))

this.viewSalesGUI();

if(arg.equals("Insert Employee"))

this.buildGUIEmployee();

if(arg.equals("Update Employee"))

this.updateGUIEmployee();

if(arg.equals("Delete Employee"))

this.deleteGUIEmployee();

if(arg.equals("View Employee"))

this.viewEmployeeGUI();

if(arg.equals("Insert Payment"))

this.buildGUIPayment();

if(arg.equals("Update Payment"))

this.updatePaymentGUI();

if(arg.equals("Delete Payment"))

this.deleteGUIPayment();

if(arg.equals("View Payment"))

this.viewPaymentGUI();

}

private void loadOrders1()

{

try

{

rs = statement.executeQuery("SELECT \* FROM order\_list");

while (rs.next())

{

ordersChoice.add(rs.getString("order\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

/\*private void dialogPopUp()

{

Frame f= new Frame();

TextField paswrd=new TextField();

errorText=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password.equals(paswrd.getText()))

{

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.add(errorText);

d.setSize(200,200);

d.setVisible(true);

}\*/

public void buildGUIMenu()

{

Frame f= new Frame();

TextField paswrd=new TextField(15);

paswrd.setEchoChar('\*');

//TextArea errorText1=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

//d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password.equals(paswrd.getText()))

{

d.setVisible(false);

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO items\_list VALUES('" + item\_IDText.getText() + "', " + "'" + item\_nameText.getText() + "'," + costText.getText() + ",'" + available\_quantityText .getText() + "')";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

item\_IDText = new TextField(15);

item\_nameText = new TextField(15);

costText = new TextField(15);

available\_quantityText = 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("Item ID:"));

first.add(item\_IDText);

first.add(new Label("Item Name:"));

first.add(item\_nameText);

first.add(new Label("Item cost:"));

first.add(costText);

first.add(new Label("Available quantity:"));

first.add(available\_quantityText );

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertButton);

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);

setLayout(new FlowLayout());

setVisible(true);

//setBackground(Color.white);

}

else

{

d.setVisible(false);

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.setLayout(new FlowLayout());

d.setSize(200,200);

d.setVisible(true);

}

public void buildGUICustomer()

{

removeAll();

ordersChoice = new Choice();

loadOrders1();

add(ordersChoice);

//When a list item is selected populate the text fields

ordersChoice.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM order\_list");

while (rs.next())

{

if (rs.getString("order\_id").equals(ordersChoice.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

order\_id1Text.setText(rs.getString("order\_id"));

//item\_idText.setText(rs.getString("item\_id"));

//ordered\_quantityText .setText(rs.getString("ordered\_quantity"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO customer VALUES('" + customer\_IDText .getText() + "', '" + customer\_nameText .getText() + "','" + customer\_PHNOText .getText() + "','" + tokenText.getText()+ "','"+ order\_id1Text.getText() + "')";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customer\_IDText = new TextField(15);

customer\_nameText = new TextField(15);

customer\_PHNOText = new TextField(15);

tokenText= new TextField(15);

order\_id1Text= new TextField(15);

order\_id1Text.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_IDText );

first.add(new Label("Customer Name:"));

first.add(customer\_nameText );

first.add(new Label("Customer PHNO:"));

first.add(customer\_PHNOText );

first.add(new Label("Token:"));

first.add(tokenText);

first.add(new Label("Order ID:"));

first.add(order\_id1Text);

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertButton);

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);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadCustomer1()

{

try

{

rs = statement.executeQuery("SELECT \* FROM customer");

while (rs.next())

{

CustomerChoice.add(rs.getString("customer\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUIPayment()

{

//customer\_ID1Text,order\_id2Text,totalText,pay\_modeText,pay\_IDText

removeAll();

CustomerChoice = new Choice();

loadCustomer1();

add(CustomerChoice);

//StringBuffer Cst;

//StringBuffer order\_quan;

//StringBuffer I\_ID;

//When a list item is selected populate the text fields

CustomerChoice.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM customer");

while (rs.next())

{

if (rs.getString("customer\_id").equals(CustomerChoice.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_ID1Text .setText(rs.getString("customer\_id"));

//customer\_nameText .setText(rs.getString("customer\_name"));

//customer\_PHNOText .setText(rs.getString("customer\_PHNO"));

//tokenText.setText(rs.getString("token"));

order\_id2Text.setText(rs.getString("order\_id"));

//totalText.setText(Integer.toString(t));

order\_quan=new StringBuffer();

I\_ID=new StringBuffer();

Cst=new StringBuffer();

int t;

rs=statement.executeQuery("SELECT ordered\_quantity FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

order\_quan.append(rs.getString("ordered\_quantity"));

}

rs=statement.executeQuery("Select item\_id FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

I\_ID.append(rs.getString("item\_id"));

}

rs=statement.executeQuery("Select cost FROM items\_list WHERE item\_id='"+new String(I\_ID)+"'");

if(rs.next())

{

Cst.append(rs.getString("cost"));

}

t=Integer.parseInt(new String(Cst))\*Integer.parseInt(new String(order\_quan));

totalText.setText(Integer.toString(t));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Insert Account Button

//totalText.setText(Integer.toString(t));

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO payment VALUES('" + customer\_ID1Text .getText() + "', '" + order\_id2Text.getText() + "','" + pay\_modeText.getText() + "','" + pay\_IDText.getText()+ "',"+ totalText.getText() + ")";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customer\_ID1Text = new TextField(15);

customer\_ID1Text.setEditable(false);

order\_id2Text = new TextField(15);

order\_id2Text.setEditable(false);

pay\_modeText= new TextField(15);

pay\_IDText= new TextField(15);

totalText = new TextField(15);

//totalText.setText(Integer.toString(t));

totalText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_ID1Text );

first.add(new Label("Order ID:"));

first.add(order\_id2Text);

first.add(new Label("Pay Mode:"));

first.add(pay\_modeText);

first.add(new Label("Pay ID:"));

first.add(pay\_IDText);

first.add(new Label("Total:"));

first.add(totalText);

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertButton);

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);

setLayout(new FlowLayout());

setVisible(true);

}

public void buildGUISales()

{ removeAll();

menuChoice = new Choice();

loadMenu1();

add(menuChoice);

//When a list item is selected populate the text fields

menuChoice.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

if (rs.getString("item\_id").equals(menuChoice.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_ID1Text.setText(rs.getString("item\_id"));

//item\_nameText.setText(rs.getString("item\_name"));

//costText.setText(rs.getString("cost"));

//available\_quantityText .setText(rs.getString("available\_quantity"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO sales VALUES('" + item\_ID1Text.getText() + "', " + "'" + quantity\_soldText.getText() + "'," + "'"+total\_amountText.getText() + "')";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

item\_ID1Text = new TextField(15);

item\_ID1Text.setEditable(false);

quantity\_soldText = new TextField(15);

total\_amountText = new TextField(15);

//ordertypeText= new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Item ID:"));

first.add(item\_ID1Text);

first.add(new Label("Quantity Sold:"));

first.add(quantity\_soldText);

first.add(new Label("Total Amount:"));

first.add(total\_amountText);

//first.add(new Label("Payment Type:"));

//first.add(ordertypeText);

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertButton);

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);

setLayout(new FlowLayout());

setVisible(true);

}

public void buildGUIEmployee()

{

Frame f= new Frame();

TextField paswrd=new TextField(15);

paswrd.setEchoChar('\*');

//TextArea errorText1=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

//d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password2.equals(paswrd.getText()))

{

d.setVisible(false);

//employee\_IDText,employee\_nameText,employee\_PHNO

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO employee\_list VALUES('" + employee\_IDText.getText() + "', " + "'" + employee\_nameText.getText() + "'," + "'"+employee\_PHNOText.getText() + "')";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

employee\_IDText = new TextField(15);

employee\_nameText = new TextField(15);

employee\_PHNOText = new TextField(15);

//ordertypeText= new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Employee ID:"));

first.add(employee\_IDText);

first.add(new Label("Employee Name:"));

first.add(employee\_nameText);

first.add(new Label("Employee PHNO:"));

first.add(employee\_PHNOText);

//first.add(new Label("Payment Type:"));

//first.add(ordertypeText);

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertButton);

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);

setLayout(new FlowLayout());

setVisible(true);

}

else

{

d.setVisible(false);

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.setLayout(new FlowLayout());

d.setSize(200,200);

d.setVisible(true);

}

private void loadMenu1()

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

menuChoice.add(rs.getString("item\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

/\*private void loadMenu2()

{

try

{

rs = statement.executeQuery("SELECT available\_quantity FROM items\_list"+"WHERE item\_id='"+item\_idText.getText()+"'");

avail\_quan.append(rs.getString("available\_quantity"));

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}\*/

public void buildGUIOrders()

{ removeAll();

menuChoice=new Choice();

loadMenu1();

add(menuChoice);

menuChoice.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

if (rs.getString("item\_id").equals(menuChoice.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

avail\_quan=new StringBuffer();

item\_idText.setText(rs.getString("item\_id"));

//item\_nameText.setText(rs.getString("item\_name"));

//costText.setText(rs.getString("cost"));

//available\_quantityText .setText(rs.getString("available\_quantity"));

rs = statement.executeQuery("SELECT available\_quantity FROM items\_list"+" WHERE item\_id='"+item\_idText.getText()+"'");

if(rs.next())

{

avail\_quan.append(rs.getString("available\_quantity"));

}

avail1.setText(new String(avail\_quan));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO order\_list VALUES('" + order\_idText .getText() + "', " + "'" + item\_idText.getText() + "'," + "'"+ordered\_quantityText.getText() + "')";

String query1="UPDATE items\_list SET available\_quantity=available\_quantity-"+ordered\_quantityText.getText()+"WHERE item\_ID='"+item\_idText.getText()+"'";

//rs = statement.executeQuery("SELECT available\_quantity FROM items\_list"+" WHERE item\_id='"+item\_idText.getText()+"'");

//avail\_quan=new StringBuffer();

// if(rs.next())

//{ //avail\_quan.append(rs.getString("available\_quantity"));

//System.out.println(avail\_quan);

int val1=Integer.parseInt(new String(avail\_quan));

int val2=Integer.parseInt(ordered\_quantityText.getText());

if(val1<val2)

{

errorText.append("ERROR: Available quantity<Ordered Quantity");

}

else

{

int i = statement.executeUpdate(query);

statement.executeUpdate(query1);

errorText.append("\nInserted " + i + " rows successfully");

}

//}

// int i = statement.executeUpdate(query);

//statement.executeUpdate(query1);

//errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

//val1=Integer.parseInt(new String(avail\_quan));

//val2=Integer.parseInt(ordered\_quantityText.getText());

order\_idText = new TextField(15);

item\_idText = new TextField(15);

item\_idText.setEditable(false);

ordered\_quantityText = new TextField(15);

//ordertypeText= new TextField(15);

avail1=new TextField(3);

avail1.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Order ID:"));

first.add(order\_idText);

first.add(new Label("Item ID:"));

first.add(item\_idText);

first.add(new Label("Ordered Quantity:"));

first.add(ordered\_quantityText );

//first.add(new Label("Payment Type:"));

//first.add(ordertypeText);

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertButton);

second.add(new Label("Available"));

second.add(avail1);

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);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadEmployee()

{

try

{

rs = statement.executeQuery("SELECT \* FROM employee\_list");

while (rs.next())

{

employeeList.add(rs.getString("employee\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateGUIEmployee()

{

Frame f= new Frame();

TextField paswrd=new TextField(15);

paswrd.setEchoChar('\*');

//TextArea errorText1=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

//d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password2.equals(paswrd.getText()))

{

d.setVisible(false);

//employee\_IDText,employee\_nameText,employee\_PHNO;

removeAll();

employeeList = new List(6);

loadEmployee();

add(employeeList);

//When a list item is selected populate the text fields

employeeList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM employee\_list");

while (rs.next())

{

if (rs.getString("employee\_id").equals(employeeList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

employee\_IDText.setText(rs.getString("employee\_id"));

employee\_nameText.setText(rs.getString("employee\_name"));

employee\_PHNOText.setText(rs.getString("employee\_PHNO"));

//available\_quantityText .setText(rs.getString("available\_quantity"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

updateButton = new Button("Modify");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE employee\_list "

+ "SET employee\_PHNO=" + employee\_PHNOText.getText()

+ " WHERE employee\_id = '" + employeeList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

employeeList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

//employee\_IDText,employee\_nameText,employee\_PHNOText;

employee\_IDText = new TextField(15);

employee\_IDText.setEditable(false);

employee\_nameText = new TextField(15);

employee\_nameText.setEditable(false);

employee\_PHNOText = new TextField(15);

//available\_quantityText = new TextField(15);

//available\_quantityText .setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Employee ID:"));

first.add(employee\_IDText);

first.add(new Label("Employee Name:"));

first.add(employee\_nameText);

first.add(new Label("Employee PHNO:"));

first.add(employee\_PHNOText);

//first.add(new Label("Quantity available:"));

//first.add(available\_quantityText );

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

else

{

d.setVisible(false);

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.setLayout(new FlowLayout());

d.setSize(200,200);

d.setVisible(true);

}

private void loadSales()

{

try

{

rs = statement.executeQuery("SELECT \* FROM sales");

while (rs.next())

{

salesList.add(rs.getString("item\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateGUISales()

{

//item\_ID1Text, quantity\_soldText, total\_amountText;

removeAll();

salesList = new List(6);

loadSales();

add(salesList);

//When a list item is selected populate the text fields

salesList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM sales");

while (rs.next())

{

if (rs.getString("item\_id").equals(salesList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_ID1Text.setText(rs.getString("item\_id"));

quantity\_soldText.setText(rs.getString("quantity\_sold"));

total\_amountText.setText(rs.getString("total\_amount"));

//available\_quantityText .setText(rs.getString("available\_quantity"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

updateButton = new Button("Modify");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE sales "

+ "SET quantity\_sold=" + quantity\_soldText.getText()+

","+"total\_amount="+ total\_amountText.getText()

+ " WHERE item\_id = '" + salesList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

salesList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

item\_ID1Text = new TextField(15);

item\_ID1Text.setEditable(false);

quantity\_soldText = new TextField(15);

//item\_nameText.setEditable(false);

total\_amountText = new TextField(15);

//available\_quantityText = new TextField(15);

//available\_quantityText .setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Item ID:"));

first.add(item\_ID1Text);

first.add(new Label("Quantity Sold:"));

first.add(quantity\_soldText);

first.add(new Label("Total Amount:"));

first.add(total\_amountText);

//first.add(new Label("Quantity available:"));

//first.add(available\_quantityText );

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadMenu()

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

menuList.add(rs.getString("item\_name"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateMenuGUI()

{

Frame f= new Frame();

TextField paswrd=new TextField(15);

paswrd.setEchoChar('\*');

//TextArea errorText1=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

//d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password.equals(paswrd.getText()))

{

d.setVisible(false);

removeAll();

menuList = new List(6);

loadMenu();

add(menuList);

//When a list item is selected populate the text fields

menuList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

if (rs.getString("item\_name").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_IDText.setText(rs.getString("item\_id"));

item\_nameText.setText(rs.getString("item\_name"));

costText.setText(rs.getString("cost"));

available\_quantityText .setText(rs.getString("available\_quantity"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

updateButton = new Button("Modify");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE items\_list "

+ "SET cost=" + costText.getText()+",available\_quantity="+ available\_quantityText.getText()

+ " WHERE item\_id = '" + item\_IDText.getText() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

menuList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

item\_IDText = new TextField(15);

item\_IDText.setEditable(false);

item\_nameText = new TextField(15);

item\_nameText.setEditable(false);

costText = new TextField(15);

available\_quantityText = new TextField(15);

//available\_quantityText .setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("Item ID:"));

first.add(item\_IDText);

first.add(new Label("Item Name:"));

first.add(item\_nameText);

first.add(new Label("Item cost:"));

first.add(costText);

first.add(new Label("Quantity available:"));

first.add(available\_quantityText );

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

else

{

d.setVisible(false);

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.setLayout(new FlowLayout());

d.setSize(200,200);

d.setVisible(true);

}

public void deleteGUIMenu()

{

Frame f= new Frame();

TextField paswrd=new TextField(15);

paswrd.setEchoChar('\*');

//TextArea errorText1=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

//d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password.equals(paswrd.getText()))

{

d.setVisible(false);

removeAll();

menuList = new List(10);

loadMenu();

add(menuList);

//When a list item is selected populate the text fields

menuList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

if (rs.getString("item\_name").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_IDText.setText(rs.getString("item\_id"));

item\_nameText.setText(rs.getString("item\_name"));

costText.setText(rs.getString("cost"));

available\_quantityText .setText(rs.getString("available\_quantity"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete menu Button

deleteRowButton = new Button("Delete");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM items\_list WHERE item\_id = '" +item\_IDText.getText() +"'");

errorText.append("\nDeleted " + i + " rows successfully");

item\_IDText.setText(null);

item\_nameText.setText(null);

costText.setText(null);

available\_quantityText .setText(null);

menuList.removeAll();

loadMenu();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

item\_IDText = new TextField(15);

item\_nameText = new TextField(15);

costText = new TextField(15);

available\_quantityText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

item\_IDText.setEditable(false);

item\_nameText.setEditable(false);

costText.setEditable(false);

available\_quantityText .setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("Item ID:"));

first.add(item\_IDText);

first.add(new Label("Item Name:"));

first.add(item\_nameText);

first.add(new Label("Item cost:"));

first.add(costText);

first.add(new Label("Quantity available:"));

first.add(available\_quantityText );

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

else

{

d.setVisible(false);

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.setLayout(new FlowLayout());

d.setSize(200,200);

d.setVisible(true);

}

private void loadOrders()

{

try

{

rs = statement.executeQuery("SELECT \* FROM order\_list");

while (rs.next())

{

ordersList.add(rs.getString("order\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateOrdersGUI()

{

removeAll();

ordersList = new List(6);

loadOrders();

add(ordersList);

//StringBuffer s;

//When a list item is selected populate the text fields

ordersList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM order\_list");

while (rs.next())

{

if (rs.getString("order\_id").equals(ordersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

avail\_quan=new StringBuffer();

order\_idText.setText(rs.getString("order\_id"));

item\_idText.setText(rs.getString("item\_id"));

ordered\_quantityText .setText(rs.getString("ordered\_quantity"));

s=new StringBuffer();

s.append(rs.getString("ordered\_quantity"));

System.out.println(s);

//ordertypeText.setText(rs.getString("ordertype"));

rs = statement.executeQuery("SELECT available\_quantity FROM items\_list"+" WHERE item\_id='"+item\_idText.getText()+"'");

if(rs.next())

{

avail\_quan.append(rs.getString("available\_quantity"));

}

avail1.setText(new String(avail\_quan));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

updateButton = new Button("Modify");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

int val1=Integer.parseInt(new String(avail\_quan));

int val2=Integer.parseInt(ordered\_quantityText.getText());

if(val1+Integer.parseInt(new String(s))<val2)

{

errorText.append("ERROR: Available quantity<Ordered Quantity");

}

else

{

String query1="UPDATE items\_list SET available\_quantity=available\_quantity+"+new String(s)+"-"+ordered\_quantityText.getText()+" WHERE item\_ID='"+item\_idText.getText()+"'";

System.out.println(query1);

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE order\_list "

+ "SET ordered\_quantity=" + ordered\_quantityText.getText()

+ " WHERE order\_id = '" + ordersList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

statement.executeUpdate(query1);

ordersList.removeAll();

loadMenu();

}

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

order\_idText = new TextField(15);

order\_idText.setEditable(false);

item\_idText = new TextField(15);

item\_idText.setEditable(false);

ordered\_quantityText = new TextField(15);

//ordertypeText = new TextField(15);

//ordertypeText.setEditable(false);

avail1=new TextField(3);

avail1.setEditable(false);;

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Order ID:"));

first.add(order\_idText);

first.add(new Label("Item ID:"));

first.add(item\_idText);

first.add(new Label("Ordered Quantity:"));

first.add(ordered\_quantityText );

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateButton);

second.add(new Label("Available"));

second.add(avail1);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIOrders()

{

removeAll();

ordersList = new List(10);

loadOrders();

add(ordersList);

//When a list item is selected populate the text fields

ordersList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM order\_list");

while (rs.next())

{

if (rs.getString("order\_id").equals(ordersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

order\_idText.setText(rs.getString("order\_id"));

item\_idText.setText(rs.getString("item\_id"));

ordered\_quantityText .setText(rs.getString("ordered\_quantity"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete orders Button

deleteRowButton = new Button("Delete");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM order\_list WHERE order\_id = '" + ordersList.getSelectedItem()+"'");

errorText.append("\nDeleted " + i + " rows successfully");

order\_idText.setText(null);

item\_idText.setText(null);

ordered\_quantityText .setText(null);

//ordertypeText.setText(null);

ordersList.removeAll();

loadOrders();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

order\_idText = new TextField(15);

item\_idText = new TextField(15);

ordered\_quantityText = new TextField(15);

//ordertypeText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

order\_idText.setEditable(false);

item\_idText.setEditable(false);

ordered\_quantityText .setEditable(false);

//ordertypeText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Order ID:"));

first.add(order\_idText);

first.add(new Label("Item ID:"));

first.add(item\_idText);

first.add(new Label("Ordered Quantity:"));

first.add(ordered\_quantityText );

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIEmployee()

{

Frame f= new Frame();

TextField paswrd=new TextField(15);

paswrd.setEchoChar('\*');

//TextArea errorText1=new TextArea(10,10);

d = new Dialog(f ,"Password", true);

//d.setLayout( new FlowLayout() );

Button b = new Button ("OK");

b.addActionListener ( new ActionListener()

{

public void actionPerformed( ActionEvent e )

{

if(Password2.equals(paswrd.getText()))

{

d.setVisible(false);

//employee\_IDText,employee\_nameText,employee\_PHNOText

removeAll();

employeeList = new List(10);

loadEmployee();

add(employeeList);

//When a list item is selected populate the text fields

employeeList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM employee\_list");

while (rs.next())

{

if (rs.getString("employee\_id").equals(employeeList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

employee\_IDText.setText(rs.getString("employee\_id"));

employee\_nameText.setText(rs.getString("employee\_name"));

employee\_PHNOText.setText(rs.getString("employee\_PHNO"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete orders Button

deleteRowButton = new Button("Delete");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM employee\_list WHERE employee\_id = '" + employeeList.getSelectedItem()+"'");

errorText.append("\nDeleted " + i + " rows successfully");

employee\_IDText.setText(null);

employee\_nameText.setText(null);

employee\_PHNOText.setText(null);

//ordertypeText.setText(null);

employeeList.removeAll();

loadOrders();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

//employee\_IDText,employee\_nameText,employee\_PHNOText

employee\_IDText = new TextField(15);

employee\_nameText = new TextField(15);

employee\_PHNOText = new TextField(15);

//ordertypeText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

employee\_IDText .setEditable(false);

employee\_nameText.setEditable(false);

employee\_PHNOText .setEditable(false);

//ordertypeText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Employee ID:"));

first.add(employee\_IDText);

first.add(new Label("Employee Name:"));

first.add(employee\_nameText);

first.add(new Label("Employee PHNO:"));

first.add(employee\_PHNOText);

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

else

{

d.setVisible(false);

}

}

});

d.add( new Label ("Password"));

d.add(paswrd);

d.add(b);

d.setLayout(new FlowLayout());

d.setSize(200,200);

d.setVisible(true);

}

public void deleteGUISales()

{

//item\_ID1Text, quantity\_soldText, total\_amountText;

removeAll();

salesList = new List(10);

loadSales();

add(salesList);

//When a list item is selected populate the text fields

salesList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM sales");

while (rs.next())

{

if (rs.getString("item\_id").equals(salesList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_ID1Text.setText(rs.getString("item\_id"));

quantity\_soldText.setText(rs.getString("quantity\_sold"));

total\_amountText.setText(rs.getString("total\_amount"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete orders Button

deleteRowButton = new Button("Delete");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM sales WHERE item\_id = '" + salesList.getSelectedItem()+"'");

errorText.append("\nDeleted " + i + " rows successfully");

item\_ID1Text.setText(null);

quantity\_soldText.setText(null);

total\_amountText.setText(null);

//ordertypeText.setText(null);

salesList.removeAll();

loadSales();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

//item\_ID1Text, quantity\_soldText, total\_amountText;

item\_ID1Text = new TextField(15);

quantity\_soldText = new TextField(15);

total\_amountText = new TextField(15);

//ordertypeText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

item\_ID1Text.setEditable(false);

quantity\_soldText.setEditable(false);

total\_amountText .setEditable(false);

//ordertypeText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Item ID:"));

first.add( item\_ID1Text);

first.add(new Label("Quantity Sold:"));

first.add(quantity\_soldText);

first.add(new Label("Total Amount:"));

first.add(total\_amountText);

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadCustomer()

{

try

{

rs = statement.executeQuery("SELECT \* FROM customer");

while (rs.next())

{

CustomerList.add(rs.getString("customer\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateCustomerGUI()

{

removeAll();

CustomerList = new List(10);

loadCustomer();

add(CustomerList);

//When a list item is selected populate the text fields

CustomerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM customer");

while (rs.next())

{

if (rs.getString("customer\_id").equals(CustomerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_IDText .setText(rs.getString("customer\_id"));

customer\_nameText .setText(rs.getString("customer\_name"));

customer\_PHNOText .setText(rs.getString("customer\_PHNO"));

tokenText.setText(rs.getString("token"));

order\_id1Text.setText(rs.getString("order\_id"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

updateButton = new Button("Modify");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE customer "

+ "SET customer\_PHNO=" + customer\_PHNOText .getText()

+ " WHERE customer\_id = '" + CustomerList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

CustomerList.removeAll();

loadCustomer();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customer\_IDText = new TextField(15);

customer\_IDText .setEditable(false);

customer\_nameText = new TextField(15);

customer\_nameText .setEditable(false);

customer\_PHNOText = new TextField(15);

tokenText = new TextField(15);

tokenText.setEditable(false);

order\_id1Text=new TextField(15);

order\_id1Text.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_IDText );

first.add(new Label("Customer Name:"));

first.add(customer\_nameText );

first.add(new Label("Customer PHNO:"));

first.add(customer\_PHNOText );

first.add(new Label("Token:"));

first.add(tokenText);

first.add(new Label("Order ID:"));

first.add(order\_id1Text);

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadPayment()

{

try

{

rs = statement.executeQuery("SELECT \* FROM payment");

while (rs.next())

{

payList.add(rs.getString("customer\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updatePaymentGUI()

{

//customer\_ID1Text,order\_id2Text,pay\_modeText,pay\_IDText,totalText;

removeAll();

payList = new List(10);

loadPayment();

add(payList);

//When a list item is selected populate the text fields

payList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM payment");

while (rs.next())

{

if (rs.getString("customer\_id").equals(payList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_ID1Text .setText(rs.getString("customer\_ID"));

order\_id2Text .setText(rs.getString("order\_ID"));

pay\_modeText .setText(rs.getString("pay\_mode"));

pay\_IDText.setText(rs.getString("pay\_ID"));

//totalText.setText(rs.getString("total"));

order\_quan=new StringBuffer();

I\_ID=new StringBuffer();

Cst=new StringBuffer();

int t;

rs=statement.executeQuery("SELECT ordered\_quantity FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

order\_quan.append(rs.getString("ordered\_quantity"));

}

rs=statement.executeQuery("Select item\_id FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

I\_ID.append(rs.getString("item\_id"));

}

rs=statement.executeQuery("Select cost FROM items\_list WHERE item\_id='"+new String(I\_ID)+"'");

if(rs.next())

{

Cst.append(rs.getString("cost"));

}

t=Integer.parseInt(new String(Cst))\*Integer.parseInt(new String(order\_quan));

totalText.setText(Integer.toString(t));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

updateButton = new Button("Modify");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE payment "

+ "SET pay\_mode=" +"'"+pay\_modeText .getText()+"'"+

",pay\_ID="+"'"+pay\_IDText.getText()+"'"

+ " WHERE customer\_id = '" + payList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

payList.removeAll();

loadCustomer();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

//customer\_ID1Text,order\_id2Text,pay\_modeText,pay\_IDText,totalText;

customer\_ID1Text = new TextField(15);

customer\_ID1Text .setEditable(false);

order\_id2Text = new TextField(15);

order\_id2Text .setEditable(false);

pay\_modeText = new TextField(15);

pay\_IDText = new TextField(15);

//pay\_IDText.setEditable(false);

totalText=new TextField(15);

totalText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_ID1Text );

first.add(new Label("Order ID:"));

first.add(order\_id2Text);

first.add(new Label("Pay Mode:"));

first.add(pay\_modeText);

first.add(new Label("Payment ID:"));

first.add(pay\_IDText);

first.add(new Label("Total:"));

first.add(totalText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUICustomer()

{

removeAll();

CustomerList = new List(10);

loadCustomer();

add(CustomerList);

//When a list item is selected populate the text fields

CustomerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM customer");

while (rs.next())

{

if (rs.getString("customer\_id").equals(CustomerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_IDText .setText(rs.getString("customer\_id"));

customer\_nameText .setText(rs.getString("customer\_name"));

customer\_PHNOText .setText(rs.getString("customer\_PHNO"));

tokenText.setText(rs.getString("token"));

order\_id1Text.setText(rs.getString("order\_id"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete Customer Button

deleteRowButton = new Button("Delete");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM Customer WHERE customer\_id = '" + CustomerList.getSelectedItem()+"'");

errorText.append("\nDeleted " + i + " rows successfully");

customer\_IDText .setText(null);

customer\_nameText .setText(null);

customer\_PHNOText .setText(null);

tokenText.setText(null);

order\_id1Text.setText(null);

CustomerList.removeAll();

loadCustomer();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

customer\_IDText = new TextField(15);

customer\_nameText = new TextField(15);

customer\_PHNOText = new TextField(15);

tokenText = new TextField(15);

order\_id1Text= new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

customer\_IDText .setEditable(false);

customer\_nameText .setEditable(false);

customer\_PHNOText .setEditable(false);

tokenText.setEditable(false);

order\_id1Text.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_IDText );

first.add(new Label("Customer Name:"));

first.add(customer\_nameText );

first.add(new Label("Customer PHNO:"));

first.add(customer\_PHNOText );

first.add(new Label("Token:"));

first.add(tokenText);

first.add(new Label("Order ID:"));

first.add(order\_id1Text);

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIPayment()

{

//customer\_ID1Text,order\_id2Text,pay\_modeText,pay\_IDText,totalText;

removeAll();

payList = new List(10);

loadPayment();

add(payList);

//When a list item is selected populate the text fields

payList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM payment");

while (rs.next())

{

if (rs.getString("customer\_id").equals(payList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_ID1Text .setText(rs.getString("customer\_id"));

order\_id2Text .setText(rs.getString("order\_id"));

pay\_modeText.setText(rs.getString("pay\_mode"));

pay\_IDText.setText(rs.getString("pay\_id"));

//totalText.setText(rs.getString("total"));

order\_quan=new StringBuffer();

I\_ID=new StringBuffer();

Cst=new StringBuffer();

int t;

rs=statement.executeQuery("SELECT ordered\_quantity FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

order\_quan.append(rs.getString("ordered\_quantity"));

}

rs=statement.executeQuery("Select item\_id FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

I\_ID.append(rs.getString("item\_id"));

}

rs=statement.executeQuery("Select cost FROM items\_list WHERE item\_id='"+new String(I\_ID)+"'");

if(rs.next())

{

Cst.append(rs.getString("cost"));

}

t=Integer.parseInt(new String(Cst))\*Integer.parseInt(new String(order\_quan));

totalText.setText(Integer.toString(t));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete Customer Button

deleteRowButton = new Button("Delete");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM payment WHERE customer\_id = '" + payList.getSelectedItem()+"'");

errorText.append("\nDeleted " + i + " rows successfully");

customer\_ID1Text .setText(null);

order\_id2Text .setText(null);

pay\_modeText .setText(null);

pay\_IDText.setText(null);

totalText.setText(null);

payList.removeAll();

loadCustomer();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

customer\_ID1Text = new TextField(15);

order\_id2Text = new TextField(15);

pay\_modeText = new TextField(15);

pay\_IDText= new TextField(15);

totalText= new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

customer\_ID1Text .setEditable(false);

order\_id2Text .setEditable(false);

pay\_modeText .setEditable(false);

pay\_IDText.setEditable(false);

totalText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_ID1Text );

first.add(new Label("Order ID:"));

first.add(order\_id2Text );

first.add(new Label("Payment Mode:"));

first.add(pay\_modeText);

first.add(new Label("Payment ID:"));

first.add(pay\_IDText);

first.add(new Label("Total:"));

first.add(totalText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewMenuGUI()

{

removeAll();

menuList = new List(10);

loadMenu();

add(menuList);

//When a list item is selected populate the text fields

menuList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM items\_list");

while (rs.next())

{

if (rs.getString("item\_name").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_IDText.setText(rs.getString("item\_id"));

item\_nameText.setText(rs.getString("item\_name"));

costText.setText(rs.getString("cost"));

available\_quantityText .setText(rs.getString("available\_quantity"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

/\*updateButton = new Button("Update Menu");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE menu "

+ "SET dishprice=" + costText.getText()

+ " WHERE dishid = '" + menuList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

menuList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

item\_IDText = new TextField(15);

item\_IDText.setEditable(false);

item\_nameText = new TextField(15);

item\_nameText.setEditable(false);

costText = new TextField(15);

costText.setEditable(false);

available\_quantityText = new TextField(15);

available\_quantityText .setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("Item ID:"));

first.add(item\_IDText);

first.add(new Label("Item Name:"));

first.add(item\_nameText);

first.add(new Label("Item cost:"));

first.add(costText);

first.add(new Label("Quantity available:"));

first.add(available\_quantityText );

Panel second = new Panel(new GridLayout(4, 1));

//second.add(updateButton);

Panel third = new Panel();

//third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewPaymentGUI()

{

//customer\_ID1Text,order\_id2Text,pay\_modeText,pay\_IDText,totalText

removeAll();

payList = new List(6);

loadPayment();

add(payList);

//When a list item is selected populate the text fields

payList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM payment");

while (rs.next())

{

if (rs.getString("customer\_id").equals(payList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_ID1Text .setText(rs.getString("customer\_id"));

order\_id2Text .setText(rs.getString("order\_id"));

pay\_modeText .setText(rs.getString("pay\_mode"));

pay\_IDText.setText(rs.getString("pay\_id"));

//totalText.setText(rs.getString("total"));

order\_quan=new StringBuffer();

I\_ID=new StringBuffer();

Cst=new StringBuffer();

int t;

rs=statement.executeQuery("SELECT ordered\_quantity FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

order\_quan.append(rs.getString("ordered\_quantity"));

}

rs=statement.executeQuery("Select item\_id FROM order\_list WHERE order\_id='"+order\_id2Text.getText()+"'");

if(rs.next())

{

I\_ID.append(rs.getString("item\_id"));

}

rs=statement.executeQuery("Select cost FROM items\_list WHERE item\_id='"+new String(I\_ID)+"'");

if(rs.next())

{

Cst.append(rs.getString("cost"));

}

t=Integer.parseInt(new String(Cst))\*Integer.parseInt(new String(order\_quan));

totalText.setText(Integer.toString(t));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

/\*//Handle Update Menu Button

updateButton = new Button("Update Customer");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE Customer "

+ "SET Customerprice=" + customer\_PHNOText .getText()

+ " WHERE Customerid = '" + CustomerList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

CustomerList.removeAll();

loadCustomer();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

//customer\_ID1Text,order\_id2Text,pay\_modeText,pay\_IDText,totalText

customer\_ID1Text = new TextField(15);

customer\_ID1Text .setEditable(false);

order\_id2Text = new TextField(15);

order\_id2Text .setEditable(false);

pay\_modeText = new TextField(15);

pay\_modeText .setEditable(false);

pay\_IDText = new TextField(15);

pay\_IDText.setEditable(false);

totalText=new TextField(15);

totalText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_ID1Text );

first.add(new Label("Order ID:"));

first.add(order\_id2Text );

first.add(new Label("Payment Mode:"));

first.add(pay\_modeText );

first.add(new Label("Payment ID:"));

first.add(pay\_IDText);

first.add(new Label("Total:"));

first.add(totalText);

Panel second = new Panel(new GridLayout(4, 1));

//second.add(updateButton);

Panel third = new Panel();

//third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewCustomerGUI()

{

//customer\_IDText, customer\_nameText, customer\_PHNOText, tokenText, order\_id1Text;

removeAll();

CustomerList = new List(6);

loadCustomer();

add(CustomerList);

//When a list item is selected populate the text fields

CustomerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM customer");

while (rs.next())

{

if (rs.getString("customer\_id").equals(CustomerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_IDText .setText(rs.getString("customer\_id"));

customer\_nameText .setText(rs.getString("customer\_name"));

customer\_PHNOText .setText(rs.getString("customer\_PHNO"));

tokenText.setText(rs.getString("token"));

order\_id1Text.setText(rs.getString("order\_id"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

/\*//Handle Update Menu Button

updateButton = new Button("Update Customer");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE Customer "

+ "SET Customerprice=" + customer\_PHNOText .getText()

+ " WHERE Customerid = '" + CustomerList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

CustomerList.removeAll();

loadCustomer();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

customer\_IDText = new TextField(15);

customer\_IDText .setEditable(false);

customer\_nameText = new TextField(15);

customer\_nameText .setEditable(false);

customer\_PHNOText = new TextField(15);

customer\_PHNOText .setEditable(false);

tokenText = new TextField(15);

tokenText.setEditable(false);

order\_id1Text=new TextField(15);

order\_id1Text.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Customer ID:"));

first.add(customer\_IDText );

first.add(new Label("Customer Name:"));

first.add(customer\_nameText );

first.add(new Label("Customer PHNO:"));

first.add(customer\_PHNOText );

first.add(new Label("Token:"));

first.add(tokenText);

first.add(new Label("Order ID:"));

first.add(order\_id1Text);

Panel second = new Panel(new GridLayout(4, 1));

//second.add(updateButton);

Panel third = new Panel();

//third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewOrdersGUI()

{

removeAll();

ordersList = new List(6);

loadOrders();

add(ordersList);

//When a list item is selected populate the text fields

ordersList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM order\_list");

while (rs.next())

{

if (rs.getString("order\_id").equals(ordersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

order\_idText.setText(rs.getString("order\_id"));

item\_idText.setText(rs.getString("item\_id"));

ordered\_quantityText .setText(rs.getString("ordered\_quantity"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

/\*//Handle Update Menu Button

updateButton = new Button("Update Order");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE menu "

+ "SET orderprice=" + ordered\_quantityText .getText()

+ " WHERE orderid = '" + ordersList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

ordersList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

order\_idText = new TextField(15);

order\_idText.setEditable(false);

item\_idText = new TextField(15);

item\_idText.setEditable(false);

ordered\_quantityText = new TextField(15);

ordered\_quantityText .setEditable(false);

//ordertypeText = new TextField(15);

//ordertypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Order ID:"));

first.add(order\_idText);

first.add(new Label("Item ID:"));

first.add(item\_idText);

first.add(new Label("Ordered quantity:"));

first.add(ordered\_quantityText );

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

//second.add(updateButton);

Panel third = new Panel();

//third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewSalesGUI()

{ //item\_ID1Text, quantity\_soldText, total\_amountText

removeAll();

salesList = new List(6);

loadSales();

add(salesList);

//When a list item is selected populate the text fields

salesList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM sales");

while (rs.next())

{

if (rs.getString("item\_id").equals(salesList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

item\_ID1Text.setText(rs.getString("item\_id"));

quantity\_soldText.setText(rs.getString("quantity\_sold"));

total\_amountText .setText(rs.getString("total\_amount"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

/\*//Handle Update Menu Button

updateButton = new Button("Update Order");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE menu "

+ "SET orderprice=" + ordered\_quantityText .getText()

+ " WHERE orderid = '" + ordersList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

ordersList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

//item\_ID1Text, quantity\_soldText, total\_amountText

item\_ID1Text = new TextField(15);

item\_ID1Text.setEditable(false);

quantity\_soldText = new TextField(15);

quantity\_soldText.setEditable(false);

total\_amountText= new TextField(15);

total\_amountText.setEditable(false);

//ordertypeText = new TextField(15);

//ordertypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Item ID:"));

first.add(item\_ID1Text);

first.add(new Label("Quantity Sold:"));

first.add(quantity\_soldText);

first.add(new Label("Total Amount:"));

first.add(total\_amountText );

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

//second.add(updateButton);

Panel third = new Panel();

//third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewEmployeeGUI()

{ // employee\_IDText,employee\_nameText,employee\_PHNOText;

removeAll();

employeeList = new List(6);

loadEmployee();

add(employeeList);

//When a list item is selected populate the text fields

employeeList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM employee\_list");

while (rs.next())

{

if (rs.getString("employee\_id").equals(employeeList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

employee\_IDText.setText(rs.getString("employee\_id"));

employee\_nameText.setText(rs.getString("employee\_name"));

employee\_PHNOText.setText(rs.getString("employee\_PHNO"));

//ordertypeText.setText(rs.getString("ordertype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

/\*//Handle Update Menu Button

updateButton = new Button("Update Order");

updateButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE menu "

+ "SET orderprice=" + ordered\_quantityText .getText()

+ " WHERE orderid = '" + ordersList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

ordersList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

//employee\_IDText,employee\_nameText,employee\_PHNOText

employee\_IDText = new TextField(15);

employee\_IDText.setEditable(false);

employee\_nameText = new TextField(15);

employee\_nameText.setEditable(false);

employee\_PHNOText= new TextField(15);

employee\_PHNOText.setEditable(false);

//ordertypeText = new TextField(15);

//ordertypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(3, 2));

first.add(new Label("Employee ID:"));

first.add(employee\_IDText);

first.add(new Label("Employee Name:"));

first.add(employee\_nameText);

first.add(new Label("Employee PHNO:"));

first.add(employee\_PHNOText);

//first.add(new Label("Order Type:"));

//first.add(ordertypeText);

Panel second = new Panel(new GridLayout(4, 1));

//second.add(updateButton);

Panel third = new Panel();

//third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void displaySQLErrors(SQLException e)

{

errorText.append("\nSQLException: " + e.getMessage() + "\n");

errorText.append("SQLState: " + e.getSQLState() + "\n");

errorText.append("VendorError: " + e.getErrorCode() + "\n");

}

public void paint(Graphics g)

{

Font f= new Font("Arial",Font.BOLD,22);

g.setFont(f);

g.drawString("BAKERY MANAGEMENT SYSTEM",100,450);

}

public static void main(String[] args)

{

InsertTables it = new InsertTables();

it.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

it.buildFrame();

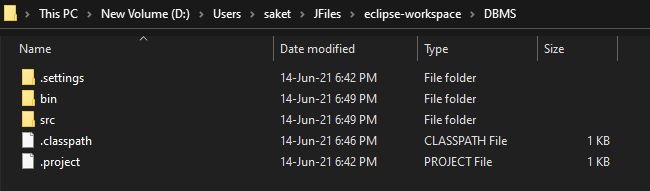
}

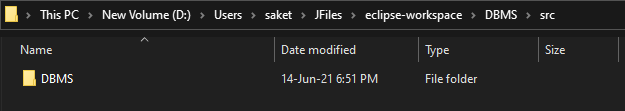
}

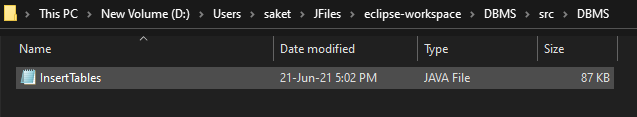
**Github links and folder structure**:

Link:

Folder structure:

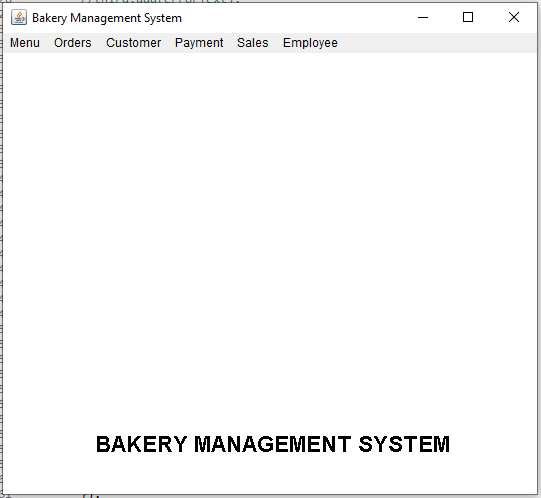


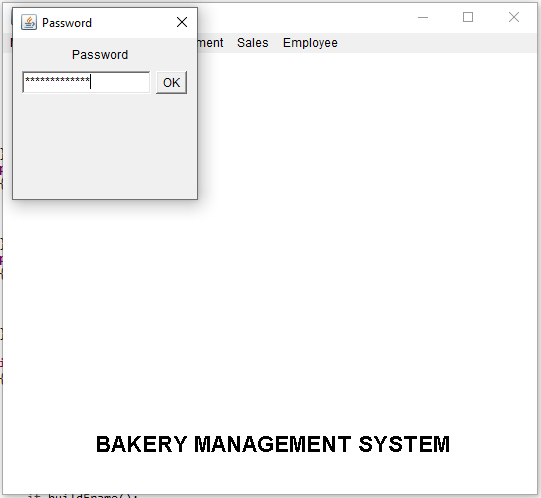


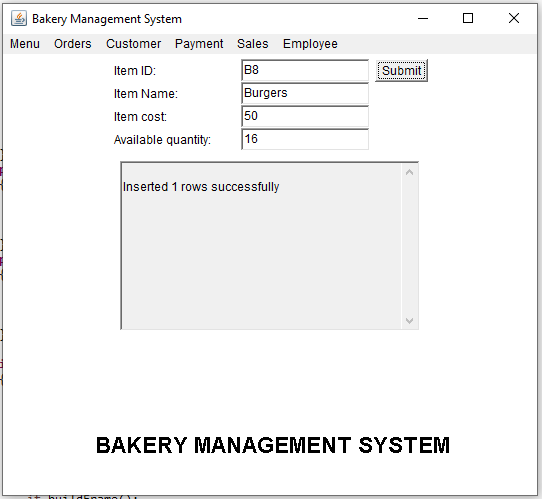


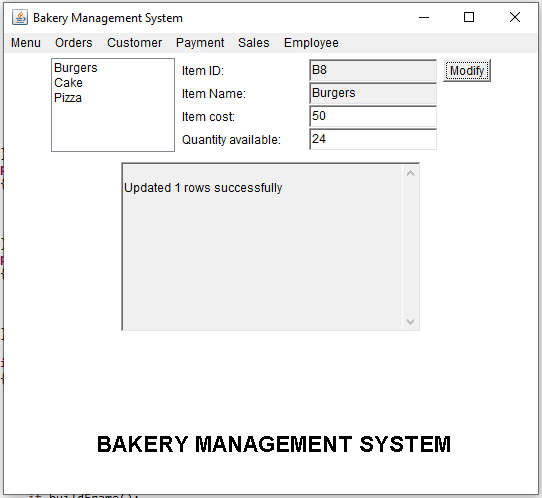
# **TESTING**

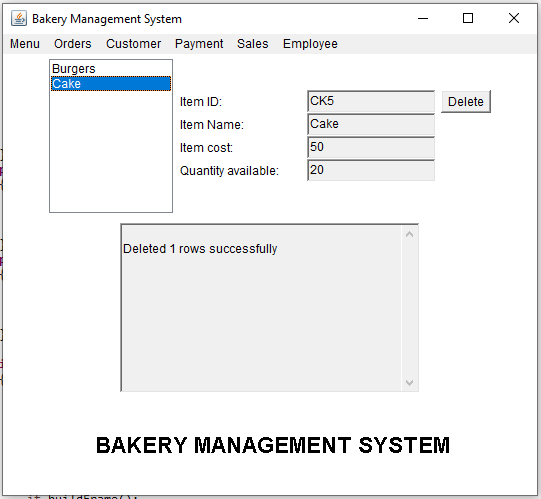
Output Screenshots:

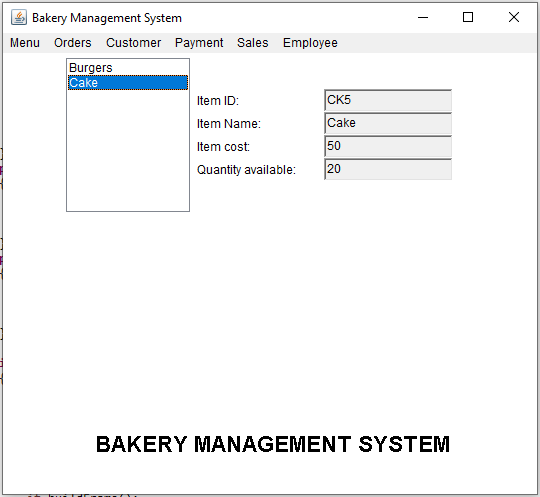


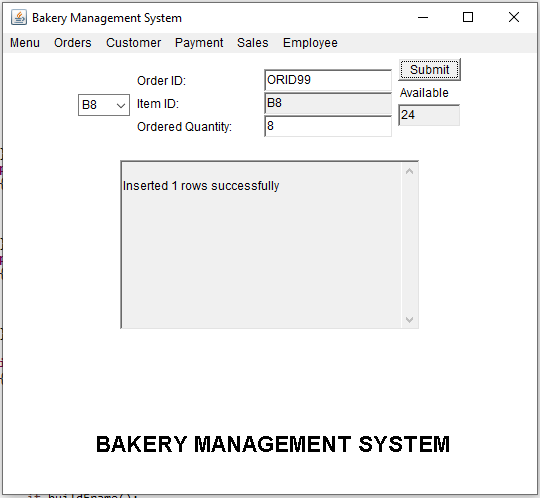


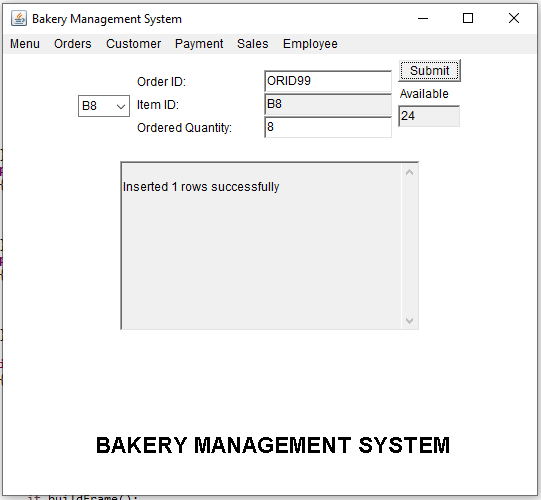


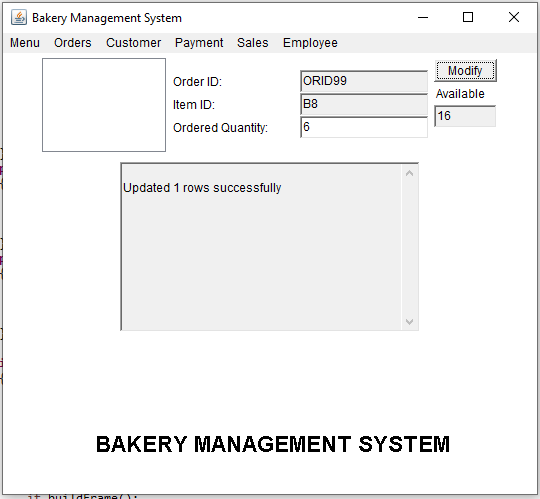


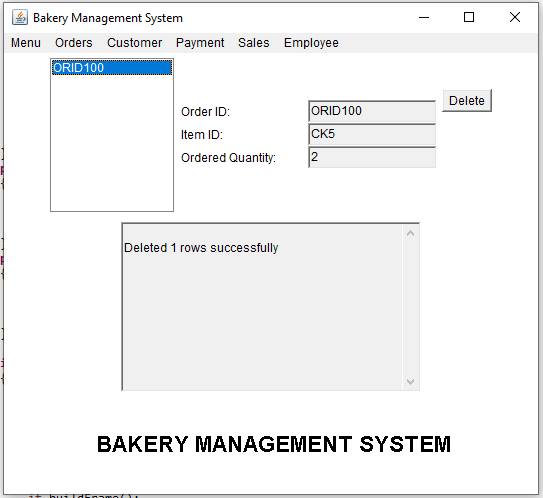


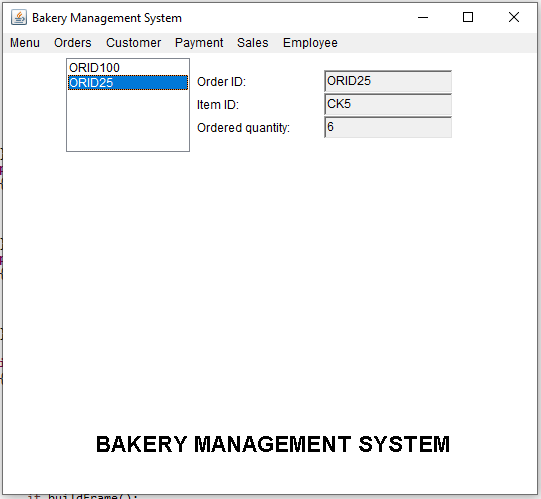


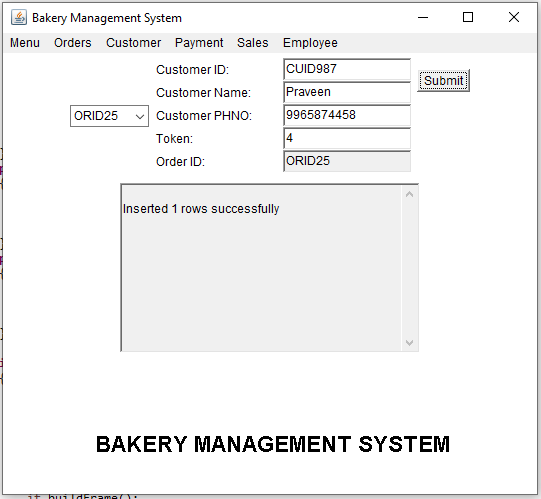


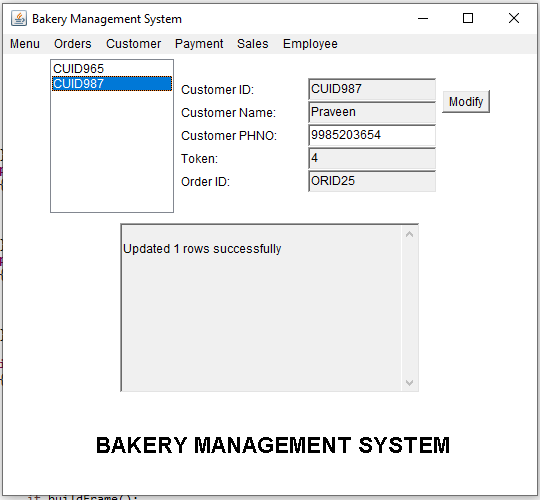


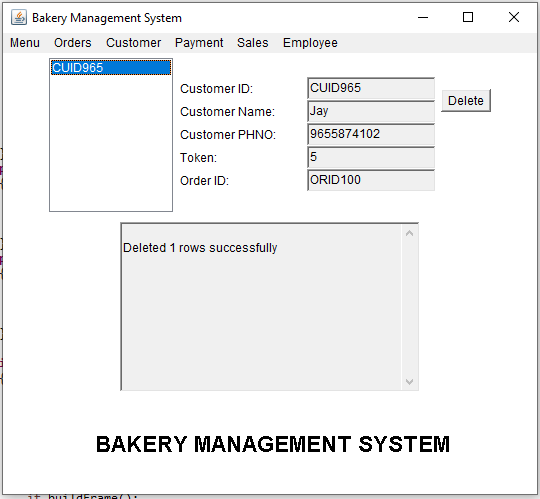


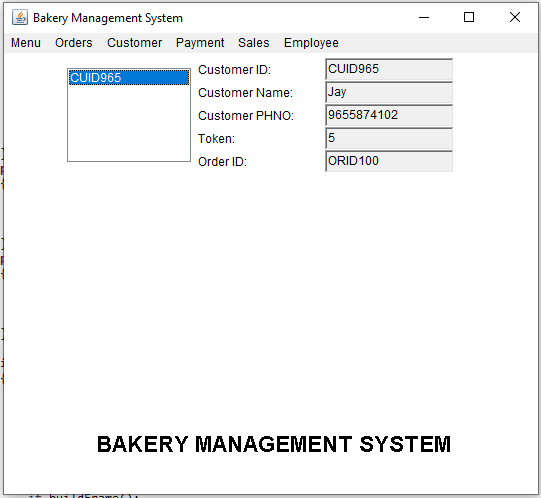


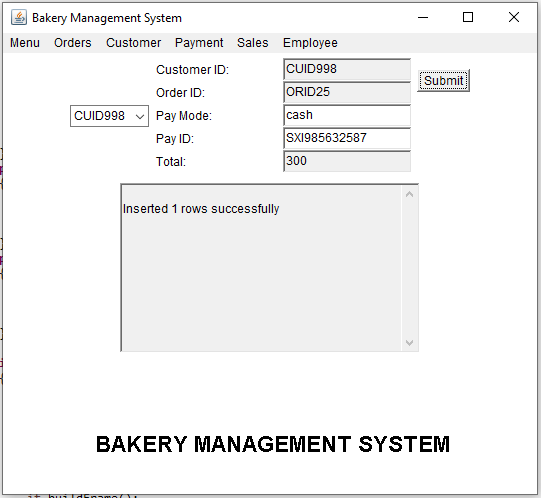


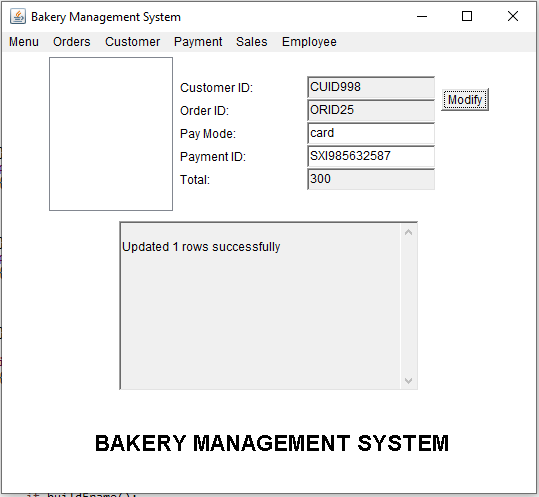


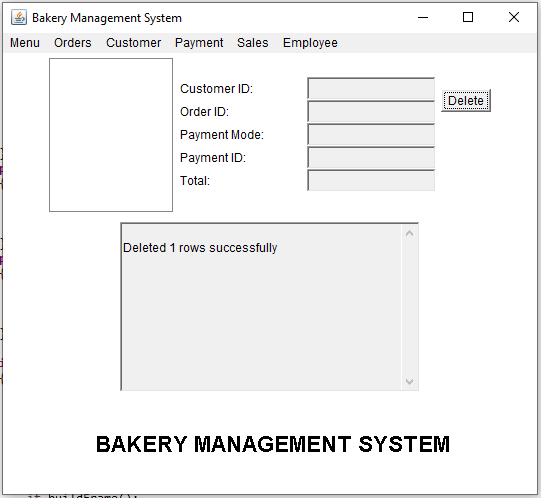


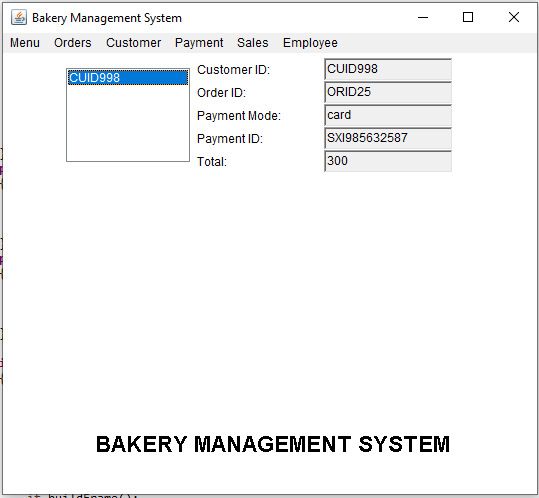


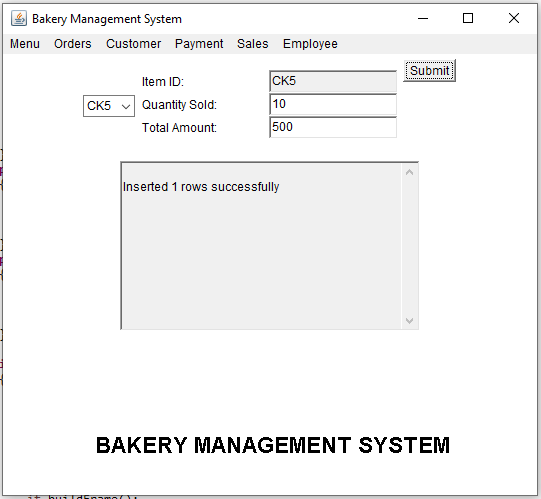


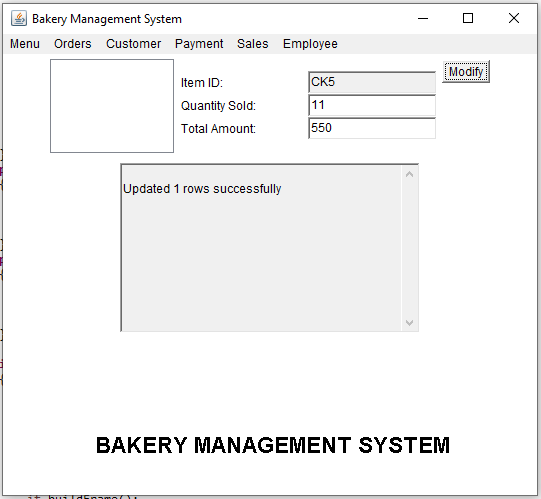


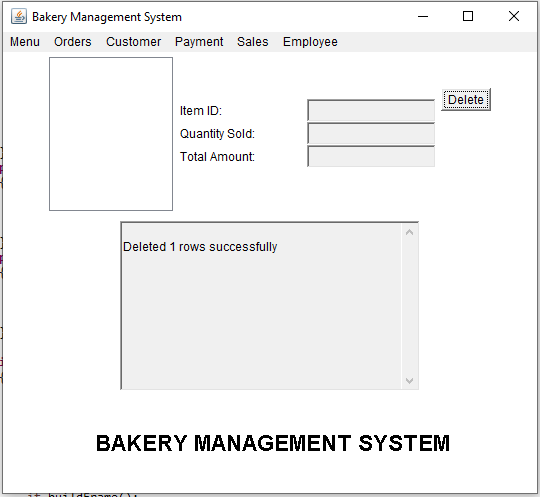


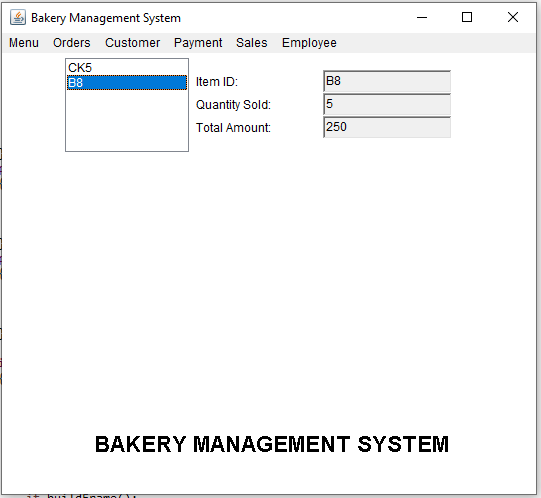


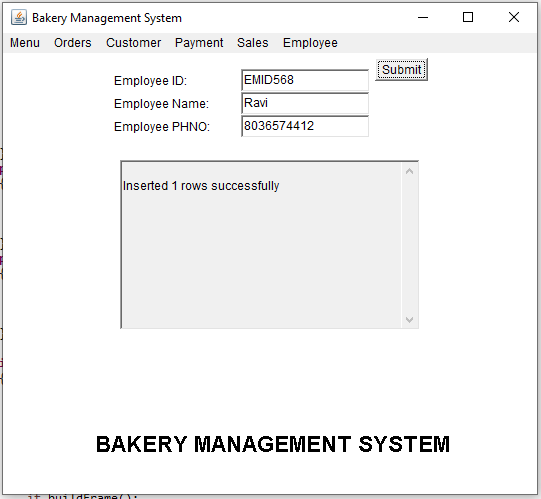


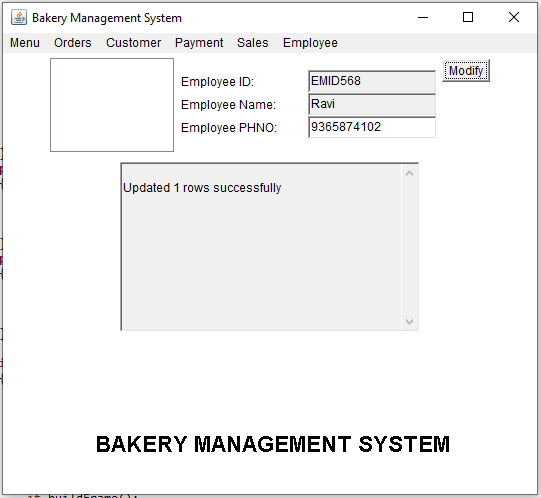


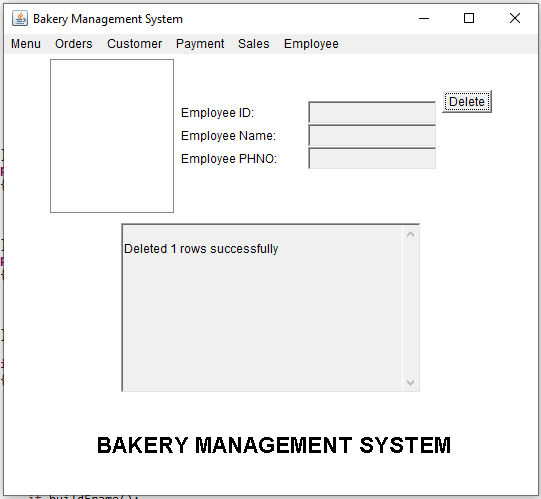


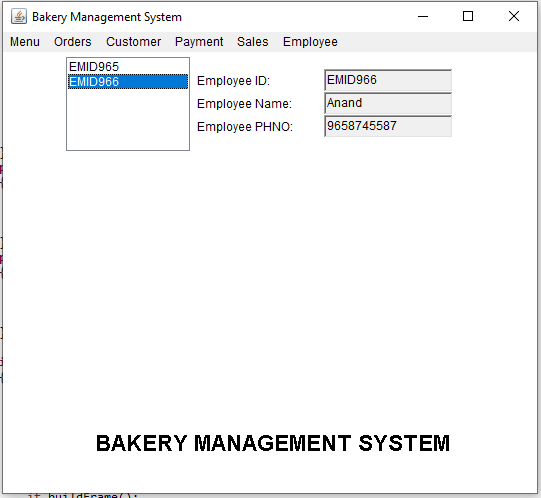






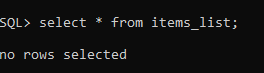




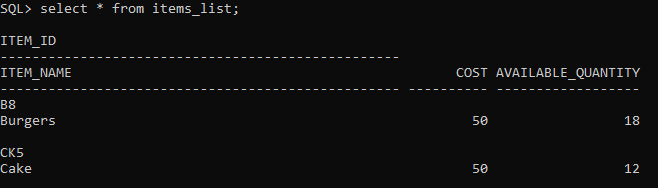


## **DATABASE OUTPUTS:**

BEFORE



AFTER



## **DISCUSSION AND FUTURE WORK**

So far this project has helped us to order the customer choice and store the details. And employee details are also stored.

In future we try to include an option which generates unique order ID, Item ID, Customer Id. Also the stock will be automatically updated.

# **REFERENCES:**

1. Abraham Silberschatz, Henry F Korth, S. Sudarshan, Database System Concepts, 6th Edition, McGraw-Hill International Edition, 2010.
2. [https://docs.oracle.com/cd/E11882\_01/server.112/e40540/intro.htm#CNCPT001](https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm)