**JAVA SWINGS BASED- Online bmi Management System.**

**- SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfillment of the Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

# INFORMATION TECHNOLOGY

By

B.Sri Kanishka Reddy<1602-19-737-111>

Under the guidance of Ms B. Leelavathy Madam.



Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

# BONAFIDE CERTIFICATE

This is to certify that this project report titled ‘**ONLINE BMI MANAGEMENT SYSTEM**’ is a project work of Mr. B Sri Kanishka Reddy bearing roll no. 1602-19-737-111 who carried out the project under my supervision in the IV semester for the academic year 2020- 2021.

Signature Signature

Internal Examiner External Examiner

**ONLINE BMI**

**MANAGEMENT SYSTEM**

**B SRI KANISHKA REDDY**

**1602-19-737-111**

**IT-B**

**ABSTRACT**

In the midst of the COVID-19 pandemic, eating healthy food remains an important part of maintaining your health. While there are no specific foods that can help protect you from the virus, a nutritious diet can boost your immune system or help you fight off symptoms. You may not be able to share meals with friends and loved ones, but there are lots of other ways to eat well and support your health at this difficult time.

Eating a healthy diet is not about strict limitations, staying unrealistically thin, or depriving yourself of the foods you love. Rather, it’s about feeling great, having more energy, improving your health, and boosting your mood.

Healthy eating doesn’t have to be overly complicated. If you feel overwhelmed by all the conflicting nutrition and diet advice out there, you’re not alone. It seems that for every expert who tells you a certain food is good for you, you’ll find another saying exactly the opposite. The truth is that while some specific foods or nutrients have been shown to have a beneficial effect on mood, it’s your overall dietary pattern that is most important. The cornerstone of a healthy diet should be to replace processed food with real food whenever possible. Eating food that is as close as possible to the way nature made it can make a huge difference to the way you think, look, and feel.

## ARCHITECTURE AND TECHNOLOGY

**Software used:** Java Eclipse, Oracle 11g Database, Java SE version 13, SQL\*Plus.

**Java SWINGS:**

**Java SWINGS** is an API to develop GUI or window-based applications in java. Java SWING components are platform-independent. It is lightweight. The javax.swing package provides classes for SWING API such as JTextField, JLabel, JTextArea, JRadioButton, JCheckBox, JChoice, JList etc.

**SQL:**

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

**REQUIREMENT ANALYSIS:**

**List of Tables:**

* Login
* customer\_login
* Customer\_account
* Orders\_From
* Selects
* Menu
* Customer\_order
* Order\_Details

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

**Login:**

* Login\_id: number(5)
* Name: varchar2(20)
* Password: varchar2(20)

**customer\_login:**

* Login\_id: number(5)
* Cust\_id: number(5)
* Date: DATE

**customer\_account:**

* Cust\_id: number(5)
* Cust\_name: varchar2(20)
* ph\_no: number(10)
* email: varchar2(20)
* address: varchar2(20)

**orders\_from:**

* Cust\_id: number(5)
* Bid: number(5)
* Date: DATE

**BMI:**

* Bid: number(5)
* Menu\_list: varchar2(20)

**Selects:**

* Bid: number(5)
* Pid: number(5)

**Menu:**

* Dish\_id: number(5)
* Dishname: varchar(20)
* Price: number(5)
* Type: varchar(20)

**Cust\_order:**

* Cust\_id: number(5)
* Date: DATE
* Order\_id: number(5)

**Order\_details:**

* Order\_id: number(5)
* totalCost: number(10)
* status: varchar2(20)

**ER DIAGRAM**

**Diagram

Description automatically generated**

**MAPPING CARDINALITY AND PARTICIPATION CONSTRAINTS**

A customer can have a single login\_id. Therefore, one to one mapping cardinality between login and customer account.

A customer can order from any number of dishes and BMIs can be searched by any number of users. So, one to many mapping cardinality between customer account and dishes.

A single BMI can contain any number of cuisines. Therefore, one to many mapping cardinality between BMI and cuisine.

A single customer can have any number of order details but all orders of one user are considered as one. So, one to many mapping cardinality between customer account and order details.

**DDL COMMANDS:**

create table **login**(

login\_id number(5),

name varchar2(20),

password varchar2(20),

primary key (login\_id));

create table **cust\_account**(

cust\_id number(5),

cust\_name varchar2(20),

ph\_no number(10),

email varchar2(20),

address varchar2(20),

primary key (cust\_id));

create table **customer\_login**(

login\_id number(5),

cust\_id number(5),

primary key(login\_id,cust\_id),

foreign key(login\_id) references login(login\_id),

foreign key(cust\_id) references cust\_account(cust\_id));

create table **BMI**(

Bid number(5),

Rname varchar2(20),

products\_list varchar2(50),

primary key(Bid));

create table **orders\_from**(

cust\_id number(5),

Bid number(5),

day date,

primary key(cust\_id,Bid),

foreign key(cust\_id) references cust\_account(cust\_id),

foreign key(Bid) references BMI(Bid));

create table **menu**(

dish\_id number(5),

dish\_name varchar2(20),

price number(5),

type varchar2(20),

primary key(dish\_id));

create table **selects**(

dish\_id number(5),

Bid number(5),

primary key(Bid,dish\_id),

foreign key(Bid) references BMI(Bid),

foreign key(dish\_id) references menu(dish\_id));

create table **order\_details**(

order\_id number(5),

totalCost number(10),

status varchar2(20),

primary key(order\_id));

create table **cust\_order**(

cust\_id number(5),

order\_id number(5),

day date,

primary key(cust\_id,order\_id),

foreign key(cust\_id) references cust\_account(cust\_id),

foreign key(order\_id) references order\_details(order\_id));

Text

Description automatically generated

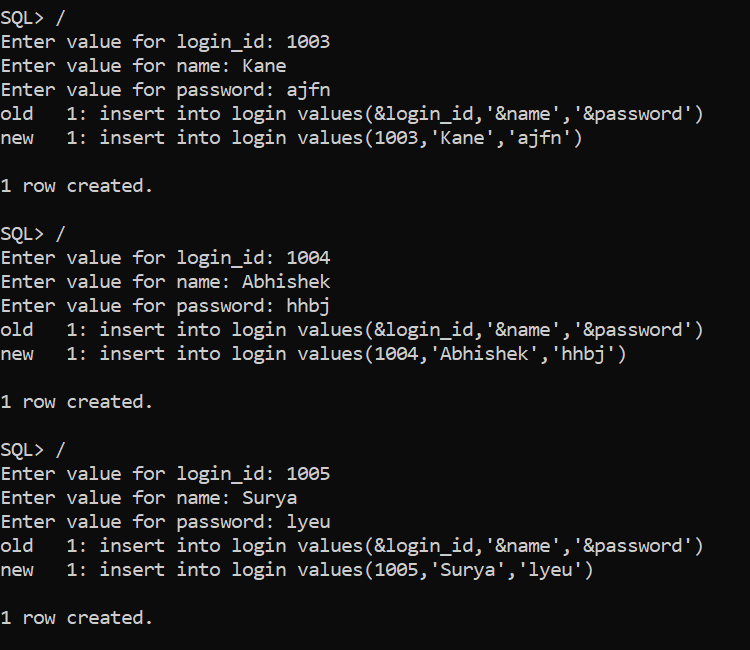
Text

Description automatically generated

**DML COMMANDS**

**Text

Description automatically generated**

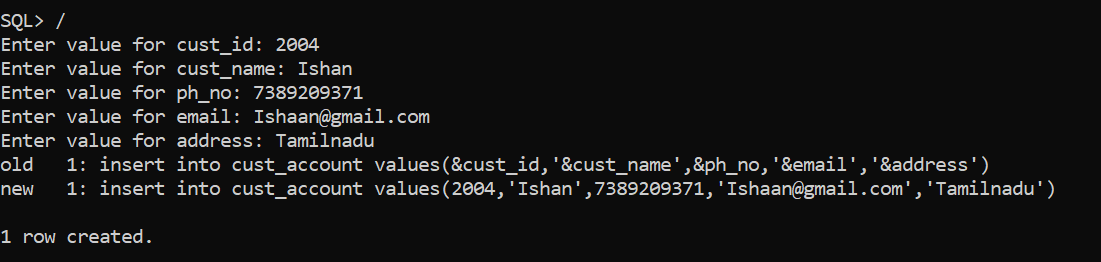
****

**Text

Description automatically generated**

**Text

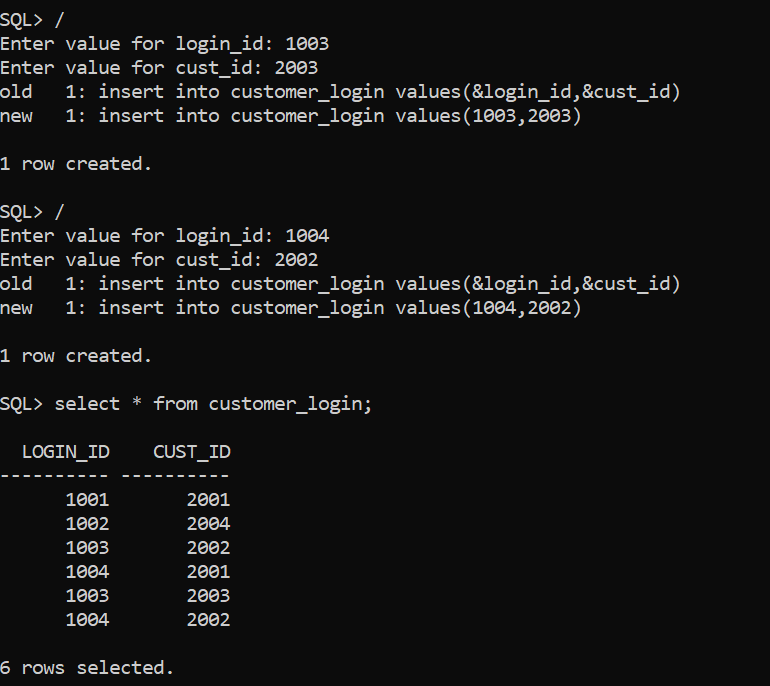
Description automatically generated**

****

****

**Text

Description automatically generated**

****

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**A screenshot of a computer

Description automatically generated with low confidence**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

Text

Description automatically generated **Text

Description automatically generated**

## 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","it19737121","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.

# LOGINPAGE (MAIN METHOD):

package dbms;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

@SuppressWarnings("serial")

public class Insert extends Frame implements ActionListener

{

MenuBar mb;

MenuItem m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16;

Menu menu,bmi,orders,customerAccount;

Button insertButton;

TextField dishidText, dishnameText, dishpriceText, dishtypeText;

TextField bmiidText, bmicategoryText, bmicountText, bmidietText;

TextField orderidText, ordernameText, orderpriceText, ordertypeText;

TextArea errorText;

Connection connection;

Statement statement;

//For updates

Button updateButton,submit;

List menuList,ordersList,bmiList,customerList;

ResultSet rs;

//TextField dishidText, dishnameText, dishpriceText, dishtypeText;

//For delete

Button deleteRowButton,Delete,modify;

TextField customeridText,customerNameText,customerPhoneNumberText,customerMailText,customerAddressText;

public Insert()

{

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","kanishka","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("Online bmi 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);

//bmi

bmi=new Menu("bmi");

m5=new MenuItem("Insert bmi");

m6=new MenuItem("Update bmi");

m7=new MenuItem("Delete bmi");

m8=new MenuItem("View bmi");

bmi.add(m5);

bmi.add(m6);

bmi.add(m7);

bmi.add(m8);

mb.add(bmi);

//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);

customerAccount=new Menu("Customer Account");

m13=new MenuItem("Insert Customer Details");

m14=new MenuItem("Update Customer Details");

m15=new MenuItem("Delete Customer Details");

customerAccount.add(m13);

customerAccount.add(m14);

customerAccount.add(m15);

mb.add(customerAccount);

//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);

}

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 bmi"))

this.buildGUIbmi();

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

this.updatebmiGUI();

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

this.deleteGUIbmi();

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

this.viewbmiGUI();

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 Customer Details"))

this.buildGUICustomerDetails();

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

this.updateCustomerAccount();

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

this.deleteCustomerDetails();

}

public void buildGUIMenu()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO menu VALUES('" + dishidText.getText() + "', " + "'" + dishnameText.getText() + "'," + dishpriceText.getText() + ",'" + dishtypeText.getText() + "')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

dishidText = new TextField(15);

dishnameText = new TextField(15);

dishpriceText = new TextField(15);

dishtypeText= 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("Dish ID:"));

first.add(dishidText);

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

first.add(dishnameText);

first.add(new Label("Dish Price:"));

first.add(dishpriceText);

first.add(new Label("Dish Type:"));

first.add(dishtypeText);

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 buildGUIbmi()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO bmi VALUES('" + bmiidText.getText() + "', " + "'" + bmicategoryText.getText() + "'," + bmicountText.getText() + ",'" + bmidietText.getText() + "')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

bmiidText = new TextField(15);

bmicategoryText = new TextField(15);

bmicountText = new TextField(15);

bmidietText= 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("bmi ID:"));

first.add(bmiidText);

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

first.add(bmicategoryText);

first.add(new Label("bmi weight:"));

first.add(bmicountText);

first.add(new Label("bmi Type:"));

first.add(bmidietText);

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 buildGUIOrders()

{ removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO Orders VALUES('" + orderidText.getText() + "', " + "'" + ordernameText.getText() + "'," + orderpriceText.getText() + ",'" + ordertypeText.getText() + "')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

orderidText = new TextField(15);

ordernameText = new TextField(15);

orderpriceText = new TextField(15);

ordertypeText= 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("Order ID:"));

first.add(orderidText);

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

first.add(ordernameText);

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

first.add(orderpriceText);

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);

}

private void loadMenu()

{

try

{

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

while (rs.next())

{

menuList.add(rs.getString("dishid"));

}

}

catch (SQLException e)

{

//displaySQLErrors(e);

}

}

public void updateMenuGUI()

{

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 menu");

while (rs.next())

{

if (rs.getString("dishid").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

dishidText.setText(rs.getString("dishid"));

dishnameText.setText(rs.getString("dishname"));

dishpriceText.setText(rs.getString("dishprice"));

dishtypeText.setText(rs.getString("dishtype"));

}

}

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 menu "

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

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

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

menuList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

dishidText = new TextField(15);

dishidText.setEditable(false);

dishnameText = new TextField(15);

dishnameText.setEditable(false);

dishpriceText = new TextField(15);

dishtypeText = new TextField(15);

dishtypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(dishidText);

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

first.add(dishnameText);

first.add(new Label("Dish Price:"));

first.add(dishpriceText);

first.add(new Label("Dish Type:"));

first.add(dishtypeText);

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 deleteGUIMenu()

{

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 menu");

while (rs.next())

{

if (rs.getString("dishid").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

dishidText.setText(rs.getString("dishid"));

dishnameText.setText(rs.getString("dishname"));

dishpriceText.setText(rs.getString("dishprice"));

dishtypeText.setText(rs.getString("dishtype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete menu Button

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM menu WHERE dishid = '" + menuList.getSelectedItem()+"'");

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

dishidText.setText(null);

dishnameText.setText(null);

dishpriceText.setText(null);

dishtypeText.setText(null);

menuList.removeAll();

loadMenu();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

dishidText = new TextField(15);

dishnameText = new TextField(15);

dishpriceText = new TextField(15);

dishtypeText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

dishidText.setEditable(false);

dishnameText.setEditable(false);

dishpriceText.setEditable(false);

dishtypeText.setEditable(false);

Panel first = new Panel();

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

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

first.add(dishidText);

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

first.add(dishnameText);

first.add(new Label("Dish Price:"));

first.add(dishpriceText);

first.add(new Label("Dish Type:"));

first.add(dishtypeText);

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 loadOrders()

{

try

{

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

while (rs.next())

{

ordersList.add(rs.getString("orderid"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateOrdersGUI()

{

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 orders");

while (rs.next())

{

if (rs.getString("orderid").equals(ordersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

orderidText.setText(rs.getString("orderid"));

ordernameText.setText(rs.getString("ordername"));

orderpriceText.setText(rs.getString("orderprice"));

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

}

}

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 orders "

+ "SET orderprice=" + orderpriceText.getText()

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

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

ordersList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

orderidText = new TextField(15);

orderidText.setEditable(false);

ordernameText = new TextField(15);

ordernameText.setEditable(false);

orderpriceText = new TextField(15);

ordertypeText = new TextField(15);

ordertypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(orderidText);

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

first.add(ordernameText);

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

first.add(orderpriceText);

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 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 orders");

while (rs.next())

{

if (rs.getString("orderid").equals(ordersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

orderidText.setText(rs.getString("orderid"));

ordernameText.setText(rs.getString("ordername"));

orderpriceText.setText(rs.getString("orderprice"));

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

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete orders Button

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM orders WHERE orderid = '" + ordersList.getSelectedItem()+"'");

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

orderidText.setText(null);

ordernameText.setText(null);

orderpriceText.setText(null);

ordertypeText.setText(null);

ordersList.removeAll();

loadOrders();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

orderidText = new TextField(15);

ordernameText = new TextField(15);

orderpriceText = new TextField(15);

ordertypeText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

orderidText.setEditable(false);

ordernameText.setEditable(false);

orderpriceText.setEditable(false);

ordertypeText.setEditable(false);

Panel first = new Panel();

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

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

first.add(orderidText);

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

first.add(ordernameText);

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

first.add(orderpriceText);

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 loadbmi()

{

try

{

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

while (rs.next())

{

bmiList.add(rs.getString("bmiid"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updatebmiGUI()

{

removeAll();

bmiList = new List(6);

loadbmi();

add(bmiList);

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

bmiList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("bmiid").equals(bmiList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

bmiidText.setText(rs.getString("bmiid"));

bmicategoryText.setText(rs.getString("bmicategory"));

bmicountText.setText(rs.getString("bmicount"));

bmidietText.setText(rs.getString("bmidiet"));

}

}

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 bmi "

+ "SET bmicount=" + bmicountText.getText()

+ " WHERE bmiid = '" + bmiList.getSelectedItem() + "'");

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

bmiList.removeAll();

loadbmi();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

bmiidText = new TextField(15);

bmiidText.setEditable(false);

bmicategoryText = new TextField(15);

bmicategoryText.setEditable(false);

bmicountText = new TextField(15);

bmidietText = new TextField(15);

bmidietText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(bmiidText);

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

first.add(bmicategoryText);

first.add(new Label("bmi Price:"));

first.add(bmicountText);

first.add(new Label("bmi Type:"));

first.add(bmidietText);

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 loadCustomerDetails()

{

try

{

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

while (rs.next())

{

customerList.add(rs.getString("cust\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void deleteGUIbmi()

{

removeAll();

bmiList = new List(10);

loadbmi();

add(bmiList);

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

bmiList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("bmiid").equals(bmiList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

bmiidText.setText(rs.getString("bmiid"));

bmicategoryText.setText(rs.getString("bmicategory"));

bmicountText.setText(rs.getString("bmicount"));

bmidietText.setText(rs.getString("bmidiet"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete bmi Button

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM bmi WHERE bmiid = '" + bmiList.getSelectedItem()+"'");

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

bmiidText.setText(null);

bmicategoryText.setText(null);

bmicountText.setText(null);

bmidietText.setText(null);

bmiList.removeAll();

loadbmi();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

bmiidText = new TextField(15);

bmicategoryText = new TextField(15);

bmicountText = new TextField(15);

bmidietText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

bmiidText.setEditable(false);

bmicategoryText.setEditable(false);

bmicountText.setEditable(false);

bmidietText.setEditable(false);

Panel first = new Panel();

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

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

first.add(bmiidText);

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

first.add(bmicategoryText);

first.add(new Label("bmi Price:"));

first.add(bmicountText);

first.add(new Label("bmi Type:"));

first.add(bmidietText);

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 updateCustomerAccount()

{

removeAll();

customerList = new List(10);

loadCustomerDetails();

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 cust\_account");

while (rs.next())

{

if (rs.getString("cust\_id").equals(customerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customeridText.setText(rs.getString("cust\_id"));

customerNameText.setText(rs.getString("cust\_name"));

customerPhoneNumberText.setText(rs.getString("ph\_no"));

customerMailText.setText(rs.getString("email"));

customerAddressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

modify = new Button("Modify");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE cust\_account SET cust\_name='" + customerNameText.getText() + "',ph\_no='" + customerPhoneNumberText.getText() + "',email='" + customerMailText.getText() + "',address='" + customerAddressText.getText()+ "' WHERE cust\_id = '" + customerList.getSelectedItem() + "' ");

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

customerList.removeAll();

loadCustomerDetails();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customeridText = new TextField(20);

customeridText.setEditable(false);

customerNameText = new TextField(20);

customerPhoneNumberText = new TextField(20);

customerMailText = new TextField(20);

customerAddressText=new TextField(20);

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(customeridText);

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

first.add(customerNameText);

first.add(new Label("Phone Number :"));

first.add(customerPhoneNumberText);

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

first.add(customerMailText);

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

first.add(customerAddressText);

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

second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteCustomerDetails()

{

removeAll();

customerList = new List(10);

loadCustomerDetails();

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 cust\_account");

while (rs.next())

{

if (rs.getString("cust\_id").equals(customerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customeridText.setText(rs.getString("cust\_id"));

customerNameText.setText(rs.getString("cust\_name"));

customerPhoneNumberText.setText(rs.getString("ph\_no"));

customerMailText.setText(rs.getString("email"));

customerAddressText.setText(rs.getString("address"));

}

}

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 cust\_account WHERE cust\_id = '" + customerList.getSelectedItem()+"'");

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

customeridText.setText(null);

customerNameText.setText(null);

customerPhoneNumberText.setText(null);

customerMailText.setText(null);

customerAddressText.setText(null);

customerList.removeAll();

loadCustomerDetails();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

customeridText = new TextField(20);

customeridText.setEditable(false);

customerNameText = new TextField(20);

customerPhoneNumberText = new TextField(20);

customerMailText = new TextField(20);

customerAddressText=new TextField(20);

customeridText.setEditable(false);

customerNameText.setEditable(false);

customerPhoneNumberText.setEditable(false);

customerMailText.setEditable(false);

customerAddressText.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(customeridText);

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

first.add(customerNameText);

first.add(new Label("Phone Number :"));

first.add(customerPhoneNumberText);

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

first.add(customerMailText);

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

first.add(customerAddressText);

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(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 menu");

while (rs.next())

{

if (rs.getString("dishid").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

dishidText.setText(rs.getString("dishid"));

dishnameText.setText(rs.getString("dishname"));

dishpriceText.setText(rs.getString("dishprice"));

dishtypeText.setText(rs.getString("dishtype"));

}

}

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 menu "

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

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

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

menuList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

dishidText = new TextField(15);

dishidText.setEditable(false);

dishnameText = new TextField(15);

dishnameText.setEditable(false);

dishpriceText = new TextField(15);

dishpriceText.setEditable(false);

dishtypeText = new TextField(15);

dishtypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(dishidText);

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

first.add(dishnameText);

first.add(new Label("Dish Price:"));

first.add(dishpriceText);

first.add(new Label("Dish Type:"));

first.add(dishtypeText);

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 viewbmiGUI()

{

removeAll();

bmiList = new List(6);

loadbmi();

add(bmiList);

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

bmiList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("bmiid").equals(bmiList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

bmiidText.setText(rs.getString("bmiid"));

bmicategoryText.setText(rs.getString("bmicategory"));

bmicountText.setText(rs.getString("bmicount"));

bmidietText.setText(rs.getString("bmidiet"));

}

}

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 bmi "

+ "SET bmicount=" + bmicountText.getText()

+ " WHERE bmiid = '" + bmiList.getSelectedItem() + "'");

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

bmiList.removeAll();

loadbmi();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

bmiidText = new TextField(15);

bmiidText.setEditable(false);

bmicategoryText = new TextField(15);

bmicategoryText.setEditable(false);

bmicountText = new TextField(15);

bmicountText.setEditable(false);

bmidietText = new TextField(15);

bmidietText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(bmiidText);

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

first.add(bmicategoryText);

first.add(new Label("bmi Price:"));

first.add(bmicountText);

first.add(new Label("bmi Type:"));

first.add(bmidietText);

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 orders");

while (rs.next())

{

if (rs.getString("orderid").equals(ordersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

orderidText.setText(rs.getString("orderid"));

ordernameText.setText(rs.getString("ordername"));

orderpriceText.setText(rs.getString("orderprice"));

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

}

}

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 menu "

+ "SET orderprice=" + orderpriceText.getText()

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

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

ordersList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

orderidText = new TextField(15);

orderidText.setEditable(false);

ordernameText = new TextField(15);

ordernameText.setEditable(false);

orderpriceText = new TextField(15);

orderpriceText.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(4, 2));

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

first.add(orderidText);

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

first.add(ordernameText);

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

first.add(orderpriceText);

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 buildGUICustomerDetails()

{

removeAll();

submit = new Button("Submit");

submit.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO cust\_account VALUES('" + customeridText.getText() + "','" + customerNameText.getText() + "'," + customerPhoneNumberText.getText() + ",'" + customerMailText.getText() + "','" + customerAddressText.getText() + "')";

int i = statement.executeUpdate(query);

errorText.append("\nCreated " + i + " Account successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customeridText = new TextField(20);

customerNameText = new TextField(20);

customerPhoneNumberText = new TextField(20);

customerMailText = new TextField(20);

customerAddressText=new TextField(20);

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(customeridText);

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

first.add(customerNameText);

first.add(new Label("Phone Number :"));

first.add(customerPhoneNumberText);

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

first.add(customerMailText);

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

first.add(customerAddressText);

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

second.add(submit);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public static void main(String[] args)

{

Insert it = new Insert();

it.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

it.buildFrame();

}

}

**Github links and folder structure**:

<https://github.com/Kani111/dbms-project>

FOLDER STRUCTURE:

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

TESTING:

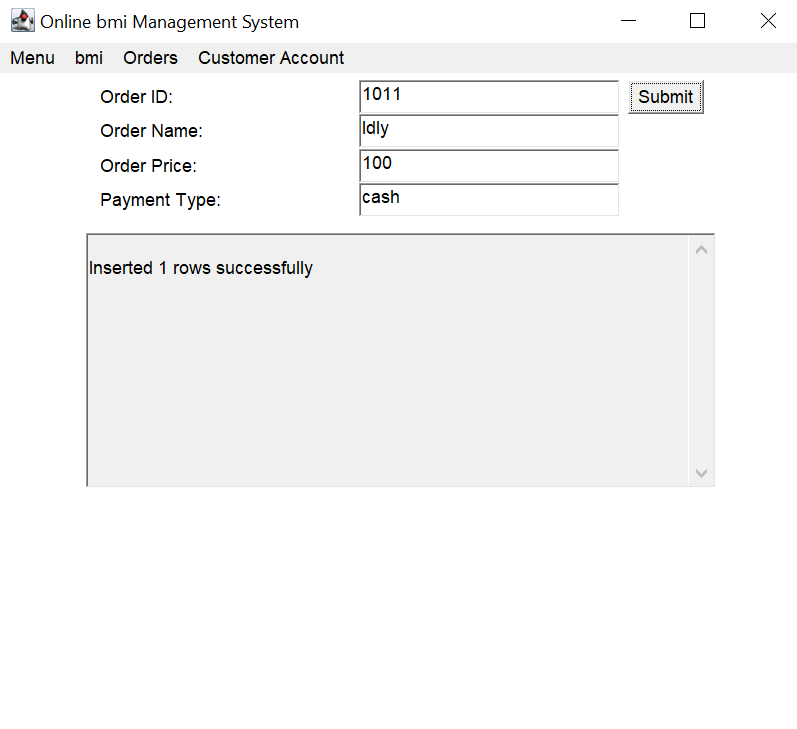
***INSERT(SUBMIT):-***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface, application

Description automatically generated***

******

***Graphical user interface, application

Description automatically generated***

***Graphical user interface, application

Description automatically generated***

***MODIFY:-***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface

Description automatically generated***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface

Description automatically generated with low confidence***

***DELETE:-***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface, application, Word

Description automatically generated***

***Graphical user interface

Description automatically generated with medium confidence***

***VIEW:-***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface, application

Description automatically generated***

***Graphical user interface

Description automatically generated***

***Graphical user interface, application

Description automatically generated***

***DATABASE OUTPUTS:-***

***BEFORE:-***

***Graphical user interface, text

Description automatically generated***

***AFTER:-***

***Graphical user interface, text

Description automatically generated with medium confidence***

## DISCUSSION AND FUTURE WORK

*so far this project has been useful in recording a person weight accordingly and put him into one category among the four categories classified and helped them in suggesting the diet which is healthy for them and making them to follow these regular diets accordingly and a person can directly get the food that is being suggested to him ot her at a reasonable price*

*In future there can be some more changes that are going to be added like displaying their body mass index with the help of their weight and height and suggesting the diet asusual but by adding the new features that gives complete description about the food that they are consuming like the amount of “Nutrients,vitamins and minerals etc……”.*

***REFERENCES:-***

* *Abraham Silberschatz, Henry F Korth, S. Sudarshan, Database System Concepts, 6th Edition, McGraw-Hill International Edition, 2010.*

* [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).

-------------------------------*THANK YOU-------------------------------*