**ONLINE RESTAURNT**

**MANAGEMENT SYSTEM**

**Sai Arvind**

**1602-19-737-092**

**IT-B**

JAVA AWT BASED- ONLINE RESTAURANT 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

Sai Arvind<1602-19-737-092>

Diagram

Description automatically generated with medium confidenceUnder the guidance of Ms B. Leelavathy

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 Restaurant Management System**’ is a project work of Sai Arvind bearing roll no. 1602-19-737-092 who carried out the project under my supervision in the IV semester for the academic year 2020- 2021.

**ABSTRACT**

Restaurant Management system, eating out it is something that we all love to do, undoubtedly we all love going out but due to covid it is not safe to go out to restaurant so we have a come up with a unique solution .We provide facilities like dine in pick up and reserve table, in reserve table you can choose the option to sanitize a table of your choice it would be sanitized and locked and u can check that.

In pick up you just need to enter your mobile number and your order would be delivered to your vehicle and u just need to tell the otp that’s all. administrator can enter name, password and create an account.

User can then login and select the restaurants and dishes that they need. The customer can order whatever they need get them easily. They can also check their order status. The application allows the user to login. The customers can maintain their account. Once the customer decides on purchasing they get a purchase order. He can also print the bill inorder to have a hard copy of the transaction.

**REQUIREMENT ANALYSIS:**

**List of Tables:**

* Login
* customer\_login
* Customer\_account
* Orders\_From
* Restraunt name
* 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)
* Rid: number(5)
* Date: DATE

**Restraunt Name:**

* Rid: number(5)
* Rname: varchar2(20)
* Menu\_list: varchar2(20)

**Selects:**

* Rid: 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 generatedMAPPING 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 restraunts can be searched by any number of users. So, one to many mapping cardinality between customer account and dishes.

A single restraunt can contain any number of cuisines. Therefore, one to many mapping cardinality between restraunt 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 **Restraunt**(

Rid number(5),

Rname varchar2(20),

products\_list varchar2(50),

primary key(Rid));

create table **orders\_from**(

cust\_id number(5),

Rid number(5),

day date,

primary key(cust\_id,Rid),

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

foreign key(Rid) references Restraunt(Rid));

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),

Rid number(5),

primary key(Rid,dish\_id),

foreign key(Rid) references Restraunt(Rid),

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

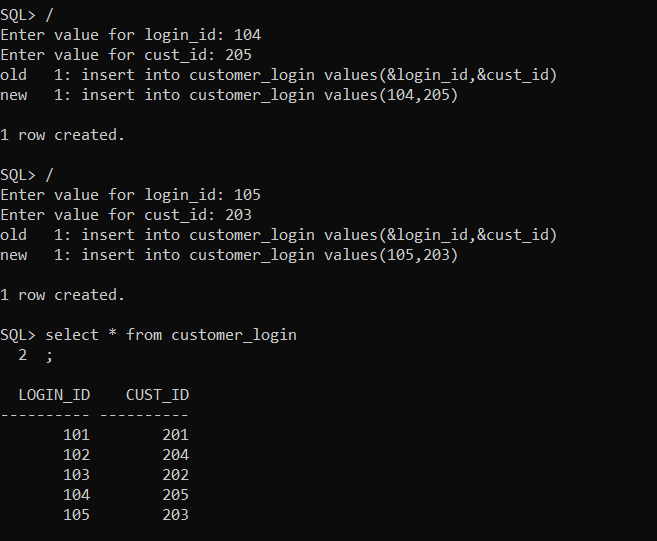
Description automatically generated**

**Text

Description automatically generated**

Text

Description automatically generated



**Text

Description automatically generated**

**Text

Description automatically generated**

**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

**CODE:-**

package DBMS;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

//1602-19-737-092 Sai Arvind

@SuppressWarnings("serial")

public class InsertTables 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,restaurant,orders,customerAccount;

Button insertButton,submit;

TextField dishidText, dishnameText, dishpriceText, dishtypeText;

TextField restaurantidText, restaurantnameText, restaurantpriceText, restauranttypeText;

TextField orderidText, ordernameText, orderpriceText, ordertypeText;

TextField customeridText,customerNameText,customerPhoneNumberText,customerMailText,customerAddressText;

TextArea errorText;

Connection connection;

Statement statement;

//For updates

Button modify;

Choice menuList,ordersList,restaurantList,customerList;

ResultSet rs;

//TextField dishidText, dishnameText, dishpriceText, dishtypeText;

//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","system","arvind");

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 restaurant 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);

//restaurant

restaurant=new Menu("Restaurant");

m5=new MenuItem("Insert Restaurant");

m6=new MenuItem("Update Restaurant");

m7=new MenuItem("Delete Restaurant");

m8=new MenuItem("View Restaurant");

restaurant.add(m5);

restaurant.add(m6);

restaurant.add(m7);

restaurant.add(m8);

mb.add(restaurant);

//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);

//Customer Details

customerAccount=new Menu("Customer Account");

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

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

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

m16=new MenuItem("View Customer Details");

customerAccount.add(m13);

customerAccount.add(m14);

customerAccount.add(m15);

customerAccount.add(m16);

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);

m16.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 Restaurant"))

this.buildGUIRestaurant();

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

this.updateRestaurantGUI();

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

this.deleteGUIrestaurant();

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

this.viewRestaurantGUI();

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();

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

this.viewCustomerDetails();

}

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 buildGUIRestaurant()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO restaurant VALUES('" + restaurantidText.getText() + "', " + "'" + restaurantnameText.getText() + "'," + restaurantpriceText.getText() + ",'" + restauranttypeText.getText() + "')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

restaurantidText = new TextField(15);

restaurantnameText = new TextField(15);

restaurantpriceText = new TextField(15);

restauranttypeText= 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("Restaurant ID:"));

first.add(restaurantidText);

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

first.add(restaurantnameText);

first.add(new Label("Avg price for 2:"));

first.add(restaurantpriceText);

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

first.add(restauranttypeText);

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();

restaurantList = new Choice();

loadRestaurant();

add(restaurantList);

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

restaurantList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("restaurantid").equals(restaurantList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

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

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

menuList = new Choice();

loadMenu1();

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("dishname").equals(menuList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

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

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

}

}

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 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);

orderidText.setEditable(false);

ordernameText = new TextField(15);

ordernameText.setEditable(false);

orderpriceText = new TextField(15);

orderpriceText.setEditable(false);

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("Restraunt ID:"));

first.add(orderidText);

first.add(new Label("Dish 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);

}

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);

}

private void loadMenu1()

{

try

{

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

while (rs.next())

{

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

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

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 Choice();

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

modify = new Button("Modify");

modify.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(modify);

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 Choice();

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");

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 Choice();

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

modify = new Button("Modify");

modify.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("Restaurant 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(modify);

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 Choice();

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");

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);

}

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 updateCustomerAccount()

{

removeAll();

customerList = new Choice();

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 Choice();

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);

}

private void loadRestaurant()

{

try

{

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

while (rs.next())

{

restaurantList.add(rs.getString("restaurantid"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateRestaurantGUI()

{

removeAll();

restaurantList = new Choice();

loadRestaurant();

add(restaurantList);

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

restaurantList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("restaurantid").equals(restaurantList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

restaurantidText.setText(rs.getString("restaurantid"));

restaurantnameText.setText(rs.getString("restaurantname"));

restaurantpriceText.setText(rs.getString("restaurantprice"));

restauranttypeText.setText(rs.getString("restauranttype"));

}

}

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 restaurant "

+ "SET restaurantprice=" + restaurantpriceText.getText()

+ " WHERE restaurantid = '" + restaurantList.getSelectedItem() + "'");

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

restaurantList.removeAll();

loadRestaurant();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

restaurantidText = new TextField(15);

restaurantidText.setEditable(false);

restaurantnameText = new TextField(15);

restaurantnameText.setEditable(false);

restaurantpriceText = new TextField(15);

restauranttypeText = new TextField(15);

restauranttypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(restaurantidText);

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

first.add(restaurantnameText);

first.add(new Label("Avg Price for 2:"));

first.add(restaurantpriceText);

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

first.add(restauranttypeText);

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

second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIrestaurant()

{

removeAll();

restaurantList = new Choice();

loadRestaurant();

add(restaurantList);

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

restaurantList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("restaurantid").equals(restaurantList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

restaurantidText.setText(rs.getString("restaurantid"));

restaurantnameText.setText(rs.getString("restaurantname"));

restaurantpriceText.setText(rs.getString("restaurantprice"));

restauranttypeText.setText(rs.getString("restauranttype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete restaurant 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 restaurant WHERE restaurantid = '" + restaurantList.getSelectedItem()+"'");

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

restaurantidText.setText(null);

restaurantnameText.setText(null);

restaurantpriceText.setText(null);

restauranttypeText.setText(null);

restaurantList.removeAll();

loadRestaurant();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

restaurantidText = new TextField(15);

restaurantnameText = new TextField(15);

restaurantpriceText = new TextField(15);

restauranttypeText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

restaurantidText.setEditable(false);

restaurantnameText.setEditable(false);

restaurantpriceText.setEditable(false);

restauranttypeText.setEditable(false);

Panel first = new Panel();

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

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

first.add(restaurantidText);

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

first.add(restaurantnameText);

first.add(new Label("Avg Price for 2:"));

first.add(restaurantpriceText);

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

first.add(restauranttypeText);

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 Choice();

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

modify = new Button("Update Menu");

modify.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(modify);

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 viewRestaurantGUI()

{

removeAll();

restaurantList = new Choice();

loadRestaurant();

add(restaurantList);

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

restaurantList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("restaurantid").equals(restaurantList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

restaurantidText.setText(rs.getString("restaurantid"));

restaurantnameText.setText(rs.getString("restaurantname"));

restaurantpriceText.setText(rs.getString("restaurantprice"));

restauranttypeText.setText(rs.getString("restauranttype"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

modify = new Button("Update Restaurant");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE restaurant "

+ "SET restaurantprice=" + restaurantpriceText.getText()

+ " WHERE restaurantid = '" + restaurantList.getSelectedItem() + "'");

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

restaurantList.removeAll();

loadRestaurant();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

restaurantidText = new TextField(15);

restaurantidText.setEditable(false);

restaurantnameText = new TextField(15);

restaurantnameText.setEditable(false);

restaurantpriceText = new TextField(15);

restaurantpriceText.setEditable(false);

restauranttypeText = new TextField(15);

restauranttypeText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(restaurantidText);

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

first.add(restaurantnameText);

first.add(new Label("Avg Price for 2:"));

first.add(restaurantpriceText);

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

first.add(restauranttypeText);

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

//second.add(modify);

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 Choice();

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

modify = new Button("Modify");

modify.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(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewCustomerDetails()

{

removeAll();

customerList = new Choice();

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);

}

}

});

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 displaySQLErrors(SQLException e)

{

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

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

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

}

public static void main(String[] args)

{

InsertTables it = new InsertTables();

it.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

it.buildFrame();

}

}

**CODE OUTPUT:-**

Graphical user interface

Description automatically generated

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

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

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

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

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Github:-

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

LINK:-

<https://github.com/Arvind004/Restaurant-Management-System-JAVA-SQL>