

# RESTAURANT MANAGEMENT SYSTEM



- IDBMS PROJECT BY:

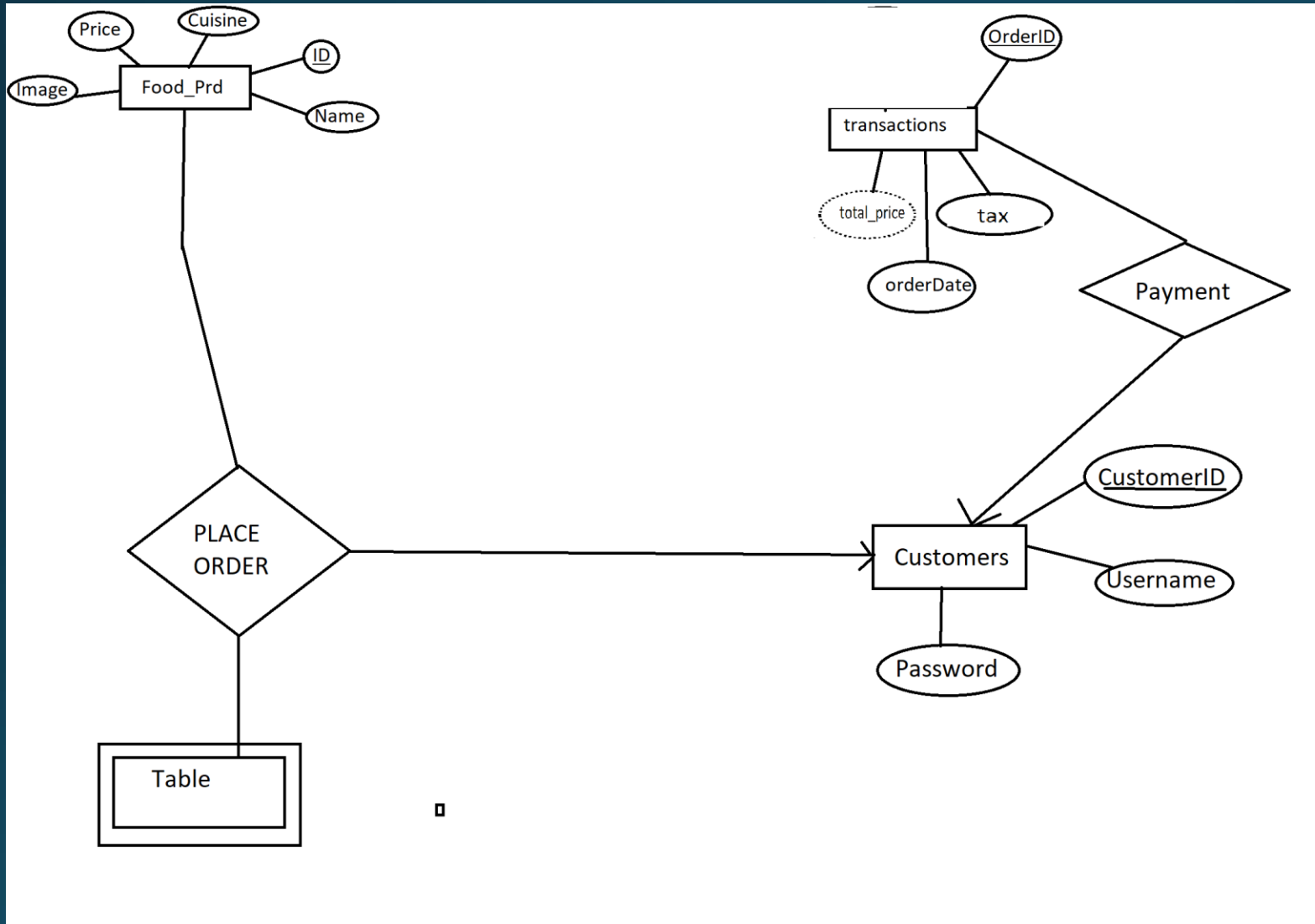
ANIMESH KULKSHRESHTHA (20UCS021)

ISHAN SHARMA (20UCS086)

HARSHAL JAIN (20UCS077)

AARYA GARG (20UCS002)

# RESTAURANT MANAGEMENT SYSTEM



ER DIAGRAM

# Overview



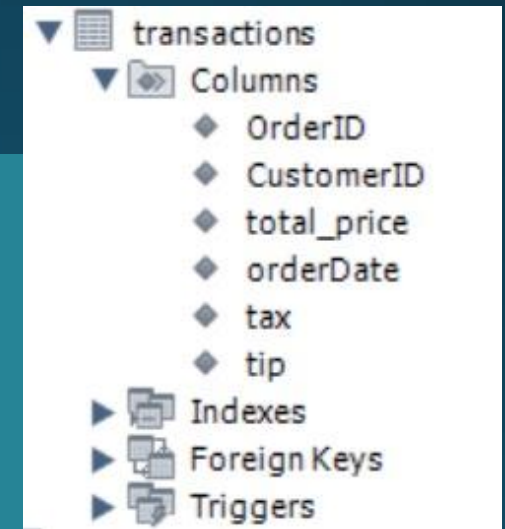
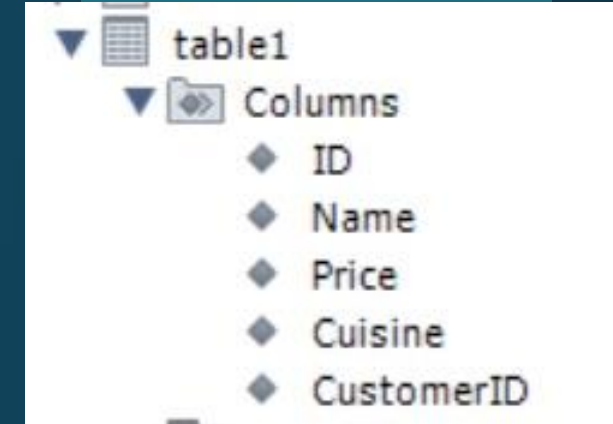
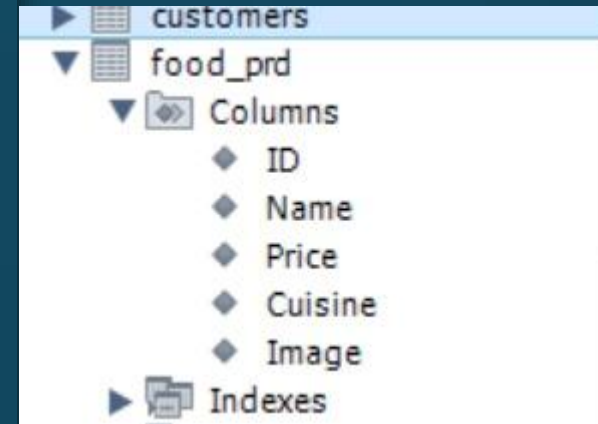
- This project is meant to help Restaurant Businesses in easing their ordering process and management..
- We have tried to include most of the basic functionalities that are provided in a general restaurant management system.
- We have used JAVA for front end and MySQL for the backend development.

# Database



# CREATING DATABASE AND TABLES

- **CREATE DATABASE FastFoodCenter;**
- **USE FastFoodCenter;**
- **CREATE TABLE Food\_Prđ (ID INT NOT NULL PRIMARY KEY,Name VARCHAR(30),Price FLOAT NOT NULL,Cuisine VARCHAR(30),Image LONGBLOB);**
- **CREATE TABLE transactions(OrderID INT NOT NULL PRIMARY KEY,CustomerID INT ,total\_price FLOAT,tip FLOAT,orderDate DATETIME,tax FLOAT,FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID));**
- **CREATE TABLE Customers(CustomerID INT NOT NULL PRIMARY KEY,Username VARCHAR(50),Password VARCHAR(50));**

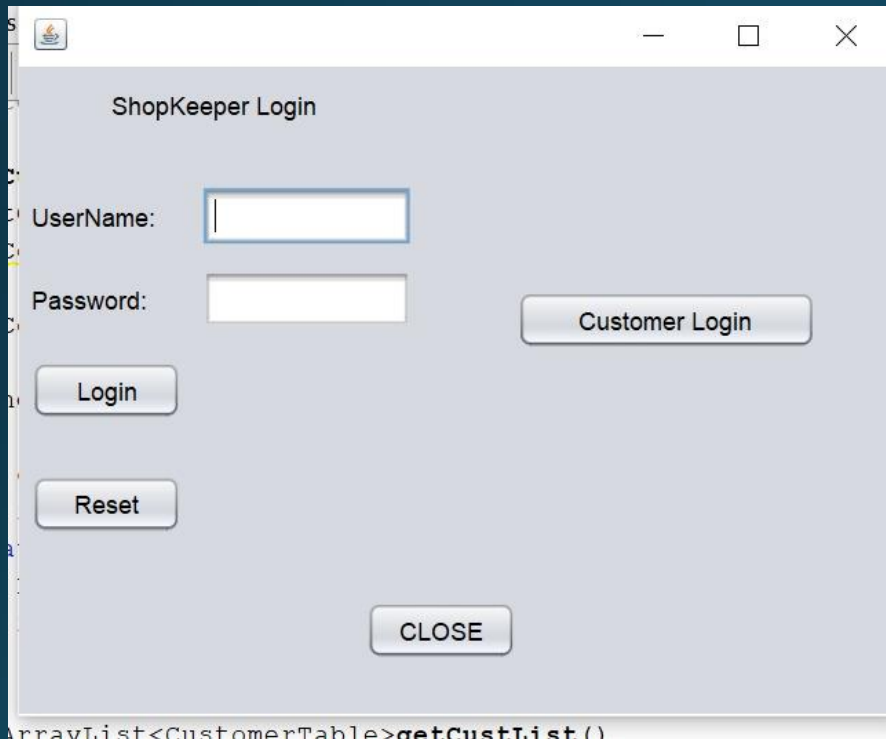


# CONNECTING THE JAVA APPLICATION WITH SQL



```
public Connection getConnection() {  
    Connection con = null;  
    try {  
        con=DriverManager.getConnection("jdbc:mysql://local  
host:3306/FastFoodCenter","FastFoodDatabase","");  
        return con;  
    }  
    catch (SQLException ex) {  
        Logger.getLogger(Products.class.getName()).log(Level  
        .SEVERE, null, ex);  
        return null;  
    }  
}
```

Shopkeeper can login with credentials to view/update database and transaction history.



The image shows a Java Swing window titled "ShopKeeper Login". It contains two text input fields: "UserName:" and "Password:". Below the "Password:" field is a "Customer Login" button. To the left of the "Password:" field are two buttons: "Login" and "Reset". At the bottom center of the window is a "CLOSE" button. The window has a standard title bar with minimize, maximize, and close buttons.

```
ArrayList<CustomerTable>getCustList()
```



If password is incorrect then it shows a dialogue box.



ID: 1202

Name: Matar Pulao

Price: 60.0

Cuisine: Gravy

Image:



ID	Name	Cuisine	Price
901	French Fires	Appetizer	60.0
902	Masala Papad	Appetizer	75.0
903	Paneer Pakoda	Appetizer	120.0
401	Chai	Beverages	20.0
402	Hot Coffee	Beverages	30.0
403	Cold Coffee	Beverages	69.0
404	Cappuccino	Beverages	49.0
405	Lemon Ice Tea	Beverages	65.0
406	Oreo Shake	Beverages	55.0
407	Smoothy	Beverages	55.0
1201	Jeera Rice	Biryani Ka Dum	40.0
1202	Matar Pulao	Biryani Ka Dum	60.0
1203	Veg Biryani	Biryani Ka Dum	115.0
101	Aloo Paratha	Breads and Parantha	35.0
102	Laccha Parantha	Breads and Parantha	30.0



Kulshrestha's

Once the shopkeeper logs in, the home page for providing accesses like:

- Insert new Menu Item
- Update details of previous Menu Item
- Delete Menu Item
- Show previous transactions



# New menu item added using insert

- INSERT INTO Food\_Prds(ID,Name,Price,Cuisine,Image) values(?,?,?,?,,?)
- Where ?,?,?,?,? are data values of the respective columns.



# Updating menu item

UpdateQuery = "UPDATE Food\_Prds SET  
Name = ?, Price = ?, Cuisine = ?, Image = ?  
WHERE ID = ?";

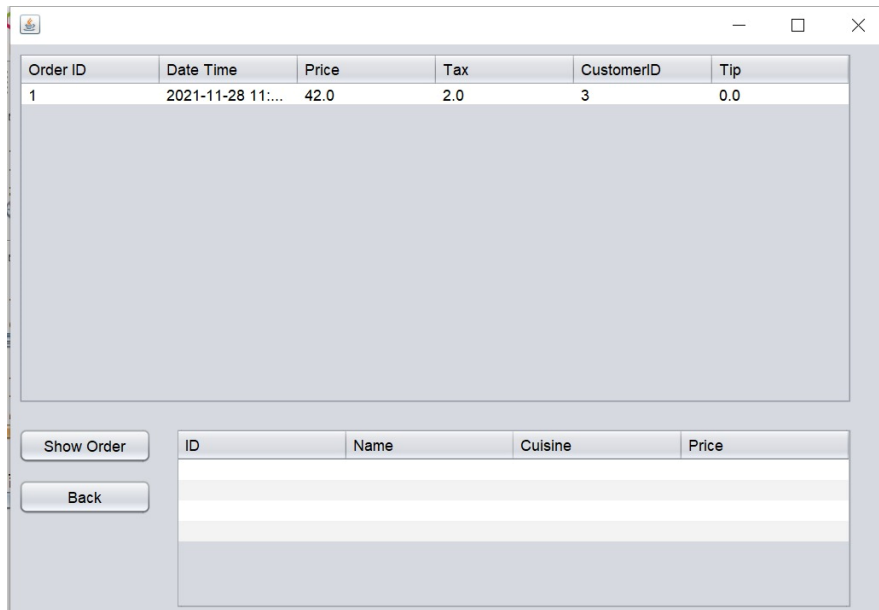
The screenshot displays a web application interface for managing a fast-food menu. On the left, a form for updating a menu item is visible, with fields for ID (308), Name (Roshogulla), Price (25.0), and Cuisine (Dessert). Below these fields is an 'Image' section with a 'Choose Image' button and a preview of a bowl of Roshogulla. In the center, a table lists the current menu items, with the item having ID 308 highlighted. At the bottom, there are navigation buttons (FIRST, PREVIOUS, NEXT, LAST), action buttons (INSERT, UPDATE, DELETE), and a 'Show Transaction' button. The right side of the interface features a decorative banner with a brick background, a white fork icon on an orange square, and the text 'Kulshrestha's'.

ID	Name	Cuisine	Price
202	Burger	Continental	45.0
203	Continental Salad	Continental	40.0
204	Crispy Corn	Continental	49.0
205	Taco	Continental	119.0
301	Brownie	Dessert	69.0
302	Cheesecake	Dessert	99.0
303	Cherry Pie	Dessert	119.0
304	Gulab Jamun	Dessert	20.0
305	Pancakes	Dessert	120.0
306	Rasmalai	Dessert	25.0
307	Waffle	Dessert	70.0
308	Roshogulla	Dessert	25.0
801	Daal Makahni	Gravy	120.0
802	Daal Fry	Gravy	90.0
803	Malai Kofta	Gravy	100.0

**Kulshrestha Fast Food Center**

**Kulshrestha's**

# Show transactions



Order ID	Date Time	Price	Tax	CustomerID	Tip
1	2021-11-28 11:...	42.0	2.0	3	0.0

Show Order

ID	Name	Cuisine	Price

Back

```
ArrayList<Transactions_Table> orderList = new ArrayList<Transactions_Table>();
Connection con = getConnection();
String Query = "SELECT * FROM Transactions ORDER BY orderDate";
Statement st;
ResultSet rs;
try {
    st = con.createStatement();
    rs = st.executeQuery(Query);
    Transactions_Table transaction;
    while(rs.next()) {
        transaction = new
        Transactions_Table(rs.getInt("OrderID"),rs.getString("orderDate"),Float.parseFl
        oat(rs.getString("total_price")),Float.parseFloat(rs.getString("tax")),rs.getInt("C
        ustomerID"),Float.parseFloat(rs.getString("Tip")));
        orderList.add(transaction);
    }
}
catch (SQLException ex) {
    Logger.getLogger(Products.class.getName()).log(Level.SEVERE, null, ex);
}
return orderList;
```

# Show orders

The screenshot shows a web application interface. At the top, there is a table with 6 columns: Order ID, Date Time, Price, Tax, CustomerID, and Tip. The first row of data shows Order ID 1, Date Time 2021-11-28 11:..., Price 42.0, Tax 2.0, CustomerID 3, and Tip 0.0. Below this table is a large empty rectangular area. At the bottom left, there are two buttons: 'Show Order' and 'Back'. To the right of these buttons is another table with 4 columns: ID, Name, Cuisine, and Price. The first row of data shows ID 203, Name Continental Salad, Cuisine Continental, and Price 40.0. Below this table is another large empty rectangular area.

Order ID	Date Time	Price	Tax	CustomerID	Tip
1	2021-11-28 11:...	42.0	2.0	3	0.0

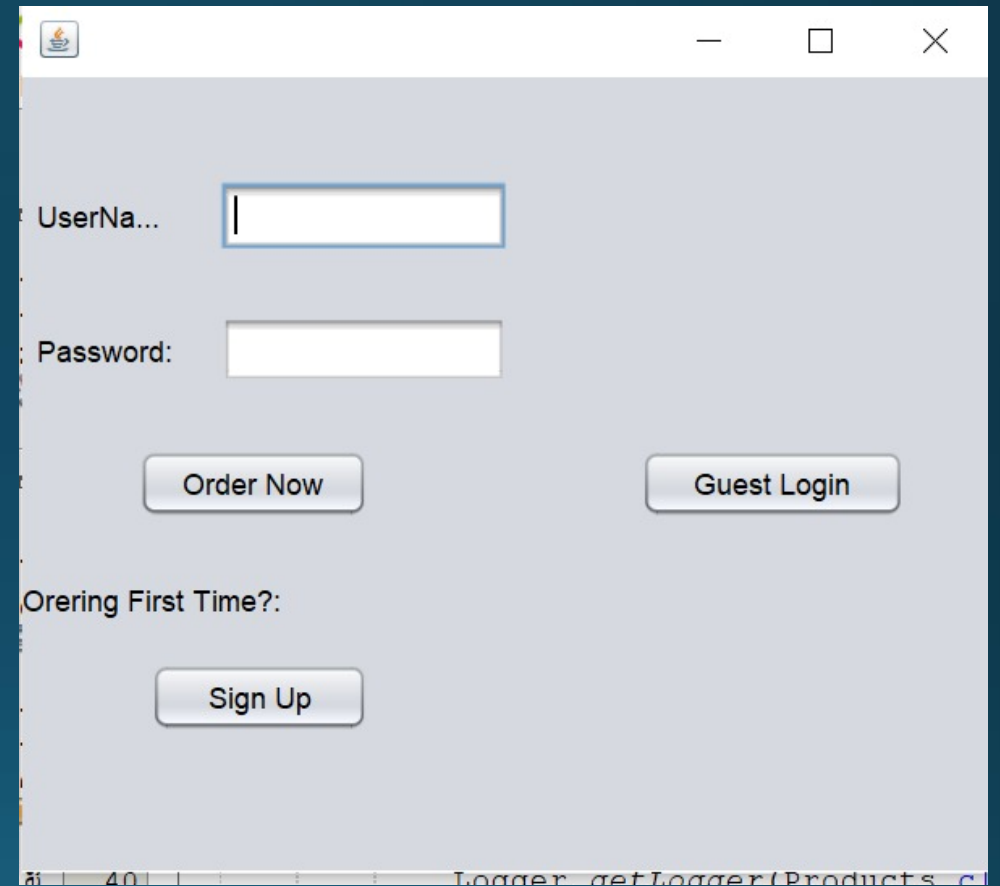
  

ID	Name	Cuisine	Price
203	Continental Salad	Continental	40.0

# QUERY

```
ArrayList<Products_Management> orderList = new ArrayList<Products_Management>();
int row = Transactions.getSelectedRow();
System.out.print("ROW:"+row);
String OrderID = Transactions.getModel().getValueAt(row,o).toString();
Connection con = getConnection();
String Query = "SELECT * FROM Table"+OrderID;
Statement st;
ResultSet rs;
try {      st = con.createStatement();
rs = st.executeQuery(Query);
Products_Management product;
while(rs.next())      {
product = new
Products_Management(rs.getInt("ID"),rs.getString("Name"),Float.parseFloat(rs.getString("Price")),rs.getString("Cuisine"),rs.getBytes("Image"),rs.getString("ImagePath"));
orderList.add(product);
}
}
catch (SQLException ex) {
Logger.getLogger(Products.class.getName()).log(Level.SEVERE, null, ex);
}
return orderList;
}
```

# Customer Signup/login



A screenshot of a web application window with a light gray background. The window has a standard title bar with a minimize icon, a maximize icon, and a close icon. The form contains the following elements:

- A label "UserNa..." followed by a text input field.
- A label "Password:" followed by a text input field.
- Two buttons: "Order Now" and "Guest Login".
- A label "Orering First Time?:" followed by a "Sign Up" button.


At the bottom of the window, there is a status bar with the text "40" and "Logger: getLogger(Product s.c)".



# Customer sign up

```
Connection con = getConnection();
if(checkInputs())
{
    try {
        String Query = "SELECT COUNT(*) FROM Customers";
        Statement st;
        ResultSet rs;
        st = con.createStatement();
        rs = st.executeQuery(Query);
        if(rs.next())
        {
            CustID = rs.getInt("COUNT(*)");
            CustID++;
        }
        else
            System.out.println("Error");
    }
    catch (SQLException ex) {
        Logger.getLogger(Products.class.getName()).log(Level.SEVERE,
            null, ex);
    }
    cis.setText(""+CustID+"");
    PreparedStatement ps;
    try {
        ps = con.prepareStatement("INSERT INTO Customers
            (CustomerID,Username>Password) values(?,?,?) ");
        ps.setString(1,""+CustID+"");
        ps.setString(2,usernamefld.getText());
        ps.setString(3,passfld.getText());
        ps.executeUpdate();
        JOptionPane.showMessageDialog(null,"Welcome
            "+usernamefld.getText()+"\n Congratulation!");
    } catch
        (SQLException ex) {
        Logger.getLogger(SignInForm.class.getName()).log(Level.SEVERE
            , null, ex);
    }
}
```

The screenshot shows a Java Swing window titled "Customer sign up". It contains three text input fields for "username:", "password:", and "confirm password:". To the right of the "username:" field is a red error message: "\*Username should contain atleast 1 number". To the right of the "password:" field is a red error message: "\*Password must be atleast 8 characters long". Below the input fields are three buttons: "Sign Up", "Continue", and "Back". At the bottom left, there is a label "Unique customer ID" and an empty text input field. The window has a standard Mac OS X title bar with a red close button, a yellow maximize button, and a green window control button. The background of the window is a light gray.



—

□

×

username:

password:

confirm password:


Sign IN

Continue

Unique customer ID

Message

×



Welcome Animesh

Congratulation!

OK

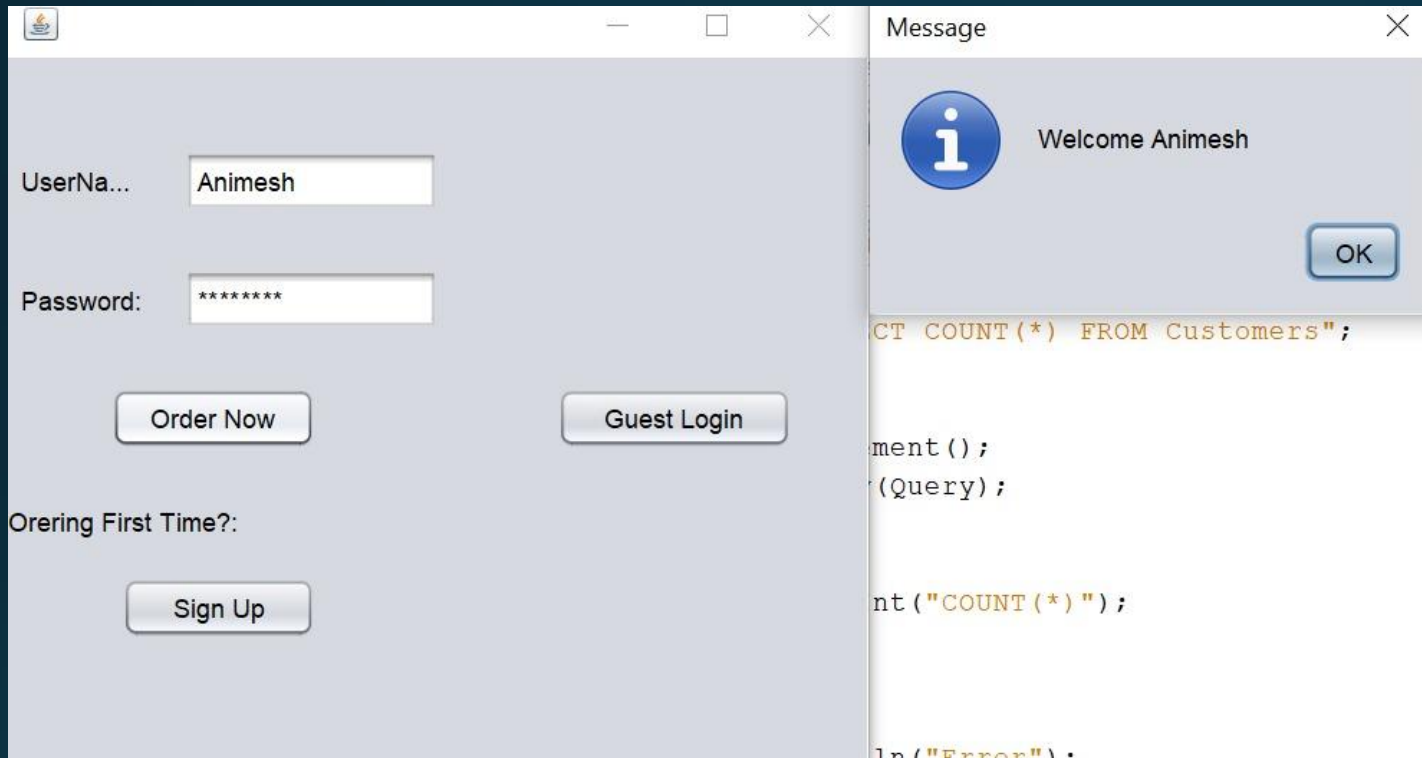
```
T COUNT(*) FROM Customers";

ent();
Query);

t("COUNT(*)");


n("Error");
,
```

# Customer Login



```
ArrayList<CustomerTable> List = getCustList();    String
usnm="",passw;
int CustID=0;
int index=0,check=0;
for(int i = 0;i<List.size();i++)    {
    usnm=List.get(i).getUsername();
    if(usnm.equals(username.getText()))
    { check = 1;
    index = i;    }    }
if(check==0)    {
    JOptionPane.showMessageDialog(null,"Username not found");
}
passw = List.get(index).getPassword();
CustID = List.get(index).getCustomerID();
if(usnm.equals(username.getText())&&passw.equals(passw.getText()))
{
    JOptionPane.showMessageDialog(null,"Welcome
    "+usnm);    }
else    {
    JOptionPane.showMessageDialog(null,"Incorrect Password or
    UserID");    username.setText(null);
    pass.setText(null);
    return;    }
close();
Customer c= new Customer(CustID);
c.setVisible(true); }
```

# Ordering



ID:

Name:

Price:

Cuisine:

ID	Name	Cuisine	Price
101	Aloo Paratha	Breads and Parantha	35.0
102	Laccha Parantha	Breads and Parantha	30.0
103	Tawa Roti	Breads and Parantha	25.0
104	Tandoori Roti	Breads and Parantha	30.0
201	Baby Corn	Continental	79.0
202	Burger	Continental	45.0
203	Continental Salad	Continental	40.0
204	Crispy Corn	Continental	49.0
205	Taco	Continental	119.0
301	Brownie	Dessert	69.0
302	Cheesecake	Dessert	99.0
303	Cherry Pie	Dessert	119.0

Customer ID:  Cuisine:

Name:


ID	Name	Cuisine	Price
203	Continental Salad	Continental	40.0

Adding to Cart

# QUERY

```
Connection con = getConnection();    if(checkInputs())    {
    try {        if(!tableexist(con))        {
        System.out.println("Creating new table");
        PreparedStatement ps1 = con.prepareStatement("CREATE TABLE Table"+OrderID+" (ID INT NOT
        NULL,Name VARCHAR(30),Price FLOAT,Cuisine VARCHAR(30),CustomerID INT,FOREIGN KEY (ID)
        REFERENCES Food_Prđ(ID),FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)) ");
        ps1.executeUpdate();        }
        PreparedStatement ps = con.prepareStatement("INSERT INTO Table"+OrderID+"
        (ID,Name,Price,Cuisine,CustomerID) values(?,?,?,?,?) ");
        int row = Cart.getSelectedRow();
        ps.setString(1,ID_input.getText());
        ps.setString(2,Name_input.getText());
        ps.setString(3,Price_input.getText());
        ps.setString(4,Cuisine_input.getText());
        ps.setInt(5,CustID);
        ps.executeUpdate();
        Show_Products_In_Cart();
        SearchOn=0;
    }
    catch (SQLException ex) {
        JOptionPane.showMessageDialog(null,ex.getMessage());
    }
}
```

Adding to Cart



ID: 702

Name: Dahi Papdi

Price: 30.0


Cuisine: Chaat Bazaar

ADD TO CART

DELETE FROM CART

ID	Name	Cuisine	Price
603	Fried Rice	Chinese	69.0
604	Honey Chilli Potato	Chinese	89.0
605	Munchurian	Chinese	80.0
701	Chhole Tikki	Chaat Bazaar	25.0
702	Dahi Papdi	Chaat Bazaar	30.0
703	Gol Gappe	Chaat Bazaar	35.0
704	Raj Kachodi	Chaat Bazaar	60.0
801	Daal Makahni	Gravy	120.0
802	Daal Fry	Gravy	90.0
803	Malai Kofta	Gravy	100.0
804	Palak Panner	Gravy	120.0
805	Rajma Masla	Gravy	55.0

Message

 Product Deleted

OK

GEN BILL

Back

ID: 3

Cuisine:

Name:

ID	Name	Cuisine	Price
203	Continental Salad	Continental	40.0
702	Dahi Papdi	Chaat Bazaar	30.0

Deleting from cart



# Query

```
if(!ID_input.getText().equals(""))
{
    try {
        Connection con = getConnection();
        PreparedStatement ps = con.prepareStatement("DELETE
        FROM Table"+OrderID+" WHERE ID = ?");
        int id = Integer.parseInt(ID_input.getText());
        ps.setInt(1,id);
        ps.executeUpdate();
        JOptionPane.showMessageDialog(null, "Product Deleted");
    }
    catch (SQLException ex) {
        Logger.getLogger(Products.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(null, "Product not
        Deleted!");    }    }
    else {
        JOptionPane.showMessageDialog(null,"Product not deleted ->
        No ID to delete");    }
    Show_Products_In_Cart();    }
```

Order ID	Date Time	Price	Tax	CustomerID	Tip
1	2021-11-28 11:...	42.0	2.0	3	0.0

Show Order

Back


ID	Name	Cuisine	Price
203	Continental Salad	Continental	40.0

Show previous orders

# Query

```
ArrayList<Transactions_Table> orderList = new
ArrayList<Transactions_Table>();
Connection con = getConnection();
System.out.println(CustID);
String Query = "SELECT * FROM Transactions WHERE
CustomerID = "+CustID+" ORDER BY orderDate";
Statement st;
ResultSet rs;
try {      st = con.createStatement();
rs = st.executeQuery(Query);
Transactions_Table transaction;
while(rs.next())      {
transaction = new
Transactions_Table(rs.getInt("OrderID"),rs.getString("orderDa
te"),Float.parseFloat(rs.getString("total_price")),Float.parseFl
oat(rs.getString("tax")),rs.getInt("CustomerID"),Float.parseFlo
at(rs.getString("Tip")));
orderList.add(transaction);      }      }
catch (SQLException ex) {
Logger.getLogger(Products.class.getName()).log(Level.SEVERE,
null, ex);      }
return orderList;
```

# Searching

Image: 

ID:

Name:

Price:

Cuisine:

ID	Name	Cuisine	Price
601	Chilli Paneer	Chinese	65.0
602	Chowmein	Chinese	30.0
604	Honey Chilli Potato	Chinese	69.0
605	Munchurian	Chinese	80.0

Customer ID:  Cuisine:

Name:

ID	Name	Cuisine	Price
----	------	---------	-------

```
if(srchbyname.getText()=="") {  
    Show_Products_In_Cuisine();  
    SearchOn = 1; }  
  
else if(srchbycuisine.getText()=="") {  
    Show_Products_In_Name();  
    SearchOn = 2; }  
  
else {  
    Show_Products_In_Cuisine();  
    SearchOn = 1; } }
```

# Searching by Namelist (QUERY)

```
Public ArrayList<Products_Management>getSearchbyNameList() {
ArrayList<Products_Management> productList = new ArrayList<Products_Management>();
Connection con = getConnection();
String Query;
if(srchbycuisine.getText()!="") {
Query = "SELECT * FROM Food_Prđ WHERE Name LIKE '%" +srchbyname.getText()+"%' AND
Cuisine LIKE '%" +srchbycuisine.getText()+"%'"; }
else {
Query = "SELECT * FROM Food_Prđ WHERE Name LIKE '%" +srchbyname.getText()+"%'";
}
Statement st;
ResultSet rs;
try { st = con.createStatement();
rs = st.executeQuery(Query);
Products_Management product;
while(rs.next()) {
product = new
Products_Management(rs.getInt("ID"),rs.getString("Name"),Float.parseFloat(rs.getString("Pri
ce")),rs.getString("Cuisine"),rs.getBytes("Image"));
productList.add(product); } }
catch (SQLException ex) {
Logger.getLogger(Products.class.getName()).log(Level.SEVERE, null, ex); }
return productList; }
```

# Searching with Cuisine list (QUERY)

```
public ArrayList<Products_Management>getSearchbyCuisineList() {
    ArrayList<Products_Management> productList = new
    ArrayList<Products_Management>();
    Connection con = getConnection();
    String Query;
    if(srchbyname.getText()!="") {
        Query = "SELECT * FROM Food_Prđ WHERE Name LIKE
        '%" +srchbyname.getText()+"%' AND Cuisine LIKE
        '%" +srchbycuisine.getText()+"%'";    }
    else {        Query = "SELECT * FROM Food_Prđ WHERE Name LIKE
        '%" +srchbycuisine.getText()+"%'";    }
    Statement st;
    ResultSet rs;
    try {        st = con.createStatement();
        rs = st.executeQuery(Query);
        Products_Management product;
        while(rs.next())    {
            product = new
            Products_Management(rs.getInt("ID"),rs.getString("Name"),Float.parseFloat(rs.g
            etString("Price")),rs.getString("Cuisine"),rs.getBytes("Image"));
            productList.add(product);        }    }
    catch (SQLException ex) {
        Logger.getLogger(Products.class.getName()).log(Level.SEVERE, null, ex);    }
    return productList; }
```



ID	Name	Cuisine	Price
203	Continental Salad	Continental	40.0

CustomerID:

Add Tip:

Date:

Total:

GST 5%:

Payable Amount:

```


Connection con = getConnection();
System.out.println("OID:"+OrderID);
String Query = "SELECT SUM(Price) AS Total FROM
Table"+OrderID;
Statement st;
ResultSet rs;    try {
st = con.createStatement();
rs = st.executeQuery(Query);
while(rs.next())    {
Float total,tax,totalwtax;
total = Float.parseFloat(rs.getString("Total"));
tax = (Float.parseFloat("0.05")*(total));
totalwtax = tax+total;
taxfld.setText(tax.toString());
totalwtaxfld.setText(totalwtax.toString());
amt.setText(total.toString());    }    }
catch (SQLException ex) {
Logger.getLogger(Products.class.getName()).log(L
evel.SEVERE, null, ex);    }

```

# Calculate Amount

ID	Name	Cuisine	Price
203	Continental Salad	Continental	40.0


Message

 Payment Complete Thankyou!

OK

CustomerID:

Add Tip:

Date:  

Calculate amt

Total:

Add Tip

GST 5%:

Payable Amount:

Pay Now

Cancel

Pay now

# QUERY

```
try {      Connection con = getConnection();      PreparedStatement ps1 =
con.prepareStatement("INSERT INTO
transactions(OrderID,total_price,orderDate,tax,CustomerID,tip)values(?,?,?,?,?,?)
");
int row = Cart.getSelectedRow();
ps1.setInt(1,OrderID);
ps1.setString(2,totalwtaxfld.getText());
Date date = new Date();
date = Date.getDate();
String pattern = "yyyy-MM-dd hh:mm:ss";
SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
String finaldate = simpleDateFormat.format(date);
ps1.setString(3,finaldate);
ps1.setString(4,taxfld.getText());
ps1.setInt(5,CustID);
ps1.setFloat(6,Float.parseFloat(add_tip.getText()));
ps1.executeUpdate();
JOptionPane.showMessageDialog(null,"Payment Complete Thankyou!");    }
catch (SQLException ex) {
Logger.getLogger(Bill.class.getName()).log(Level.SEVERE, null, ex);
JOptionPane.showMessageDialog(null,"Payment
Failed!","Error",JOptionPane.ERROR_MESSAGE);    }
close(); }
```



THANK  
YOU