



Team 3		Phase 2
Students		Topic
تسنيم المسالمة	4442	Database System for a coffee shop called "Flame"
لمى أبو سعدة	4442	
رغد باسلم	4442	
شهد الطليان	4442	
ندى الملق	4442	

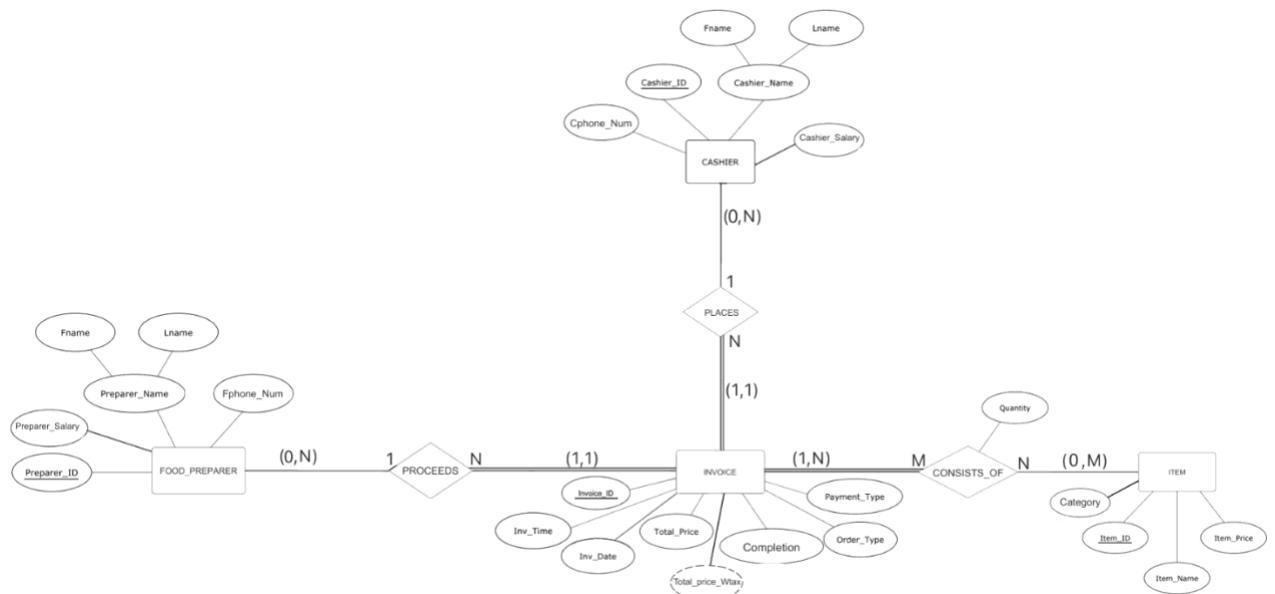
1. Overview:

Database System for a Coffee Shop called “Flame”. It’s helps to streamline the coffee shop’s daily operations, focusing on different perspectives of two users, **The Cashier** who’s responsible for placing the invoice, and **The Food preparer** who’s responsible for processing the invoice by preparing the item, also responsible for updating the completion of the invoice preparation after preceding it once the items are ready.

2. The requirements:

- The Cashier is identified by a unique Cashier ID. For each cashier, there is: a phone number, Name (First Name, Last Name), and salary must be recorded.
- The Invoice is identified by a unique Invoice ID. For each invoice there is: invoice date, time, Completion state, total price, total price with tax, order type (example: eat-in or take-away), and payment method (example: VISA, cash, or MADA) must be recorded.
 - each invoice is placed by one cashier.
 - Food preparers can update the completion state of the invoice after preceding it once the items are ready.
- The Food **preparer** is identified by a unique Preparer ID. For each food preparer, there is: a phone number, name (first and last name), and salary must be recorded.
 - Each food preparer proceeds invoices, and each invoice can be processed by only one food preparer.
- The Item is identified by a unique Item ID. Each item has a name and a price.
 - Each invoice must consist of at least one item, and each item can appear in multiple invoices.
 - Each invoice can have multiple quantity of a certain item.

3. The ER diagram:



4. Relational Schema:

CASHIER

<u>Cashier_ID</u>	Fname	Lname	Cphone_Num	Cashier_Salary
-------------------	-------	-------	------------	----------------

FOOD_PREPARER

<u>Preparer_ID</u>	Fname	Lname	Fphone_Num	Preparer_Salary
--------------------	-------	-------	------------	-----------------

INVOICE

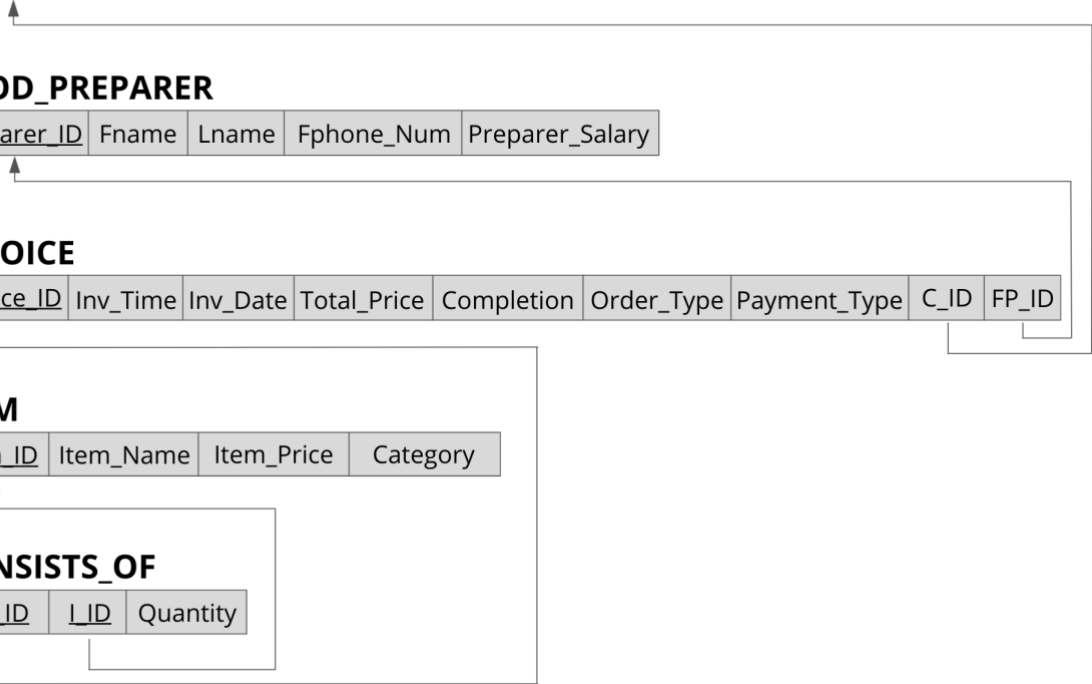
<u>Invoice_ID</u>	Inv_Time	Inv_Date	Total_Price	Completion	Order_Type	Payment_Type	C_ID	FP_ID
-------------------	----------	----------	-------------	------------	------------	--------------	------	-------

ITEM

<u>Item_ID</u>	Item_Name	Item_Price	Category
----------------	-----------	------------	----------

CONSISTS_OF

<u>Inv_ID</u>	<u>I_ID</u>	Quantity
---------------	-------------	----------



5. System users:

This system has 2 users' group. we will be showing all of them which they are cashier, and food preparer.

5.1. Cashier View

This user group has access to essential parts of the database that facilitate customer transactions and invoice management. (Each Cashier can view his full Info only)

Their key responsibilities include: The Cashier at "Flame" coffee shop is responsible for accurately entering customer orders into the system, add the payments type, and issuing invoices. They also ensure the order details are correct before sending them to the Food Preparer for preparation, contributing to fast and organized service for customers.

CASHIER

Cashier_ID	Fname	Lname	CPhone_Num	Cashier_Salary
------------	-------	-------	------------	----------------

INVOICE

Invoice_ID	Inv_Time	Inv_Date	Total_Price	Order_Type	Payment_Type	C_ID
------------	----------	----------	-------------	------------	--------------	------

ITEM

Item_ID	Item_Name	Category	Item_Price
---------	-----------	----------	------------

CONSISTS_OF

Inv_ID	I_ID	Quantity
--------	------	----------

5.1.1. Insert Operations

Insert Invoices
<p>Insert invoices</p> <pre>INSERT INTO INVOICE VALUES (1, '12:30:00', '2024-10-10', 100, TRUE, 'Dine-In', 'Card', 1, 1), (2, '13:45:00', '2024-10-10', 200, FALSE, 'Takeout', 'Cash', 2, 2), (3, '14:15:00', '2024-10-11', 150, TRUE, 'Takeout', 'Card', 1, 3);</pre>
Insert items to invoices (into CONSISTS_OF table)
<pre>INSERT INTO CONSISTS_OF (Inv_ID, I_ID, Quantity) VALUES (1, 1, 2), (1, 4, 1), (2, 2, 1), (3, 3, 3), (3, 4, 2);</pre>

5.1.2. Updates operations

Update operation (update Total_price for a certain Invoice)	
<pre>UPDATE INVOICE SET Total_Price = 100 where Invoice_ID == 1;</pre>	
Update	
<pre>UPDATE ITEM SET Item_Price = 15 WHERE Item_ID = '1';</pre>	

5.1.3. Delete operations

Removes a specific item from a specific invoice.
<pre>DELETE FROM Consist_Of WHERE Inv_ID = 1 AND I_ID = 2;</pre>
Removes a particular invoice completely from the system.
<pre>DELETE FROM INVOICE WHERE Invoice_ID = 3;</pre>

5.1.4. Retrieve operations

The first operation retrieves the Total_Price from the INVOICE table for an invoice where the ID is '2'. Essentially, this allows the cashier to find out the total amount charged on the specific invoice number '2'.

The second operation retrieves the item_Price for the product named 'Iced coffee' from the ITEM table. This helps the cashier find the price of "Iced coffee" in the database

```
SELECT Total_Price
From INVOICE
WHERE invoice_ID ='2';

SELECT item_Price
From ITEM
WHERE Item_Name ='Iced coffee';
```

This operation retrieves the Invoice ID, the first name, and the last name of the cashier, along with the invoice time and total price for all invoices handled by the cashier with ID = 1.

```
SELECT
    INVOICE.Invoice_ID,
    CASHIER.Fname,
    CASHIER.Lname,
    INVOICE.Inv_Time,
    INVOICE.Total_Price
FROM
    CASHIER
JOIN
    INVOICE ON CASHIER.Cashier_ID = INVOICE.C_ID
WHERE
    CASHIER.Cashier_ID = 1;
```

retrieve all the INVOICE info including the the first and last names of the related Food_Preparer and Cashier

```
SELECT
    I.Invoice_ID,
    I.Inv_Time,
    I.Inv_Date,
    I.Total_Price,
    I.Completion,
    I.Order_Type,
    I.Payment_Type,
    C.Fname AS Cashier_First_Name,
    C.Lname AS Cashier_Last_Name,
    FP.Fname AS Preparer_First_Name,
    FP.Lname AS Preparer_Last_Name
FROM
    INVOICE I
JOIN
    CASHIER C ON I.C_ID = C.Cashier_ID
JOIN
    FOOD_PREPARER FP ON I.FP_ID = FP.Preparer_ID;
```

5.2. FOOD PREPARER VIEW

This user group has access to specific parts of the database that support food preparation and order fulfillment. (each Food Preparer can view his full Info only)

Their key responsibilities include: The Food Preparer at "Flame" coffee shop is responsible for preparing orders accurately and efficiently based on what is received from the cashier. After completing the preparation, they update the system to confirm the order is complete, ensuring a smooth workflow and timely service for customers.

FOOD_PREPARER

Preparer_ID	Fname	Lname	Fphon_Num	Preparer_Salary
-------------	-------	-------	-----------	-----------------

INVOICE

Invoice_ID	Inv_Time	Inv_Date	Completion	Order_Type	FP_ID
------------	----------	----------	------------	------------	-------

ITEM

Item_ID	Item_Name	Category	Item_Price
---------	-----------	----------	------------

CONSISTS_OF

Inv_ID	I_ID	Quantity
--------	------	----------

5.2.1. Insert Operations

Insert new Item

```
INSERT INTO ITEM
VALUES (1, 'Iced coffee', 12, 'Cold Drinks'),
       (2, 'Black coffee', 10, 'Hot Drinks'),
       (3, 'Spanish latte', 14, 'Hot Drinks'),
       (4, 'water', 1, 'Cold Drinks');
```

Insert new food preparer

```
INSERT INTO FOOD_PREPARER
VALUES (1, 'Lama', 'Mohammed', '5551112', NULL);
```

5.2.2. Update Operations

Update the phone number for the food preparer

```
UPDATE FOOD_PREPARER  
SET Fphone_Num = '5652222'  
WHERE Preparer_ID='2';
```

Update the completion attribute

```
UPDATE INVOICE  
SET Completion = true  
WHERE Invoice_ID == 2;
```

5.2.3. Delete Operations

Delete
<pre>DELETE FROM ITEM WHERE Item_ID = 3;</pre>
Delete
<pre>DELETE FROM FOOD_PREPARER WHERE Preparer_ID = 2;</pre>

5.2.4. Retrieve Operations

The first retrieves the quantity of a specific item (with Item_ID = 1) from the
Consist_Of table

The second operation will retrieve whether the invoice with ID 2 is marked as complete (TRUE) or incomplete (FALSE) , the completion status for invoice 2 is FALSE.

```
Select Quantity
From Consist_Of
Where I_ID=1;

SELECT completion
From INVOICE
WHERE Invoice_ID ='2';
```

This operation retrieves the Invoice ID and Order Type for all orders handled by a specific food preparer.

```
SELECT INVOICE.Invoice_ID, INVOICE.Order_Type AS OrderType
FROM FOOD_PREPARER
JOIN INVOICE ON FOOD_PREPARER.Preparer_ID = INVOICE.FP_ID
WHERE FOOD_PREPARER.Preparer_ID = 1;
```

retrieve INVOICE ID including the the first and last names of the related Food_Preparer and Cashier and number of item

```
SELECT
    I.Invoice_ID,
    C.Fname AS Cashier_First_Name,
    FP.Fname AS Preparer_First_Name,
    COUNT(CO.I_ID) AS Number_of_Items
FROM
    INVOICE I
JOIN
    CASHIER C ON I.C_ID = C.Cashier_ID
JOIN
    FOOD_PREPARER FP ON I.FP_ID = FP.Preparer_ID
JOIN
    consist_of CO ON I.Invoice_ID = CO.Inv_ID
GROUP BY
    I.Invoice_ID, C.Fname, FP.Fname;
```

6. Implementation

6.1. Connection part:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost/flare"; // Use your database name here
    private static final String USER = "root"; // Replace with your database username
    private static final String PASSWORD = "1234"; // Replace with your database password

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

6.2. The code:

To access the java files, go to Flame > src. The Main class is FlameUI.java.

6.2.1. Check ID in Welcome Page

```
private void signbttActionPerformed(java.awt.event.ActionEvent evt) {  
    try{  
        Connection con = DriverManager.getConnection(DB_URL, USER, PASS);  
  
        int id;  
        try {  
            id = Integer.parseInt(IDtextf.getText());  
        } catch (NumberFormatException e) {  
            JOptionPane.showMessageDialog(null, "Please enter a valid numerical ID.", "Invalid Input", JOptionPane.ERROR_MESSAGE);  
            return;  
        }  
  
        if(P1ComboBox.getSelectedItem().toString().equals("Cashier")){  
            String sql = "SELECT * FROM cashier WHERE Cashier_ID = " + id;  
            Statement stmt = con.createStatement();  
            ResultSet rs = stmt.executeQuery(sql);  
            if(rs.next()){  
                ID = id;  
                EmployeeType = "Cashier";  
                GenerateItems(); //added by Raghad  
                BaseLayout.removeAll();  
                BaseLayout.add(Page2Panel);  
                BaseLayout.repaint();  
                BaseLayout.validate();  
            }else{  
                JOptionPane.showMessageDialog(this, "ID is incorrect");  
                IDtextf.setText("");  
            }  
        }  
  
        else if(P1ComboBox.getSelectedItem().toString().equals("Food Preparer")){  
            String sq = "SELECT * FROM food_preparer WHERE preparer_ID = " + id;  
            Statement stm = con.createStatement();  
            ResultSet r = stm.executeQuery(sq);  
            if(r.next()){  
                ID = id;  
                EmployeeType = "Food Preparer";  
                BaseLayout.removeAll();  
                BaseLayout.add(Page3Panel);  
                BaseLayout.repaint();  
                BaseLayout.validate();  
            }else{  
                JOptionPane.showMessageDialog(this, "ID is incorrect");  
                IDtextf.setText("");  
                P1ComboBox.setSelectedIndex(0);  
            }  
        }  
  
        con.close();  
    } catch (Exception e) //close try  
    {System.out.println(e.getMessage());  
    }  
}
```


6.2.2. Retrieve items and adding them to the menu in Cashier page

In ItemDAO.java file

```
public static List<Item> getAllItems() throws SQLException {
    List<Item> items = new ArrayList<>();
    String query = "SELECT * FROM item";
    try (Connection conn = DatabaseConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery()) {

        while (rs.next()) {
            Item item = new Item(
                rs.getInt("Item_ID"),
                rs.getString("Item_Name"),
                rs.getInt("Item_Price"),
                rs.getString("Category")
            );
            item.setItemId(rs.getInt("Item_ID"));
            item.setName(rs.getString("Item_Name"));
            item.setPrice(rs.getInt("Item_Price"));
            item.setCategory(rs.getString("Category"));
            items.add(item);
        }
    }
    return items;
}
```

In FlameUI.java file:

```
public void GenerateItems(){
    P2P4Panel1.removeAll();
    P2P4Panel1.repaint();
    P2P4Panel1.revalidate();

    P2P4Panel2.removeAll();
    P2P4Panel2.repaint();
    P2P4Panel2.revalidate();

    P2P4Panel3.removeAll();
    P2P4Panel3.repaint();
    P2P4Panel3.revalidate();
    List<Item> list = null;
    try {
        list = ItemDAO.getAllItems();
    } catch (SQLException ex) {
        Logger.getLogger(FlameUI.class.getName()).log(Level.SEVERE, null, ex);
    }

    for (Item list1 : list) {
        AddItemToMenu(list1.getName(), (int)list1.getPrice(), list1.getItemId(), list1.getCategory());
    }
}
```

```

public void AddItemToMenu(String IName,int IPrice,int IId,String CTGRY) {
    javax.swing.JPanel jPanelItem = new javax.swing.JPanel();
    jPanelItem.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0), 2));
    jPanelItem.setLayout(new java.awt.BorderLayout(10, 10));
    jPanelItem.setSize(new Dimension(100,100));

    javax.swing.JLabel ItemName = new javax.swing.JLabel();
    ItemName.setText(IName);
    |

    javax.swing.JLabel ItemPrice = new javax.swing.JLabel();
    ItemPrice.setText(Integer.toString(IPrice));

    javax.swing.JButton ItemButton = new javax.swing.JButton();
    ItemButton.setText("ADD");
    ItemButton.putClientProperty("ID", IId);
    ItemButton.putClientProperty("Name", IName);
    ItemButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            AddItemToMenu(((int)ItemButton.getClientProperty("ID")), ((String)ItemButton.getClientProperty("Name")), IPrice);
        });
    });
    jPanelItem.add(ItemName, java.awt.BorderLayout.PAGE_START);
    jPanelItem.add(ItemPrice, java.awt.BorderLayout.CENTER);
    jPanelItem.add(ItemButton, java.awt.BorderLayout.PAGE_END);
    if(CTGRY.equalsIgnoreCase("Hot Drinks")){
        P2P4Panel1.add(jPanelItem);
        P2P4Panel1.repaint();
        P2P4Panel1.revalidate();
    }else if(CTGRY.equalsIgnoreCase("Cold Drinks")){
        P2P4Panel2.add(jPanelItem);
        P2P4Panel2.repaint();
        P2P4Panel2.revalidate();
    }
}

```

```

}else if(CTGRY.equalsIgnoreCase("Cold Drinks")){
    P2P4Panel2.add(jPanelItem);
    P2P4Panel2.repaint();
    P2P4Panel2.revalidate();
}else{
    P2P4Panel3.add(jPanelItem);
    P2P4Panel3.repaint();
    P2P4Panel3.revalidate();
}
}

```

6.2.3. Insert new Invoice and insert its items to the DB

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    int InvID = 0;  
    InvT1 = (String)jComboBox2.getSelectedItem();  
    InvT2 = (String)jComboBox3.getSelectedItem();  
    PaymentTypeDialog.dispose();  
    String insertQuery = "INSERT INTO INVOICE (Inv_Time,Inv_Date ,Total_Price ,Completion ,Order_Type ,Payment_Type ,C_ID,FP_ID ) VALUES (?,?,?,?,?,?,?,?)";  
    //insert invoice  
    try {Connection conn = DriverManager.getConnection(DB_URL,USER,PASS );  
        PreparedStatement pstmt = conn.prepareStatement(insertQuery, PreparedStatement.RETURN_GENERATED_KEYS) {  
            LocalDateTime currentTime = LocalDateTime.now();  
            LocalDate currentDate = LocalDate.now();  
            pstmt.setTime(1, java.sql.Time.valueOf(currentTime));  
            pstmt.setDate(2, java.sql.Date.valueOf(currentDate));  
            pstmt.setInt(3,Integer.parseInt(TP2.getText()));  
            pstmt.setBoolean(4, false);  
            pstmt.setString(5,InvT2);  
            pstmt.setString(6,InvT1);  
            pstmt.setInt(7,ID);  
            pstmt.setNull(8,java.sql.Types.INTEGER);  
            pstmt.executeUpdate();  
            try (ResultSet generatedKeys = pstmt.getGeneratedKeys()) {  
                if (generatedKeys.next()) {  
                    InvID = generatedKeys.getInt(1);  
                }  
            }  
        }  
        pstmt.close();  
        conn.close();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```
//insert items to invoice  
DefaultTableModel model1 = (DefaultTableModel) InvItemsListT.getModel();  
for (int i = 0; i < model1.getRowCount(); i++) {  
    int currentQuantity = (int) model1.getValueAt(i, 1);  
    int currentID = (int) model1.getValueAt(i, 0);  
    insertQuery = "INSERT INTO Consist_Of (Inv_ID, I_ID, Quantity) VALUES (?,?,?)";  
    //insert invoice  
    try {Connection conn = DriverManager.getConnection(DB_URL,USER,PASS );  
        PreparedStatement pstmt = conn.prepareStatement(insertQuery) {  
            LocalDateTime currentTime = LocalDateTime.now();  
            LocalDate currentDate = LocalDate.now();  
            pstmt.setInt(1, InvID);  
            pstmt.setInt(2, currentID);  
            pstmt.setInt(3,currentQuantity);  
  
            pstmt.executeUpdate();  
            pstmt.close();  
            conn.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
    JOptionPane.showMessageDialog(null, "Invoice has been added successfully!");  
    for (int i = model1.getRowCount() - 1; i >= 0; i--) {  
        model1.removeRow(i);  
    }  
    TP2.setText(" ");  
    TPT2.setText(" ");  
}
```

6.2.4. Retrieve and Update the employee information (Whether Cashier or Food Preparer)

Update:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    String phoneNumber = jTextField4.getText();  
    if (phoneNumber.length() != 9)  
        JOptionPane.showMessageDialog(null, "The Phone Number Must Be 9 digits.", "Invalid Input", JOptionPane.ERROR_MESSAGE);  
    else  
        try {  
            if (EmployeeType.equals("Cashier")) {  
                Connection con = DriverManager.getConnection("jdbc:mysql://localhost/flame", "root", "1234");  
                String query = "UPDATE cashier SET Fname=?, Lname=?, Cphone_Num=? WHERE cashier_id=?";  
                PreparedStatement ps = con.prepareStatement(query);  
                ps.setString(1, jTextField2.getText());  
                ps.setString(2, jTextField3.getText());  
                ps.setString(3, jTextField4.getText());  
                ps.setInt(4, ID);  
                ps.executeUpdate();  
                con.close();  
            } else {  
                Connection con = DriverManager.getConnection("jdbc:mysql://localhost/flame", "root", "1234");  
                String query = "UPDATE food_preparer SET Fname=?, Lname=?, Fphone_Num=? WHERE Preparer_ID=?";  
                PreparedStatement ps = con.prepareStatement(query);  
                ps.setString(1, jTextField2.getText());  
                ps.setString(2, jTextField3.getText());  
                ps.setString(3, jTextField4.getText());  
                ps.setInt(4, ID);  
                ps.executeUpdate();  
                con.close();  
            }  
        }  
    }  
}
```

Retrieve Cashier information:

```
private void P2P1Button1ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        MyInfoDialog.setLocationRelativeTo(this); // Center relative to the main frame  
        MyInfoDialog.pack();  
        MyInfoDialog.setVisible(true);  
        Connection con = DriverManager.getConnection(DB_URL, USER, PASS);  
        String sql = "SELECT * FROM cashier WHERE Cashier_ID = " + ID;  
        Statement stmt = con.createStatement();  
        ResultSet rs = stmt.executeQuery(sql);  
        if (rs.next()) {  
            jTextField1.setText(Integer.toString(rs.getInt(1)));  
            jTextField2.setText(rs.getString(2));  
            jTextField3.setText(rs.getString(3));  
            jTextField4.setText(rs.getString(4));  
            jTextField5.setText(rs.getString(5));  
        }  
    } catch (SQLException ex) {  
        Logger.getLogger(FlameUI.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

Retrieve food preparer information:

```

private void btnMyInformationActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        MyInfoDialog.setLocationRelativeTo(this); // Center relative to the main frame
        MyInfoDialog.pack();
        MyInfoDialog.setVisible(true);
        Connection con = DriverManager.getConnection(DB_URL,USER,PASS);
        String sql = "SELECT * FROM food_preparer WHERE Preparer_ID = " + ID;
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        if(rs.next()){
            jTextField1.setText(Integer.toString(rs.getInt(1)));
            jTextField2.setText(rs.getString(2));
            jTextField3.setText(rs.getString(3));
            jTextField4.setText(rs.getString(4));
            jTextField5.setText(rs.getString(5));
        }
    } catch (SQLException ex) {
        Logger.getLogger(FlameUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

6.2.5.Retrieve Invoices

```

private void loadInvoices() {
    try {
        String url = FlameUI.DB_URL;
        String user = FlameUI.USER;
        String password = FlameUI.PASS;
        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            String query = "SELECT * FROM INVOICE";
            PreparedStatement stmt = conn.prepareStatement(query);
            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                int invoiceId = rs.getInt("Invoice_ID");
                java.sql.Date invDate = rs.getDate("Inv_Date");
                int totalPrice = rs.getInt("Total_Price");
                String orderType = rs.getString("Order_Type");
                String completion = Boolean.toString(rs.getBoolean("Completion"));
                tableModel.addRow(new Object[]{invoiceId, invDate, totalPrice, orderType, completion});
            }
            conn.close();
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Error loading invoices: " + ex.getMessage());
    }
}

```

Retrieve completed invoices:

```

boolean completedOnly = completedOnlyCheckBox.isSelected();
if (completedOnly) {
    try {
        String url = FlameUI.DB_URL;
        String user = FlameUI.USER;
        String password = FlameUI.PASS;
        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            String query = "SELECT * FROM INVOICE WHERE Completion=true";
            PreparedStatement stmt = conn.prepareStatement(query);
            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                int invoiceId = rs.getInt("Invoice_ID");
                java.sql.Date invDate = rs.getDate("Inv_Date");
                int totalPrice = rs.getInt("Total_Price");
                String orderType = rs.getString("Order_Type");
                String completion = Boolean.toString(rs.getBoolean("Completion"));
                tableModel.addRow(new Object[]{invoiceId, invDate, totalPrice, orderType, completion});
            }
            conn.close();
        }
    }
}

```

Retrieve Uncompleted invoices and show the number of items in it

```

private void populateWaitingList() {
    String sql = "SELECT I.Invoice_ID, COUNT(CO.I_ID) AS Number_of_Items FROM INVOICE I JOIN consist_of CO ON I.Invoice_ID = CO.Inv_ID WHERE I.Completion = false";
    try {
        Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        PreparedStatement stmt = conn.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            waitingListModel.addRow(new Object[] {
                rs.getInt("I.Invoice_ID"),
                rs.getInt("Number_of_Items")
            });
        }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "noo " + e.getMessage()); // Handle the exception
    }
}

```

The full String sql is

SELECT I.Invoice_ID, COUNT(CO.I_ID) AS Number_of_Items

FROM INVOICE I

JOIN consist_of CO

ON I.Invoice_ID = CO.Inv_ID

WHERE I.Completion = false

GROUP BY I.Invoice_ID;

6.2.6.Update completion of the invoice

```

private void completeInvoice() {
    int selectedRow = currentInvoice.getSelectedRow();
    if (selectedRow != -1) {
        int invoiceId = Integer.parseInt(invoiceLabel.getText().substring(9));
        String sql = "UPDATE INVOICE SET Completion = true, FP_ID = ? WHERE Invoice_ID = ?";
        try {
            Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
            PreparedStatement stmt = conn.prepareStatement(sql) {
                stmt.setInt(1, ID);
                stmt.setInt(2, invoiceId);
                stmt.executeUpdate();
                // Refresh the tables
                tableModel.setRowCount(0);
                waitingListModel.setRowCount(0);
                invoiceLabel.setText("");
                populateWaitingList();
                populateTable();
            } catch (SQLException ex) {
                ex.printStackTrace(); // Handle the exception
                JOptionPane.showMessageDialog(this, "not complete" + ex.getMessage());
            }
        }
    }
}

```

```

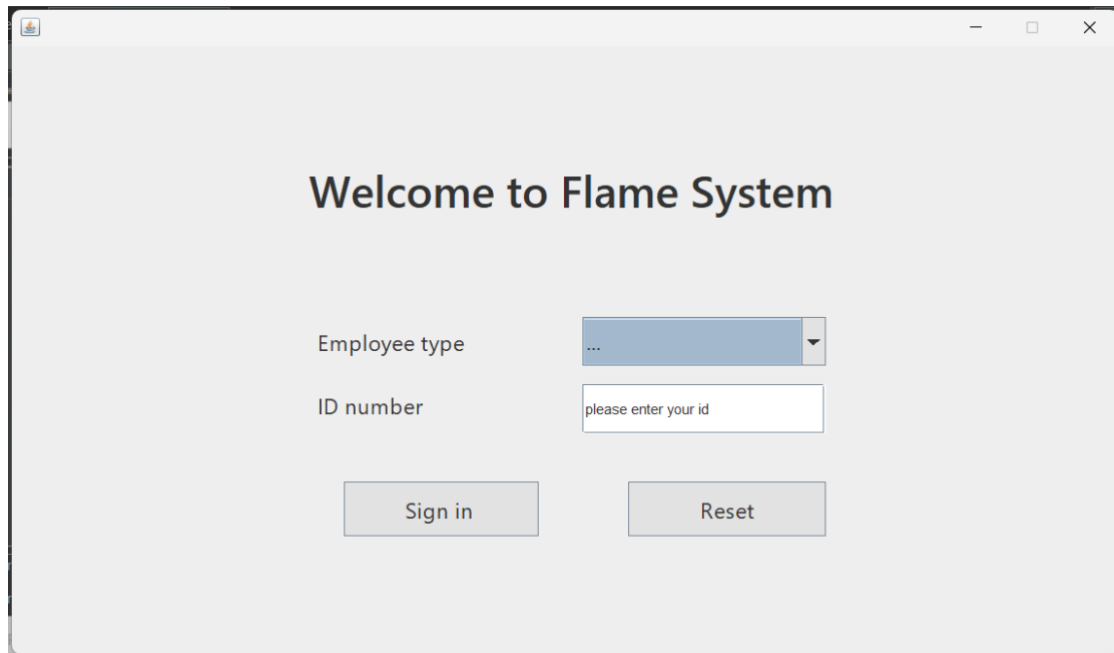
private void completeInvoice() {
    int selectedRow = currentInvoice.getSelectedRow();
    if (selectedRow != -1) {
        int invoiceId = Integer.parseInt(invoiceLabe.getText().substring(9));
        String sql = "UPDATE INVOICE SET Completion = true WHERE Invoice_ID = ?";
        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
            PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, invoiceId);
            stmt.executeUpdate();
            // Refresh the tables
            tableModel.setRowCount(0);
            waitingListModel.setRowCount(0);
            invoiceLabe.setText("");
            populateWaitingList();
            populateTable();
        } catch (SQLException ex) {
            ex.printStackTrace(); //Handle the exception
            JOptionPane.showMessageDialog(this, "not complete" + ex.getMessage());
        }
    }
}

```

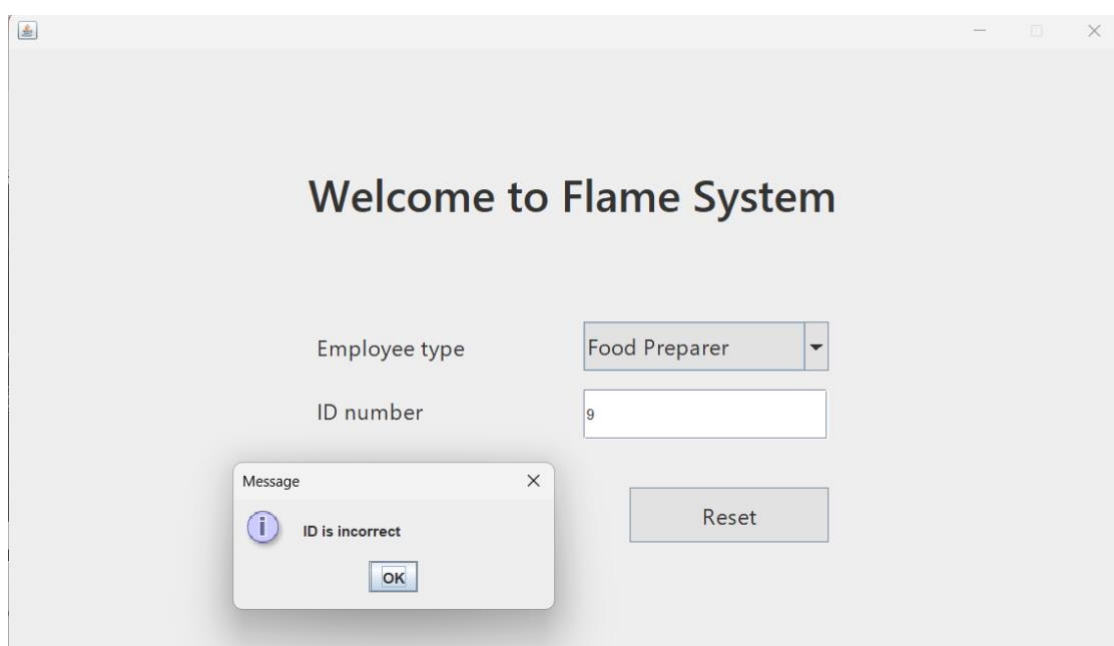
6.3. GUI Screenshot

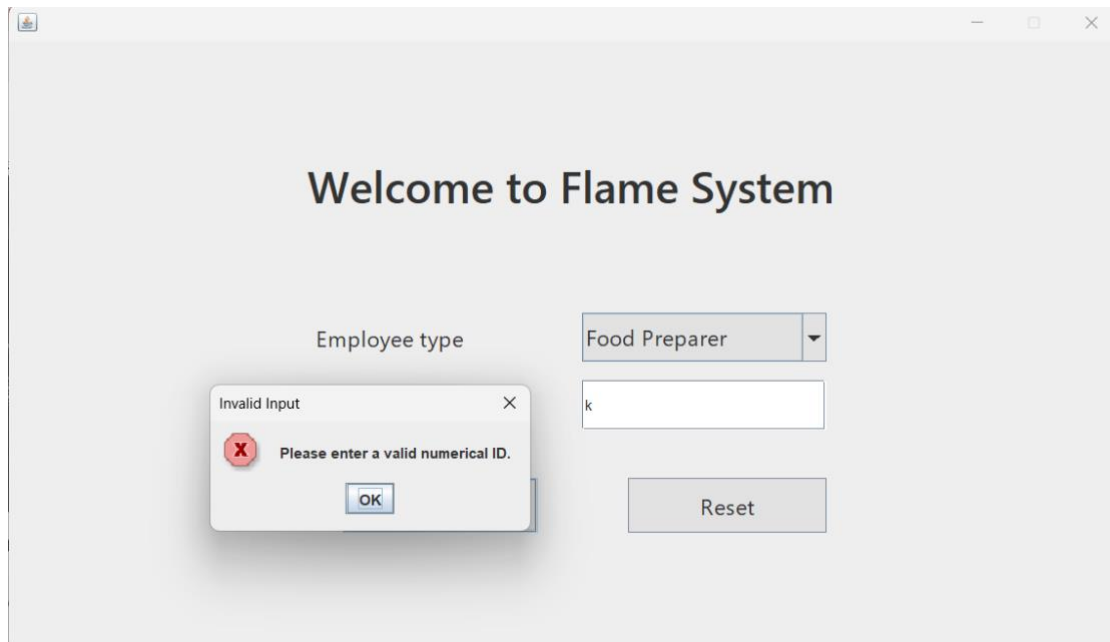
6.3.1. Welcome Page

The first GUI will appear for both users, the cashier and the food preparer.



if the user enters the wrong information it shows a message to inform them.

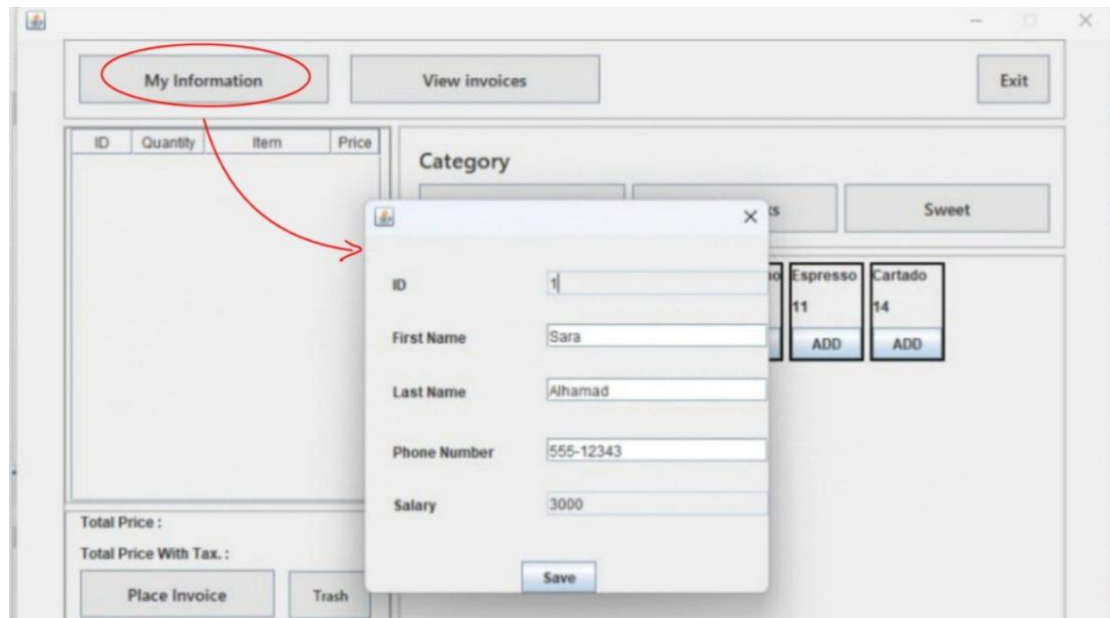




-once the user enters their information correctly it either sends them to the cashier or food preparer page.

6.3.2. Cashier Page

The "my information" button in the Cashier interface opens a window displaying the user's information (ID, name, phone number, salary) and he can only modify his name and phone number.



The "save" button updates the new information. When you click on it, the change will be saved and the length of the phone number will be verified to be 9 digits:



If the phone number is not 9 digits, a message will appear:



Before update:

flame.cashier: 2 rows total (exact)

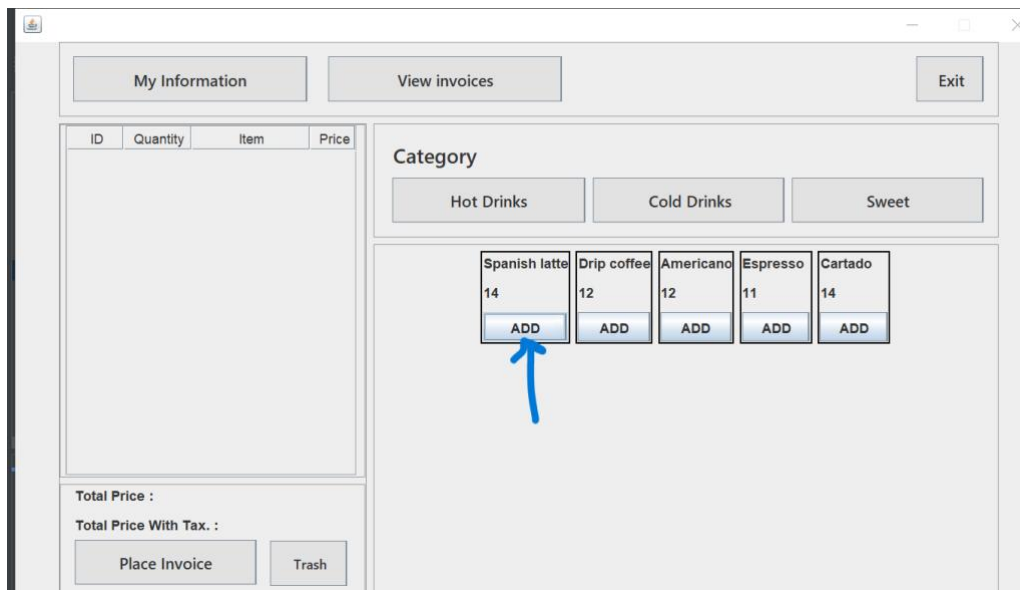
#	Cashier_ID	Fname	Lname	Cphone_Num	Cashier_Salary
1	1	lama	hosam	555-44444	3,000
2	2	Nora	Alsaleh	555-5678	3,500

After update:

#	Cashier_ID	Fname	Lname	Cphone_Num	Cashier_Salary
1	1	sara	ahmad	555-33333	3,000
2	2	Nora	Alsaleh	555-5678	3,500

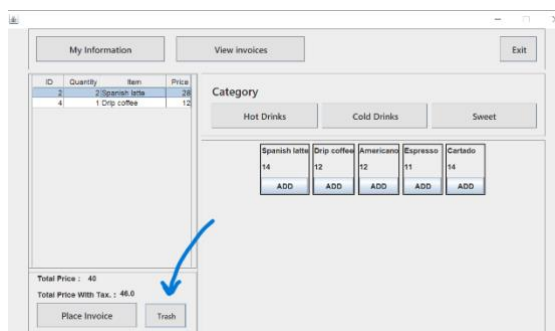
Adding items and place Invoice:

To item to the invoice, click add. And to increase the quantity, click Add again.

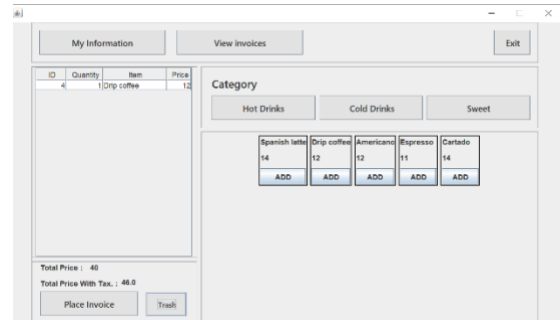


To delete a specific item from the list, select the item, then click Trash.

Before:

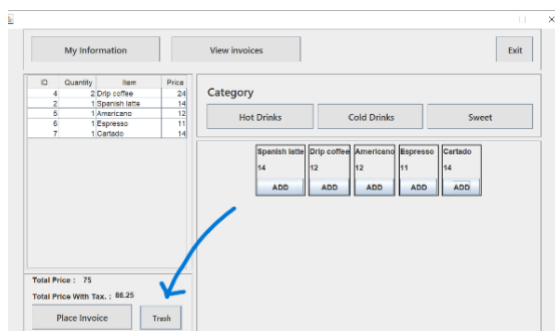


After:

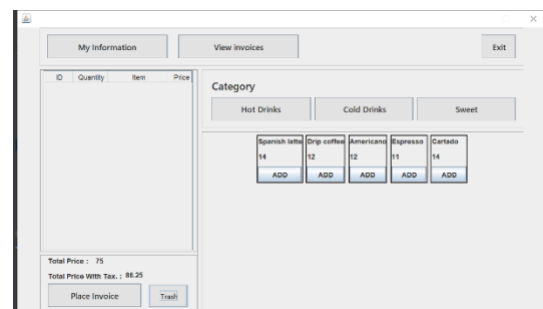


To delete the whole list, just click Trash without selecting.

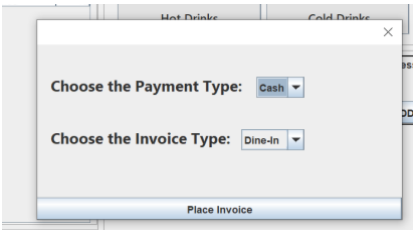
Before:



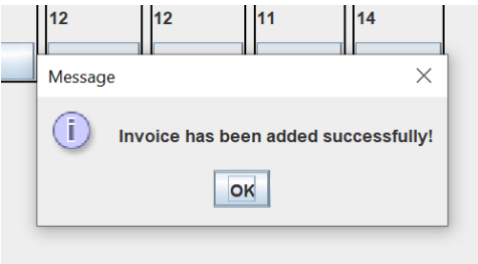
. After:



When Place Invoice is clicked and the invoice's item list is not empty, a Dialog will shows up. The cashier will select the payment type and invoice type.



After clicking Place Invoice button in the dialog, the invoice and its will be added to the DB successfully.



DB Before adding Invoice:

#	Invoice_ID	Inv_Time	Inv_Date	Total_Price	Completion	Order_Type	Payment_Type	C_ID	FP_ID
1	1	12:39:00	2024-10-10	100	1	Dine-In	Card	1	1
2	2	13:45:00	2024-10-10	200	1	Takeout	Cash	2	(NULL)
3	3	14:15:00	2024-10-11	150	1	Takeout	Card	1	3
4	4	02:31:18	2024-11-15	49	1	Dine-In	Cash	1	1
5	5	02:34:15	2024-11-15	38	0	Takeout	Card	1	(NULL)
6	6	02:35:54	2024-11-15	38	0	Dine-In	Cash	1	(NULL)
7	7	02:39:14	2024-11-15	42	0	Dine-In	Cash	1	(NULL)
8	8	02:39:56	2024-11-15	42	0	Dine-In	Cash	1	(NULL)
9	9	02:41:40	2024-11-15	28	1	Dine-In	Cash	1	(NULL)
10	10	02:43:07	2024-11-15	27	1	Takeout	Card	1	(NULL)
11	11	02:44:57	2024-11-15	49	1	Takeout	Card	1	(NULL)
12	12	02:58:58	2024-11-15	42	1	Dine-In	Cash	1	(NULL)
13	13	21:29:00	2024-11-16	28	0	Dine-In	Cash	1	1
14	14	21:29:06	2024-11-16	12	1	Dine-In	Cash	1	1

DB after adding Invoice:

#	Invoice_ID	Inv_Time	Inv_Date	Total_Price	Completion	Order_Type	Payment_Type	C_ID	FP_ID
1	1	12:39:00	2024-10-10	100	1	Dine-In	Card	1	1
2	2	13:45:00	2024-10-10	200	1	Takeout	Cash	2	(NULL)
3	3	14:15:00	2024-10-11	150	1	Takeout	Card	1	3
4	4	02:31:18	2024-11-15	49	1	Dine-In	Cash	1	1
5	5	02:34:15	2024-11-15	38	0	Takeout	Card	1	(NULL)
6	6	02:35:54	2024-11-15	38	0	Dine-In	Cash	1	(NULL)
7	7	02:39:14	2024-11-15	42	0	Dine-In	Cash	1	(NULL)
8	8	02:39:56	2024-11-15	42	0	Dine-In	Cash	1	(NULL)
9	9	02:41:40	2024-11-15	28	1	Dine-In	Cash	1	(NULL)
10	10	02:43:07	2024-11-15	27	1	Takeout	Card	1	(NULL)
11	11	02:44:57	2024-11-15	49	1	Takeout	Card	1	(NULL)
12	12	02:58:58	2024-11-15	42	1	Dine-In	Cash	1	(NULL)
13	13	21:29:00	2024-11-16	28	0	Dine-In	Cash	1	1
14	14	21:29:06	2024-11-16	12	1	Dine-In	Cash	1	1
15	15	02:51:00	2024-11-17	26	0	Dine-In	Cash	1	(NULL)

#	Inv_ID	I_ID	Quantity
1	1	1	2
2	1	4	1
3	2	2	1
4	3	3	3
5	3	4	2
6	9	2	2
7	10	3	1
8	10	8	1
9	10	1	1
10	11	2	1
11	11	4	1
12	11	5	1
13	11	6	1
14	12	2	3
15	13	2	2
16	14	5	1

#	Inv_ID	I_ID	Quantity
1	1	1	2
2	1	4	1
3	2	2	1
4	3	3	3
5	3	4	2
6	9	2	2
7	10	3	1
8	10	8	1
9	10	1	1
10	11	2	1
11	11	4	1
12	11	5	1
13	11	6	1
14	12	2	3
15	13	2	2
16	14	5	1
17	16	2	1
18	16	4	1

6.3.3. Food Preparer Page

- Adding new item :

The screenshot shows a software interface for a food preparer. At the top, there are three buttons: 'My Information', 'View invoices', and 'Create new Item' (which is circled in red). To the right is an 'Exit' button. Below these buttons is a section labeled 'Invoice #4' containing a table with the following data:

Item_ID	Item_Name	Category	Quantity
2	Spanish latte	Hot Drinks	1

Below the table is a 'Complete' button. At the bottom of the page is an 'Invoice Waitlist' section with a table:

Invoice_ID	Completion
4	false
5	false
3	false
7	false

-once the food preparer presses the (Create new item) button it opens a new page to add a new item to the database.

The screenshot shows a 'Create New Item' dialog box. It has three input fields: 'Item Name' with the value 'brownie', 'Price' with the value '12', and 'Category' with a dropdown menu showing 'Sweets'. Below these fields are 'Add' and 'Clear' buttons. A 'Message' dialog box is overlaid on top, displaying an information icon and the text 'insertion successful.' with an 'OK' button.

-When the user enters the wrong format of the price:

The screenshot shows a web application window titled "Create New Item". It contains three input fields: "Item Name:" with the value "water", "Price:" with the value "jbkgjg", and "Category:" with a dropdown menu showing "Cold Drinks". An "Exit" button is in the top right, and a "Clear" button is at the bottom right. An "Invalid Input" error dialog is displayed in the center, with a red 'X' icon and the message "Please enter a valid numerical price." and an "OK" button.

-The item database before creating a new item:

#	Item_ID	Item_Name	Item_Price	Category
1	3	water	1	Cold Drinks
2	2	Spanish latte	14	Hot Drinks
3	17	matcha	21	Cold Drinks
4	13	Marble Cake	8	Sweets
5	12	lemon Cake	8	Sweets
6	1	Iced coffee	12	Cold Drinks
7	10	Ice Latte	17	Cold Drinks
8	8	Ice Americano	14	Cold Drinks
9	6	Espresso	11	Hot Drinks
10	4	Drip coffee	12	Hot Drinks
11	11	Cookie	8	Sweets
12	9	Cold brew	20	Cold Drinks
13	16	cold brew	16	Cold Drinks
14	14	chocolate baka	12	Sweets
15	15	chocolate baka	12	Sweets
16	7	Cartado	14	Hot Drinks
17	5	Americano	12	Hot Drinks

-After:

#	Item_ID	Item_Name	Item_Price	Category
1	3	water	1	Cold Drinks
2	2	Spanish latte	14	Hot Drinks
3	17	matcha	21	Cold Drinks
4	13	Marble Cake	8	Sweets
5	12	lemon Cake	8	Sweets
6	1	Iced coffee	12	Cold Drinks
7	10	Ice Latte	17	Cold Drinks
8	8	Ice Americano	14	Cold Drinks
9	6	Espresso	11	Hot Drinks
10	4	Drip coffee	12	Hot Drinks
11	11	Cookie	8	Sweets
12	9	Cold brew	20	Cold Drinks
13	16	cold brew	16	Cold Drinks
14	14	chocolate baka	12	Sweets
15	15	chocolate baka	12	Sweets
16	7	Cartado	14	Hot Drinks
17	18	brownie	12	Sweets
18	5	Americano	12	Hot Drinks

- It also adds it to the cashier's view

The screenshot shows a web application for a bakery. The interface is divided into several sections:

- Top Navigation Bar:** Contains three buttons: "My Information", "View invoices", and "Exit".
- Main Section:**
 - Category:** A section with three buttons: "Hot Drinks", "Cold Drinks", and "Sweet". A red arrow points to the "Sweet" button.
 - Items:** Below the "Sweet" button, there are six items displayed in a row:

Item	Price	Action
Cookie	8	ADD
lemon Cake	8	ADD
Marble Cake	8	ADD
chocolate baka	12	ADD
chocolate baka	12	ADD
brownie	12	ADD
- Left Sidebar:**
 - A table with columns: ID, Quantity, Item, Price.
 - Below the table, there are two sections: "Total Price :" and "Total Price With Tax. :".
 - At the bottom, there are two buttons: "Place Invoice" and "Trash".

- **completion process for each invoice**

once he done preparing the food for the current invoice items he can just press the (Complete) button then the next uncompleted invoice will appear immediately, also the (Invoice Waitlist) table will be updated automatically

The screenshot shows a window titled 'View invoices' with a menu bar containing 'My Information', 'View invoices', 'Create new Item', and 'Exit'. The main area displays 'Invoice #9' on the left and a table of items on the right. The table has columns: Item_ID, Item_Name, Category, and Quantity. The first row shows Item_ID 7, Item_Name 'Cartado', Category 'Hot Drinks', and Quantity 1. Below the table is a 'Complete' button, which is circled in red. A red arrow points from the 'Complete' button to the 'Invoice Waitlist' table below. The 'Invoice Waitlist' table has columns: Invoice_ID and Completion. It shows two rows: Invoice_ID 9 with Completion 'false', and Invoice_ID 10 with Completion 'false'.

Item_ID	Item_Name	Category	Quantity
7	Cartado	Hot Drinks	1

Complete

Invoice_ID	Completion
9	false
10	false

The screenshot shows the same 'View invoices' window, but now displaying 'Invoice #10'. The table of items shows Item_ID 12, Item_Name 'Cartado', Category 'Sweets', and Quantity 1. The 'Complete' button is still circled in red. A red arrow points from the 'Complete' button to the 'Invoice Waitlist' table below. The 'Invoice Waitlist' table now shows only one row: Invoice_ID 10 with Completion 'false'.

Item_ID	Item_Name	Category	Quantity
12	Cartado	Sweets	1

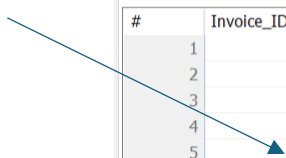
Complete

Invoice_ID	Completion
10	false

-The complete state for the invoice with ID#9 before it's updated to be true

#	Invoice_ID 🔑	Inv_Time	Inv_Date	Total_Price	Completion	Order_Type	Payment_Type	C_ID 🔑	FP_ID 🔑
1	1	12:30:00	2024-10-10	100	1	Dine-In	Card	1	1
2	2	13:45:00	2024-10-10	200	1	Takeout	Cash	2	(NULL)
3	3	14:15:00	2024-10-11	150	1	Takeout	Card	1	3
4	4	02:31:18	2024-11-15	49	1	Dine-In	Cash	1	1
5	5	02:34:15	2024-11-15	38	0	Takeout	Card	1	(NULL)
6	6	02:35:54	2024-11-15	38	0	Dine-In	Cash	1	(NULL)
7	7	02:39:14	2024-11-15	42	0	Dine-In	Cash	1	(NULL)
8	8	02:39:56	2024-11-15	42	0	Dine-In	Cash	1	(NULL)

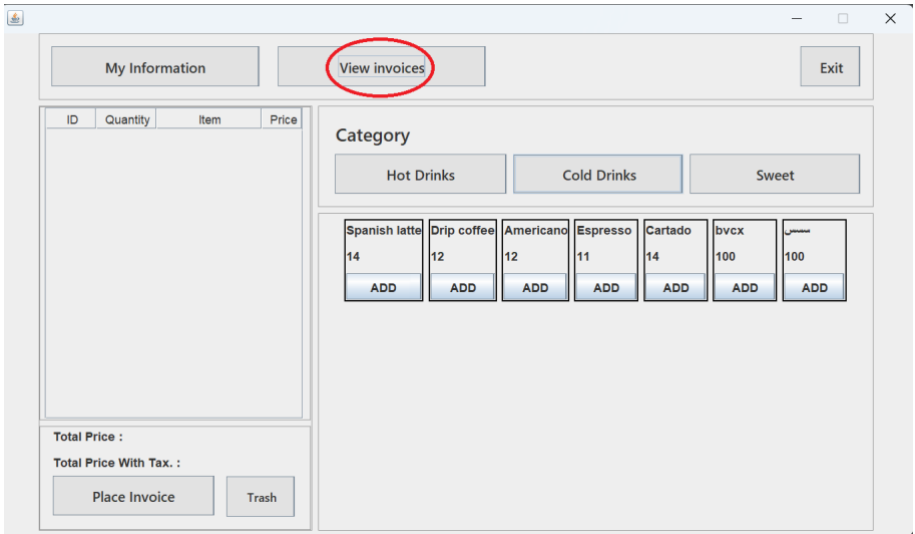
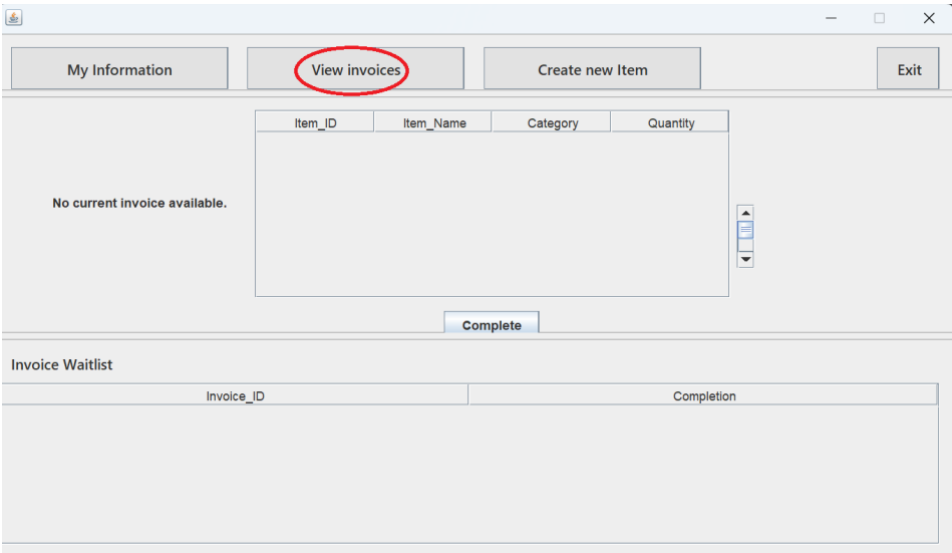
-After updating it to be true



#	Invoice_ID 🔑	Inv_Time	Inv_Date	Total_Price	Completion	Order_Type	Payment_Type	C_ID 🔑	FP_ID 🔑
1	1	12:30:00	2024-10-10	100	1	Dine-In	Card	1	1
2	2	13:45:00	2024-10-10	200	1	Takeout	Cash	2	(NULL)
3	3	14:15:00	2024-10-11	150	1	Takeout	Card	1	3
4	4	02:31:18	2024-11-15	49	1	Dine-In	Cash	1	1
5	5	02:34:15	2024-11-15	38	1	Takeout	Card	1	1
6	6	02:35:54	2024-11-15	38	0	Dine-In	Cash	1	(NULL)
7	7	02:39:14	2024-11-15	42	0	Dine-In	Cash	1	(NULL)

View Invoices:

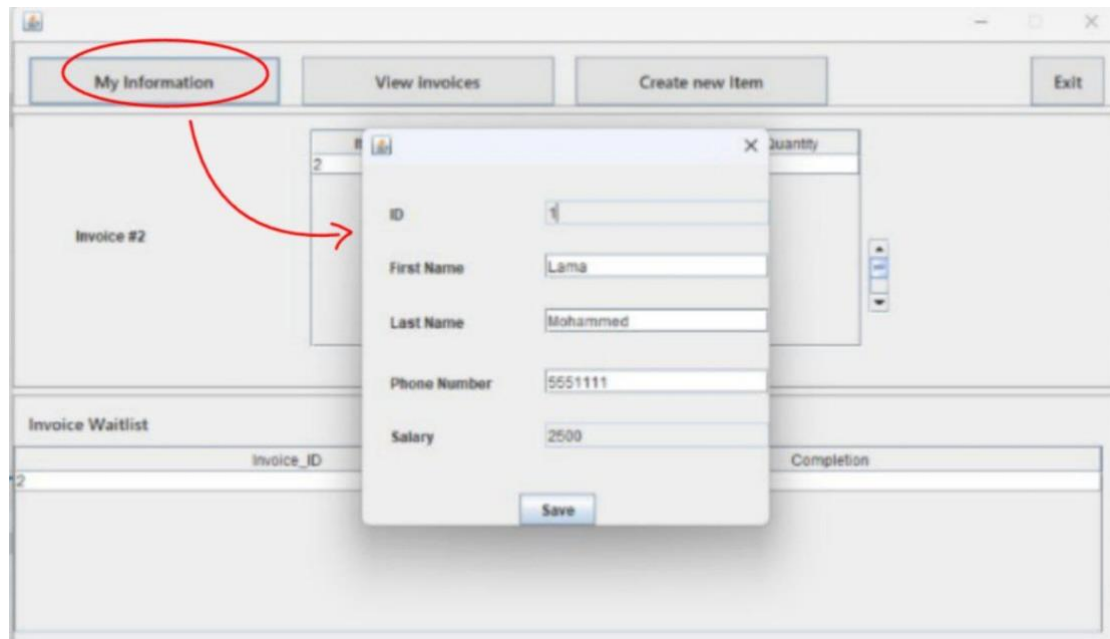
The "View Invoices" button in the FoodPreparer and Cashier view opens a window displaying a list of invoices stored in the database. Data is loaded using JDBC by executing an SQL query to fetch all invoices and displaying them in a table with details (ID, date, price, order type, and completion status). The window also provides a filter option to view only completed invoices through a checkbox and an apply button, making it easier for users to manage and review invoice details efficiently.



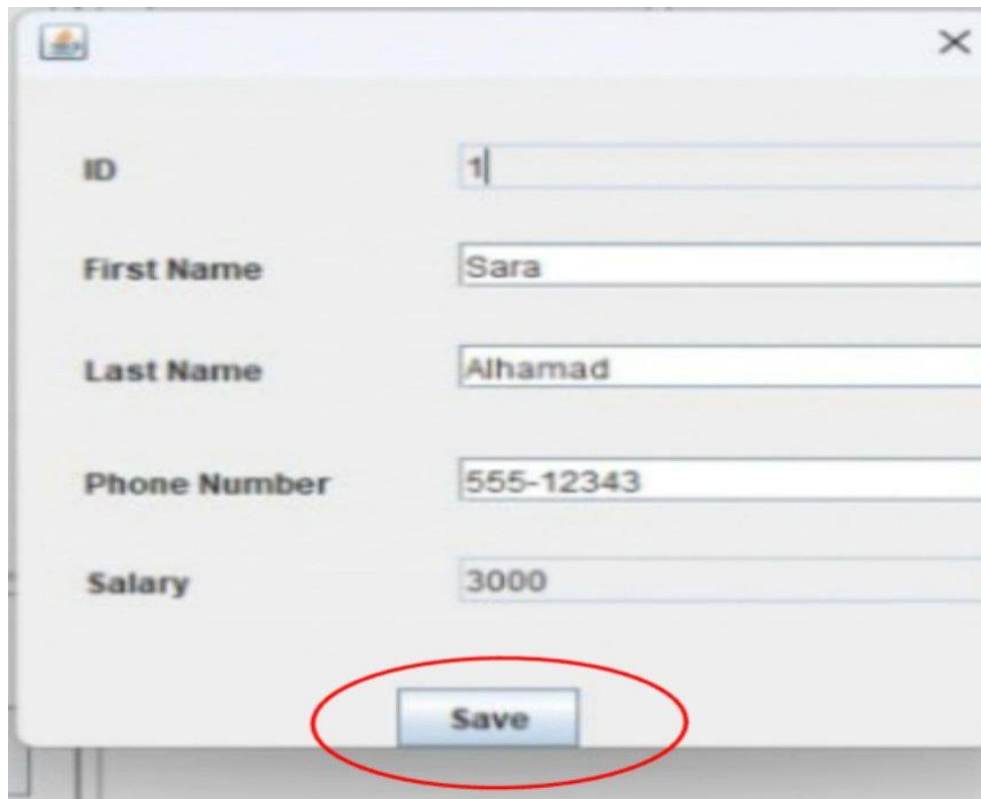
Invoices				
Filter: <input checked="" type="checkbox"/> Only completed invoices <input type="button" value="Apply"/>				
Invoice_ID	Inv_Date	Total_Price	Order_Type	Completion
1	2024-10-10	100	Dine-In	true
2	2024-10-10	200	Takeout	true
3	2024-10-11	150	Takeout	true

Update My Information:

The "my information" button in the food preparer interface opens a window displaying the user's information (ID, name, phone number, salary) and he can modify his name and phone number.



The “save” button updates the new information. When you click on it, the change will be saved and the length of the phone number will be verified to be 9 digits:



A screenshot of a user interface form. The form has five input fields: ID (containing '1'), First Name (containing 'Sara'), Last Name (containing 'Alhamad'), Phone Number (containing '555-12343'), and Salary (containing '3000'). Below the fields is a 'Save' button, which is circled in red. The form is enclosed in a window with a close button (X) in the top right corner.

If the phone number is not 9 digits, a message will appear:

