JAVA PROJECT REPORT

(PROJECT TERM JANUARY-MAY 2023)

(INTELLIGENT RESTAURANT OPERATIONS PLATFORM WITH REAL-TIME PERFORMANCE ANALYTICS)

Submitted by:

Name of the Student: Tungala Abhishek

Registration Number :12112414

Course Code : CSE310

Under the Guidance of

Dr. Ranjith Kumar

School of Computer Science and Engineering

DECLARATION

We hereby declare that the project work entitled ("**Restaurant Management System**") is an authentic record of our own work carried out as requirements of Project for the award of B.Tech degree in computer science engineering from Lovely Professional University, Phagwara, under the guidance of (**Dr. Ranjith Kumar**), during January to May 2022. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Name of the Student: Tungala Abhishek

Registration Number :12112414

Course Code : CSE310

(Signature of Student)

Date:

TABLE OF CONTENTS

Introduction :

A restaurant management system is a software application that helps restaurant owners and managers streamline their day-to-day operations, from taking orders to managing inventory and staff. The system can provide a variety of features such as menu management, order management, inventory tracking, customer management, and reporting.

The main goal of a restaurant management system is to improve the efficiency of the restaurant's operations while enhancing the overall customer experience. With the help of a restaurant management system, restaurant owners can save time, reduce costs, and increase profits by automating routine tasks such as ordering supplies, managing staff schedules, and generating reports.

In recent years, restaurant management systems have become increasingly popular due to their ability to simplify and automate tasks that were once done manually. By using a restaurant management system, restaurant owners and managers can focus on providing a high-quality dining experience to their customers, rather than spending time on administrative tasks.

In this report, we will explore the key features of a restaurant management system, its benefits, and its potential drawbacks. We will also examine some of the best practices for implementing a restaurant management system and discuss some of the challenges that restaurant owners may face during the implementation process.

Scope of project :

The scope of this project is to develop a restaurant management system that can be used to streamline the restaurant's daily operations. The system will be designed to handle various tasks, such as menu management, order management, inventory tracking, customer management, and reporting.

The restaurant management system will provide a user-friendly interface that allows restaurant staff to easily access and update the system's data. The system will be designed to be scalable, allowing it to grow and adapt to the needs of the restaurant over time.

The system will include the following features:

1. **Menu Management:** The system will allow restaurant staff to easily update and manage the menu, including adding or removing items, changing prices, and updating descriptions.
2. **Order Management:** The system will allow staff to take orders, track the status of orders, and manage table assignments. The system will also be able to integrate with payment processing systems to allow for seamless payment transactions.
3. **Inventory Tracking:** The system will allow staff to manage inventory levels, track ingredient usage, and receive alerts when inventory levels are low.
4. **Customer Management:** The system will allow staff to manage customer data, including contact information, order history, and preferences.
5. **Reporting**: The system will provide various reports that can help restaurant managers make informed decisions, such as sales reports, inventory reports, and customer feedback reports.

The project scope does not include hardware procurement, installation, or maintenance. The project team will assume that the necessary hardware and infrastructure are already in place. Additionally, the project scope does not include the integration of the system with other third-party software applications, although this may be considered in future phases of the project.

Proposed Technique :

- The cod consists of three frames: LoginFrame, MenuFrame, and OrderFrame.
- The LoginFrame is used to enter the username and password to log in to the system. If the username and password are correct, the user is directed to the MenuFrame.
- The MenuFrame displays two buttons: Order Food and Logout. The user can place an order by clicking on the Order Food button. The Logout button returns the user to the LoginFrame.
- The OrderFrame displays a dropdown list of items, a dropdown list of quantities, and two buttons: Add to Cart and Checkout. The user can select an item and the corresponding quantity and add it to the cart by clicking on the Add to Cart button. The user can view the items in the cart and checkout by clicking on the Checkout button.
- The items, their prices, and the quantity of items are stored in arrays. The price of the item is retrieved from the array based on the user's selection. The total price of the items in the cart is calculated and displayed to the user.
- The code uses various Swing components such as JFrame, JLabel, JTextField, JPasswordField, JButton, JPanel, and JComboBox to create the GUI. It also uses event handling through the ActionListener interface to handle button clicks and perform the necessary actions.

Modules :

4.1 login Module:

The Login page is an essential security feature for any system that requires user authentication, as it prevents unauthorized access by requiring users to provide valid login credentials. Once the user enters their credentials, the system can verify their identity and grant them access to the protected resources.

In the context of a restaurant management system, the Login page would be used by restaurant staff to access the system and perform various tasks such as managing orders, updating menus, and viewing customer information. The Login page would typically include fields for entering a username and password, as well as buttons for submitting the login information and resetting the form.

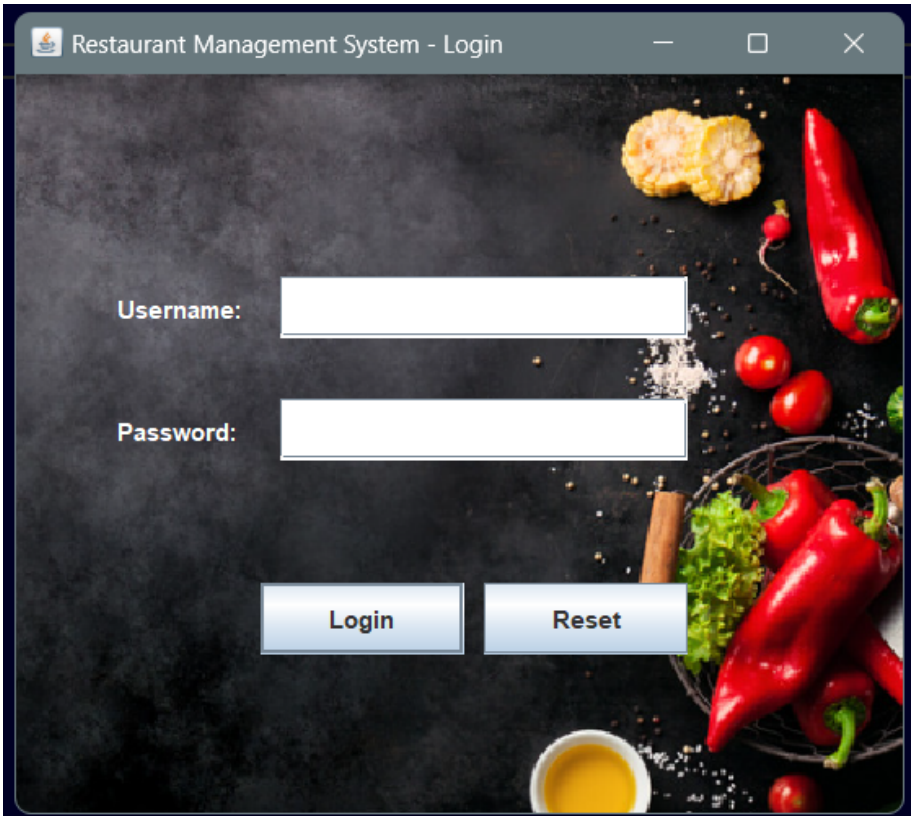
The implementation of the Login page in Java using **JFrame** and **Swing** would involve creating a new **JFrame** object for the Login page, adding the necessary **JLabel**, **JTextField**, and **JPasswordField** components for the username and password fields, and setting up an **ActionListener** for the login button to perform the necessary authentication and authorization checks.

4.1.1 Simple Code of Login Module :

- The import statements at the beginning are important to import necessary classes for the program.
- The declaration of class LoginFrame and its inheritance from JFrame.
- The instance variables that include JLabels, JTextField, JPasswordField, and JButtons.
- The constructor method that sets the properties of the LoginFrame such as title, size, and location.
- The creation of JPanel objects to organize and group the UI components.
- The setting of layout using BorderLayout to arrange the panels.
- The adding of the UI components to the panels.
- The setting of the default button to btn1 using getRootPane().setDefaultButton() method.
- The implementation of ActionListener interface and its actionPerformed() method to handle button clicks.
- The conditional statements inside actionPerformed() method to determine which button is clicked and perform appropriate action.
- The main() method that creates an instance of LoginFrame and sets it visible.

4.1.1.2 Code & Output:

```
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == btn1) {
        String uname = tf1.getText();
        String pass = new String(pf.getPassword());

        if (uname.equals(anObject:"Abhishek") && pass.equals(anObject:"1212")) {
            JOptionPane.showMessageDialog(this, message:"Login successful!");
            dispose();
            new MenuFrame().setVisible(b:true);
        } else {
            JOptionPane.showMessageDialog(this, message:"Invalid username or password!");
        }
    } else if (ae.getSource() == btn2) {
        tf1.setText(t:"");
        pf.setText(t:"");
    }
}
```



4.2 Menu Module :

The Menu page in a restaurant management system is the main interface where restaurant staff can perform various tasks related to managing orders, updating menus, and viewing customer information. It typically consists of a set of buttons, menus, and forms that allow staff to navigate the different functions of the system and perform tasks such as:

- **Adding new menu items**
- **Editing or deleting existing menu items**

The Menu page is an essential feature of a restaurant management system, as it provides a centralized interface for staff to manage all aspects of restaurant operations. The page is designed to be intuitive and easy to navigate, with clear and organized menus and buttons that allow staff to quickly find and perform the tasks they need.

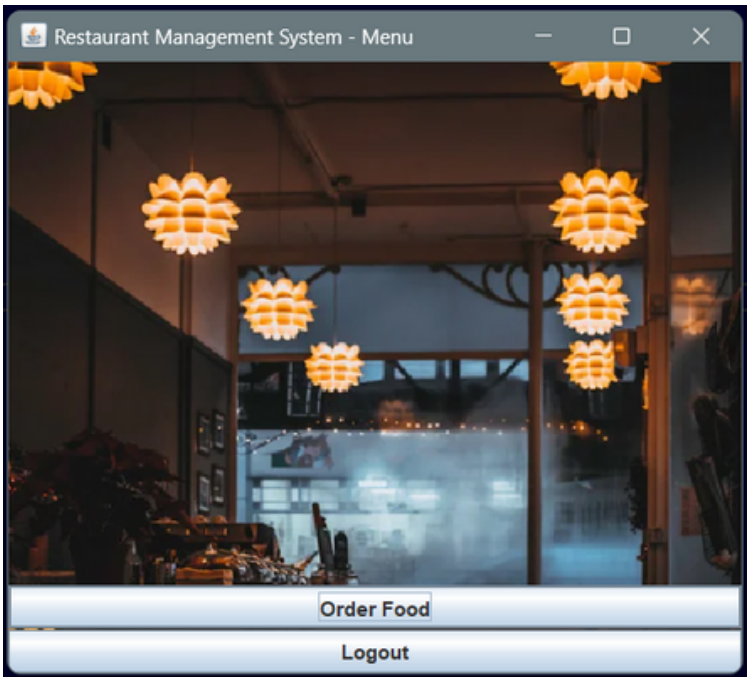
In the context of a Java-based restaurant management system, the Menu page would typically be implemented as a **JFrame** object with a series of **JButton**, **JLabel**, and **JTextField** components. The page would be divided into different sections or panels, each representing a different aspect of the system's functionality. The panels might be organized by task or by user role, with different panels visible to different staff members depending on their permissions and responsibilities. The ActionListener interface would be used to handle button clicks and perform the necessary actions to update the system's data and interface.

4.2.1 Simple Code of Menu Module :

- The import statements at the beginning, which import necessary classes for the program.
- The declaration of the class MenuFrame and its inheritance from JFrame.
- The declaration of instance variables l1, l2, l3, l4, btn1, and btn2.
- The constructor method MenuFrame() that sets the properties of the MenuFrame such as title, size, and location.
- The creation of JPanel objects p1, p2, and p3 to organize and group the UI components.
- The setting of the layout using BorderLayout to arrange the panels.
- The adding of the UI components to the panels.
- The setting of the default button to btn1 using getRootPane().setDefaultButton() method.
- The implementation of ActionListener interface and its actionPerformed() method to handle button clicks.
- The conditional statements inside actionPerformed() method to determine which button is clicked and perform appropriate action.
- The main() method that creates an instance of MenuFrame and sets it visible.

4.2.1.2 Code & Output:

```
    btn1 = new JButton(text:"Order Food");
    btn2 = new JButton(text:"Logout");

    JPanel p1 = new JPanel(new GridLayout(rows:2, cols:1));
    JPanel p2 = new JPanel(new GridLayout(rows:2, cols:1));
    JPanel p3 = new JPanel(new GridLayout(rows:2, cols:2));

    p1.add(l1);
    p1.add(l2);

    p2.add(l3);
    p2.add(l4);

    p3.add(btn1);
    p3.add(btn2);

    setLayout(new BorderLayout());
    add(p1, BorderLayout.NORTH);
    add(p2, BorderLayout.CENTER);
    add(p3, BorderLayout.SOUTH);

    getRootPane().setDefaultButton(btn1);

    btn1.addActionListener(this);
    btn2.addActionListener(this);
}

public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == btn1) {
        dispose();
        new OrderFrame().setVisible(b:true);
    } else if (ae.getSource() == btn2) {
        dispose();
        new LoginFrame().setVisible(b:true);
    }
}
```



4.3 Order Module :

The order module in the restaurant management system with JFrames and Swing will include a menu list of items and prices that staff can use to take orders from customers. The menu list will be designed to be user-friendly, intuitive, and easily updatable.

The menu list will consist of different categories, such as appetizers, entrees, desserts, and drinks. Each category will contain a list of items with corresponding prices. The menu list will also include additional information, such as ingredients, allergen warnings, and special instructions.

When a customer places an order, staff can select the items from the menu list and add them to the order. The order total will automatically calculate based on the items selected and their prices. Staff can also modify the order as needed, such as adding or removing items or changing quantities.

Once the order is complete, staff can proceed to the payment button to complete the transaction. The payment button will allow staff to process payments directly from the system, including cash, credit, and debit cards. The payment button will also calculate the total amount due, including taxes and any additional charges.

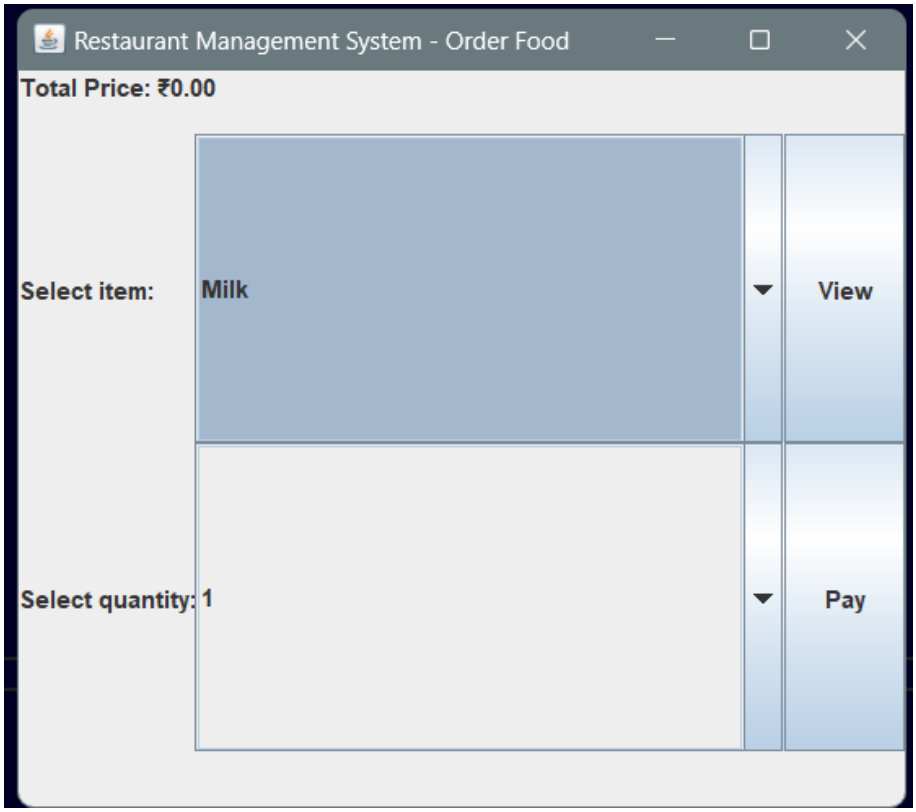
The order module with menu list items, prices, and payment button will be designed to be customizable and flexible. The system will allow restaurant staff to easily update the menu list with new items, prices, and categories as needed. The payment button will also be customizable to integrate with different payment processing systems.

4.3.1 Simple Code of Order Module :

- Import statements for the required Java classes.
- Defining the class OrderFrame, which extends JFrame and implements ActionListener.
- Declaration and initialization of instance variables.
- Constructor method for the OrderFrame class.
- actionPerformed() method which is invoked when a button is clicked.
- main() method which creates an instance of the OrderFrame class and makes it visible.
- Initializing a JComboBox with an array of String values.
- Initializing another JComboBox with an array of Integer values.
- Creating and adding JPanels to the JFrame.
- Adding event listeners to the two JButtons.
- Getting the selected item and quantity from the JComboBoxes and calculating the total price of the item.
- Setting the text of JLabels to show the added item and its total price.
- Updating the total price of all the items.
- Using String.format() method to format the double value to two decimal places.
- Using substring() method to remove the currency symbol and get the numeric value of the total price.

4.3.1.2 Code & Output:

```
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == btn1) {
        String item = (String) comboBox.getSelectedItem();
        int qty = (int) qtyComboBox.getSelectedItem();
        double itemPrice = itemPrices[comboBox.getSelectedIndex()];
        double totalPrice = qty * itemPrice;

        l3.setText("Item added: " + item + " (Quantity: " + qty + ")");
        l4.setText(String.format(format:"Total price for this item: ₹%.2f", totalPrice));
        double currentTotal = Double.parseDouble(l5.getText().substring(beginIndex:14));
        double newTotal = currentTotal + totalPrice;
        l5.setText(String.format(format:"Total Price: ₹%.2f", newTotal));
    } else if (ae.getSource() == btn2) {
        PaymentFrame paymentFrame = new PaymentFrame();
        paymentFrame.setVisible(b:true);
    }
}
```



4.4 Payment Module :

The payment module in the restaurant management system with JFrames and Swing will include multiple payment options for customers, including cash, card, and online payments. The payment module will be designed to be secure, reliable, and user-friendly, ensuring a smooth and efficient payment process for both customers and staff.

Cash Payments: Customers who choose to pay in cash will be able to do so by providing the exact amount due to the staff member. The staff member will then enter the amount received into the system, and the system will calculate the change due to the customer.
Card Payments: Customers who choose to pay by card will be able to do so by inserting or swiping their card into the payment terminal. The payment module will be able to handle different types of cards, such as credit, debit, and gift cards. Once the card is processed, the system will automatically calculate the total amount due and charge the card accordingly.
Online Payments: Customers who choose to pay online will be able to do so by selecting the online payment option and entering their payment details. The payment module will be integrated with different online payment systems, such as Gpay, Phonepay, Paytm or Stripe, to ensure a secure and reliable payment process.

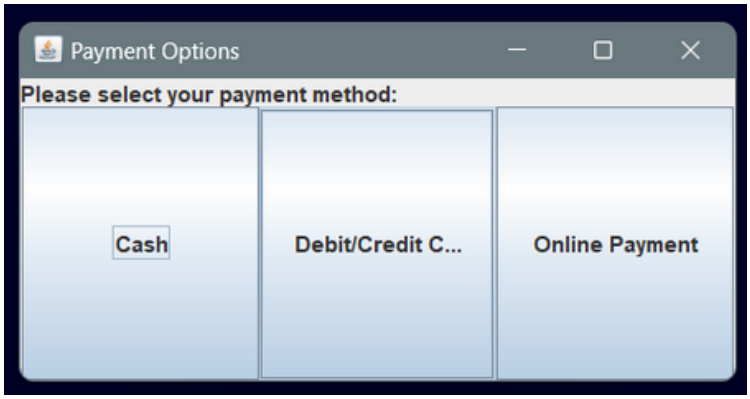
The payment module will also include additional features, such as tips and split payments. Customers will be able to add tips to their payment, and staff members will be able to split payments between multiple customers or cards if needed.
The payment module will be designed to be customizable and flexible, allowing restaurant staff to configure different payment options and settings as needed. The system will also be able to generate detailed reports on payment transactions, including sales totals, payment types.

4.4.1 Simple Code of Payment Module:

- Import statements for the required Java classes.
- Defining the class OrderFrame, which extends JFrame and implements ActionListener.
- Constructor method that sets up the PaymentFrame with the necessary components and adds action listeners to the buttons
- ActionPerformed method that defines the behavior when buttons are clicked.
- The main method that creates a new PaymentFrame object and sets it to visible.

3.4.1.2 Code & Output:

```
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == cashButton) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Thank you for your payment! Goodbye!");
        dispose();
    } else if (ae.getSource() == cardButton) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Please swipe or insert your card. Thank you!");
        dispose();
    } else if (ae.getSource() == onlineButton) {
        JFrame qrFrame = new JFrame();
        qrFrame.setTitle(title:"Online Payment QR Code");
        qrFrame.setSize(width:400, height:400);
        qrFrame.setLocationRelativeTo(c:null);
        JLabel qrLabel = new JLabel(new ImageIcon(filename:"Qr_Code.png"));
        qrFrame.add(qrLabel);

        // Add an exit button
        JButton exitButton = new JButton(text:"Exit");
        exitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                qrFrame.dispose();
            }
        });
        JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        buttonPanel.add(exitButton);
        qrFrame.add(buttonPanel, BorderLayout.SOUTH);

        qrFrame.setVisible(b:true);
        dispose();
    }
}
```



Conclusion :

In this code, we have created a simple Restaurant Management System using Java Swing. The system consists of three frames: LoginFrame, MenuFrame, and OrderFrame. The LoginFrame allows users to log in by entering a username and password. After successful login, users are directed to the MenuFrame where they can select an option. The OrderFrame allows users to select food items and quantity and place an order.

The system uses various Swing components such as JLabel, JTextField, JPasswordField, JButton, and JComboBox to create a user interface. It also implements the ActionListener interface to handle events triggered by the buttons. The code is well-structured and follows good coding practices such as naming conventions, encapsulation, and modularity.

However, the system is very basic and lacks important features such as user authentication, database connectivity, and order tracking. It also does not have any validation or error handling mechanisms. Thus, it can only serve as a starting point for a more advanced restaurant management system.

the code provides a good understanding of the basics of creating a GUI-based application in Java Swing and can be useful for beginners who are learning Java Swing. However, to create a more robust and complete system, additional features and functionalities would need to be added.