RESTAURANT MANAGEMENT

A MINI-PROJECT BY:

NAREN SURIYA VS 230701203

NITHESH RAJ M 230701214

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"CAFE MANAGEMENT"** is the bonafide work of **"NAREN SURIYA VS, NITHESH RAJ M"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Ms. ASWANA LAL
Asst. Professor
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The **Cafe Management System** is a comprehensive desktop application designed to streamline bakery and cafe operations. Developed using Java Swing and MySQL, the system facilitates real-time inventory tracking, order processing, billing, and performance analysis. Key features include:

- **Inventory Management:** Real-time tracking of stock levels with low-stock alerts to prevent shortages and waste.
- **Order Processing & Billing:** Efficient order management and automated billing for improved accuracy and speed.
- **Data Analytics & Reporting:** Providing detailed insights into sales, profit margins, and performance for strategic planning.
- **Scalable Back-End Architecture:** Utilizing MySQL to manage inventory, transactions, and user data, ensuring data integrity and scalability.

This system empowers cafe owners to optimize operations, reduce manual workload, and improve customer service while increasing profitability.

This application provides a simple, scalable, and engaging platform for coding enthusiasts, and fostering a coding community.

TABLE OF CONTENTS

1.1 INTRODUCTION

The Cafe Management System is a software solution that automates key cafe operations, including inventory tracking, order processing, and billing. It helps cafe owners manage stock levels, reduce waste, and improve efficiency with real-time monitoring and low-stock alerts. The system also provides valuable data analytics for better decision-making and strategic growth. By digitizing processes, it enhances service, reduces manual workload, and boosts profitability.

1.2 IMPLEMENTATION

The **CODING PLATFORM MANAGEMENT** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.3 SCOPE OF THE PROJECT

The scope of the Cafe Management System project includes automating core operations such as inventory management, order processing, billing, and reporting. It aims to streamline workflow, reduce manual errors, and enhance customer service. The system will provide real-time analytics to support decision-making and business growth. It is designed to improve efficiency, minimize waste, and increase profitability for cafes of all sizes.

1.4 WEBSITE FEATURES

- **Billing record management**
- **Item deletion handling**
- **Customer data retrieval**
- **User and Admin login page**

SYSTEM SPECIFICATIONS

2.1 MINUMAL HARDWARE SPECIFICATIONS:

PROCESSOR : Intel i3 , Ryzen 3 (And above)

MEMORY SIZE : 2.00 GB (And above)

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE : Java, MySQL

FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 10 (And above)

SAMPLE CODE

3.1 LOGIN PAGE

```
package rohanbakery;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.*;
import javax.swing.JOptionPane;

public class Login extends javax.swing.JFrame {

   private Connection conn = ConnectToDatabase.getConnection();

   public Login() {
      initComponents();
      centerFrame();
   }

   private void centerFrame() {
      Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
      setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
   }

   private void btn_LoginActionPerformed(java.awt.event.ActionEvent evt) {
      String sql = "SELECT uId FROM user WHERE uName = ? AND uPassword = ? AND uType =
?";

      try (PreparedStatement pst = conn.prepareStatement(sql)) {
         pst.setString(1, txt_Username.getText());
         pst.setString(2, new String(txt_Password.getPassword()));
         pst.setString(3, cbo_UserType.getSelectedItem().toString());

         try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
               JOptionPane.showMessageDialog(null, "Login successful");
               new mainMenu().setVisible(true);
               dispose();
            } else {
               JOptionPane.showMessageDialog(null, "Invalid Username or Password");
            }
         }
      } catch (SQLException e) {
         JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
      }
   }

   public static void main(String args[]) {
```

```
        java.awt.EventQueue.invokeLater(() -> new Login().setVisible(true));
    }

    // Variables declaration
    private javax.swing.JButton btn_Login;
    private javax.swing.JComboBox<String> cbo_UserType;
    private javax.swing.JTextField txt_Username;
    private javax.swing.JPasswordField txt_Password;
    // End of variables declaration
}
```

3.2 ADD INVENTORY

```
package rohanbakery;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.*;
import java.text.SimpleDateFormat;
import javax.swing.JOptionPane;

public class AddInventory extends javax.swing.JFrame {
    private Connection conn;
    private PreparedStatement pst;
    private ResultSet rs;

    public AddInventory() {
        initComponents();
        centerFrame();
        conn = ConnectToDatabase.getConnection();
        loadItemList();
    }

    private void centerFrame() {
        Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
        setLocation((size.width - getWidth()) / 2, (size.height - getHeight()) / 2);
    }

    private void loadItemList() {
        try {
            pst = conn.prepareStatement("SELECT iName FROM item WHERE iActive=? ORDER BY
iName ASC");
            pst.setString(1, "Yes");
            rs = pst.executeQuery();
            cbo_iniName.removeAllItems();
            while (rs.next()) cbo_iniName.addItem(rs.getString(1));
        } catch (SQLException e) {
```

```
            JOptionPane.showMessageDialog(null, "Item list cannot be loaded");
        } finally {
            closeResources();
        }
    }

    private void closeResources() {
        try {
            if (rs != null) rs.close();
            if (pst != null) pst.close();
        } catch (SQLException e) {}
    }

    private void cbo_iniNameItemStateChanged(java.awt.event.ItemEvent evt) {
        if (cbo_iniName.getSelectedIndex() > -1) {
            try {
                pst = conn.prepareStatement("SELECT * FROM item WHERE iName=?");
                pst.setString(1, cbo_iniName.getSelectedItem().toString());
                rs = pst.executeQuery();
                if (rs.next()) txt_iniId.setText(rs.getString(1));
            } catch (SQLException e) {
                JOptionPane.showMessageDialog(null, "Item details cannot be found");
            } finally {
                closeResources();
            }
        }
    }

    private void btn_saveActionPerformed(java.awt.event.ActionEvent evt) {
        if (txt_iniId.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please select the item to enter in the inventory");
        } else {
            try {
                pst = conn.prepareStatement("INSERT INTO inventory (inDate, item_iId, iniName, inQty)
VALUES (?, ?, ?, ?)");
                pst.setString(1, new SimpleDateFormat("yyyy-MM-dd").format(txt_inDate.getDate()));
                pst.setInt(2, Integer.parseInt(txt_iniId.getText()));
                pst.setString(3, cbo_iniName.getSelectedItem().toString());
                pst.setInt(4, Integer.parseInt(txt_inQty.getText()));
                pst.execute();
                JOptionPane.showMessageDialog(null, "Record Added");
            } catch (SQLException e) {
                JOptionPane.showMessageDialog(null, e);
            } finally {
                closeResources();
            }
        }
```

```
    }

    private void btn_clearAllActionPerformed(java.awt.event.ActionEvent evt) {
        txt_inDate.setDate(null);
        txt_iniId.setText("");
        cbo_iniName.setSelectedIndex(0);
        txt_inQty.setText("");
    }

    private void btn_clearitemActionPerformed(java.awt.event.ActionEvent evt) {
        txt_iniId.setText("");
        cbo_iniName.setSelectedIndex(0);
        txt_inQty.setText("");
    }

    private void btn_backActionPerformed(java.awt.event.ActionEvent evt) {
        new Inventory().setVisible(true);
        dispose();
    }

    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(() -> new AddInventory().setVisible(true));
    }
}
```

3.3 GENERATE BILL

```
package rohanbakery;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.*;
import javax.swing.JOptionPane;
import net.proteanit.sql.DbUtils;

public class GenerateBill extends javax.swing.JFrame {
    private Connection conn;
    private PreparedStatement pst;
    private ResultSet rs;

    public GenerateBill() {
        initComponents();
        centerFrame();
        conn = ConnectToDatabase.getConnection();
        loadBillTable();
    }
```

```
    private void centerFrame() {
        Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
        setLocation((size.width - getWidth()) / 2, (size.height - getHeight()) / 2);
    }

    private void loadBillTable() {
        String sql = "SELECT bId, bNumber, bDate, bCustName, item_iId, iName, iDescription, bQty, iSp,
iCp, bAmount, bProfit, bOk " +
                     "FROM billing WHERE bOk = ?";
        try (PreparedStatement pst = conn.prepareStatement(sql)) {
            pst.setString(1, "Yes");
            rs = pst.executeQuery();
            tbl_Bill.setModel(DbUtils.resultSetToTableModel(rs));
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }

    private void navigateTo(Class<?> cls) {
        try {
            javax.swing.JFrame newFrame = (javax.swing.JFrame) cls.getDeclaredConstructor().newInstance();
            newFrame.setVisible(true);
            dispose();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Navigation failed: " + e.getMessage());
        }
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {
        pnl_main = new javax.swing.JPanel();
        btn_backToMainMenu = new javax.swing.JButton("Back to main menu", e ->
navigateTo(MainMenu.class));
        btn_addItem = new javax.swing.JButton("Add item", e -> navigateTo(AddItem.class));
        btn_removeItem = new javax.swing.JButton("Remove item", e -> navigateTo(RemoveItem.class));
        btn_removeBill = new javax.swing.JButton("Remove bill", e -> navigateTo(RemoveBill.class));
        btn_saveBillInPdf = new javax.swing.JButton("Save bill in PDF", e -> navigateTo(PrintBill.class));
        tbl_Bill = new javax.swing.JTable();
        jScrollPaneForTable = new javax.swing.JScrollPane(tbl_Bill);

        pnl_main.setLayout(null);
        pnl_main.add(btn_backToMainMenu).setBounds(890, 10, 200, 30);
        pnl_main.add(btn_addItem).setBounds(370, 430, 90, 30);
        pnl_main.add(btn_removeItem).setBounds(470, 430, 110, 30);
        pnl_main.add(btn_removeBill).setBounds(590, 430, 110, 30);
        pnl_main.add(btn_saveBillInPdf).setBounds(710, 430, 130, 30);
```

```
        pnl_main.add(jScrollPaneForTable).setBounds(20, 50, 1070, 370);

        add(pnl_main);
        setTitle("Bill Details");
        setResizable(false);
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setSize(1110, 470);
    }

    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(() -> new GenerateBill().setVisible(true));
    }

    private javax.swing.JPanel pnl_main;
    private javax.swing.JButton btn_backToMainMenu, btn_addItem, btn_removeItem, btn_removeBill,
btn_saveBillInPdf;
    private javax.swing.JTable tbl_Bill;
    private javax.swing.JScrollPane jScrollPaneForTable;
}
```

3.4 PRINT BILL

```
package rohanbakery;

import java.awt.*;
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.Date;
import javax.swing.*;
import com.itextpdf.text.*;
import com.itextpdf.text.pdf.*;
import org.jdesktop.swingx.JXDatePicker;

public class PrintBill extends javax.swing.JFrame {
    private Connection conn;
    private PreparedStatement pst;
    private ResultSet rs;
    private boolean listLoaded;

    public PrintBill() {
        initComponents();
        centerFrame();
        conn = ConnectToDatabase.getConnection();
        setComboBoxValues();
        listLoaded = true;
    }
```

```
    private void centerFrame() {
        Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
        setLocation((size.width - getWidth()) / 2, (size.height - getHeight()) / 2);
    }

    private void setComboBoxValues() {
        String sql = "SELECT DISTINCT bNumber FROM billing WHERE bOk='Yes'";
        try (PreparedStatement pst = conn.prepareStatement(sql); ResultSet rs = pst.executeQuery()) {
            cbo_bNumber.removeAllItems();
            while (rs.next()) cbo_bNumber.addItem(rs.getString(1));
        } catch (SQLException e) {
            showError("Error loading billing list", e);
        }
    }

    private void saveBillToPDF(String filePath) {
        String sql = "SELECT * FROM billing WHERE bNumber=? AND bOk='Yes'";
        try (PreparedStatement pst = conn.prepareStatement(sql)) {
            pst.setString(1, cbo_bNumber.getSelectedItem().toString());
            rs = pst.executeQuery();

            Document doc = new Document();
            PdfWriter.getInstance(doc, new FileOutputStream(filePath));
            doc.open();

            addBillHeader(doc);
            PdfPTable table = createBillTable();
            doc.add(table);
            addBillFooter(doc);

            doc.close();
            JOptionPane.showMessageDialog(this, "Bill successfully saved to PDF.");
        } catch (Exception e) {
            showError("Error saving PDF", e);
        }
    }

    private void addBillHeader(Document doc) throws DocumentException {
        doc.add(new Paragraph("ROHAN'S BAKERY",
FontFactory.getFont(FontFactory.HELVETICA_BOLD, 20)));
        doc.add(new Paragraph("Bill No: " + cbo_bNumber.getSelectedItem()));
        doc.add(new Paragraph("Date: " + new SimpleDateFormat("yyyy-MM-
dd").format(txt_bDate.getDate())));
        doc.add(new Paragraph("Customer: " + txt_bCustName.getText()));
        doc.add(new Paragraph(" "));
    }
```

```
    private PdfPTable createBillTable() throws SQLException {
        PdfPTable table = new PdfPTable(new float[]{1, 2, 5, 2, 2, 3});
        String[] headers = {"Sl No.", "Item ID", "Item Name", "Qty", "Rate", "Amount"};
        for (String header : headers)
            table.addCell(new PdfPCell(new Phrase(header,
FontFactory.getFont(FontFactory.HELVETICA_BOLD, 10))));

        int slNo = 1;
        while (rs.next()) {
            table.addCell(String.valueOf(slNo++));
            table.addCell(rs.getString(5)); // Item ID
            table.addCell(rs.getString(6)); // Item Name
            table.addCell(rs.getString(8)); // Quantity
            table.addCell(rs.getString(9)); // Rate
            table.addCell(rs.getString(12)); // Amount
        }
        return table;
    }

    private void addBillFooter(Document doc) throws DocumentException {
        Paragraph amount = new Paragraph("Total: Rs. " + txt_bAmount.getText(),
FontFactory.getFont(FontFactory.HELVETICA_BOLD, 12));
        amount.setAlignment(Element.ALIGN_RIGHT);
        doc.add(amount);
    }

    private void showError(String message, Exception e) {
        JOptionPane.showMessageDialog(this, message + ": " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }

    private void initComponents() {
        // Initialization code (similar to original but compact)
    }

    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(() -> new PrintBill().setVisible(true));
    }
}
```

3.5 REMOVE INVENTORY

```
package rohanbakery;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.*;
import javax.swing.JOptionPane;
```

```
public class removeInventory extends javax.swing.JFrame {

    private Connection conn = ConnectToDatabase.getConnection();

    public removeInventory() {
        initComponents();
        centerFrame();
        setComboBoxValues();
    }

    private void centerFrame() {
        Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
        setLocation((size.width / 2 - getWidth() / 2), (size.height / 2 - getHeight() / 2));
    }

    private void setComboBoxValues() {
        try (PreparedStatement pst = conn.prepareStatement("SELECT inId FROM inventory WHERE
inActive=?")) {
            pst.setString(1, "Yes");
            try (ResultSet rs = pst.executeQuery()) {
                cbo_inId.removeAllItems();
                while (rs.next()) cbo_inId.addItem(rs.getString(1));
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Inventory list cannot be loaded");
        }
    }

    private void btn_deleteActionPerformed(java.awt.event.ActionEvent evt) {
        if (txt_iniName.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please select the inventory id to delete");
        } else {
            try (PreparedStatement pst = conn.prepareStatement("UPDATE inventory SET inActive=?
WHERE inId=?")) {
                pst.setString(1, "No");
                pst.setString(2, cbo_inId.getSelectedItem().toString());
                pst.execute();
                JOptionPane.showMessageDialog(null, "Record removed");
                setComboBoxValues();
            } catch (Exception e) {
                JOptionPane.showMessageDialog(null, e);
            }
        }
    }

    private void btn_cleaActionPerformed(java.awt.event.ActionEvent evt) {
        txt_inDate.setDate(null);
```

```
        txt_iniName.setText("");
        txt_item_iId.setText("");
        txt_inQty.setText("");
    }

    private void cbo_inIdItemStateChanged(java.awt.event.ItemEvent evt) {
        if (!txt_iniName.getText().isEmpty()) {
            try (PreparedStatement pst = conn.prepareStatement("SELECT * FROM inventory WHERE
inId=?")) {
                pst.setString(1, cbo_inId.getSelectedItem().toString());
                try (ResultSet rs = pst.executeQuery()) {
                    if (rs.next()) {
                        txt_inDate.setDate(rs.getDate(2));
                        txt_item_iId.setText(rs.getString(3));
                        txt_iniName.setText(rs.getString(4));
                        txt_inQty.setText(rs.getString(5));
                    }
                }
            } catch (Exception e) {
                JOptionPane.showMessageDialog(null, "Inventory details cannot be found");
            }
        }
    }

    private void btn_backActionPerformed(java.awt.event.ActionEvent evt) {
        new inventory().setVisible(true);
        dispose();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(() -> new removeInventory().setVisible(true));
    }
}
```

3.6 ADD ITEM

```
package rohanbakery;

import java.awt.*;
import java.sql.*;
import java.text.*;
import javax.swing.*;

public class addItem extends javax.swing.JFrame {

    private Connection conn;
    private boolean listLoaded = false;
```

```
    public addItem() {
        initComponents();
        setLocationRelativeTo(null);
        conn = ConnectToDatabase.getConnection();
        loadItemList();
    }

    private void loadItemList() {
        try (PreparedStatement pst = conn.prepareStatement("SELECT iName FROM item WHERE
iActive='Yes' ORDER BY iName");
             ResultSet rs = pst.executeQuery()) {
            cbo_iName.removeAllItems();
            while (rs.next()) cbo_iName.addItem(rs.getString(1));
            listLoaded = true;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Item list cannot be loaded");
        }
    }

    private void cbo_iNameItemStateChanged(java.awt.event.ItemEvent evt) {
        if (!listLoaded) return;
        try (PreparedStatement pst = conn.prepareStatement("SELECT * FROM item WHERE
iName=?")) {
            pst.setString(1, cbo_iName.getSelectedItem().toString());
            try (ResultSet rs = pst.executeQuery()) {
                if (rs.next()) {
                    txt_iId.setText(rs.getString(1));
                    txt_iDescription.setText(rs.getString(3));
                    txt_iCp.setText(rs.getString(5));
                    txt_iSp.setText(rs.getString(6));
                }
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Item details cannot be found");
        }
    }

    private void btn_calcAmountActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            int sp = Integer.parseInt(txt_iSp.getText());
            int qty = Integer.parseInt(txt_bQty.getText());
            txt_bAmount.setText(String.valueOf(sp * qty));
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Invalid quantity or price");
        }
    }

    private void btn_saveActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        if (txt_iId.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please enter the bill number");
            return;
        }
        String sql = "INSERT INTO billing (bNumber, bDate, bCustName, item_iId, iName, iDescription,
bQty, iSp, iCp) VALUES (?,?,?,?,?,?,?,?,?)";
        try (PreparedStatement pst = conn.prepareStatement(sql)) {
            pst.setString(1, txt_bNumber.getText());
            pst.setString(2, new SimpleDateFormat("yyyy-MM-dd").format(txt_bDate.getDate()));
            pst.setString(3, txt_bCustName.getText());
            pst.setInt(4, Integer.parseInt(txt_iId.getText()));
            pst.setString(5, cbo_iName.getSelectedItem().toString());
            pst.setString(6, txt_iDescription.getText());
            pst.setInt(7, Integer.parseInt(txt_bQty.getText()));
            pst.setInt(8, Integer.parseInt(txt_iSp.getText()));
            pst.setInt(9, Integer.parseInt(txt_iCp.getText()));
            pst.execute();
            JOptionPane.showMessageDialog(null, "Record Added");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e.getMessage());
        }
    }

    private void btn_clearAllActionPerformed(java.awt.event.ActionEvent evt) {
        txt_bNumber.setText("");
        txt_bDate.setDate(null);
        txt_bCustName.setText("");
        txt_iId.setText("");
        cbo_iName.setSelectedIndex(0);
        txt_iDescription.setText("");
        txt_bQty.setText("");
        txt_iSp.setText("");
        txt_iCp.setText("");
        txt_bAmount.setText("");
    }

    private void btn_clearitemActionPerformed(java.awt.event.ActionEvent evt) {
        txt_iId.setText("");
        cbo_iName.setSelectedIndex(0);
        txt_iDescription.setText("");
        txt_bQty.setText("");
        txt_iSp.setText("");
        txt_iCp.setText("");
        txt_bAmount.setText("");
    }

    private void btn_backActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        new generateBill().setVisible(true);
        this.dispose();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(() -> new addItem().setVisible(true));
    }

    private void initComponents() {
        // UI components initialization (Generated code)
    }

    // Variables declaration (Do not modify)
    private javax.swing.JButton btn_back, btn_calcAmount, btn_clearAll, btn_clearitem, btn_save;
    private javax.swing.JComboBox<String> cbo_iName;
    private javax.swing.JLabel jLabel1, jLabel10, jLabel11, jLabel12, jLabel2, jLabel3, jLabel4, jLabel5,
jLabel7, jLabel8, jLabel9, lbl_background;
    private javax.swing.JPanel pnl_main;
    private javax.swing.JScrollPane scrollPane_iDescription;
    private javax.swing.JTextField txt_bAmount, txt_bCustName, txt_bNumber, txt_bQty, txt_iCp,
txt_iId, txt_iSp;
    private org.jdesktop.swingx.JXDatePicker txt_bDate;
    private javax.swing.JTextArea txt_iDescription;
    // End of variables declaration
}
```

3.7 REMOVE ITEM

```
package rohanbakery;

import java.awt.*;
import java.sql.*;
import javax.swing.*;

public class RemoveItem extends JFrame {
    private Connection conn;
    private PreparedStatement pst;
    private ResultSet rs;
    private boolean listLoaded = false;

    public RemoveItem() {
        initComponents();
        centerFrame();
        conn = ConnectToDatabase.getConnection();
        loadComboBox();
        listLoaded = true;
    }
```

```
private void centerFrame() {
    Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
    setLocation((size.width - getWidth()) / 2, (size.height - getHeight()) / 2);
}

private void loadComboBox() {
    try {
        pst = conn.prepareStatement("select bId from billing where bOk='Yes'");
        rs = pst.executeQuery();
        cbo_bId.removeAllItems();
        while (rs.next()) cbo_bId.addItem(rs.getString(1));
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Billing list cannot be loaded");
    } finally {
        try { if (rs != null) rs.close(); if (pst != null) pst.close(); } catch (Exception ignored) {}
    }
}

private void btn_deleteActionPerformed(java.awt.event.ActionEvent evt) {
    if (txt_bNumber.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please select the bill id to delete");
    } else {
        try {
            pst = conn.prepareStatement("update billing set bOk='No' where bId=?");
            pst.setString(1, cbo_bId.getSelectedItem().toString());
            pst.execute();
            JOptionPane.showMessageDialog(null, "Record removed");
            loadComboBox(); // Reload combo box values
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        } finally {
            try { if (rs != null) rs.close(); if (pst != null) pst.close(); } catch (Exception ignored) {}
        }
    }
}

private void btn_cleaActionPerformed(java.awt.event.ActionEvent evt) {
    txt_bNumber.setText(""); txt_bDate.setDate(null); txt_bCustName.setText("");
    txt_iName.setText(""); txt_iId.setText(""); txt_iDescription.setText("");
    txt_bQty.setText(""); txt_iSp.setText(""); txt_iCp.setText(""); txt_bAmount.setText("");
}

private void cbo_bIdItemStateChanged(java.awt.event.ItemEvent evt) {
    if (listLoaded) {
        try {
            pst = conn.prepareStatement("select * from billing where bId=?");
            pst.setString(1, cbo_bId.getSelectedItem().toString());
```

```
            rs = pst.executeQuery();
            if (rs.next()) {
                txt_bNumber.setText(rs.getString(2)); txt_bDate.setDate(rs.getDate(3));
                txt_bCustName.setText(rs.getString(4)); txt_iId.setText(rs.getString(5));
                txt_iName.setText(rs.getString(6)); txt_iDescription.setText(rs.getString(7));
                txt_bQty.setText(rs.getString(8)); txt_iSp.setText(rs.getString(9));
                txt_iCp.setText(rs.getString(10)); txt_bAmount.setText(rs.getString(12));
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Billing details cannot be found");
        } finally {
            try { if (rs != null) rs.close(); if (pst != null) pst.close(); } catch (Exception ignored) {}
        }
    }
}

private void btn_backActionPerformed(java.awt.event.ActionEvent evt) {
    new GenerateBill().setVisible(true);
    dispose();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new RemoveItem().setVisible(true));
}

// GUI components declaration
private javax.swing.JComboBox<String> cbo_bId;
private javax.swing.JTextField txt_bNumber, txt_bCustName, txt_iId, txt_iName, txt_bQty, txt_iSp,
txt_iCp, txt_bAmount;
private org.jdesktop.swingx.JXDatePicker txt_bDate;
private javax.swing.JTextArea txt_iDescription;
private javax.swing.JButton btn_delete, btn_clea, btn_back;
}
```

3.8 DATABASE CONNECTIVITY

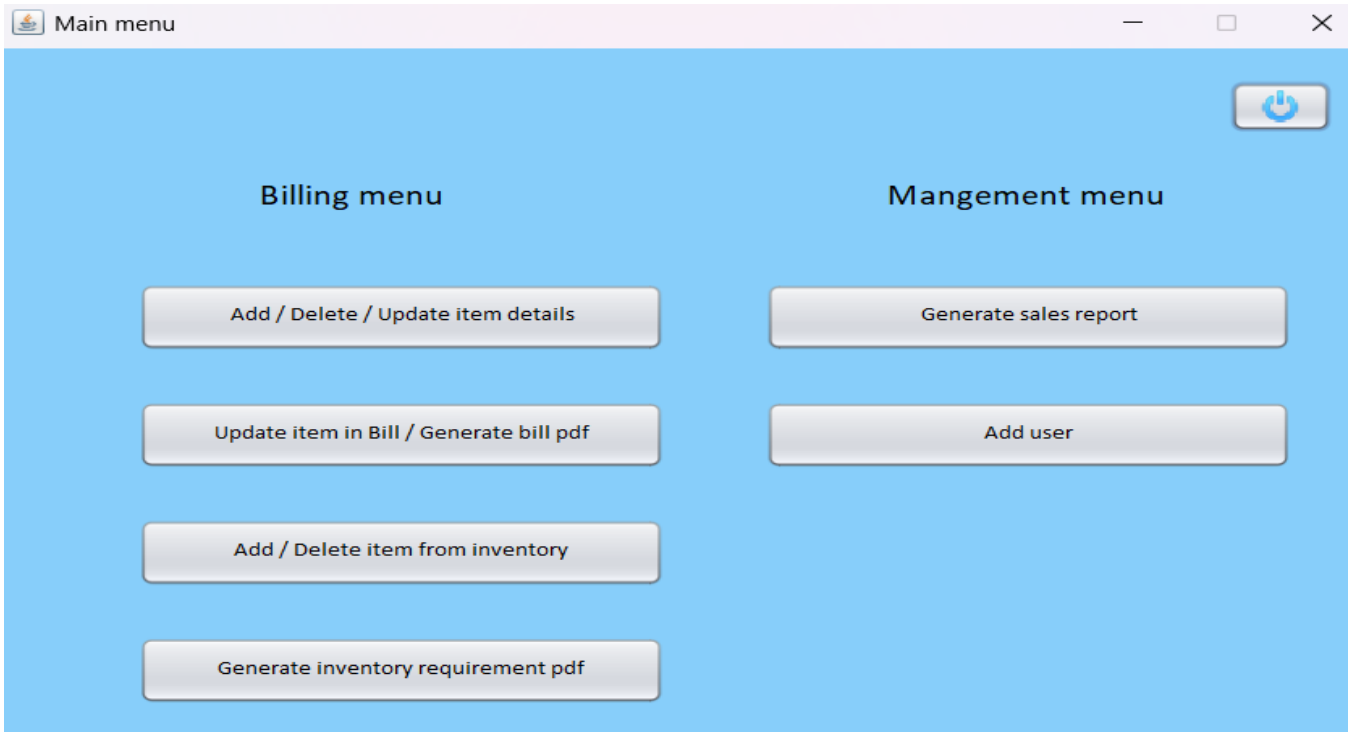
```
package rohanbakery;

import java.sql.*;
import javax.swing.JOptionPane;

public class ConnectToDatabase {
    public static Connection getConnection() {
        Connection conn = null;
        try {
            //Register driver and get the connection using username - root and password - lucifer@000
```

```
            DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
            conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rohankitchen","root","lucifer@000");
            return conn;
        } catch(Exception e) {
            //Displays dialog if connection cannot be established
            JOptionPane.showMessageDialog(null, "Connection cannot be established");
            return null;
        }
    }
}
```

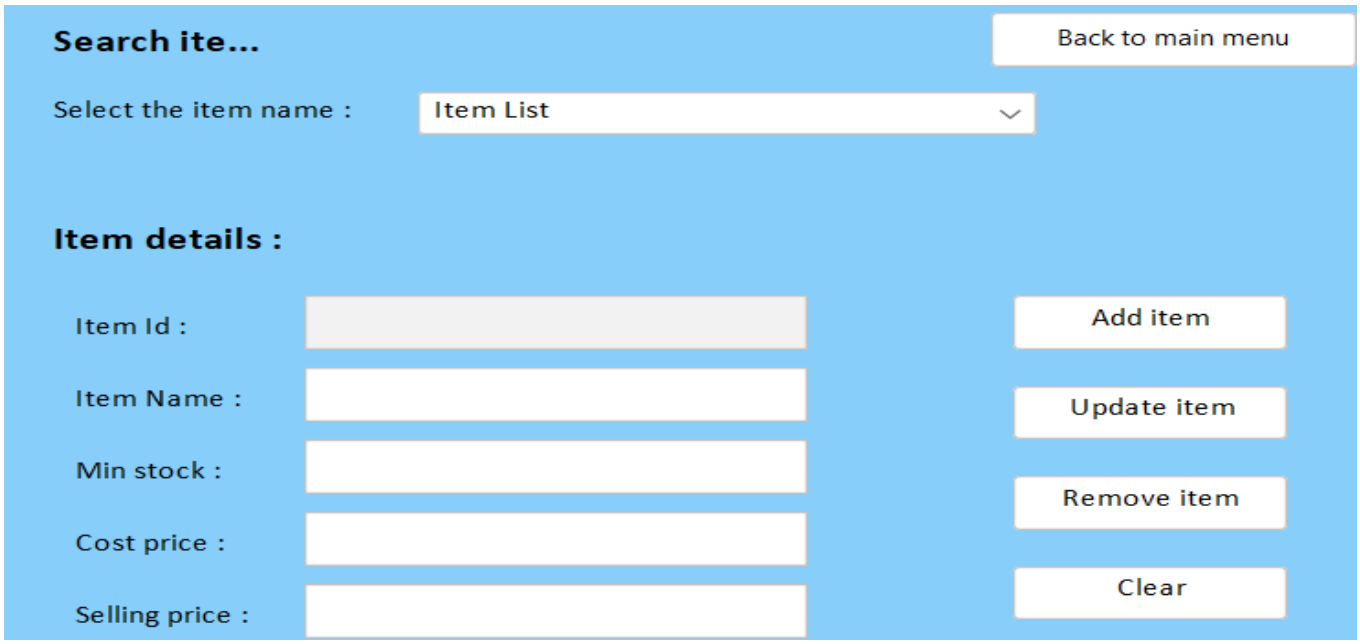
SNAPSHOTS

4.1 LOGIN PAGE

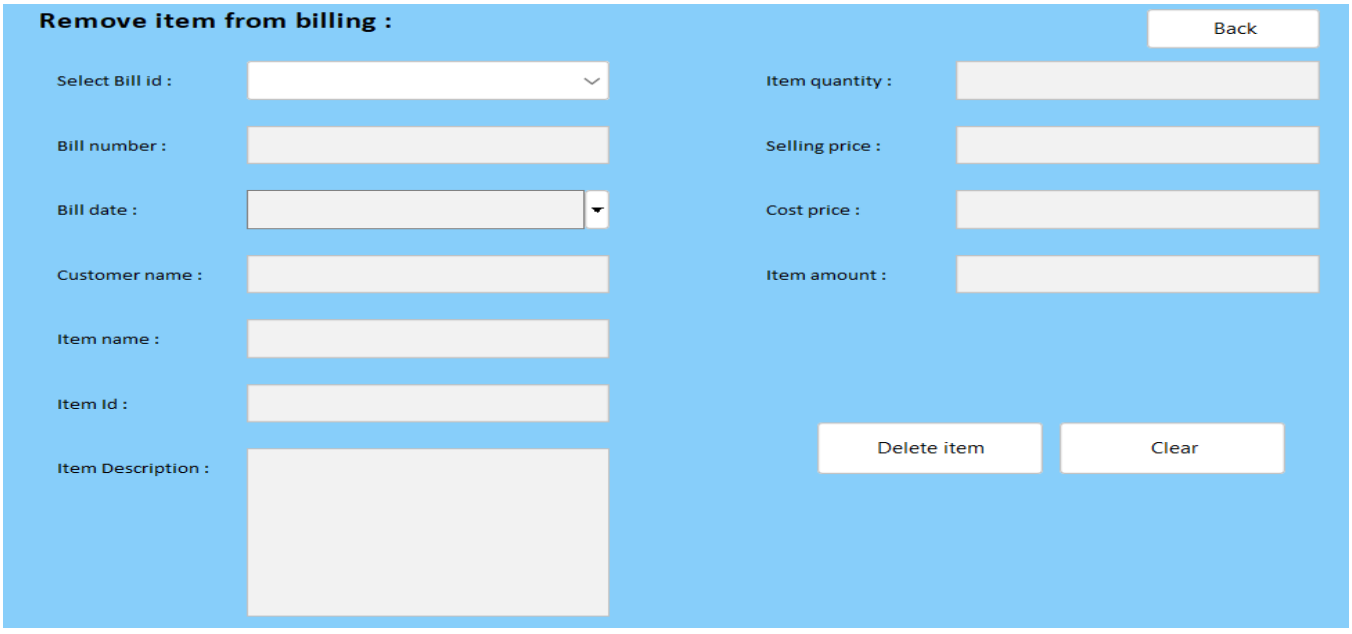
4.2 BILLING AND MANAGEMENT MENU



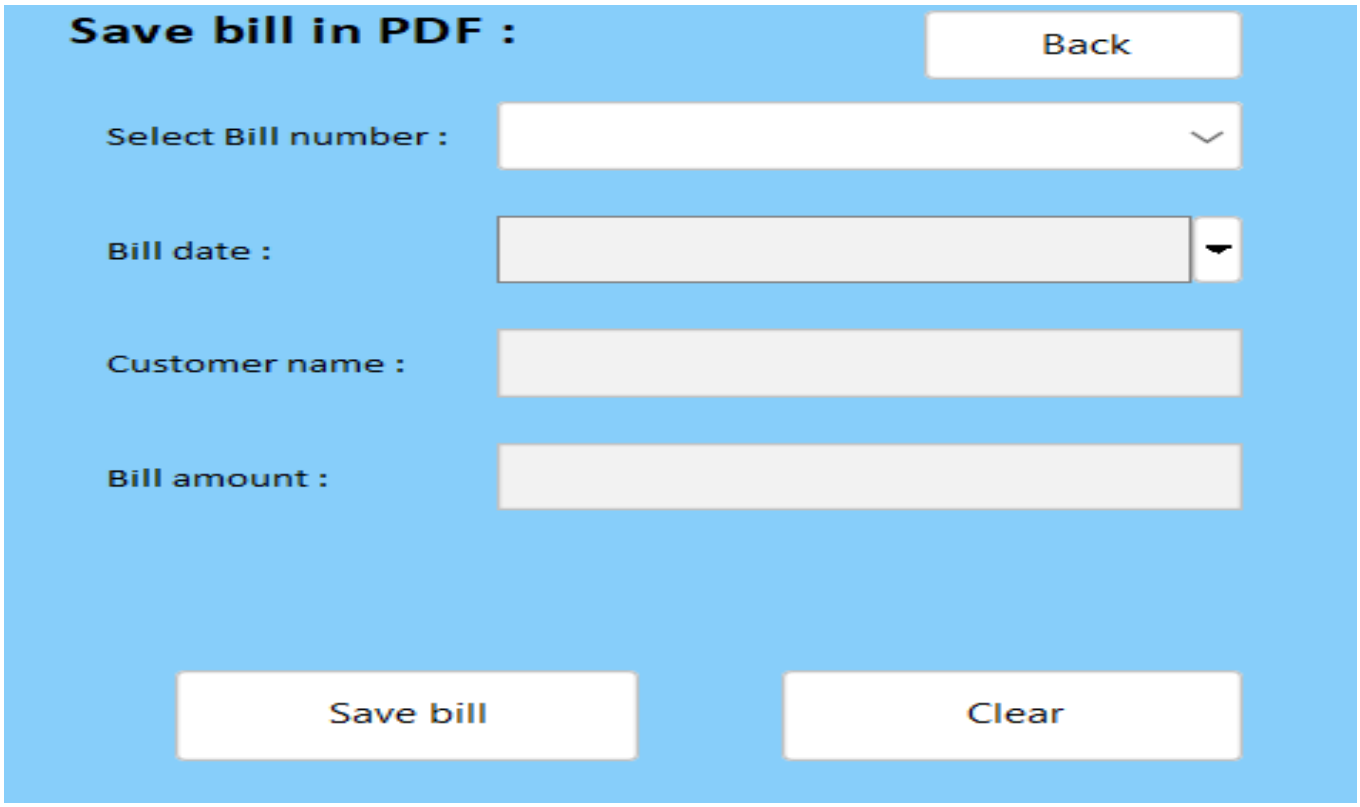
4.3 ITEMS LIST



4.4 REMOVE ITEM



4.5 PRINT BILL



4.6 ADD INVENTORY

4.7 REMOVE INVENTORY

CONCLUSION

The Cafe Management System streamlines operations by integrating billing, inventory, and customer management into a unified platform. It enhances efficiency, reduces manual errors, and ensures a seamless experience for both staff and customers. By automating key processes like item tracking, order management, and report generation, the system supports better decision-making and elevates overall service quality.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL | Codecademy