**CAFE MANAGEMENT SYSTEM**

**A MINI PROJECT REPORT**

**Submitted by**

**Krithika PV 230701157**

**Keerthivarman AB 230701151**

In partial fulfilment for the award of the degree OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE



RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 25

**BONAFIDE CERTIFICATE**

Certified that this project report “**CAFE MANAGEMENT SYSTEM**” is the bonafide work of

**“KRITHIKA PV (230701157), KEERTHIVARMAN AB (230701151)”**

who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**SIGNATURE**

**Mrs. K. MAHESMEENA,**

**Assistant Professor,**

**Computer Science and Engineering,**

**Rajalakshmi Engineering College,**

**(Autonomous),**

**Thandalam, Chennai - 602 105**

**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**

# Chapter 1

1. **INTRODUCTION** 
   1. Introduction
   2. Objectives
   3. Modules

# Chapter 2

1. **SURVEY OF TECHNOLOGIES** 
   1. Software Description .................................... 11-13
   2. Languages .................................................... 14
      1. Java .......................................................... 14-15
      2. SQL ........................................................... 15

## Chapter 3 3 REQUIREMENTS AND ANALYSIS

3.1 Requirement Specification ............................ 16-19

3.1.1 Functional Requirements ......................... 19-21

3.2 Hardware and Software Requirements ........... 22-24

3.3 ER Diagram .................................................. 25

**Chapter 4 PROGRAM CODE**

4.1 Back End Implementation

4.2 Front End Implementation

**Chapter 5 RESULTS AND DISCUSSION**

5.1 Results and Discussion

**Chapter 6 CONCLUSION**

6.1 Conclusion

**Chapter 7 REFERENCES**

7.1 References

**Chapter 1: INTRODUCTION**

### 1.1 INTRODUCTION

The **Cafe Management System** is an advanced software solution tailored to meet the unique operational needs of cafes, ensuring seamless management and enhanced productivity. It automates key cafe operations such as inventory tracking, order processing, and billing, providing a comprehensive platform for efficient day-to-day management. This system is designed to simplify complex tasks, allowing cafe owners and staff to focus on delivering exceptional customer experiences.

**Inventory Management:**  
One of the standout features of the Cafe Management System is its ability to track inventory in real time. It keeps an accurate record of stock levels, helping cafe owners avoid overstocking or running out of essential items. Low-stock alerts notify managers when supplies are nearing depletion, ensuring timely reordering and reducing the risk of service disruptions. This proactive approach not only minimizes waste but also contributes to cost savings by preventing over-purchasing and spoilage.

**Order Processing and Billing:**  
The system streamlines order processing, allowing staff to manage orders efficiently, whether they are dine-in, takeout, or online. It integrates with point-of-sale (POS) devices to facilitate swift and accurate billing, reducing the chances of errors. Customers benefit from faster service, while the cafe enjoys smoother operations and improved customer satisfaction.

**Data Analytics and Reporting:**  
The Cafe Management System offers robust data analytics capabilities, providing valuable insights into sales patterns, customer preferences, and operational performance. These analytics empower cafe owners to make informed decisions, identify trends, and strategize for growth. With access to detailed reports, owners can evaluate the success of promotional campaigns, optimize menu offerings, and plan for future expansions.

**Enhancing Service and Profitability:**  
By digitizing manual processes, the Cafe Management System significantly reduces the workload of cafe staff. This automation not only improves service speed and accuracy but also boosts overall profitability by minimizing errors and operational inefficiencies. With less time spent on administrative tasks, staff can dedicate more energy to customer engagement, fostering loyalty and repeat business.

1.2 OBJECTIVES

 **Reduce operational costs** by minimizing waste and optimizing resource use.

 **Provide data-driven insights** to support strategic decision-making and growth.

 **Ensure secure data management** for staff, inventory, and customer information.

 **Improve staff coordination** through role-based task management and notifications.

 Boost sales by streamlining promotions, loyalty programs, and menu updates.

**1.3 MODULES**

1. **User Authentication and Authorization Module**  
   This module manages secure login and access control for cafe staff and administrators. It ensures role-based access to features, protecting sensitive data and preventing unauthorized actions.
2. **Inventory Management Module**  
   This module tracks stock levels of ingredients and supplies. It provides real-time updates, low-stock alerts, and options to add, update, or remove inventory items for efficient resource management.
3. **Order Management Module**  
   This module handles customer orders, whether dine-in, takeaway, or online. It allows staff to process orders, customize items, and link them to table numbers or delivery requests.
4. **Billing and Payment Module**  
   This module generates accurate bills, processes various payment methods (cash, card, digital wallets), and issues receipts. It also maintains transaction records for audit and reporting purposes.
5. **Menu Management Module**  
   This module allows administrators to create and update menu items, set prices, and manage daily specials or promotional offers. It ensures that the menu reflects real-time availability.
6. **Table Management Module**  
   This module provides a visual representation of table occupancy, tracks reservations, and manages seating arrangements to optimize space usage and enhance the customer experience.
7. **Employee Management Module**  
   This module maintains records of staff details, roles, and work schedules. It supports attendance tracking and shift assignment for smooth operations.
8. **Reporting and Analytics Module**  
   This module generates insights into sales trends, inventory usage, and peak operational hours. It supports performance evaluation, allowing managers to make data-driven decisions.

**Chapter 2: SURVEY OF TECHNOLOGIES**

### 2.1 SOFTWARE DESCRIPTION

The **Cafe Management System** utilizes a combination of technologies to ensure robust and efficient functionality. The backend is supported by a relational database management system (RDBMS), while the frontend features an interactive and user-friendly interface built using Java Swing. Middleware technologies enable seamless communication between the backend and frontend.

#### 2.1.1 Java

* **Role**: Java serves as the primary programming language for both backend logic and GUI development.
* **Usage**:
  + Backend: Handles operations like student registration, profile management, and academic updates.
  + Frontend: A JFrame-based GUI provides an intuitive and interactive user experience.
  + Middleware: Java Database Connectivity (JDBC) ensures seamless communication with the MySQL database.
* **Advantages**:
  + Platform independence for cross-platform compatibility.
  + Built-in security features to protect sensitive student data.

#### 2.1.2 MySQL

* **Role**: MySQL is used as the relational database for storing and managing all student-related information.
* **Usage**:
  + Stores student profiles, attendance, grades, and course information.
  + Efficient SQL queries enable quick retrieval and management of large datasets.
* **Advantages**:
  + Reliable, open-source database management.
  + Ensures data integrity and supports complex queries for academic reporting.

## Chapter 3: REQUIREMENTS AND ANALYSIS

### 3.1 REQUIREMENT SPECIFICATION

**3.1.1 Functional Requirements**

### • Inventory Management

### Allow real-time tracking of stock levels for ingredients and other supplies.

### Provide low-stock alerts to notify staff when inventory needs replenishment.

### Support adding, updating, and deleting inventory items with details such as item name, quantity, cost, and expiration date.

**• Menu Management**

* Enable managers to create, update, and remove menu items, including pricing, descriptions, and categories.
* Allow for daily specials and promotional items to be highlighted.

**• User Authentication and Authorization**

* Enable secure registration and login for cafe staff, including managers, waitstaff, and kitchen staff.
* Implement role-based access control to restrict features based on user roles (e.g., manager vs. staff).
* Maintain session details for logged-in users to ensure secure and uninterrupted operation.

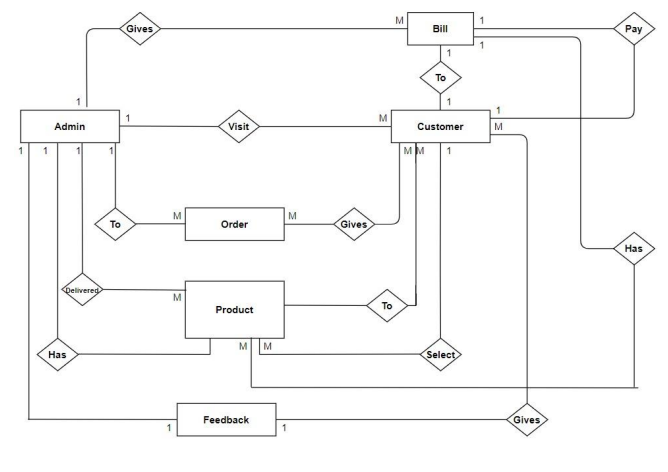
### 3.2 HARDWARE AND SOFTWARE REQUIREMENTS Hardware Requirements

* **Processor**: Intel Core i3 or equivalent for smooth processing.
* **RAM**: 4 GB or higher to handle concurrent database operations.
* **Storage**: At least 500 MB for application files and database storage.
* **Monitor Resolution**: 1024 x 768 or higher.

### Software Requirements

* **Operating System**: Windows 10 or higher.
* **Frontend**: Java Swing (JFrame-based interface).
* **Backend**: MySQL for database management.
* **IDE**: NetBeans for development.
* **Version Control**: Git for code versioning and collaboration.

### 3.3 ER DIAGRAM



## Chapter 4 :PROGRAM CODE

4.1 **Back End Implementation**

CREATE TABLE `user` (

  `uId` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'Unique use id for each user',

  `uName` varchar(45) DEFAULT NULL COMMENT 'Username for login page of the user',

  `uPassword` varchar(45) DEFAULT NULL COMMENT 'Password for the login page of the user',

  `employee\_eId` int(11) NOT NULL,

  `uActive` varchar(3) DEFAULT 'Yes' COMMENT 'Whether the user is still active. It is used to avoid deletring the details of the user that have left the bakery',

  `uType` varchar(45) DEFAULT NULL COMMENT 'Type of the employee\n- Admin\n- Manager\n- Billing executive',

  PRIMARY KEY (`uId`),

  KEY `user\_employee\_fk\_idx` (`employee\_eId`),

  CONSTRAINT `user\_employee\_fk` FOREIGN KEY (`employee\_eId`) REFERENCES `employee` (`eId`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

INSERT INTO `user` VALUES (1,'Admin','Password',1,'Yes','Admin');

Table Item:

CREATE TABLE `item` (

  `iId` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'Unique Id of the item',

  `iName` varchar(45) DEFAULT NULL COMMENT 'Name of the item',

  `iDescription` varchar(45) DEFAULT NULL COMMENT 'Description of the item',

  `iMinStock` int(11) DEFAULT '5' COMMENT 'Minimum stock of item to be on the showboard. Used to check when more qty of the item is required',

  `iCp` int(11) DEFAULT NULL COMMENT 'The expense for making the item (Cost price)',

  `iSp` int(11) DEFAULT NULL COMMENT 'The selling price of the item',

  `iActive` varchar(3) DEFAULT 'Yes' COMMENT 'Whether the item is still active. It is used to avoid deletring the item when it is removed from the meny',

  PRIMARY KEY (`iId`)

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8;

INSERT INTO `item` VALUES (1,'Black forest cake','Black forest cake small piece cake',10,18,30,'Yes'),(2,'Dark chocolate cake','Dark chocolate small piece cake',10,11,22,'Yes'),(3,'Apple cake','Apple cake',10,12,20,'Yes'),(5,'Choco cup','Chocolate filled delight in a cup',5,5,10,'Yes'),(7,'g','w',1,1,1,'No'),(8,'h','h',1,1,1,'No'),(9,'Kaju Barfi','Bardis made from Kajus',10,5,25,'Yes');

Table Inventory:

CREATE TABLE `inventory` (

  `inId` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'Unique id for each entry of the inventory',

  `inDate` date DEFAULT NULL,

  `item\_iId` int(11) NOT NULL,

  `iniName` varchar(45) DEFAULT NULL,

  `inQty` int(11) DEFAULT NULL COMMENT 'Quantity of the item added in the stock',

  `inActive` varchar(3) DEFAULT 'Yes',

  PRIMARY KEY (`inId`),

  KEY `inventory\_item\_fk1\_idx` (`item\_iId`),

  CONSTRAINT `inventory\_item\_fk1` FOREIGN KEY (`item\_iId`) REFERENCES `item` (`iId`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

INSERT INTO `inventory` VALUES (1,'2017-10-03',3,'Apple cake',10,'Yes'),(2,'2017-10-03',2,'Dark chocolate cake',10,'Yes'),(3,'2017-10-03',8,'h',10,'Yes');

Table Employee:

CREATE TABLE `employee` (

  `eId` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'Unique Id for each employee in the employee table',

  `eName` varchar(45) DEFAULT NULL COMMENT 'Name of the employee',

  `eAddress` varchar(45) DEFAULT NULL COMMENT 'Address of the employee',

  `ePhoneNo` varchar(10) DEFAULT NULL COMMENT 'Phone No of the employee',

  `eDateOfBirth` date DEFAULT NULL COMMENT 'Dare of Birth of the employee',

  `eType` varchar(45) DEFAULT NULL COMMENT 'Type of the employee\n- Admin\n- Manager\n- Billing executive',

  `eImage` blob COMMENT 'Image of the employee',

  `eActive` varchar(3) DEFAULT 'Yes' COMMENT 'Whether the emplyee is still active. It is used to avoid deletring the details of the employee the have left the bakery',

  PRIMARY KEY (`eId`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

INSERT INTO `employee` VALUES (1,'Ajeesh',NULL,NULL,NULL,NULL,NULL,'Yes'),(2,'Yasir',NULL,NULL,NULL,NULL,NULL,'Yes'),(3,'Arya',NULL,NULL,NULL,NULL,NULL,'Yes');

Table Billing:

CREATE TABLE `billing` (

  `bId` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'Unique id for each entry for the bill',

  `bNumber` varchar(45) DEFAULT NULL COMMENT 'Bill number',

  `bDate` date DEFAULT NULL COMMENT 'Biiling date',

  `bCustName` varchar(45) DEFAULT NULL,

  `item\_iId` int(11) NOT NULL,

  `iName` varchar(45) DEFAULT NULL,

  `iDescription` varchar(45) DEFAULT NULL COMMENT 'Description fo the item. This is a foreighn key form the item table',

  `bQty` int(11) DEFAULT NULL COMMENT 'Qty of the item sold',

  `iSp` int(11) DEFAULT NULL COMMENT 'Selling price of the item taken from the item table',

  `iCp` int(11) DEFAULT NULL COMMENT 'Cost price of the item taken from the item table',

  `bOk` varchar(5) DEFAULT 'Yes' COMMENT 'Table to know if row is active or has been deleted\nYes means row is active.\nNo means row has been deleted.',

  `bAmount` int(11) GENERATED ALWAYS AS ((`bQty` \* `iSp`)) STORED COMMENT 'Amount of that particular item',

  `bProfit` int(11) GENERATED ALWAYS AS (((`bQty` \* `iSp`) - (`bQty` \* `iCp`))) STORED,

  PRIMARY KEY (`bId`),

  KEY `billing\_item\_fk1\_idx` (`item\_iId`),

  CONSTRAINT `billing\_item\_fk1` FOREIGN KEY (`item\_iId`) REFERENCES `item` (`iId`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8;

INSERT INTO `billing` (`bId`, `bNumber`, `bDate`, `bCustName`, `item\_iId`, `iName`, `iDescription`, `bQty`, `iSp`, `iCp`, `bOk`) VALUES (1,'RB/1/17-18','2017-10-01','Yasir',1,'Black forest cake','Black forest cake small piece cake',10,15,10,'No'),(2,'RB/2/17-18','2017-10-02','Ajeesh',2,'Dark chocolate cake','Dark chocolate small piece cake',5,22,11,'Yes'),(3,'RB/2/17-18','2017-10-02','Ajeesh',5,'dfsdfdsf','dfghdfg',2,10,5,'Yes'),(4,'RB/2/17-18','2017-10-02','Ajeesh',3,'Apple cake','Apple cake',5,20,12,'Yes'),(5,'RB/3/17-18','2017-10-20','Ajeesh',3,'Apple cake','Apple cake',5,20,12,'Yes'),(6,'RB/3/17-18','2017-10-20','Ajeesh',9,'Kaju Barfi','Bardis made from Kajus',2,25,5,'Yes'),(7,'RB/3/17-18','2017-10-20','Ajeesh',5,'Choco cup','Chocolate filled delight in a cup',6,10,5,'Yes'),(8,'RB/3/17-18','2017-10-20','Ajeesh',1,'Black forest cake','Black forest cake small piece cake',7,30,18,'Yes');

**4.2 Front End Implementation**

## LOGIN PAGE

package keerthi;

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

}

## ADD INVENTORY

package keerthi;

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));

}

}

**GENERATE BILL**

package keerthi;

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;

}

## PRINT BILL

package keerthi;

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));

}

}

## REMOVE INVENTORY

package keerthi;

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));

}

}

## ADD ITEM

package keerthi;

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

}

## REMOVE ITEM

package keerthi;

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;

}

**DATABASE CONNECTIVITY**

package keerthi;

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;

        }

    }

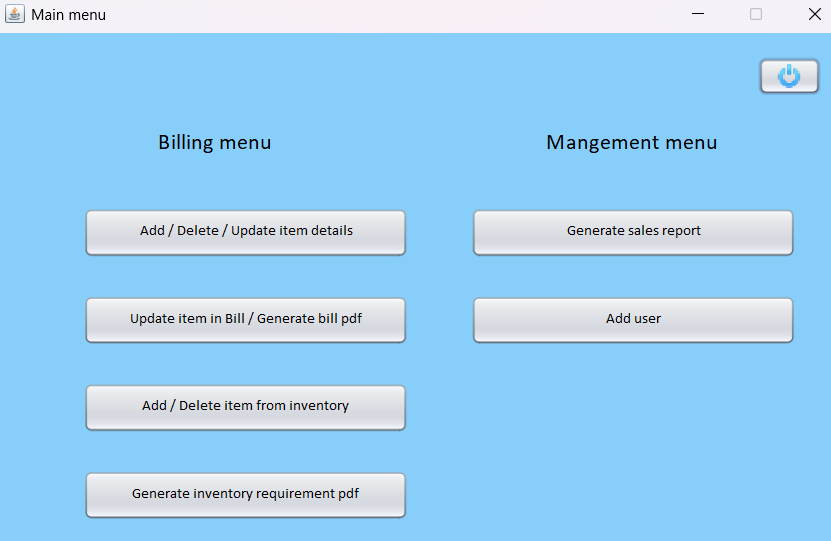
}

# Chapter 5: RESULT AND DISCUSSION

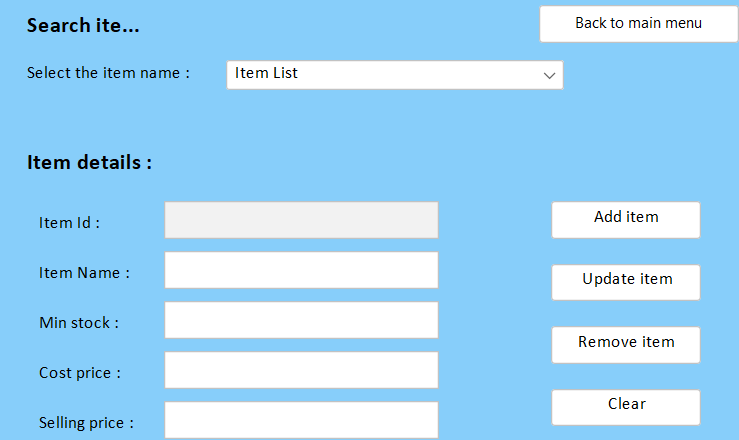
## 5.1 LOGIN PAGE

## 

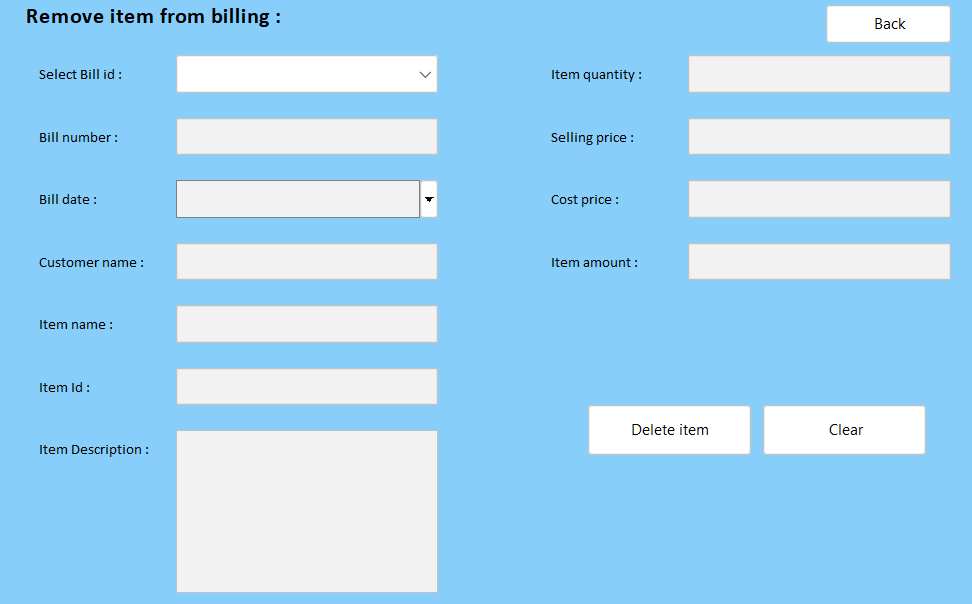
## 

**5.2 BILLING AND MANAGEMENT MENU**

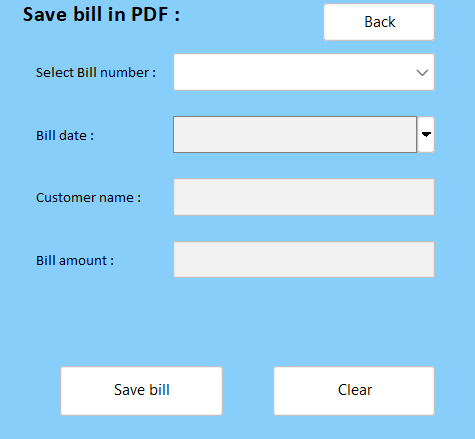
**5.3 ITEMS LIST**



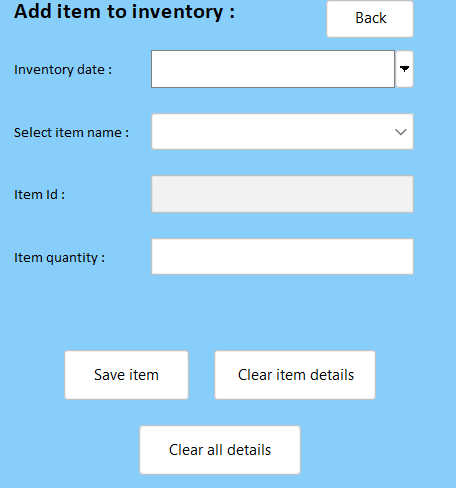
## 5.4 REMOVE ITEM



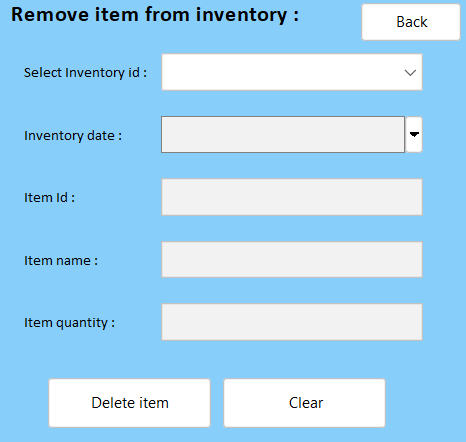
**5.5 PRINT BILL**



## 5.6 ADD INVENTORY



## 5.7 REMOVE INVENTORY



# Chapter 6: Conclusion

### 6.1 CONCLUSIONN

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.

**Chapter 7: REFERENCE**

# 7.1 REFERENCES

1. [**https://www.javatpoint.com/java-tutorial**](https://www.javatpoint.com/java-tutorial)

1. <https://www.wikipedia.org/>

1. <https://www.w3schools.com/sql/>

1. [SQL | Codecademy](https://www.codecademy.com/resources/docs/sql)