ABSTRACT

This project on Bakery Management System is generate the bill payment process of Bakery System. The aim of the project is to develop an efficient, user-friendly software solution that automates key bakery operations, with a focus on managing sales, profit analysis, inventory tracking, and billing. The system will provide real-time processing of sales transactions, accurately track inventory usage and stock levels, calculate profits, and generate detailed bills and invoices in PDF format for customers. It allows storing and retrieving data related to the bakery products and make transactions related to bill etc. For data storage and retrieval we use MySQL Database. The system we have implemented manages different objects viz.

. Admin

. Manager

. Billing Executive

Each of these accesses a database schema which has corresponding tables.

Languages Used - Java Core

Concept Used - Swing

IDE Used - NetBeans

Database Used - MySQL

CONTENTS

CHAPTER 1:

INTRODUCTION

The Bakery Management System is a comprehensive software solution designed to streamline and automate essential bakery operations. As bakeries continue to grow in popularity, managing day-to-day tasks like inventory tracking, order processing, billing, and profit analysis can become complex and time-consuming. Traditional manual methods of managing these operations can lead to errors, inefficiencies, and delays, ultimately impacting customer satisfaction and business profitability. This system addresses these challenges by providing a centralized platform to manage key functions. It allows bakery owners and staff to monitor inventory levels in real-time, ensuring that ingredients are available when needed, while minimizing waste through low-stock alerts and efficient stock management. The Bakery Management System not only improves operational efficiency and accuracy but also offers valuable data analytics to support strategic planning and growth. By digitizing and automating key processes, this system empowers bakeries to provide better service, reduce manual workload, and increase profitability, making it an essential tool for modern bakery management.

1.1 Problem Definition:

This project on Bakery Management System is generate the bill payment process of Bakery System. Generating bills manually is time-consuming and error-prone, leading to discrepancies in pricing and delayed customer service. Additionally, a lack of automated revenue tracking makes it difficult for bakery owners to monitor profits, costs, and sales trends accurately. For data storage and retrieval we use the MySQL database. It enables us to add any number of records in our database from the frontend which is Java core. Any changes made in the frontend will be reflected at the backend.

1.2 Need

A bakery requires efficient tracking of ingredients and stock to avoid shortages or waste. This system will enable real-time inventory tracking, low-stock alerts, and usage analytics, ensuring the bakery always has the necessary ingredients available while minimizing excess. The Bakery Management System will streamline order processing, allowing bakery staff to handle customer orders with ease, reducing wait times, and improving the customer experience. Manual billing can be time-consuming and prone to errors. The system will automatically generate itemized bills in PDF format for easy, accurate, and quick transactions, improving both customer satisfaction and staff productivity. This system will provide detailed sales reports and profit analysis, enabling bakery owners to make data-driven decisions, identify popular items, and optimize their pricing and marketing strategies. Quick order processing, accurate billing, and availability of fresh ingredients contribute significantly to customer satisfaction. By automating these tasks, the system enables the bakery to deliver a more seamless and pleasant experience for its customers. � Real-time reporting on inventory, sales, and profits provides valuable insights for strategic planning and business growth. With access to accurate and updated data, bakery owners can make informed decisions on inventory purchasing, menu planning, and marketing. This system provides a scalable solution that can handle an expanding customer base, more extensive inventory, and increased sales volume, supporting the bakery's growth.

A few factors that directs us to develop a new system are given below -:

1) Faster System
2) Accuracy
3) Reliability
4) Informative

CHAPTER 2

REQUIREMENTS

2.1 Software Requirement Specifications

Operating System Front End Back End Server Documentation : Windows 10

Frontend Software: Java NetBeans 8.2 : JDK 8

Backend Software: MySQL

2.2 Hardware Requirement Specifications

Computer Processor Core i3 Processor Speed 2.3 GHz Processor Hard Disk 400 GB or more RAM Min 2GB

CHAPTER 3

IMPLEMENTATION

3.1 Backend Implementation

MYSQL

MySQL is an open-source relational database management system (RDBMS). A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

Table User:

```
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',N
ULL,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');
```

3.2 Frontend Implementation

Java Core

Core Java is the part of Java programming language that is used for creating or developing a general-purpose application. It uses only one tier architecture that is why it is called as 'stand alone' application.Core java programming covers the swings, socket, awt, thread concept, collection object and classess.

Swings

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

3.3 Creating Mainframe Class:

```
package ajeesh;

import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JOptionPane;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class AddUser extends javax.swing.JFrame {

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

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

```
    private void initComponents() {

        lbl_username = new javax.swing.JLabel();
        lbl_password = new javax.swing.JLabel();
        lbl_userType = new javax.swing.JLabel();
        txt_username = new javax.swing.JTextField();
        txt_password = new javax.swing.JPasswordField();
        cmb_userType = new javax.swing.JComboBox<>();
        btn_addUser = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Add New User");

        lbl_username.setText("Username:");
        lbl_password.setText("Password:");
        lbl_userType.setText("User Type:");

        cmb_userType.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Admin",
"Manager", "Billing executive" }));

        btn_addUser.setText("Add User");
        btn_addUser.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btn_addUserActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
          .addGroup(layout.createSequentialGroup()
            .addGap(30, 30, 30)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addComponent(lbl_username)
              .addComponent(lbl_password)
              .addComponent(lbl_userType))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
              .addComponent(txt_username)
              .addComponent(txt_password)
              .addComponent(cmb_userType, 0, 150, Short.MAX_VALUE)
              .addComponent(btn_addUser, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap(30, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(layout.createSequentialGroup()
            .addGap(20, 20, 20)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(lbl_username)
              .addComponent(txt_username, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(lbl_password)
              .addComponent(txt_password, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(lbl_userType)
```

```
                .addComponent(cmb_userType, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addComponent(btn_addUser)
            .addContainerGap(20, Short.MAX_VALUE))
        );

        pack();
    }

    private void btn_addUserActionPerformed(java.awt.event.ActionEvent evt) {
        String username = txt_username.getText();
        String password = new String(txt_password.getPassword());
        String userType = cmb_userType.getSelectedItem().toString();

        if (username.isEmpty() || password.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please fill all fields", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bakery",
"root", null);
            String sql = "INSERT INTO user (uName, uPassword, employee_eId, uActive, uType) VALUES (?,
?, ?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, username);
            stmt.setString(2, password);
            stmt.setInt(3, 1); // Assuming `employee_eId` is set to 1 by default
            stmt.setString(4, "Yes");
            stmt.setString(5, userType);
            int rowsInserted = stmt.executeUpdate();
            if (rowsInserted > 0) {
                JOptionPane.showMessageDialog(this, "User added successfully!");
                this.dispose();  // Close the AddUser form
            }
            stmt.close();
            conn.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error adding user: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    private javax.swing.JButton btn_addUser;
    private javax.swing.JComboBox<String> cmb_userType;
    private javax.swing.JLabel lbl_password;
    private javax.swing.JLabel lbl_userType;
    private javax.swing.JLabel lbl_username;
    private javax.swing.JPasswordField txt_password;
    private javax.swing.JTextField txt_username;
}
```

3.4 JAVA DATABASE CONNECTIVITY:

```
package ajeesh;

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

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

CHAPTER 4

SNAPSHOTS

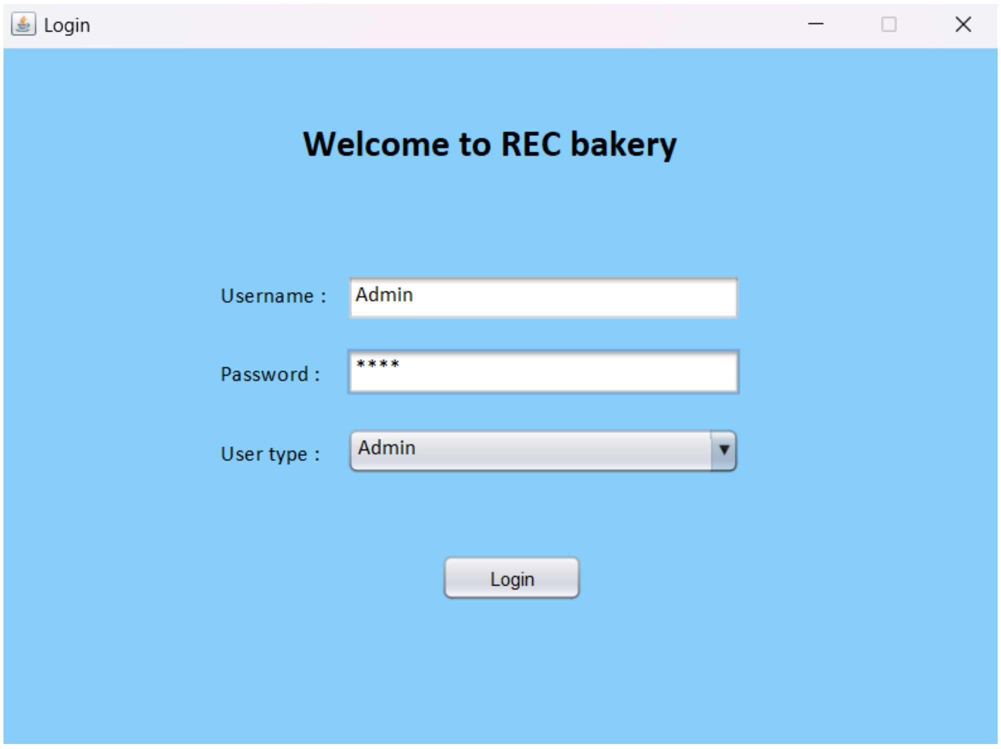


Fig 4.1: Login Operation

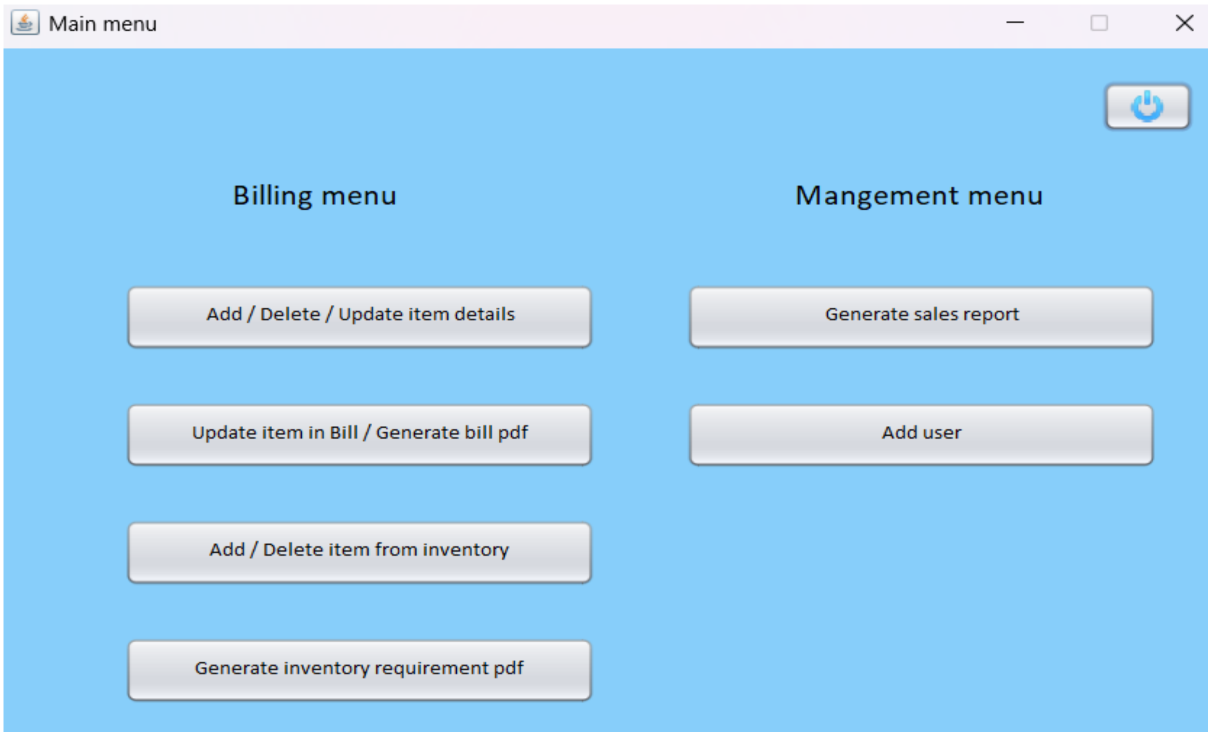
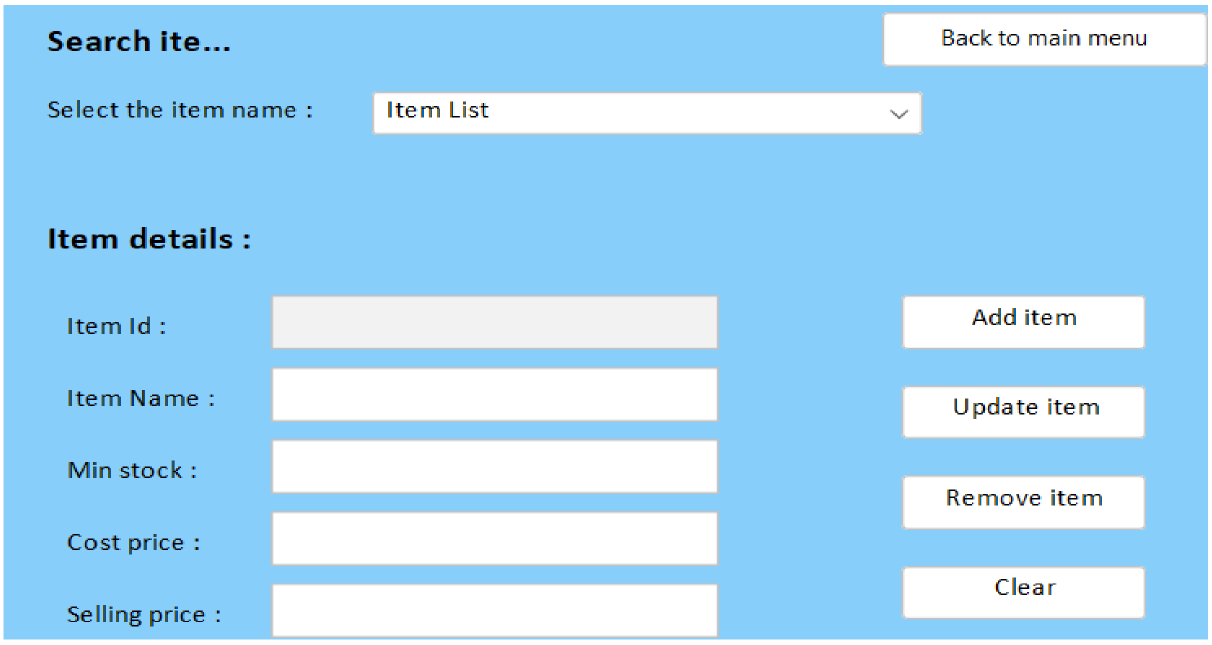

Main menu
Billing menu
Mangement menu
Add / Delete / Update item details
Update item in Bill / Generate bill pdf
Add / Delete item from inventory
Generate inventory requirement pdf
Generate sales report
Add user

Fig:4.2 BILLING AND MANAGEMENT MENU


Search ite...

Back to main menu

Select the item name : Item List

Item details :

Item Id :

Item Name :

Min stock :

Cost price :

Selling price :

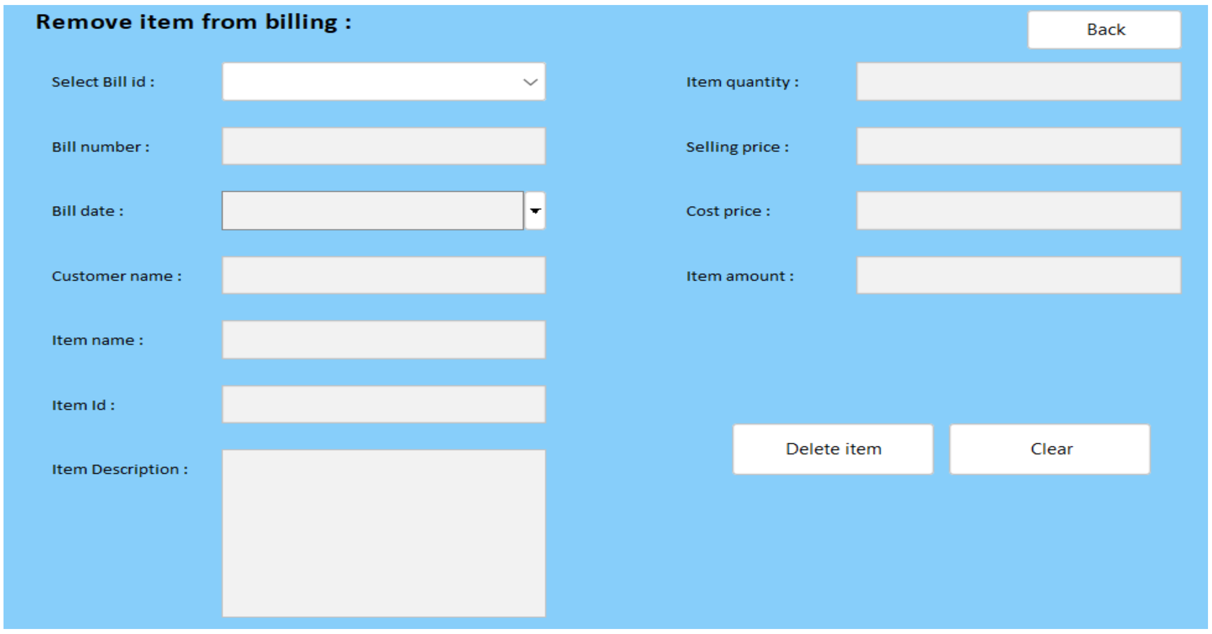
Add item

Update item

Remove item

Clear

Fig:4.3 ITEM LISTS


Remove item from billing :

Back

Select Bill id :

Bill number :

Bill date :

Customer name :

Item name :

Item Id :

Item Description :

Item quantity :

Selling price :

Cost price :

Item amount :

Delete item

Clear

Fig:4.4 DELETE ITEM

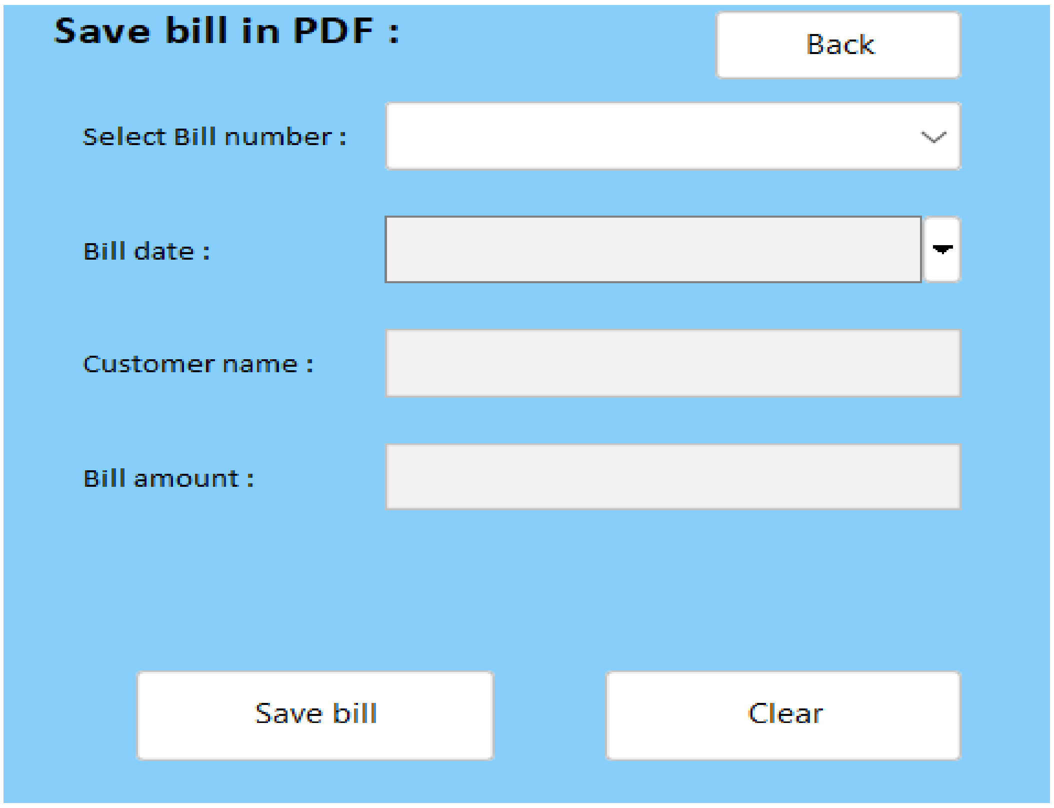


Fig:4.5 PRINT BILL

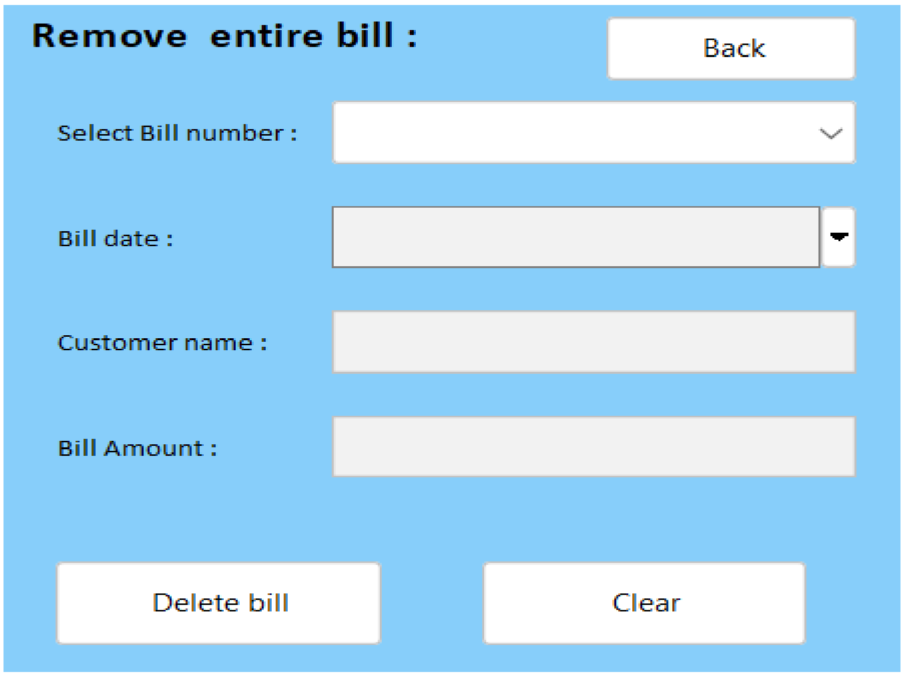
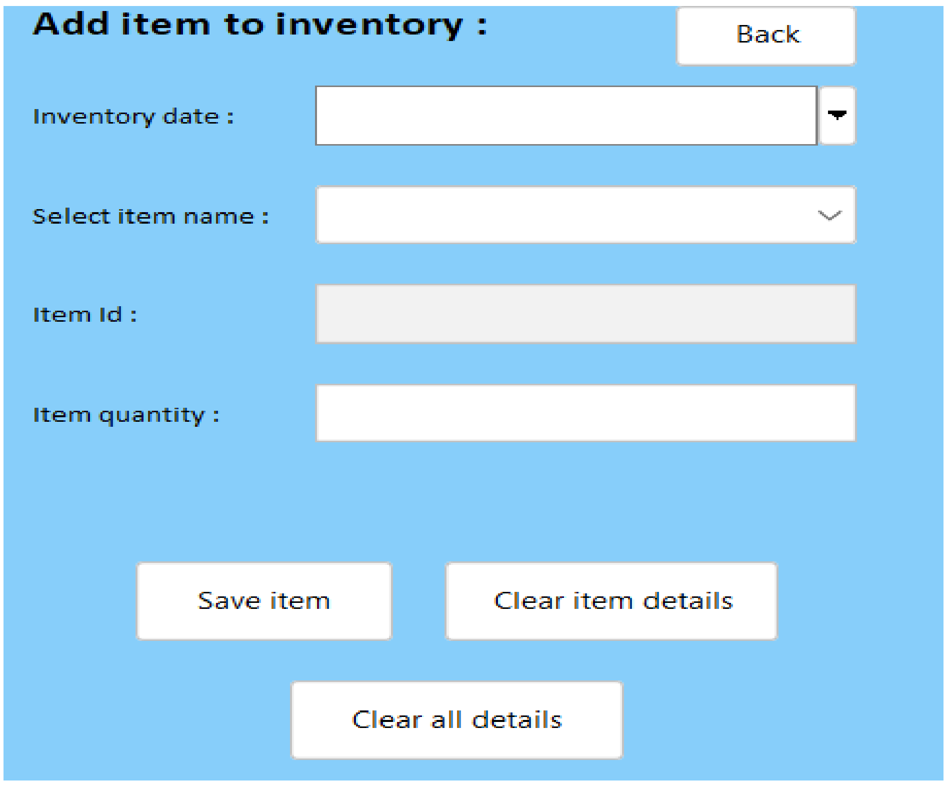


Fig:4.6 DELETE BILL


Add item to inventory :

Back

Inventory date :

Select item name :

Item Id :

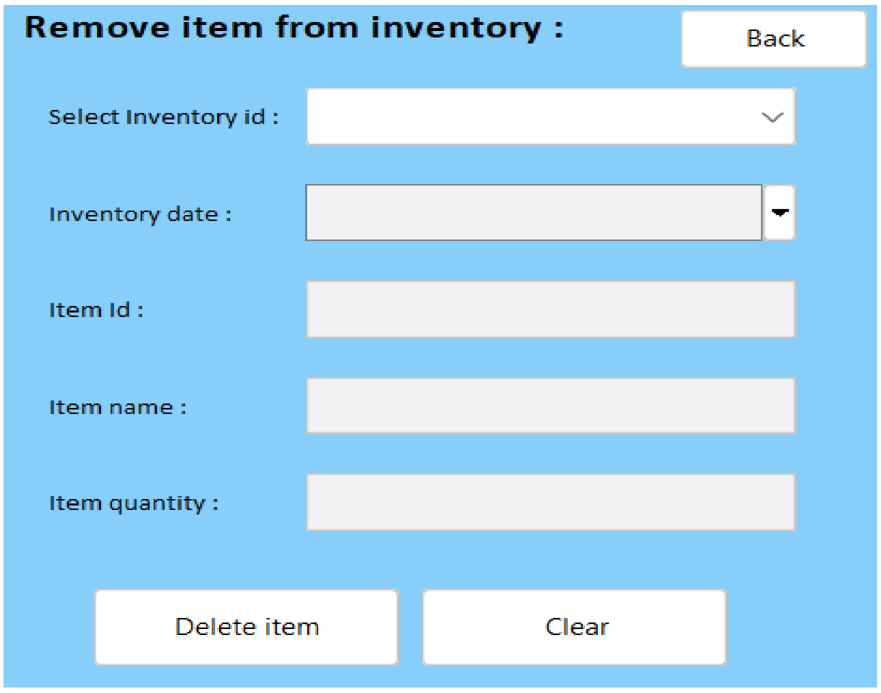
Item quantity :

Save item

Clear item details

Clear all details

Fig:4.7 ADD INVENTORY


Remove item from inventory :

Back

Select Inventory id :

Inventory date :

Item Id :

Item name :

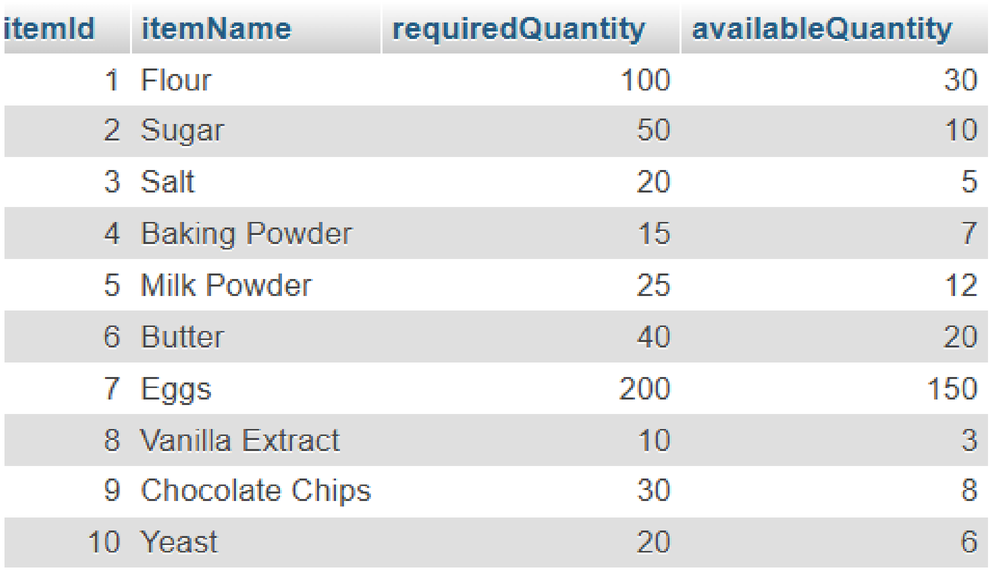
Item quantity :

Delete item

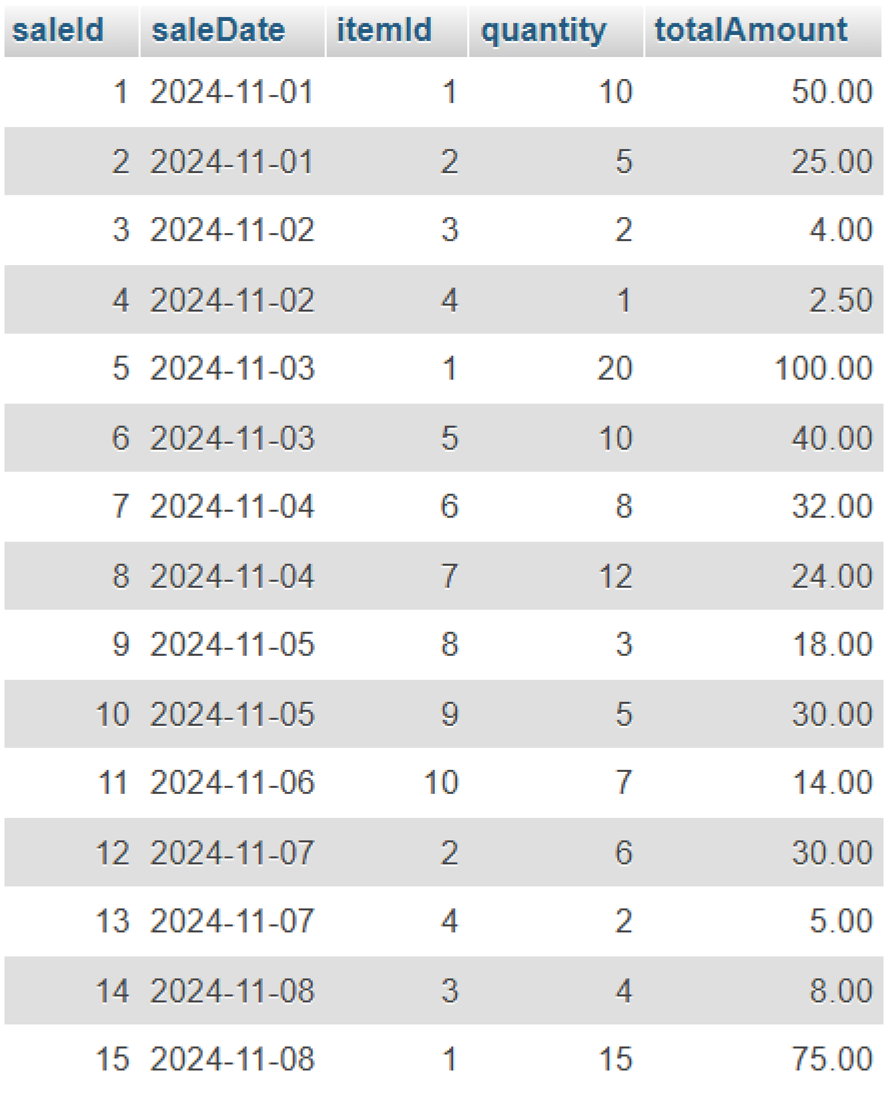
Clear

Fig:4.8 DELETE INVENTORY

Fig:4.9 Back End Records

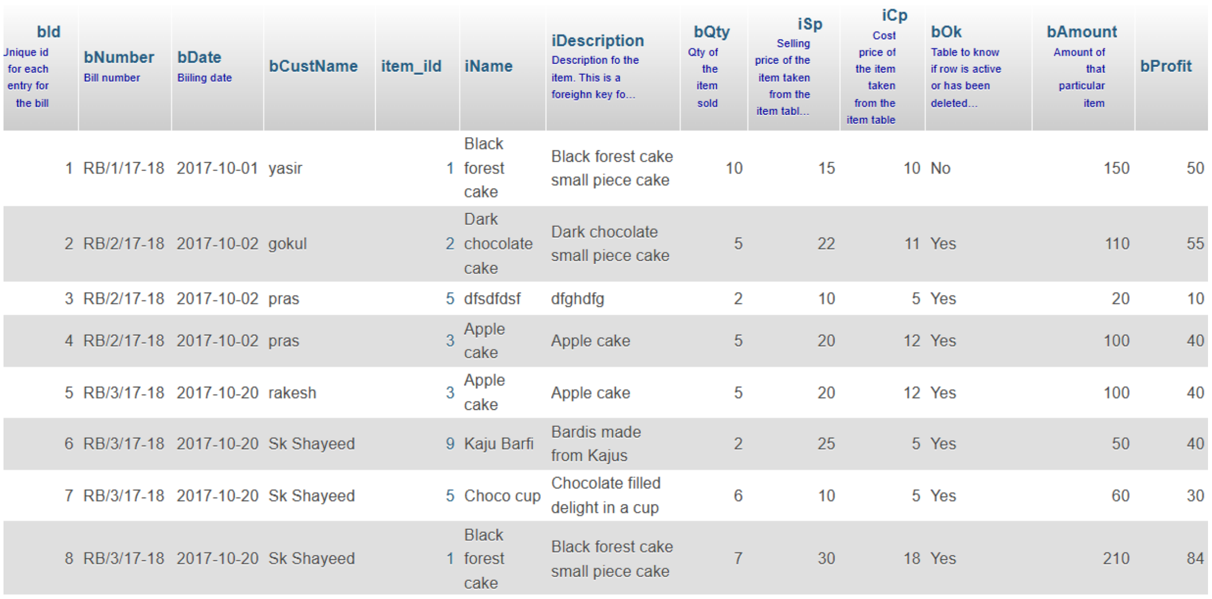


itemId	itemName	requiredQuantity	availableQuantity
1	Flour	100	30
2	Sugar	50	10
3	Salt	20	5
4	Baking Powder	15	7
5	Milk Powder	25	12
6	Butter	40	20
7	Eggs	200	150
8	Vanilla Extract	10	3
9	Chocolate Chips	30	8
10	Yeast	20	6



saleId	saleDate	itemId	quantity	totalAmount
1	2024-11-01	1	10	50.00
2	2024-11-01	2	5	25.00
3	2024-11-02	3	2	4.00
4	2024-11-02	4	1	2.50
5	2024-11-03	1	20	100.00
6	2024-11-03	5	10	40.00
7	2024-11-04	6	8	32.00
8	2024-11-04	7	12	24.00
9	2024-11-05	8	3	18.00
10	2024-11-05	9	5	30.00
11	2024-11-06	10	7	14.00
12	2024-11-07	2	6	30.00
13	2024-11-07	4	2	5.00
14	2024-11-08	3	4	8.00
15	2024-11-08	1	15	75.00

Fig: 4.10 Back End



bId Unique id for each entry for the bill	bNumber Bill number	bDate Biiling date	bCustName	item_iId	iName	iDescription Description fo the item. This is a foreighn key fo...	bQty Qty of the item sold	iSp Selling price of the item taken from the item tabl...	iCp Cost price of the item taken from the item table	bOk Table to know if row is active or has been deleted...	bAmount Amount of that particular item	bProfit
1	RB/1/17-18	2017-10-01	yasir	1	Black forest cake	Black forest cake small piece cake	10	15	10	No	150	50
2	RB/2/17-18	2017-10-02	gokul	2	Dark chocolate cake	Dark chocolate small piece cake	5	22	11	Yes	110	55
3	RB/2/17-18	2017-10-02	pras	5	dfsdfdsf	dfghdfg	2	10	5	Yes	20	10
4	RB/2/17-18	2017-10-02	pras	3	Apple cake	Apple cake	5	20	12	Yes	100	40
5	RB/3/17-18	2017-10-20	rakesh	3	Apple cake	Apple cake	5	20	12	Yes	100	40
6	RB/3/17-18	2017-10-20	Sk Shayeed	9	Kaju Barfi	Bardis made from Kajus	2	25	5	Yes	50	40
7	RB/3/17-18	2017-10-20	Sk Shayeed	5	Choco cup	Chocolate filled delight in a cup	6	10	5	Yes	60	30
8	RB/3/17-18	2017-10-20	Sk Shayeed	1	Black forest cake	Black forest cake small piece cake	7	30	18	Yes	210	84

CONCLUSION

The Bakery Management System effectively streamlines bakery operations by automating key tasks like inventory tracking, order processing, billing, and financial reporting. This automation reduces errors, enhances efficiency, and improves customer satisfaction. Real-time inventory management minimizes waste, while automated billing speeds up transactions and ensures accuracy. Financial insights provided by the system enable bakery owners to make data-driven decisions to boost profitability and support growth. Overall, the system is a scalable, comprehensive solution that empowers bakeries to operate more efficiently and focus on delivering quality products and services, establishing a strong foundation for future success.