NAME : ABHISHEK S ROLL NUMBER : 230701006

CLASS : CSE “A”

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**

|  |  |  |
| --- | --- | --- |
| **CHAPTERS** |  | **PAGE NO** |
| Chapter 1 | Introduction  1.1Problem Definition 1  1.2 Need 2 |  |
| Chapter 2 | Requirements  2.1 Software Requirement Specifications 3  2.2 Hardware Requirement Specifications 3 |  |
| Chapter 3 | Entity Relationship Diagram 4  3.1 Entity relationship diagram 5 |  |
| Chapter 4 | Schema Diagram 6  4.1 Schema diagram 7 |  |
| Chapter 5 | Implementation  5.1 Backend Implementation 8  5.2 Frontend implemenatation 9  5.3 Creating mainframe class 10 - 13 |  |
| Chapter 6 | Snapshots 14 - 19 |  |
|  | Conclusion |  |
|  | References |  |

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. **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. **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

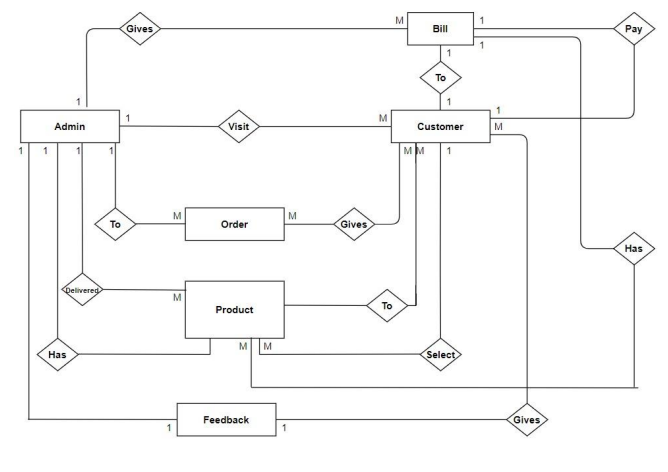
**2.2 Hardware Requirement Specifications**

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

# **CHAPTER 3**

# ***ENTITY RELATIONSHIP DIAGRAM***

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. If the application is primarily a database application, the entity-relationship approach can be used effectively for modeling some parts of the problem. The main focus in ER modeling is the Data Items in the system and the relationship between them. It aims to create conceptual scheme for the Data from the user’s perspective. The model thus created is independent of any database model. The ER models are frequently represented as ER diagram. Here we present the ER diagram of the above mentioned project.



# **CHAPTER 4**

## **SCHEMA DIAGRAM**

*4.1 SCHEMA DIAGRAM*

A database schema is the skeleton structure that represents the logical view of the entire database. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It’s the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories −

* Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
* Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

# **CHAPTER 5**

***IMPLEMENTATION***

5.1 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',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');

5.2 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.

5.3 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(); // Set frame in the center of the screen

}

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;

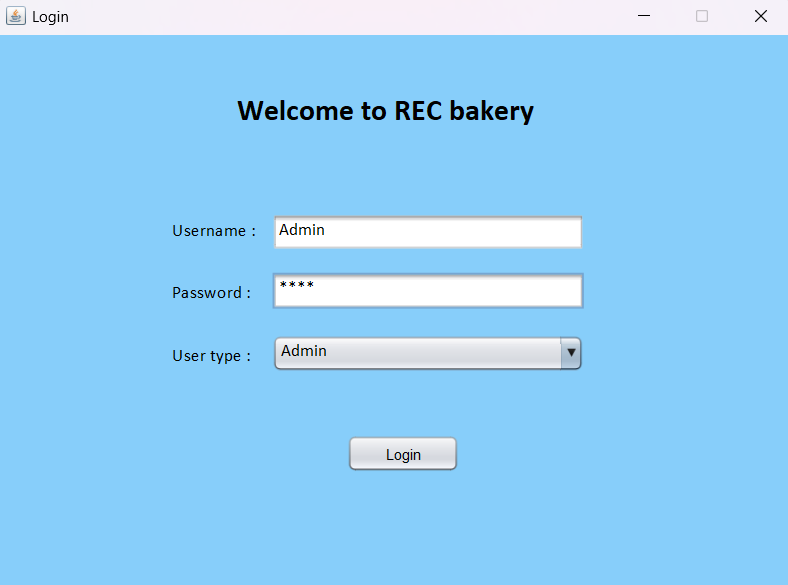
private javax.swing.JPasswordField txt\_password;

private javax.swing.JTextField txt\_username;

}

# **CHAPTER 6**

**SNAPSHOTS**



**Fig 6.1: Login Operation**

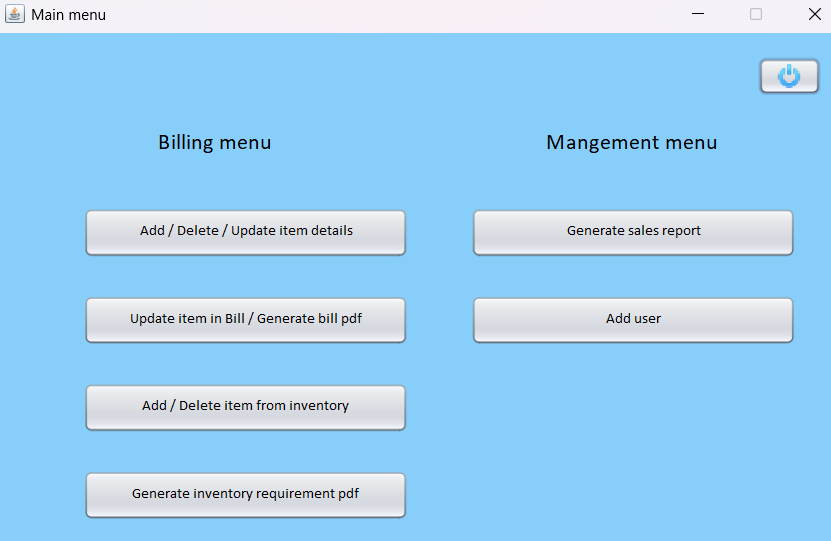


Fig:6.2 BILLING AND MANAGEMENT MENU

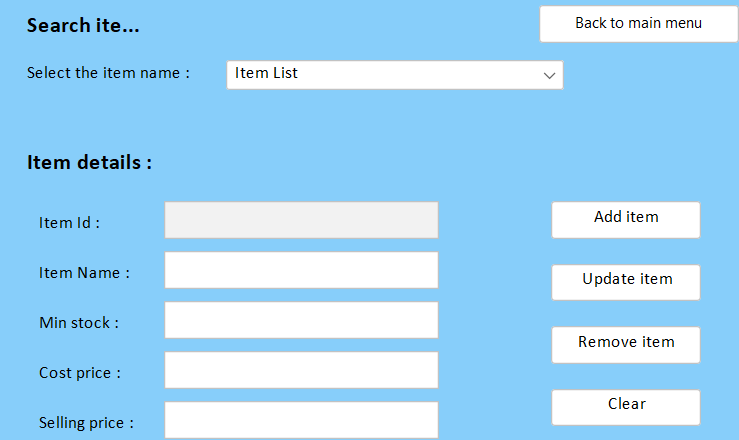


Fig:6.3 ITEM LISTS

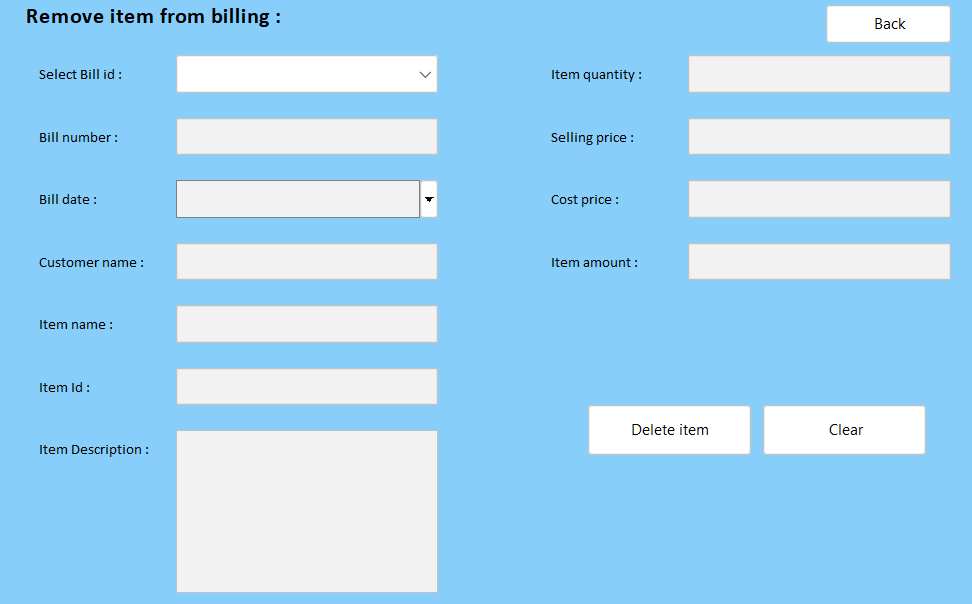


Fig:6.4 DELETE ITEM

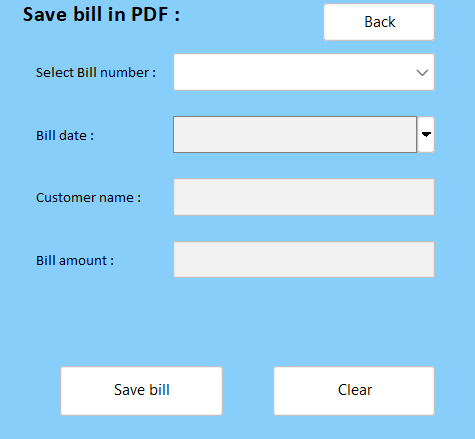


Fig:6.5 PRINT BILL

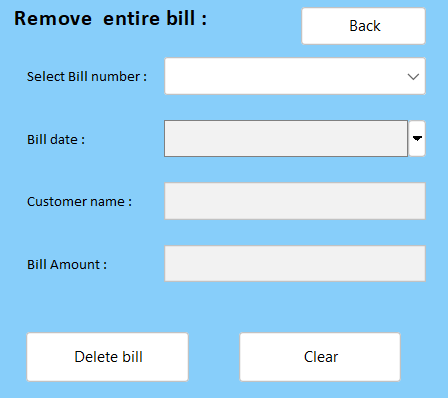


Fig:6.6 DELETE BILL

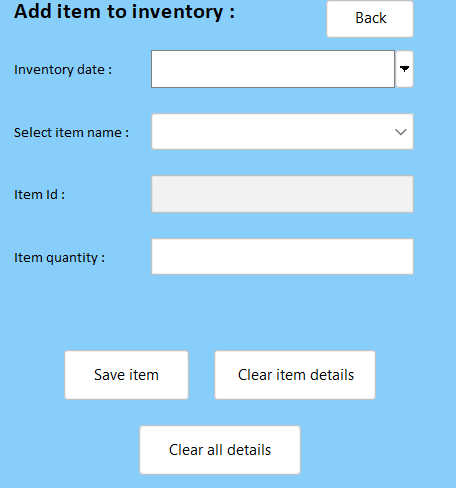


Fig:6.7 ADD INVENTORY

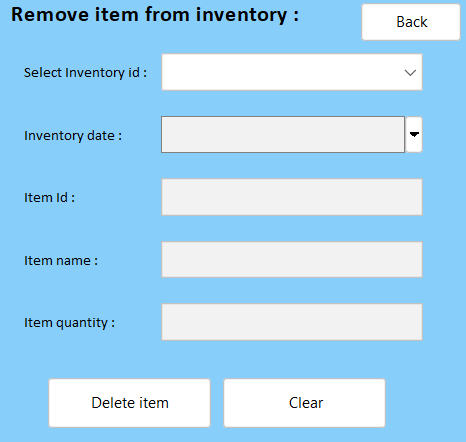


Fig:6.8 DELETE INVENTORY

Fig:6.9 Back End Records



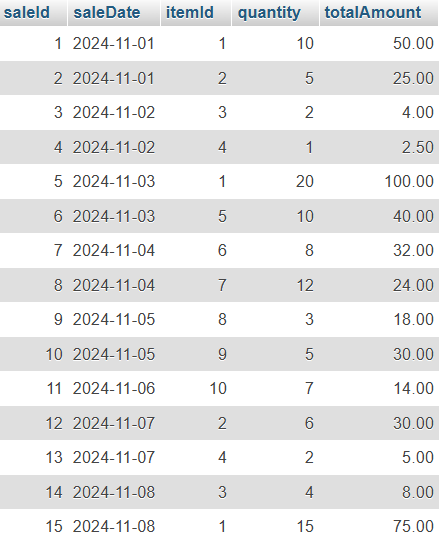
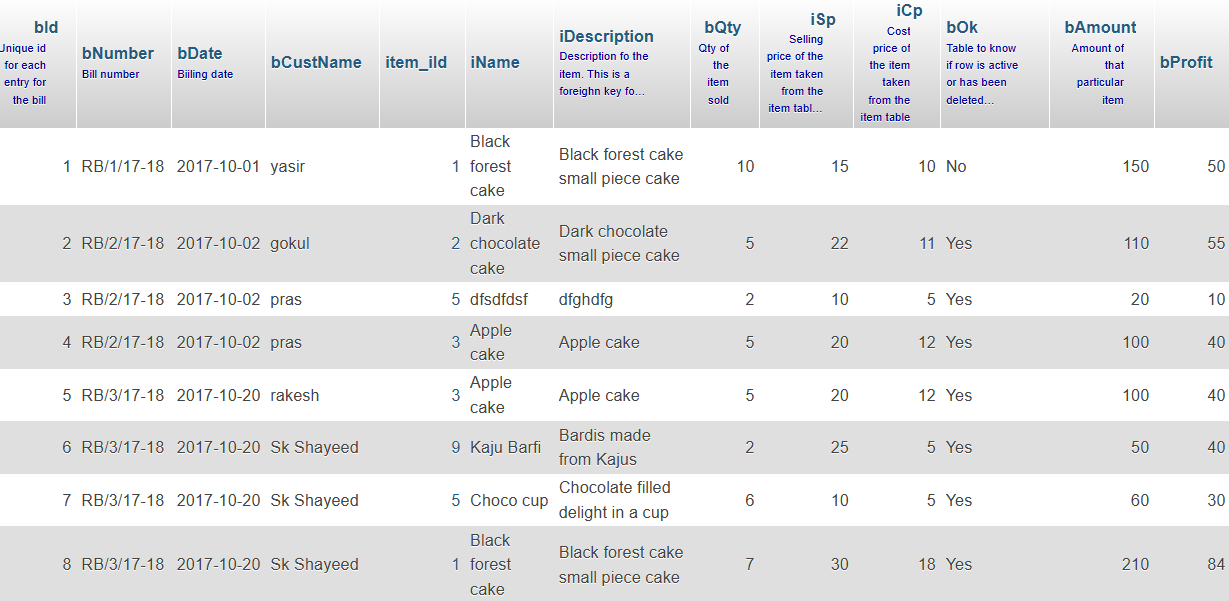


Fig: 6.10 Back End



**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.