SEVA MANDAL EDUCATION SOCIETY'S

SMT.KAMLABEN GAMBHICHAND SHAH DEPARTMENT OF COMPUTER APPLICATIONS

UNDER

DR.BMN COLLEGE OF HOME SCIENCE (Autonomous)

NAAC RE-ACCREDITED 'A+' GRADE WITH
CGPA 3.69/4 UGC STATUS-COLLEGE WITH
POTENTIAL FOR EXCELLENCE
338, R.A. KIDWAI ROAD, MATUNGA, MUMBAI-19

2023-2024

PROJECT ON

"Cake Shop Management System"

FOR

"BETTER BE BUTTER"

UNDER THE GUIDANCE
OF PROF.NITIN PAWAR

Submitted in partial fulfilment of the requirements for the Degree Bachelor of Computer Application (BCA)

Submission Date: 21st April 2024

Submitted By:

Aastha Kushwah (145)

Siddhi Prabhu (148)

Sneha Prabhu (149)

Aayesha Shindagi (151)

SEVA MANDAL EDUCATION SOCIETY'S

SMT.KAMLABEN GAMBHICHAND SHAH DEPARTMENT OF COMPUTER APPLICATIONS

UNDER

DR.BMN COLLEGE OF HOME SCIENCE (Autonomous)

338, R.A. KIDWAI ROAD MATUNGA, MUMBAI-19

CERTIFICATE

This is to certify that Ms. Aastha Kushwah, Ms. Siddhi Prabhu, Ms. Sneha Prabhu, Ms. Aayesha Shindagi has completed the project titled CAKE SHOP MANAGEMENT SYSTEM satisfactorily, and submitted the project report as per the guidelines of the S.N.D.T. Women's University, Mumbai.

Project Guide	**Head / Principal**
Signature:	**Signature:**
Name: Mr. Nitin Pawar	**Name:** Prof. Dr. Mala Pandurang
Date:	**Date:**
Organization Name and Seal:	**College Seal:**
Internal Examiner	**External Examiner**
Signature:	**Signature:**
Name:	**Name:**
Date:	**Date:**

BETTER BE BUTTER

This is to certify that **Ms. Aastha Kushwah, Ms. Siddhi Prabhu, Ms. Sneha Prabhu, Ms. Aayesha Shindagi** has satisfactorily completed the project titled **CAKE SHOP MANAGEMENT SYSTEM** under my guidance/supervision. The project work was carried during the period 01/01/2024 to 20/04/2024.

Signature of the Supervisor/Guide
Designation
Company Seal

ACKNOWLEDGEMENT

We would like to express our deep gratitude to a range of people who provided us valuable support in this project. Their enthusiastic guidance and encouragement have helped us in making this project a great success.

Our sincere sentiments of appreciation and gratitude to our Principal, Prof. Dr. Mala Pandurang for the approval of project.

We want to thank Mr. Shahjahan Khan, Head of the Department of Smt. Kamalaben Gambhichand Shah Department of Computer Applications.

Also, we would like to express our heartiest gratitude to our teachers Mr. Nitin Pawar and Ms. Manjot Kaur for their valuable suggestions and encouragement and also undivided and continuous support during tenure of this project.

We would like to thank Mr. Neffqriti Monteiro and Mr. Victor Monteiro for the reference and for taking the time from his busy schedule to provide us with necessary details about their cake shop business.

Finally, a vote of thanks to the department of Bachelor of Computer Applications, the staff, both teaching and non-teaching for their co-operation extended to us.

INDEX

INTRODUCTION

PROJECT TITLE:
Desktop Application for Cake Shop Management System.

OBJECTIVES:
The purpose of the Cake Shop Management System is to help the user to get overall analysis of his business and also provide an efficient order processing system.

DEVELOPMENT TOOLS AND TECHNOLOGY USED:
C# Windows Application in Cake Shop Management System is a Desktop based Windows Application which we have developed in C# .NET platform MySQL Database.

Language: C# .NET

Database: MySQL

Software’s: Visual Studio 2022, MySQL

ORGANIZATION PROFILE

Name of the shop: Better Be Butter

Owner: Neffriti Monteiro, Victor Monteiro.

Date Of establishment: 2023

Address: Kalyan

Better Be Butter is a venture by Neffriti Monteiro.

They specialise in popular Flavour Cakes, Special Occasion Cakes, Kids Cakes, Picture Cakes & Theme Fondant Cakes.
Better Be Butter constantly endeavours to make our ‘BIG’ & ‘SPECIAL’ occasion even more enjoyable by serving us to “BEST”.

Better Be Butter delivers cakes only in Kalyan.

They also provide a Chocolates, pastries, and Decorative items.

CURRENT SYSTEM AND DRAWBACKS

CURRENT SYSTEM:

Currently there is no software for managing order in cake shop. The daily order processing is carried out manually by the owner or staff.

There are many problems in current Manual system:

Shop Owner face many difficulties while analyzing the record of his business in day-to-day life. He has to do a lot of paper work for maintaining his information.

DRAWBACKS:

The analysis of the current Manual system really has exposed some problems are follows:

- The Current Manual System is time consuming.
- There is no Data Security.
- There is Data Redundancy Issue in current System.
- Preparation of reports is not an easy work.
- Maintaining information and retrieving information to our needs are limited.

PROPOSED SYSTEM

To overcome the problems of current manual system we are proposing a complete Automated system using C#.net as a frontend and SQL as a backend.

Characteristics of the proposed system:

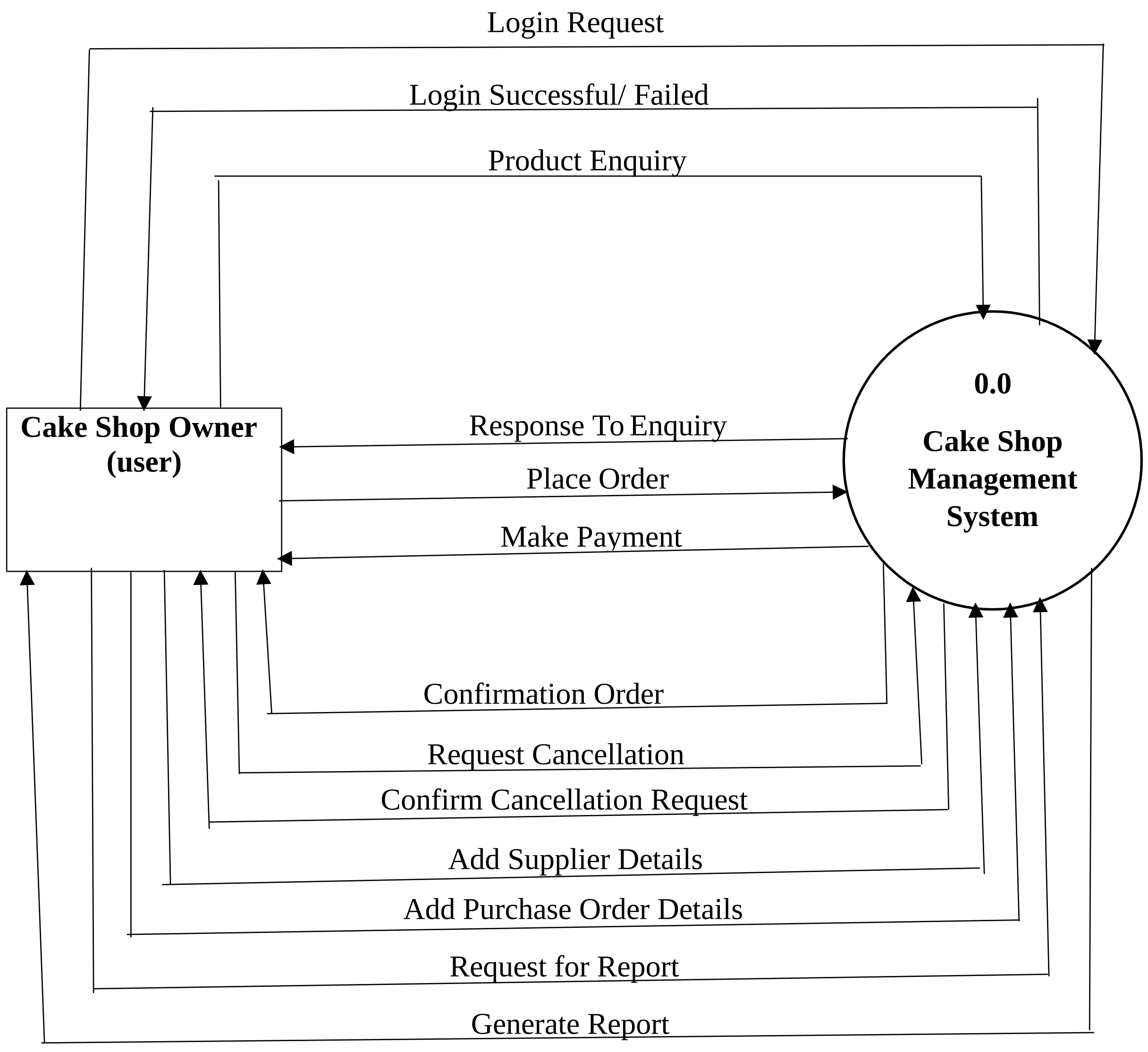
- **User Friendly:** The proposed system is user-friendly because the Retrieval and storing of data is fast and data is maintained efficiently.
- **Reports are Easily Generated:** Reports are easily generated in proposed system So, any type of reports can be generated in proposed system, which help the shop owner in decision making activity.
- **Computer Operator Control:** Computer Operator Control Will be there no errors. Moreover, storing and retrieving of information is easy.
- **No or Very Few paper work:** The proposed System either does not require paper work or a very few paper work is required. All the data is inserted into the computer immediately and bills and reports can be generated through computers. Since all the data kept in a database no data of the shop can be destroyed. Moreover, work becomes very easy because there is no need to keep data on papers.

ADVANTAGES OF THE PROPOSED SYSTEM:

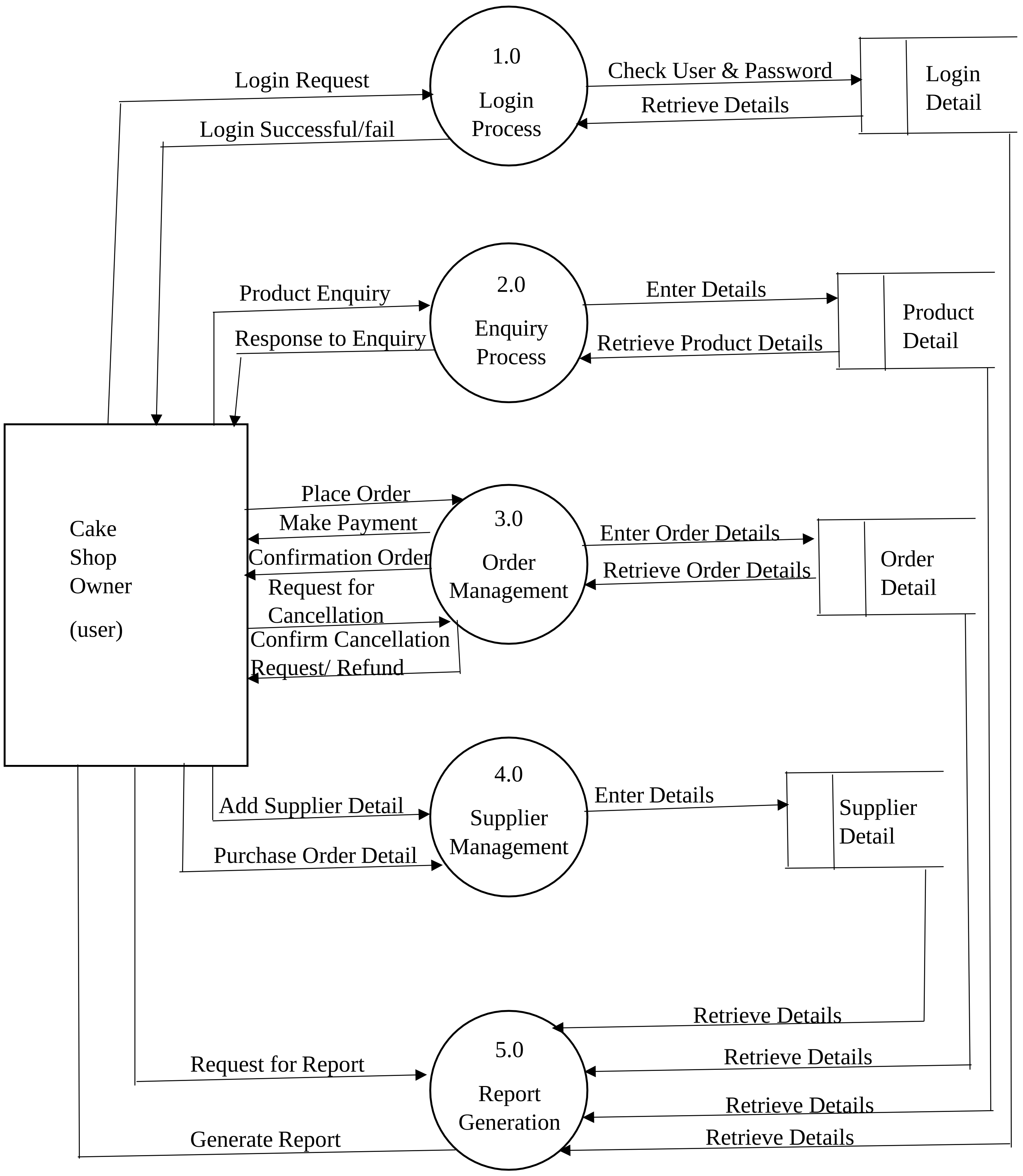
- Computerization will be helpful in reducing extra manpower.
- Data stored in computer is easily accessible than current manual system.
- Computerization make searching easy and instantaneous. Also, the results obtained are consistent.
- Proposed system will definitely reduce paper work and thus reduce possibility of human error.
- The graphical user interface makes the application more attractive and easily understandable to the user.

DATA FLOW DIGRAM

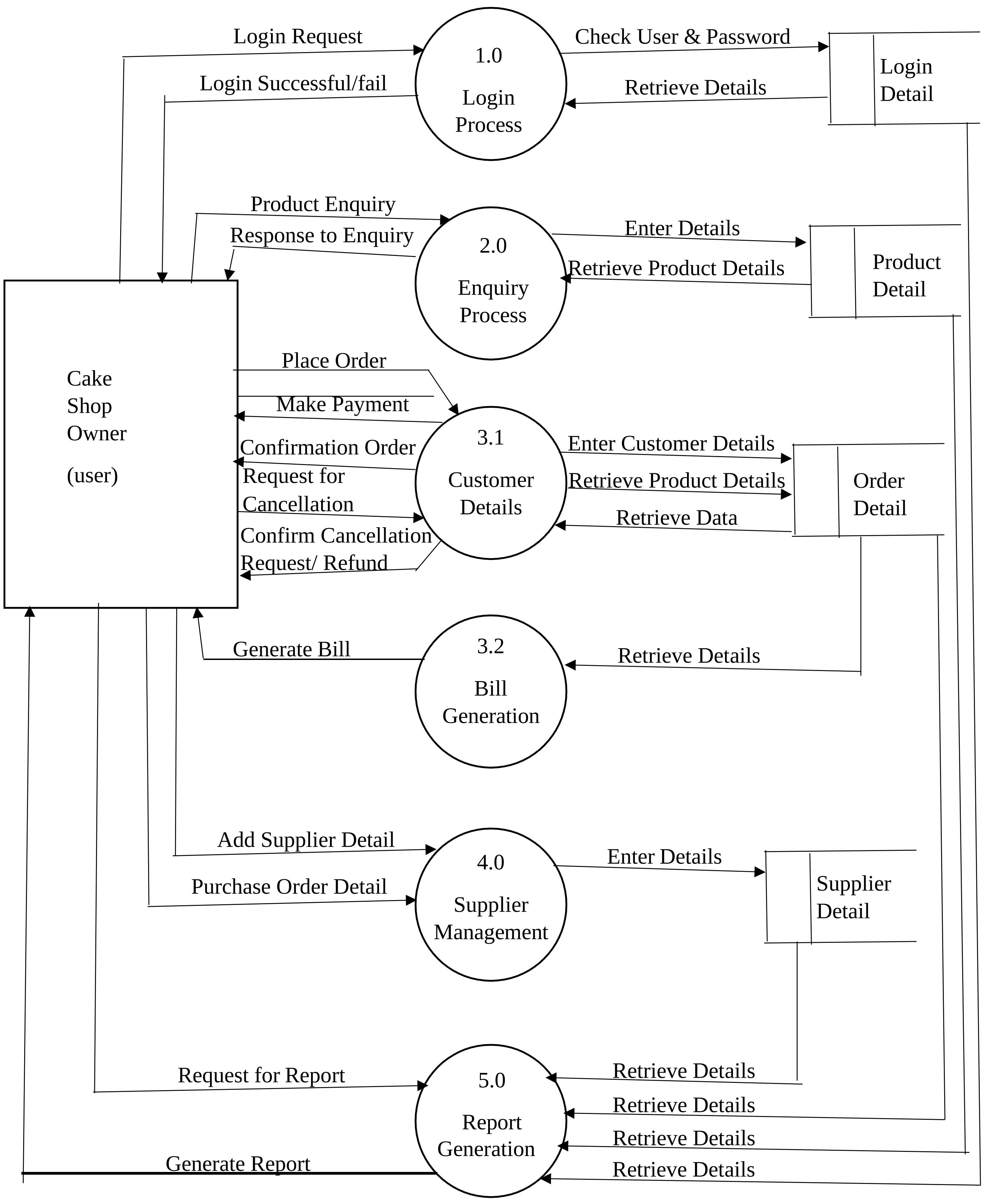
LEVEL 0


Login Request
Login Successful/ Failed
Product Enquiry
Cake Shop Owner (user)
Response To Enquiry
Place Order
Make Payment
0.0
Cake Shop Management System
Confirmation Order
Request Cancellation
Confirm Cancellation Request
Add Supplier Details
Add Purchase Order Details
Request for Report
Generate Report

LEVEL 1


1.0
Login Process
Login Request
Login Successful/fail
Check User & Password
Retrieve Details
Login Detail
2.0
Enquiry Process
Product Enquiry
Response to Enquiry
Enter Details
Retrieve Product Details
Product Detail
Cake Shop Owner (user)
3.0
Order Management
Place Order
Make Payment
Confirmation Order
Request for Cancellation
Confirm Cancellation Request/ Refund
Enter Order Details
Retrieve Order Details
Order Detail
4.0
Supplier Management
Add Supplier Detail
Purchase Order Detail
Enter Details
Supplier Detail
5.0
Report Generation
Request for Report
Generate Report
Retrieve Details
Retrieve Details
Retrieve Details
Retrieve Details

LEVEL 2


Login Request
Login Successful/fail
1.0
Login Process
Check User & Password
Retrieve Details
Login Detail
Product Enquiry
Response to Enquiry
2.0
Enquiry Process
Enter Details
Retrieve Product Details
Product Detail
Cake Shop Owner
(user)
Place Order
Make Payment
Confirmation Order
Request for Cancellation
Confirm Cancellation Request/ Refund
3.1
Customer Details
Enter Customer Details
Retrieve Product Details
Retrieve Data
Order Detail
Generate Bill
3.2
Bill Generation
Retrieve Details
Add Supplier Detail
Purchase Order Detail
4.0
Supplier Management
Enter Details
Supplier Detail
Request for Report
Generate Report
5.0
Report Generation
Retrieve Details
Retrieve Details
Retrieve Details
Retrieve Details

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Types of Normalization:

1NF: A relation is in 1NF if it contains an atomic value.

2NF: A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

3NF: A relation will be in 3NF if it is in 2NF and no transition dependency exists.

BCNF: A stronger definition of 3NF is known as Boyce Codd's normal form

4NF: A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.

5NF: A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Our Project Database is in the 1NF.

DATABASE DESIGN

Table name: login

It will use to store the login details of users.

Sr. no	Field Name	Field Type	Null/ not null	Description
1	Username	varchar(50)	Not null	It's a primary key, is used to store name of the user.
2	Password	varchar(255)	Not null	A password is a string of characters used to verify the identity of a User.

Table name: customers

It will store the customer Details.

Sr No.	Field Name	Field type	Null / not null	Description
1	Customerid	Bigint AI PK	Not null	It's use to store the customer's id.
2	CustomerName	varchar(50)	Not null	It's use to store the customer's name.
3	Contact	char(10)	null	It's use to store the customer's contact number.

4	EmailId	varchar(255)	null	It’s use to store the customer’s email-id.
5	Dob	date	null	It’s use to store the customer’s email-id.

Table name: order

It will store Order Details of Product’s.

Sr No.	Field Name	Field type	Null / notnull	Description
1	Category	varchar(15)	Not null	It’s used to store the category.
2	Item	varchar(50)	Not null	It’s used to store the Item
3	Order_id	int PK	Not null	It’s use to store the Order id.
4	Customer_name	varchar(10)	Not null	It’s use to store the Customer’s Name.
5	Price	int	Not null	It’s use to store the Product’s Price.
6	Total	int	Not null	It’s use to store the Total Amount
7	Subtotal	int	Null	It’s use to store the Subtotal Amount.
8	Discount	int	Null	It’s use to store the Discount of product
9	Net	int	Null	It’s use to store the Net Amount.
10	Paid	int	Not null	It’s use to store the Paid Amount
11	Balance	int	Not null	It’s use to store the balance.

12	Date	date	Not null	It is used to store date.

Table name: supplierdetails

It will use to store the details of suppliers.

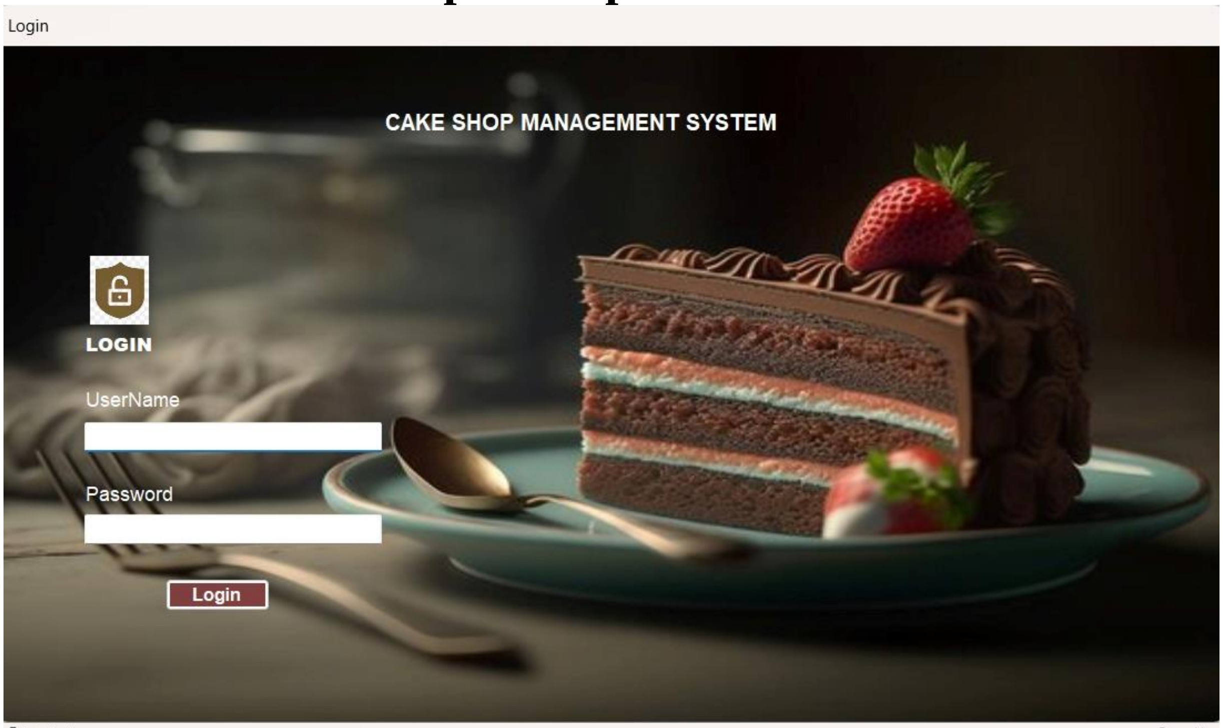
Sr. no	Field Name	Field Type	Null / not null	Description
1	Supplier_id	int AI PK	Not null	It's use to store the supplier's id.
2	Supplier_name	varchar(50)	Not null	It's use to store the supplier's name.
3	Contact	char(10)	Not null	It's use to store the supplier's contact.
4	Emailid	varchar(255)	Null	It's use to store the supplier's email-id.
5	Address	varchar(255)	Null	It's use to store the supplier's address.
6	Bank_name	varchar(50)	Null	It's use to store the supplier's
7	Account_number	varchar(50)	Not null	A unique identifier assigned to a bank account
8	IFSC_code	varchar(20)	Not null	An alphanumeric code used to identify a specific bank branch
9	Branch	varchar(50)	Not null	The name or location of the bank branch associated with the account

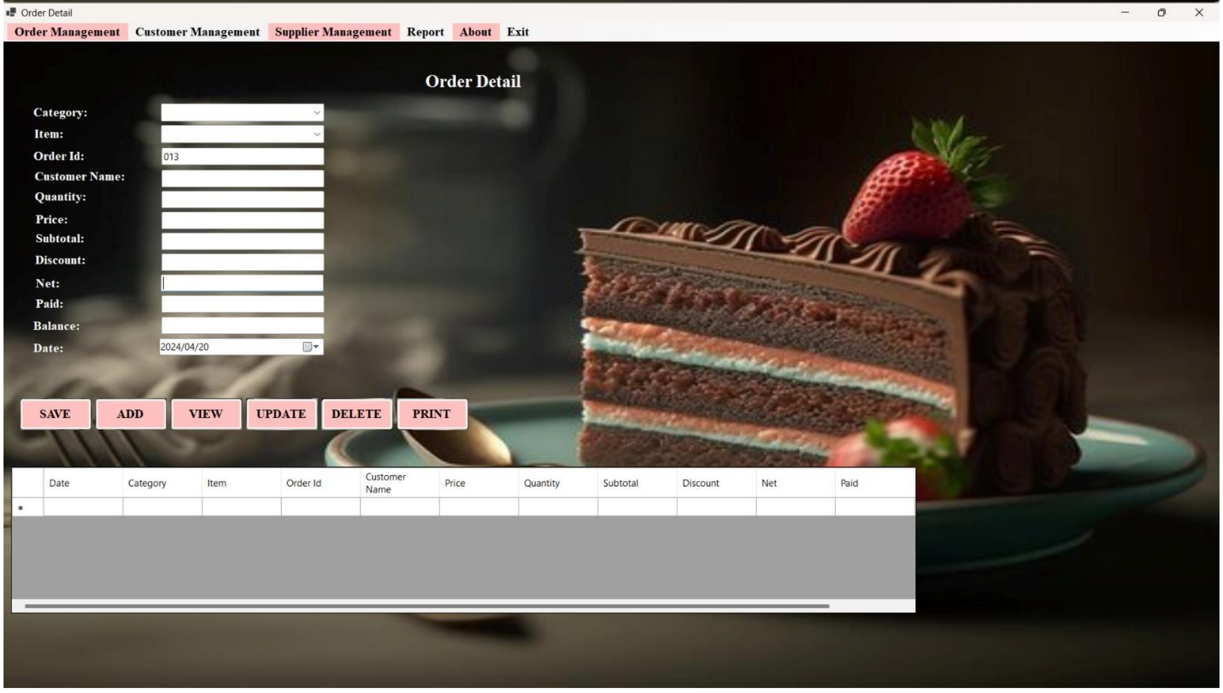
Table name: supplierproduct

It will store the product type, product id, product name which has been received from Suppliers.

Sr. no	Field Name	Field Type	Null / not null	Description
1	Supplier_id	int AI PK	Not null	It's use to Store the supplier's id.
2	Product_id	int	Not null	A unique numerical identifier for a product
3	Product_category	varchar(50)	Not null	A descriptive category that classifies a product
4	Subcategory	varchar(50)	Not null	A descriptive category that further classifies a product
5	Price	int	Not null	It's use to Store the Price of Products.
6	quantity	int	Not null	The quantity of a particular product
7	Date	date	Not null	It's use to store the the total amount.

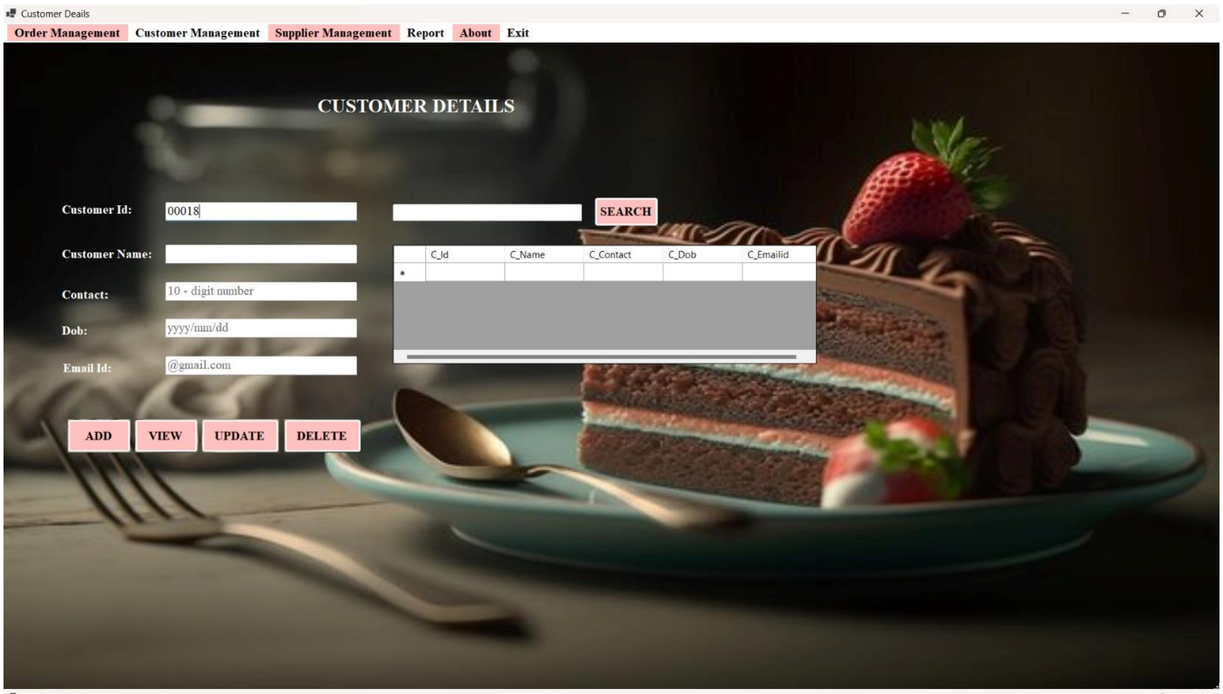
Input Output Screen





Category:	
Item:	
Order Id:	013
Customer Name:	
Quantity:	
Price:	
Subtotal:	
Discount:	
Net:	
Paid:	
Balance:	
Date:	2024/04/20

	Date	Category	Item	Order Id	Customer Name	Price	Quantity	Subtotal	Discount	Net	Paid
*											



Customer Deails

Order Management | Customer Management | Supplier Management | Report | About | Exit

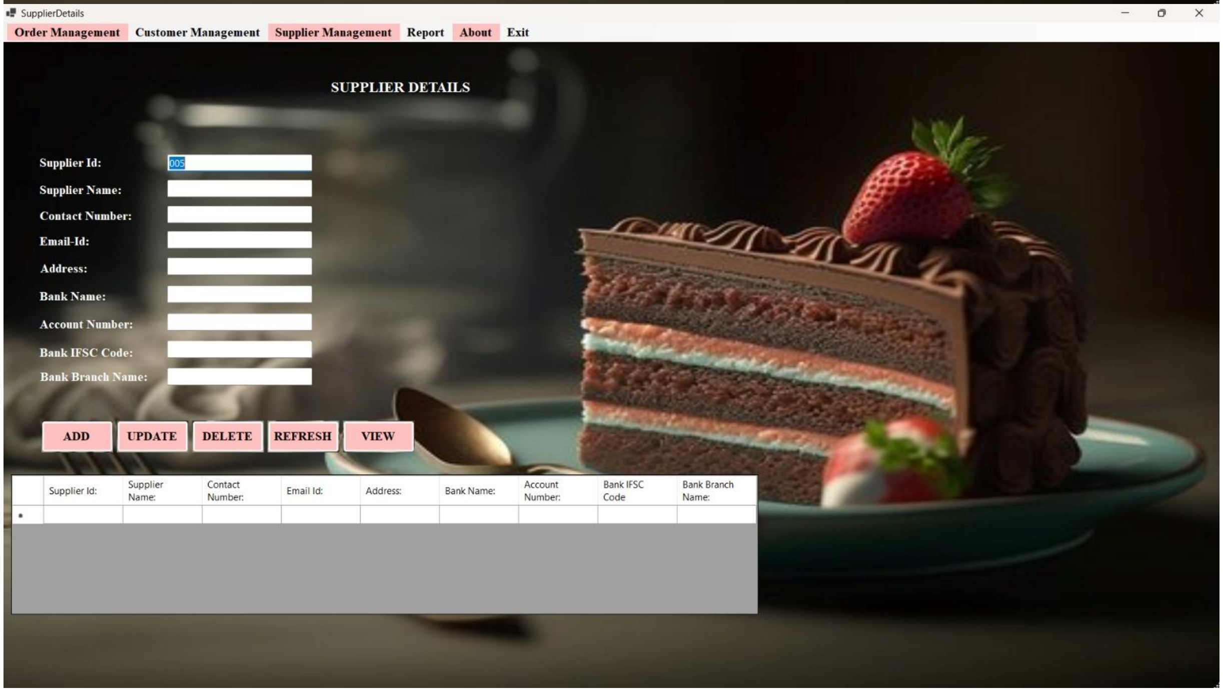
CUSTOMER DETAILS

Customer Id:	00018
Customer Name:	
Contact:	10 - digit number
Dob:	yyyy/mm/dd
Email Id:	@gmail.com

SEARCH

	C_Id	C_Name	C_Contact	C_Dob	C_Emailid
*					

ADD | VIEW | UPDATE | DELETE



SupplierDetails

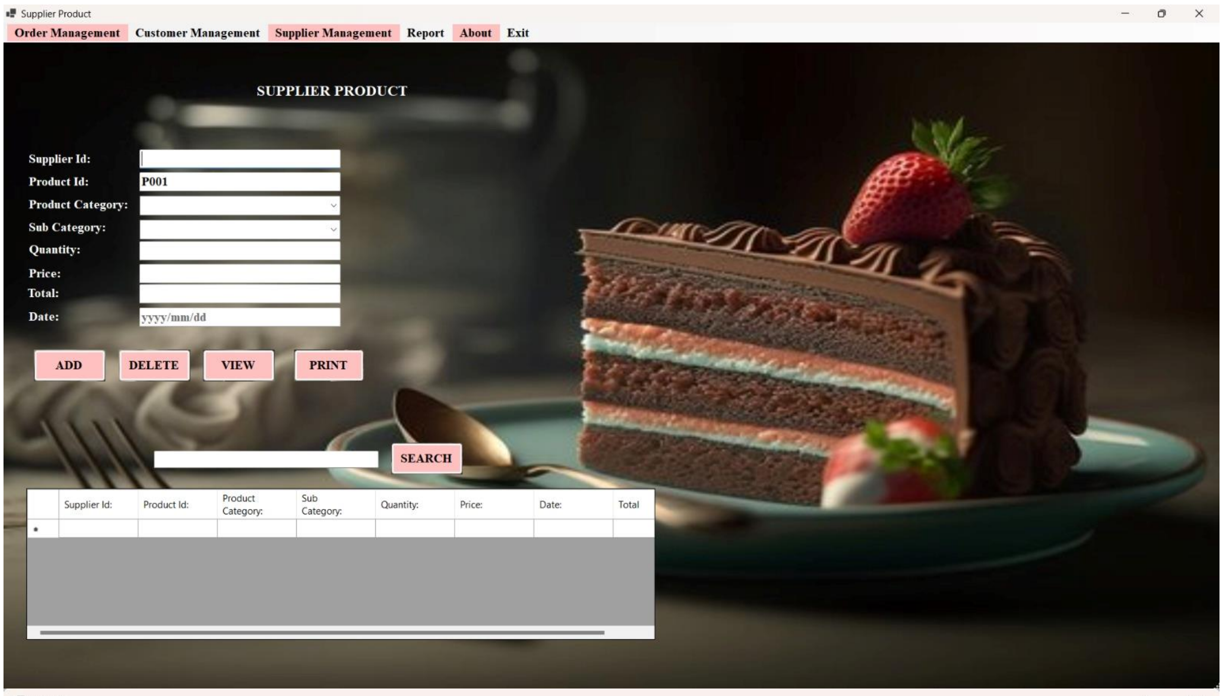
Order Management | Customer Management | Supplier Management | Report | About | Exit

SUPPLIER DETAILS

Supplier Id:	005
Supplier Name:	
Contact Number:	
Email-Id:	
Address:	
Bank Name:	
Account Number:	
Bank IFSC Code:	
Bank Branch Name:	

ADD | UPDATE | DELETE | REFRESH | VIEW

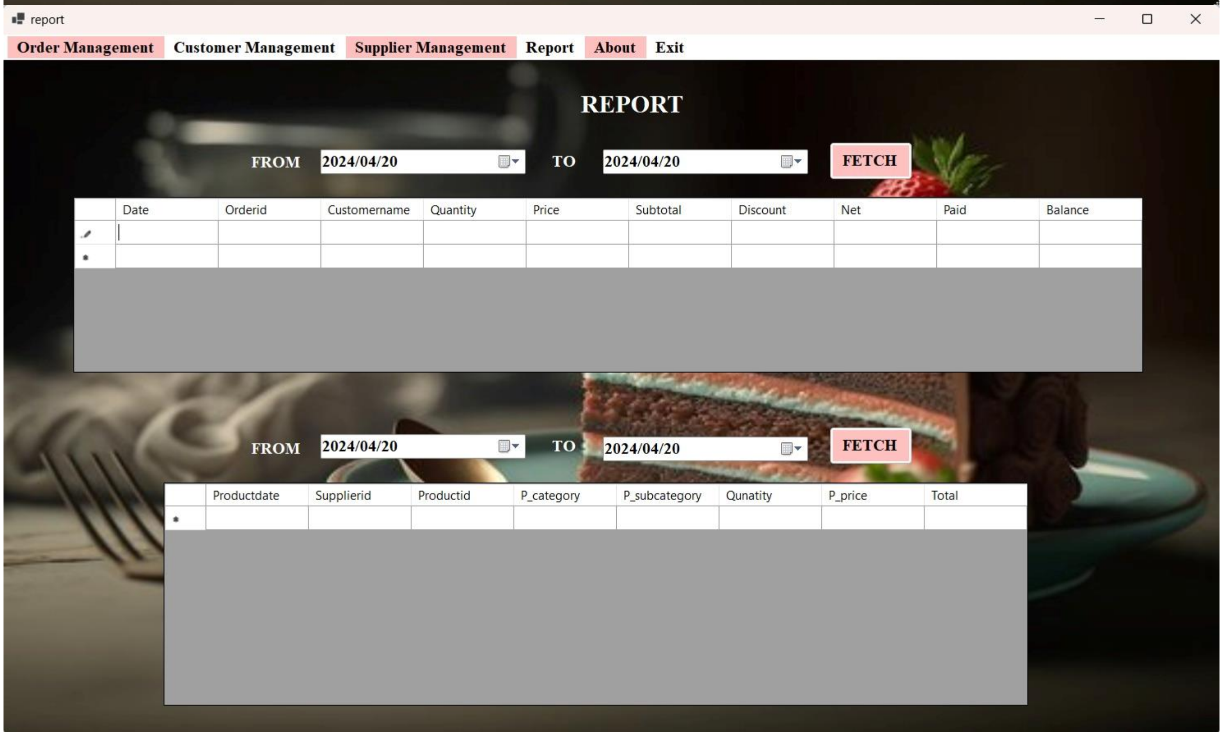
	Supplier Id:	Supplier Name:	Contact Number:	Email Id:	Address:	Bank Name:	Account Number:	Bank IFSC Code	Bank Branch Name:
*									



SUPPLIER PRODUCT

Supplier Id:	
Product Id:	P001
Product Category:	
Sub Category:	
Quantity:	
Price:	
Total:	
Date:	yyyy/mm/dd

	Supplier Id:	Product Id:	Product Category:	Sub Category:	Quantity:	Price:	Date:	Total
*								



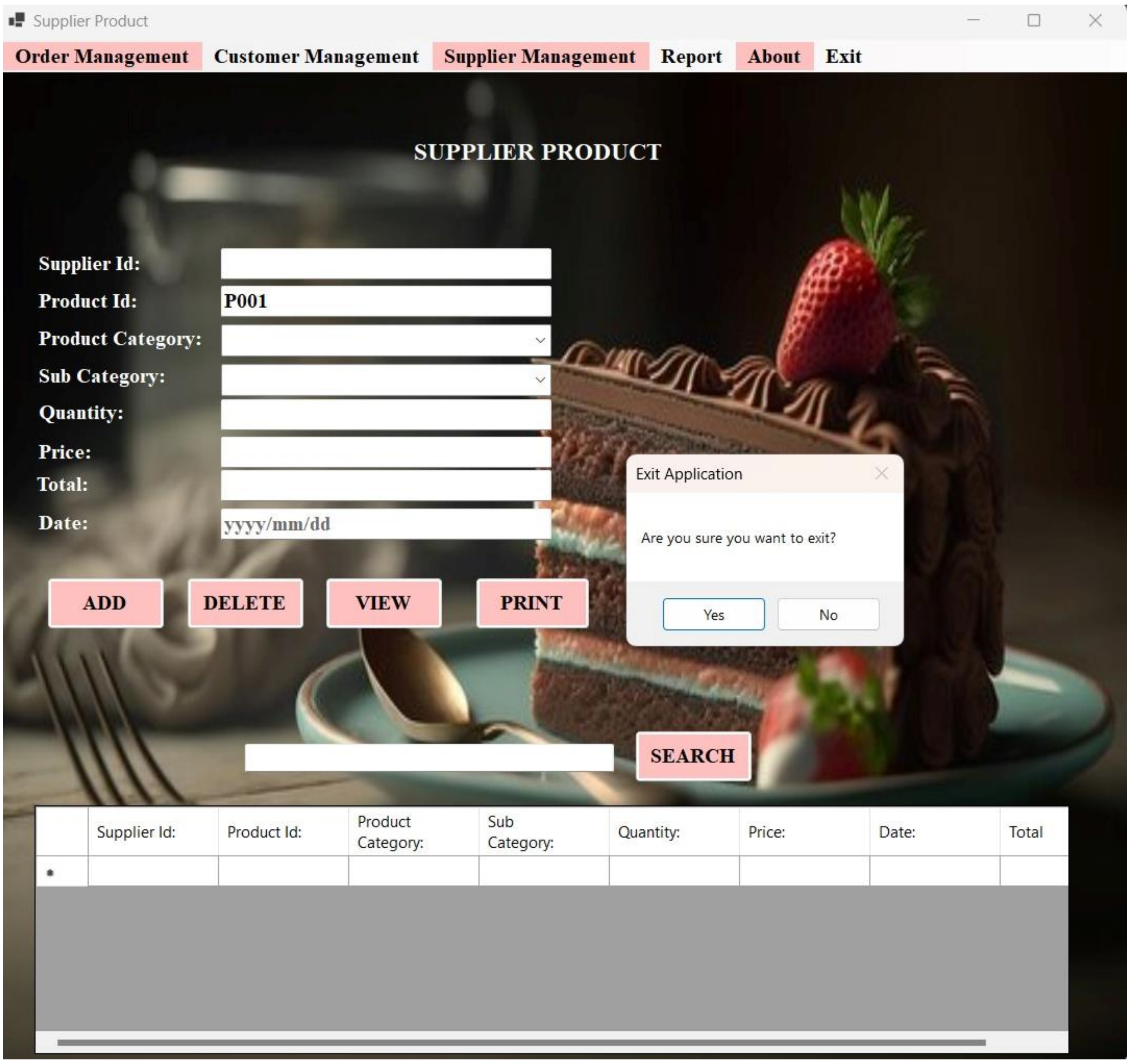
REPORT

FROM 2024/04/20 TO 2024/04/20

	Date	Orderid	Customername	Quantity	Price	Subtotal	Discount	Net	Paid	Balance
*										

FROM 2024/04/20 TO 2024/04/20

	Productdate	Supplierid	Productid	P_category	P_subcategory	Qunatity	P_price	Total
*								


Supplier Product

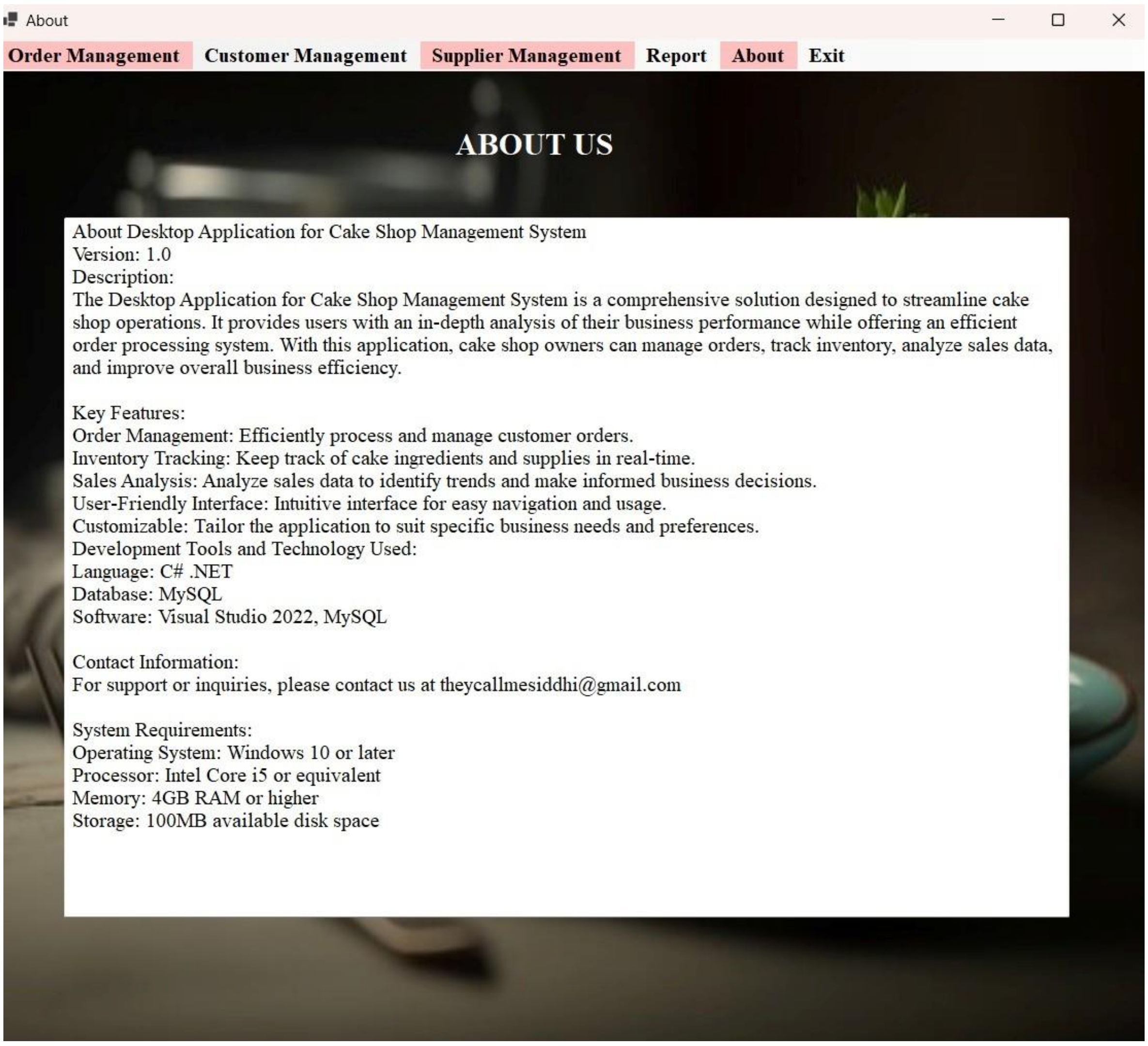
SUPPLIER PRODUCT

Supplier Id:	
Product Id:	P001
Product Category:	
Sub Category:	
Quantity:	
Price:	
Total:	
Date:	yyyy/mm/dd

Exit Application

Are you sure you want to exit?

	Supplier Id:	Product Id:	Product Category:	Sub Category:	Quantity:	Price:	Date:	Total
*								



ABOUT US

About Desktop Application for Cake Shop Management System
Version: 1.0
Description:
The Desktop Application for Cake Shop Management System is a comprehensive solution designed to streamline cake shop operations. It provides users with an in-depth analysis of their business performance while offering an efficient order processing system. With this application, cake shop owners can manage orders, track inventory, analyze sales data, and improve overall business efficiency.

Key Features:
Order Management: Efficiently process and manage customer orders.
Inventory Tracking: Keep track of cake ingredients and supplies in real-time.
Sales Analysis: Analyze sales data to identify trends and make informed business decisions.
User-Friendly Interface: Intuitive interface for easy navigation and usage.
Customizable: Tailor the application to suit specific business needs and preferences.
Development Tools and Technology Used:
Language: C# .NET
Database: MySQL
Software: Visual Studio 2022, MySQL

Contact Information:
For support or inquiries, please contact us at theycallmesiddhi@gmail.com

System Requirements:
Operating System: Windows 10 or later
Processor: Intel Core i5 or equivalent
Memory: 4GB RAM or higher
Storage: 100MB available disk space

CODING

LOGIN

```
using MySql.Data.MySqlClient;
using BCrypt.Net;
using Mysqlx.Crud;
using Org.BouncyCastle.Crypto.Generators;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace Betterbebutter15
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }
        string sql;
        MySqlDataReader read;
        string str = "server=localhost; uid=root; pwd=Aastha@1978;
database=betterbebutter15";
        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {


        }

        private void LoginButton_Click(object sender, EventArgs e)
        {
            using (MySqlConnection conn = new MySqlConnection(str))
            {
                conn.Open();

                MySqlCommand cmd = conn.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "SELECT COUNT(*) FROM users WHERE username =
@username AND password = @password";

                // Using parameters to prevent SQL injection
                cmd.Parameters.AddWithValue("@username", UserNameTxt.Text);
                cmd.Parameters.AddWithValue("@password", PasswordTxt.Text);

                int userCount = Convert.ToInt32(cmd.ExecuteScalar());

                if (UserNameTxt.Text == "Staff" && PasswordTxt.Text == "123")
                {
                    new orderdetail().Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Invalid username or password. Please try again.");
                    UserNameTxt.Clear();
                    PasswordTxt.Clear();
                    UserNameTxt.Focus();
                }

                conn.Close();
            }
        }



        private void Form1_Load_1(object sender, EventArgs e)
        {

        }

        private void UserNameTxt_TextChanged(object sender, EventArgs e)
        {
        }
    }
}
```

CUSTOMER DETAIL

```
using MySql.Data.MySqlClient;
using Mysqlx.Crud;
using PdfSharp.Diagnostics;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;


namespace Betterbebutter15
{

    public partial class CustomerDetail : Form
    {
        private static int nextCustomerid = 1;
        public CustomerDetail()
        {
            InitializeComponent();
            Dob.Validating += Dob_Validating;
            contact.Validating += contact_Validating;
            Customerid.Text = GenerateUniqueCustomerNumber();
            Customerid = Customerid;

            button1.TextChanged += button1_TextChanged;
        }
        string sql;
        MySqlDataReader read;
        string str = "server=localhost; uid=root; pwd=Aastha@1978;
database=betterbebutter15";

        private void ADD_Click(object sender, EventArgs e)

        {
            if (string.IsNullOrWhiteSpace(Customerid.Text) ||
            string.IsNullOrWhiteSpace(contact.Text) ||
            string.IsNullOrWhiteSpace(Dob.Text))
            {
                MessageBox.Show("Please fill in all required fields.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            string customerid = GenerateUniqueCustomerNumber();
            string sql = "INSERT INTO Customers(Customerid, CustomerName, contact, Dob,
EmailId) VALUES(@Customerid, @CustomerName, @contact, @Dob, @EmailId)";

            using (MySqlConnection conn = new MySqlConnection(str))
            {
                try
                {
                    conn.Open();
                    MySqlCommand cmd = new MySqlCommand(sql, conn);
                    cmd.Parameters.AddWithValue("@Customerid", customerid);
                    cmd.Parameters.AddWithValue("@CustomerName", customername.Text);
                    cmd.Parameters.AddWithValue("@contact", contact.Text);
                    cmd.Parameters.AddWithValue("@Dob", Dob.Text);
                    cmd.Parameters.AddWithValue("@EmailId", Emailid.Text);

                    int rowsAffected = cmd.ExecuteNonQuery();

                    if (rowsAffected > 0)
                    {
                        MessageBox.Show("Record added successfully");

                        customername.Clear();
                        contact.Clear();
                        Dob.Clear();
                        Emailid.Clear();
                        Customerid.Text = GenerateUniqueCustomerNumber();
                    }
                    else
                    {
                        MessageBox.Show("Failed to add record");
                    }
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error adding record: {ex.Message}");
        }
    }


}

private void VIEW_Click(object sender, EventArgs e)
{
    MySqlConnection conn = new MySqlConnection(str);
    conn.ConnectionString = str;
    conn.Open();
    sql = "SELECT * FROM customers";
    MySqlCommand cmd = new MySqlCommand(sql, conn);
    read = cmd.ExecuteReader();
    dataGridView1.Rows.Clear();
    while (read.Read())
    {
        dataGridView1.Rows.Add(read[0], read[1], read[2], read[3], read[4]);
    }
    conn.Close();


}

private void UPDATE_Click(object sender, EventArgs e)
{
        }

            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();

            string sql = "UPDATE CustomerS SET CustomerName = @CustomerName WHERE
Customerid = @Customerid";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@Customerid", Customerid.Text);
            cmd.Parameters.AddWithValue("@Customername", customername.Text);

            MessageBox.Show("Record updated");
            cmd.ExecuteNonQuery();
            VIEW_Click(sender, e);
            ClearTextBoxes();
        }

        private void ClearTextBoxes()
        {
            Customerid.Clear();
            customername.Clear();
            contact.Clear();
            Dob.Clear();
            Emailid.Clear();

        }
        private bool ValidateFields()
```

```
{
    // Validate Contact
    if (string.IsNullOrWhiteSpace(contact.Text))
    {
        MessageBox.Show("Contact cannot be empty");
        return false;
    }

    // Validate Dob
    if (string.IsNullOrWhiteSpace(Dob.Text))
    {
        MessageBox.Show("Dob cannot be empty");
        return false;
    }

    return true;
}



private void gridView()
{
}

private void Customerid_TextChanged(object sender, EventArgs e)
{


}
private void customerToolStripMenuItem_Click(object sender, EventArgs e)
{
    CustomerDetail form = new CustomerDetail();
    form.ShowDialog();
}
private void Customer_Load(object sender, EventArgs e)
{

}

private string GenerateUniqueCustomerNumber()
{
    string nextCustomerId = ""; // Default value
    MySqlConnection conn = new MySqlConnection(str);

    try
    {
        conn.Open();

        // Query to get the maximum customer ID from the "Customers" table
        string sql = "SELECT MAX(Customerid) FROM Customers";
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        object result = cmd.ExecuteScalar();

        if (result != null && result != DBNull.Value)
        {
            string lastCustomerId = result.ToString();
            int lastIdNumericPart;
            if (int.TryParse(lastCustomerId, out lastIdNumericPart))
            {
                int nextId = lastIdNumericPart + 1;

                // Ensure the next ID has only 5 digits
```

```
                    if (nextId < 100000)
                    {
                        nextCustomerId = nextId.ToString("D5");
                    }
                    else
                    {
                        MessageBox.Show("Maximum customer ID limit reached.");
                    }
                }
                else
                {
                    // If the retrieved value is not a valid integer, handle it here
                    // For example, you can generate a new ID starting from a default value
                    nextCustomerId = "00001";
                    MessageBox.Show("The last customer ID retrieved is not a valid integer.
Generating new ID starting from default value.");
                }
            }
            else
            {
                // If no previous customer IDs exist, start with 00001
                nextCustomerId = "00001";
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error generating next customer ID: " + ex.Message);
        }
        finally
        {
            conn.Close();
```

```
    }

    return nextCustomerId;
}




private void Address_Leave(object sender, EventArgs e)
{

}

private void productToolStripMenuItem_Click(object sender, EventArgs e)
{



    orderdetail form = new orderdetail();
    form.ShowDialog();

}

private void contact_Click(object sender, EventArgs e)
{

}

private void Dob_Validating(object sender, CancelEventArgs e)
{
    // Trim the input to remove leading or trailing whitespaces
    string inputDob = Dob.Text.Trim();
    // Check if the input is empty
    if (string.IsNullOrWhiteSpace(inputDob))
    {
        // Set an error message for an empty Dob
        errorProvider1.SetError(Dob, "Dob cannot be empty");
        e.Cancel = true; // Cancel the event to prevent the focus from changing
    }
    else
    {
        // Clear any existing error message
        errorProvider1.SetError(Dob, string.Empty);
    }
}
private void contact_Validating(object sender, CancelEventArgs e)
{
    // Trim the input to remove leading or trailing whitespaces
    string inputcontact = contact.Text.Trim();

    // Check if the input is empty
    if (string.IsNullOrWhiteSpace(inputcontact))
    {
        // Set an error message for an empty contact
        errorProvider1.SetError(contact, "contact cannot be empty");
        e.Cancel = true; // Cancel the event to prevent the focus from changing
    }
    else
    {
        // Clear any existing error message
        errorProvider1.SetError(contact, string.Empty);
    }
}

private void contact_Leave_1(object sender, EventArgs e)
{
    // Trim the input to remove leading or trailing whitespaces
    string inputcontact = contact.Text.Trim();

    // Check if the input is empty
    if (string.IsNullOrEmpty(inputcontact))
    {
        MessageBox.Show("Mobile Number cannot be empty");
        return;
    }

    // Use a verbatim string (@) to avoid escaping characters
    Regex ex = new Regex(@"^\+?[0-9\s-]+$");
    bool isValid = ex.IsMatch(inputcontact);
    string digitsOnly = new string(inputcontact.Where(char.IsDigit).ToArray());



    if (!isValid || digitsOnly.Length != 10)
    {
        MessageBox.Show("Please Enter a Valid Mobile Number");
    }

}

private void Emailid_Leave_1(object sender, EventArgs e)
{
    // Trim the input to remove leading or trailing whitespaces
    string inputEmail = Emailid.Text.Trim();

    // Check if the input is empty
    if (string.IsNullOrEmpty(inputEmail))
    {
        MessageBox.Show("Email cannot be empty");
        return;
    }

    // Use a regular expression for email validation
    string emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
    Regex emailRegex = new Regex(emailPattern);
    bool isValidEmail = emailRegex.IsMatch(inputEmail);

    if (!isValidEmail)
    {
        MessageBox.Show("Please Enter a Valid Email Address");
    }
    // No else part, so no message will be shown when the email is valid


}

private void customername_Leave(object sender, EventArgs e)
{
    // Trim the input to remove leading or trailing whitespaces
    string inputName = customername.Text.Trim();
```

```
            // Check if the input is empty
            if (string.IsNullOrEmpty(inputName))
            {
                MessageBox.Show("Fill the Valid Name");
                return;
            }

            // Use a regular expression to allow only letters and spaces
            string namePattern = "^[a-zA-Z]+( [a-zA-Z]+)*$";
            Regex nameRegex = new Regex(namePattern);
            bool isValidName = nameRegex.IsMatch(inputName);

            if (!isValidName)
            {
                MessageBox.Show("Please Enter a Valid Name (only letters with spaces
allowed)");
            }
        }

        private void customerToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            CustomerDetail form = new CustomerDetail();
            form.ShowDialog();
        }

        private void DELETE_Click(object sender, EventArgs e)
        {
            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();
```

```
    string sql = "DELETE FROM Customers WHERE Customerid = @Customerid";
    MySqlCommand cmd = new MySqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@Customerid", Customerid.Text);

    MessageBox.Show("Record deleted");
    cmd.ExecuteNonQuery();
    VIEW_Click(sender, e);

}

private void REFRESH_Click(object sender, EventArgs e)
{

}

private void menuStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
{

}

private void Emailid_TextChanged(object sender, EventArgs e)
{

}




private void Dob_TextChanged(object sender, EventArgs e)
{
}

private void supplierDetailToolStripMenuItem_Click(object sender, EventArgs e)
{
   SupplierDetails form = new SupplierDetails();
   form.ShowDialog();
}

private void supplierProductToolStripMenuItem_Click_1(object sender, EventArgs e)
{
   SupplierProduct form = new SupplierProduct();
   form.ShowDialog();
}

private void reportToolStripMenuItem_Click(object sender, EventArgs e)
{

}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{

}
```

```
    private void exitToolStripMenuItem_Click_1(object sender, EventArgs e)
    {
        // ExitApplication();
    }



    private void menuStrip1_ItemClicked_1(object sender, ToolStripItemClickedEventArgs
e)
    {


    }

    private void orderManagementToolStripMenuItem_Click(object sender, EventArgs e)
    {
        orderdetail form = new orderdetail();
        form.ShowDialog();
    }

    private void customerManagementToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        CustomerDetail form = new CustomerDetail();
        form.ShowDialog();
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {

        ExitApplication();
    }
```

```
        private void ExitApplication()
        {



            DialogResult result = MessageBox.Show("Are you sure you want to exit?", "Exit
Application", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                Application.Exit();
            }

        }

        private void button1_Click(object sender, EventArgs e)
        {
            string searchText = textBox1.Text.Trim();

            if (string.IsNullOrEmpty(searchText))
            {
                MessageBox.Show("Please enter a search term.", "Search Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            string connectionString =
"server=localhost;uid=root;pwd=Aastha@1978;database=betterbebutter15";
            using (MySqlConnection conn = new MySqlConnection(connectionString))
            {
                try
                {
                    conn.Open();
                    string query = "SELECT * FROM Customers WHERE CustomerName LIKE
@searchText";
                    MySqlCommand cmd = new MySqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@searchText", "%" + searchText + "%");

                    MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);
                    DataTable dt = new DataTable();
                    adapter.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
                    {
                        MessageBox.Show("No records found.", "Search Result",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        dataGridView1.DataSource = null; // Clear the DataGridView
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error searching data: " + ex.Message, "Search Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        private void button1_TextChanged(object sender, EventArgs e)
        {
        }
        private void reportToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            report Form = new report();
            Form.ShowDialog();
        }

        private void aboutToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            About Form = new About();
            Form.ShowDialog();
        }

        private void supplierManagementToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void supplierDetailsToolStripMenuItem_Click(object sender, EventArgs e)
        {

            SupplierDetails form = new SupplierDetails();
            form.ShowDialog();
        }

        private void supplierProductToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SupplierProduct form = new SupplierProduct();
            form.ShowDialog();
```

```
        }

        private void customername_Validating(object sender, CancelEventArgs e)
        {
            string inputName = customername.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputName))
            {
                MessageBox.Show("Customer name cannot be empty", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                e.Cancel = true; // Cancel the event to prevent the focus from changing
                return;
            }

            // Use a regular expression to allow only letters and spaces
            string namePattern = "^[a-zA-Z]+( [a-zA-Z]+)*$";
            Regex nameRegex = new Regex(namePattern);
            bool isValidName = nameRegex.IsMatch(inputName);

            if (!isValidName)
            {
                MessageBox.Show("Please enter a valid name (only letters with spaces allowed)",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                e.Cancel = true; // Cancel the event to prevent the focus from changing
            }
        }
    }
```

```
}
```

ORDER DETAIL

```
using MySql.Data.MySqlClient;
using Mysqlx.Crud;
using PdfSharp.Pdf;
using PdfSharp.Drawing;
using PdfSharp.Fonts;
using PdfSharp.Fonts.OpenType;
using System.IO;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Drawing.Printing;

namespace Betterbebutter15
{
    public partial class orderdetail : Form

    {
```

```
private static int nextOrderNumber = 1;
private DataTable? orderItems;
private Dictionary<string, decimal> itemPrices;
public orderdetail()
{
    InitializeComponent();

    OrderIdTxt.Text = GenerateOrderId();



    itemPrices = new Dictionary<string, decimal>();
    itemPrices.Add("Chocolate Cake", 400.00m);
    itemPrices.Add("Vanilla Cake", 300.00m);
    itemPrices.Add("Strawberry Cake", 350.00m);
    itemPrices.Add("Red Velvet Cake", 450.00m);
    itemPrices.Add("Black Forest Cake", 400.00m);
    itemPrices.Add("Tiramisu Cake", 500.00m);
    itemPrices.Add("Cheesecake", 500.00m);
    itemPrices.Add("Mousse Cake", 450.00m);
    itemPrices.Add("Pineapple Cake", 400.00m);
    itemPrices.Add("Coffee Cake", 500.00m);

    itemPrices.Add("Croissant", 60.00m);
    itemPrices.Add("Danish Pastry", 50.00m);
    itemPrices.Add("Éclair", 40.00m);
    itemPrices.Add("Cinnamon Roll", 40.00m);
    itemPrices.Add("Cream Puff", 30.00m);
    itemPrices.Add("Palmier", 40.00m);
    itemPrices.Add("Mille-Feuille", 60.00m);
    itemPrices.Add("Macaron", 30.00m);
itemPrices.Add("Carrot Cake", 400.00m);
itemPrices.Add("Lemon Cake", 400.00m);
itemPrices.Add("Zucchini Cake", 450.00m);
itemPrices.Add("Banana Cake", 350.00m);
itemPrices.Add("Almond Flour Cake", 440.00m);
itemPrices.Add("Coconut Flour Cake", 430.00m);
itemPrices.Add("Vegan Chocolate Cake", 460.00m);
itemPrices.Add("Gluten-Free Cake", 450.00m);
itemPrices.Add("Sugar-Free Cake", 440.00m);
itemPrices.Add("Whole Wheat Cake", 420.00m);
itemPrices.Add("Chia Seed Cake", 430.00m);
itemPrices.Add("Quinoa Cake", 440.00m);
itemPrices.Add("Avocado Cake", 420.00m);

itemPrices.Add("Edible Flowers", 150.00m);
itemPrices.Add("Sugar Sprinkles", 70.00m);
itemPrices.Add("Fondant Decorations", 160.00m);
itemPrices.Add("Edible Gold Leaf", 200.00m);
itemPrices.Add("Balloons", 50.00m);
itemPrices.Add("Birthday Caps", 40.00m);
itemPrices.Add("Confetti", 60.00m);
itemPrices.Add("Party Hats", 40.00m);
itemPrices.Add("Candles", 50.00m);
itemPrices.Add("Cake Toppers", 100.00m);
itemPrices.Add("Glitter", 90.00m);
itemPrices.Add("Ribbons", 70.00m);
itemPrices.Add("Banners", 140.00m);
itemPrices.Add("Garlands", 130.00m);
itemPrices.Add("Table Centerpieces", 180.00m);
```

```
// Populate ItemComboBox with items
foreach (var item in itemPrices.Keys)
{
   ItemComboBox.Items.Add(item);
}
ItemComboBox.SelectedIndexChanged += ItemComboBox_SelectedIndexChanged;

orderItems = new DataTable();
orderItems.Columns.Add("Category");
orderItems.Columns.Add("Item");
//orderItems.Columns.Add("Order_id");
orderItems.Columns.Add("Customer_name");
orderItems.Columns.Add("Price", typeof(decimal));
orderItems.Columns.Add("Quantity", typeof(int));
orderItems.Columns.Add("Subtotal", typeof(decimal));
orderItems.Columns.Add("Discount", typeof(decimal));
orderItems.Columns.Add("Net", typeof(decimal));
orderItems.Columns.Add("Paid", typeof(decimal));
orderItems.Columns.Add("Balance", typeof(decimal));
orderItems.Columns.Add("Date", typeof(DateTime));

CategoryComboBox.DropDownStyle = ComboBoxStyle.DropDownList;
ItemComboBox.DropDownStyle = ComboBoxStyle.DropDownList;



CategoryComboBox.Validating += CategoryComboBox_Validating;
ItemComboBox.Validating += ItemComboBox_Validating;

DiscountTxt.TextChanged += DiscountTxt_TextChanged;
```

```
    PaidTxt.TextChanged += PaidTxt_TextChanged;
    PriceTxt.TextChanged += PriceTxt_TextChanged;
    QuantityTxt.TextChanged += QuantityTxt_TextChanged;

    QuantityTxt.KeyPress += QuantityTxt_KeyPress;
    PriceTxt.KeyPress += PriceTxt_KeyPress;
    DiscountTxt.KeyPress += DiscountTxt_KeyPress;
    PaidTxt.KeyPress += PaidTxt_KeyPress;

    QuantityTxt.Validating += QuantityTxt_Validating;
    PriceTxt.Validating += PriceTxt_Validating;


}



private string GenerateOrderId()
{
    string nextOrderNumber = ""; // Default value
    MySqlConnection conn = new MySqlConnection(str);

    try
    {
        conn.Open();

        string sql = "SELECT MAX(order_id) FROM Order"; // Corrected table name
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        object result = cmd.ExecuteScalar();
    if (result != null && result != DBNull.Value)
    {
        int lastOrderId = Convert.ToInt32(result);
        int nextId = lastOrderId + 1;

        // Ensure the next ID has only 3 digits
        if (nextId < 1000)
        {
            nextOrderNumber = nextId.ToString("D3");
        }
        else
        {
            MessageBox.Show("Maximum order ID limit reached.");
        }
    }
    else
    {
        // If no previous order IDs exist, start with 001
        nextOrderNumber = "001";
    }
}
catch (Exception ex)
{
    MessageBox.Show("Error generating next order ID: " + ex.Message);
}
finally
{
    conn.Close();
}
```

```
            return nextOrderNumber;
        }

        string sql;
        MySqlDataReader read;
        string str = "server=localhost; uid=root; pwd=Aastha@1978;
database=betterbebutter15";

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void DeleteButton_Click(object sender, EventArgs e)
        {


            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();

            string sql = "DELETE FROM Order WHERE Order_id = @Order_id";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@Order_id", OrderIdTxt.Text);

            MessageBox.Show("Record deleted");
            cmd.ExecuteNonQuery();
            ViewButton_Click(sender, e);


        }
        private void supplierIToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void AddButton_Click(object sender, EventArgs e)
        {



            if (!orderItems.Columns.Contains("Order_id"))
            {
                orderItems.Columns.Add("Order_id");
            }



            DataRow newRow = orderItems.NewRow();


            newRow["Category"] = CategoryComboBox.Text;
            newRow["Item"] = ItemComboBox.Text;
            newRow["Order_id"] = OrderIdTxt.Text;
            newRow["Customer_name"] = CustomerNameTxt.Text;
            newRow["Price"] = decimal.Parse(PriceTxt.Text); // Assuming PriceTxt.Text is in
decimal format
            newRow["Quantity"] = int.Parse(QuantityTxt.Text); // Assuming QuantityTxt.Text is
in integer format
            newRow["Subtotal"] = decimal.Parse(SubtotalTxt.Text);
            newRow["Discount"] = decimal.Parse(DiscountTxt.Text);
            newRow["Net"] = decimal.Parse(NetTxt.Text);
            newRow["Paid"] = decimal.Parse(PaidTxt.Text);
            newRow["Balance"] = decimal.Parse(BalanceTxt.Text);
```

```
    newRow["Date"] = date.Value;

    orderItems.Rows.Add(newRow);

    MessageBox.Show("Item added to order.");

    // OrderIdTxt.Clear();
    CustomerNameTxt.Clear();
    PriceTxt.Clear();
    QuantityTxt.Clear();
    SubtotalTxt.Clear();
    DiscountTxt.Clear();
    NetTxt.Clear();
    PaidTxt.Clear();
    BalanceTxt.Clear();

    CategoryComboBox.SelectedIndex = -1;
    ItemComboBox.SelectedIndex = -1;

}



private void ViewButton_Click(object sender, EventArgs e)
{
    MySqlConnection conn = new MySqlConnection(str);
    conn.ConnectionString = str;
    conn.Open();
    sql = "SELECT * FROM Order";
    MySqlCommand cmd = new MySqlCommand(sql, conn);
```

```
            read = cmd.ExecuteReader();
            OrderDetailDataGridView.Rows.Clear();
            while (read.Read())
            {
                OrderDetailDataGridView.Rows.Add(read[0], read[1], read[2], read[3], read[4],
read[5], read[6], read[7], read[8], read[9], read[10], read[11]);
            }
            conn.Close();


        }


        private void UpdateButton_Click(object sender, EventArgs e)
        {
            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();


            string sql = "UPDATE  SET Customer_name = @Custome_name WHERE Order_id
= @Order_id";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@Order_id", OrderIdTxt.Text);
            cmd.Parameters.AddWithValue("@Customer_name", CustomerNameTxt.Text);


            MessageBox.Show("Record updated");
            cmd.ExecuteNonQuery();
            ViewButton_Click(sender, e);
            ClearTextBoxes();
        }


        private void ClearTextBoxes()
        {
            orderItems.Clear();
            CustomerNameTxt.Clear();
            PriceTxt.Clear();
            QuantityTxt.Clear();
            SubtotalTxt.Clear();
            DiscountTxt.Clear();
            NetTxt.Clear();
            PaidTxt.Clear();
            BalanceTxt.Clear();

        }



        private void orderdetail_Load(object sender, EventArgs e)
        {

        }

        private void orderManagementToolStripMenuItem_Click(object sender, EventArgs e)
        {
            orderdetail form = new orderdetail();
            form.ShowDialog();
        }

        private void customerManagementToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            CustomerDetail form = new CustomerDetail();
            form.ShowDialog();
        }
```

```
private void supplierDetailToolStripMenuItem_Click(object sender, EventArgs e)
{
   SupplierDetails form = new SupplierDetails();
   form.ShowDialog();
}

private void supplierProductToolStripMenuItem_Click(object sender, EventArgs e)
{
   SupplierProduct form = new SupplierProduct();
   form.ShowDialog();
}

private void reportToolStripMenuItem_Click(object sender, EventArgs e)
{
   report Form = new report();
   Form.ShowDialog();
}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
   About Form = new About();
   Form.ShowDialog();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
   ExitApplication();
}
private void ExitApplication()
{
```

```
            DialogResult result = MessageBox.Show("Are you sure you want to exit?", "Exit Application", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                Application.Exit();
            }


        }

        private void CategoryComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            ItemComboBox.Items.Clear();
            if (CategoryComboBox.SelectedItem != null)
            {
                // Get the selected category from CategoryComboBox
                string selectedCategory = CategoryComboBox.SelectedItem.ToString();

                // Populate ItemComboBox based on the selected category
                switch (selectedCategory)
                {
                    case  "Cake":
                        ItemComboBox.Items.Add("Chocolate Cake");
                        ItemComboBox.Items.Add("Vanilla Cake");
                        ItemComboBox.Items.Add("Strawberry Cake");
                        ItemComboBox.Items.Add("Red Velvet Cake");
                        ItemComboBox.Items.Add("Black Forest Cake");
                        ItemComboBox.Items.Add("Tiramisu Cake");
                        ItemComboBox.Items.Add("Cheesecake");
                        ItemComboBox.Items.Add("Mousse Cake");
```

```
    ItemComboBox.Items.Add("Red Velvet Cake");
    ItemComboBox.Items.Add("Pineapple Cake");
    ItemComboBox.Items.Add("Coffee Cake");
    // Add more cake items as needed
    break;
case "Pastries":
    ItemComboBox.Items.Add("Croissant");
    ItemComboBox.Items.Add("Danish Pastry");
    ItemComboBox.Items.Add("Éclair");
    ItemComboBox.Items.Add("Cinnamon Roll");
    ItemComboBox.Items.Add("Cream Puff");
    ItemComboBox.Items.Add("Palmier");
    ItemComboBox.Items.Add("Mille-Feuille");
    ItemComboBox.Items.Add("Macaron");
    // Add more pastry items as needed
    break;
case "Healthier cake":
    ItemComboBox.Items.Add("Carrot Cake");
    ItemComboBox.Items.Add("Lemon Cake");
    ItemComboBox.Items.Add("Zucchini Cake");
    ItemComboBox.Items.Add("Banana Cake");
    ItemComboBox.Items.Add("Almond Flour Cake");
    ItemComboBox.Items.Add("Coconut Flour Cake");
    ItemComboBox.Items.Add("Vegan Chocolate Cake");
    ItemComboBox.Items.Add("Gluten-Free Cake");
    ItemComboBox.Items.Add("Sugar-Free Cake");
    ItemComboBox.Items.Add("Whole Wheat Cake");
    ItemComboBox.Items.Add("Chia Seed Cake");
    ItemComboBox.Items.Add("Quinoa Cake");
    ItemComboBox.Items.Add("Avocado Cake");
```

```
                break;
            case "Decoratives":
                ItemComboBox.Items.Add("Edible Flowers");
                ItemComboBox.Items.Add("Sugar Sprinkles");
                ItemComboBox.Items.Add("Fondant Decorations");
                ItemComboBox.Items.Add("Edible Gold Leaf");
                ItemComboBox.Items.Add("Balloons");
                ItemComboBox.Items.Add("Birthday Caps");
                ItemComboBox.Items.Add("Confetti");
                ItemComboBox.Items.Add("Party Hats");
                ItemComboBox.Items.Add("Candles");
                ItemComboBox.Items.Add("Cake Toppers");
                ItemComboBox.Items.Add("Glitter");
                ItemComboBox.Items.Add("Ribbons");
                ItemComboBox.Items.Add("Banners");
                ItemComboBox.Items.Add("Garlands");
                ItemComboBox.Items.Add("Table Centerpieces");
                break;
            default:
                break;
        }
    }
}

private void CategoryComboBox_Validating(object sender, CancelEventArgs e)
{
    if (string.IsNullOrWhiteSpace(CategoryComboBox.Text))
    {
        errorProvider1.SetError(CategoryComboBox, "Category cannot be empty");
        e.Cancel = true;
    }
    else
    {
        errorProvider1.SetError(CategoryComboBox, string.Empty);
    }
}

private void ItemComboBox_Validating(object sender, CancelEventArgs e)
{
    if (string.IsNullOrWhiteSpace(ItemComboBox.Text))
    {
        errorProvider1.SetError(ItemComboBox, "Item cannot be empty");
        e.Cancel = true;
    }
    else
    {
        errorProvider1.SetError(ItemComboBox, string.Empty);
    }
}

private void CustomerNameTxt_Leave(object sender, EventArgs e)
{
    string inputName = CustomerNameTxt.Text.Trim();
    string namePattern = @"^[a-zA-Z]+(?: [a-zA-Z]+)*$";
    Regex nameRegex = new Regex(namePattern);
    bool isValidName = nameRegex.IsMatch(inputName);

    if (!isValidName)
    {
        errorProvider1.SetError(CustomerNameTxt, "Please Enter a Valid Name (only
letters with limited spaces allowed)");
```

```
            }
            else
            {
                errorProvider1.SetError(CustomerNameTxt, ""); // Clear any previous error
message
            }
        }

        private void PriceTxt_TextChanged(object sender, EventArgs e)
        {
            CalculateSubtotal();
        }

        private void QuantityTxt_TextChanged(object sender, EventArgs e)
        {
            CalculateSubtotal();
        }

        private void DiscountTxt_TextChanged(object sender, EventArgs e)
        {
            CalculateNet();
        }

        private void PaidTxt_TextChanged(object sender, EventArgs e)
        {
            CalculateBalance();
        }

        private void CalculateSubtotal()
        {
            if (!string.IsNullOrWhiteSpace(PriceTxt.Text) &&
!string.IsNullOrWhiteSpace(QuantityTxt.Text))
            {
                if (decimal.TryParse(PriceTxt.Text, out decimal price) &&
int.TryParse(QuantityTxt.Text, out int quantity))
                {
                    decimal subtotal = price * quantity;
                    SubtotalTxt.Text = subtotal.ToString();
                }
            }
        }

        private void CalculateNet()
        {
            if (!string.IsNullOrWhiteSpace(SubtotalTxt.Text) &&
!string.IsNullOrWhiteSpace(DiscountTxt.Text))
            {
                if (decimal.TryParse(SubtotalTxt.Text, out decimal subtotal) &&
decimal.TryParse(DiscountTxt.Text, out decimal discount))
                {
                    decimal net = subtotal - discount;
                    NetTxt.Text = net.ToString();
                }
            }
        }

        private void CalculateBalance()
        {
            if (!string.IsNullOrWhiteSpace(NetTxt.Text) &&
!string.IsNullOrWhiteSpace(PaidTxt.Text))
            {
```

```
            if (decimal.TryParse(NetTxt.Text, out decimal net) &&
decimal.TryParse(PaidTxt.Text, out decimal paid))
            {
                decimal balance = net - paid;
                BalanceTxt.Text = balance.ToString();
            }
        }
    }

    private void QuantityTxt_KeyPress(object sender, KeyPressEventArgs e)
    {
        // Allowing only numeric input and control keys (e.g., Backspace, Delete)
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }

    private void PriceTxt_KeyPress(object sender, KeyPressEventArgs e)
    {
        // Allowing only numeric input, decimal point, and control keys (e.g., Backspace,
Delete)
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != '.'))
        {
            e.Handled = true;
        }

        // Allowing only one decimal point
        if ((e.KeyChar == '.') && ((sender as
System.Windows.Forms.TextBox).Text.IndexOf('.') > -1))
        {
            e.Handled = true;
        }
    }

    private void DiscountTxt_KeyPress(object sender, KeyPressEventArgs e)
    {
        // Allowing only numeric input, decimal point, and control keys (e.g., Backspace,
Delete)
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != '.'))
        {
            e.Handled = true;
        }

        // Allowing only one decimal point
        if ((e.KeyChar == '.') && ((sender as
System.Windows.Forms.TextBox).Text.IndexOf('.') > -1))
        {
            e.Handled = true;
        }
    }

    private void PaidTxt_KeyPress(object sender, KeyPressEventArgs e)
    {
        // Allowing only numeric input, decimal point, and control keys (e.g., Backspace,
Delete)
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != '.'))
        {
            e.Handled = true;
        }

        // Allowing only one decimal point
```

```
            if ((e.KeyChar == '.') && ((sender as
System.Windows.Forms.TextBox).Text.IndexOf('.') > -1))
            {
                e.Handled = true;
            }
        }

        private void QuantityTxt_Validating(object sender, CancelEventArgs e)
        {
            if (string.IsNullOrWhiteSpace(QuantityTxt.Text))
            {
                e.Cancel = true;
                errorProvider1.SetError(QuantityTxt, "Quantity cannot be empty.");
            }
            else
            {
                errorProvider1.SetError(QuantityTxt, ""); // Clear any existing error message
            }
        }

        private void PriceTxt_Validating(object sender, CancelEventArgs e)
        {
            if (string.IsNullOrWhiteSpace(PriceTxt.Text))
            {
                e.Cancel = true;
                errorProvider1.SetError(PriceTxt, "Price cannot be empty.");
            }
            else
            {
                errorProvider1.SetError(PriceTxt, ""); // Clear any existing error message
            }
```

```
}

private void SaveButton_Click(object sender, EventArgs e)
{
    if (orderItems.Rows.Count == 0)
    {
        MessageBox.Show("Please add at least one item to the order before saving.");
        return; // Exit the method without saving the record
    }



    foreach (DataRow row in orderItems.Rows)
    {
        if (string.IsNullOrWhiteSpace(row["Category"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Item"].ToString()) ||
   //string.IsNullOrWhiteSpace(row["Customer_name"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Price"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Quantity"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Subtotal"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Discount"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Net"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Paid"].ToString()) ||
   string.IsNullOrWhiteSpace(row["Balance"].ToString()))
        {
            MessageBox.Show("Fill the required fields");
            return; // Exit the method without saving the record
        }

        MySqlConnection conn = new MySqlConnection(str);
        conn.Open();
        sql = "INSERT INTO Order (Category, Item, Order_id, Customer_name, Price, 
Quantity, Subtotal, Discount, Net, Paid, Balance) " +
            "VALUES (@Category, @Item, @Order_id, @Customer_name, @Price, 
@Quantity, @Subtotal, @Discount, @Net, @Paid, @Balance)";


        MySqlCommand cmd = new MySqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@Category", row["Category"]);
        cmd.Parameters.AddWithValue("@Item", row["Item"]);
        cmd.Parameters.AddWithValue("@Order_id", row["Order_id"]);
        cmd.Parameters.AddWithValue("@Customer_name", row["Customer_name"]);
        cmd.Parameters.AddWithValue("@Price", row["Price"]);
        cmd.Parameters.AddWithValue("@Quantity", row["Quantity"]);
        cmd.Parameters.AddWithValue("@Subtotal", row["Subtotal"]);
        cmd.Parameters.AddWithValue("@Discount", row["Discount"]);
        cmd.Parameters.AddWithValue("@Net", row["Net"]);
        cmd.Parameters.AddWithValue("@Paid", row["Paid"]);
        cmd.Parameters.AddWithValue("@Balance", row["Balance"]);
        cmd.Parameters.AddWithValue("@Date", row["Date"]);


        try
        {
            cmd.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }



        conn.Close();
    }
```

```
        MessageBox.Show("Order saved.");

        // Clear the orderItems DataTable after saving
        orderItems.Clear();
        CustomerNameTxt.Clear();
        PriceTxt.Clear();
        QuantityTxt.Clear();
        SubtotalTxt.Clear();
        DiscountTxt.Clear();
        NetTxt.Clear();
        PaidTxt.Clear();
        BalanceTxt.Clear();
        CategoryComboBox.SelectedIndex = -1;
        ItemComboBox.SelectedIndex = -1;

        // Generate a new Order ID
        string newOrderId = GenerateOrderId();
        OrderIdTxt.Text = newOrderId;
    }

    private void ItemComboBox_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (ItemComboBox.SelectedItem != null)
        {
            string selectedItem = ItemComboBox.SelectedItem.ToString();

            // Check if the selected item exists in the itemPrices dictionary
            if (itemPrices.ContainsKey(selectedItem))
            {
                decimal price = itemPrices[selectedItem];
                PriceTxt.Text = price.ToString("0.00"); // Display the price in the PriceTxt
TextBox
            }
        }
    }

    private void CustomerNameTxt_Validating(object sender, CancelEventArgs e)
    {
        string inputName = CustomerNameTxt.Text.Trim();
        string namePattern = "^[a-zA-Z]+( [a-zA-Z]+)*$";
        Regex nameRegex = new Regex(namePattern);
        bool isValidName = nameRegex.IsMatch(inputName);

        if (!isValidName)
        {
            errorProvider1.SetError(CustomerNameTxt, "Please Enter a Valid Name (only
letters with spaces allowed)");
            e.Cancel = true; // Prevent focus from moving to the next control
        }
        else
        {
            errorProvider1.SetError(CustomerNameTxt, string.Empty);
        }
    }

    private void OrderDetailDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {

    }
```

```
        private void PrintButton_Click(object sender, EventArgs e)
        {
            PrintDocument printDocument = new PrintDocument();
            printDocument.PrintPage += new
PrintPageEventHandler(PrintDocument_PrintPage);

            try
            {
                PrintDialog printDialog = new PrintDialog();
                printDialog.Document = printDocument;

                if (printDialog.ShowDialog() == DialogResult.OK)
                {
                    printDocument.Print();
                    MessageBox.Show("Invoice printed successfully.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error printing invoice: {ex.Message}");
            }
        }

        private void PrintDocument_PrintPage(object sender, PrintPageEventArgs e)
        {
            // Define the font, brush, and string format for drawing text
            Font font = new Font("Arial", 12);
            SolidBrush brush = new SolidBrush(Color.Black);
            StringFormat stringFormat = new StringFormat();

            // Define the invoice details to print
            string invoiceDetails =
               $"Invoice Details:\n" +
                        $"Category: {CategoryComboBox.Text}\n" +
                        $"Item: {ItemComboBox.Text}\n" +
                        $"Order ID: {OrderIdTxt.Text}\n" +
                        $"Customer Name: {CustomerNameTxt.Text}\n" +
                        $"Price: {PriceTxt.Text}\n" +
                        $"Quantity: {QuantityTxt.Text}\n" +
                        $"Subtotal: {SubtotalTxt.Text}\n" +
                        $"Discount: {DiscountTxt.Text}\n" +
                        $"Net: {NetTxt.Text}\n" +
                        $"Paid: {PaidTxt.Text}\n" +
                        $"Balance: {BalanceTxt.Text}"+
                        $"Date: {date.Value}";
            // Define the rectangle to draw the invoice details
            RectangleF rectangle = new RectangleF(100, 100, 400, 300); // Adjust the position
and size as needed


            // Draw the invoice details on the print document
            e.Graphics.DrawString(invoiceDetails, font, brush, rectangle, stringFormat);
        }





        private void OrderIdTxt_TextChanged(object sender, EventArgs e)
        {


        }


        private void CustomerNameTxt_TextChanged(object sender, EventArgs e)
        {

        }
    }

}
```

Supplier Detail

```
using MySql.Data.MySqlClient;
using Mysqlx.Crud;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Betterbebutter15
{
    public partial class SupplierDetails : Form
    {
        private static int nextSupplierNumber = 1;
        public SupplierDetails()
```

```
{
    InitializeComponent();

    SupplierIdTxt.Text = GenerateSupplierId();




}
private string GenerateSupplierId()
{
    string nextSupplierId = ""; // Default value
    MySqlConnection conn = new MySqlConnection(str);

    try
    {
        conn.Open();

        // Query to get the maximum supplier ID from the "Supplierdetail" table
        string sql = "SELECT MAX(Supplier_id) FROM Supplierdetail";
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        object result = cmd.ExecuteScalar();

        if (result != null && result != DBNull.Value)
        {
            int lastSupplierId = Convert.ToInt32(result);
            int nextId = lastSupplierId + 1;

            // Ensure the next ID has only 3 digits
            if (nextId < 1000)
            {
                nextSupplierId = nextId.ToString("D3");
            }
            else
            {
                MessageBox.Show("Maximum supplier ID limit reached.");
            }
        }
        else
        {
            // If no previous supplier IDs exist, start with 001
            nextSupplierId = "001";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error generating next supplier ID: " + ex.Message);
    }
    finally
    {
        conn.Close();
    }

    return nextSupplierId;
}



string sql;
MySqlDataReader read;
string str = "server=localhost; uid=root; pwd=Aastha@1978;
database=betterbebutter15";
private void SupplierDetails_Load(object sender, EventArgs e)
```

```
        {
            SupplierNameTxt.Validating += SupplierNameTxt_Validating;
            ContactNumberTxt.Validating += ContactNumberTxt_Validating;
            EmailIdTxt.Validating += EmailIdTxt_Validating;
            AddressTxt.Validating += AddressTxt_Validating;
            BankNameTxt.Validating += BankNameTxt_Validating;
            AccountNumberTxt.Validating += AccountNumberTxt_Validating;
            BankIFSCCodeTxt.Validating += BankIFSCCodeTxt_Validating;
            BankBranchNameTxt.Validating += BankBranchNameTxt_Validating;



        }



        private void AddButton_Click(object sender, EventArgs e)
        {
            bool isValidSupplierId = ValidateField(SupplierIdTxt, "Supplier ID cannot be
empty.");
            bool isValidSupplierName = ValidateField(SupplierNameTxt, "Supplier name cannot
be empty.");
            bool isValidContactNumber = ValidateField(ContactNumberTxt, "Contact number
cannot be empty.");
            bool isValidEmailId = ValidateField(EmailIdTxt, "Email ID cannot be empty.");
            bool isValidAddress = ValidateField(AddressTxt, "Address cannot be empty.");
            bool isValidBankName = ValidateField(BankNameTxt, "Bank name cannot be
empty.");
            bool isValidAccountNumber = ValidateField(AccountNumberTxt, "Account number
cannot be empty.");
            bool isValidBankIFSCCode = ValidateField(BankIFSCCodeTxt, "IFSC code cannot
be empty.");
            bool isValidBankBranchName = ValidateField(BankBranchNameTxt, "Branch name
cannot be empty.");
```

```
            // If all fields are valid, proceed with adding the record
            if (isValidSupplierId && isValidSupplierName && isValidContactNumber &&
isValidEmailId &&
                isValidAddress && isValidBankName && isValidAccountNumber &&
isValidBankIFSCCode &&
                isValidBankBranchName)
            {



                MySqlConnection conn = new MySqlConnection(str);
                conn.ConnectionString = str;
                conn.Open();
                sql = "insert into
Supplierdetail(Supplier_id,Supplier_name,Contact,Emailid,Address,Bank_name,Account_nu
mber,IFSC_code,Branch)
values(@Supplier_id,@Supplier_name,@Contact,@Emailid,@Address,@Bank_name,@Acc
ount_number,@IFSC_code,@Branch)";
                MySqlCommand cmd = new MySqlCommand(sql, conn);
                cmd.Parameters.AddWithValue("@Supplier_id", SupplierIdTxt.Text);
                cmd.Parameters.AddWithValue("@Supplier_name", SupplierNameTxt.Text);
                cmd.Parameters.AddWithValue("@Contact", ContactNumberTxt.Text);
                cmd.Parameters.AddWithValue("@Emailid", EmailIdTxt.Text);
                cmd.Parameters.AddWithValue("@Address", AddressTxt.Text);
                cmd.Parameters.AddWithValue("@Bank_name", BankNameTxt.Text);
                cmd.Parameters.AddWithValue("@Account_number", AccountNumberTxt.Text);
                cmd.Parameters.AddWithValue("@IFSC_code", BankIFSCCodeTxt.Text);
                cmd.Parameters.AddWithValue("@Branch", BankBranchNameTxt.Text);
                MessageBox.Show("record added");
                cmd.ExecuteNonQuery();
                SupplierIdTxt.Clear();
                SupplierNameTxt.Clear();
                ContactNumberTxt.Clear();
```

```
        EmailIdTxt.Clear();
        AddressTxt.Clear();
        BankNameTxt.Clear();
        AccountNumberTxt.Clear();
        BankIFSCCodeTxt.Clear();
        BankBranchNameTxt.Clear();
        SupplierIdTxt.Text = GenerateSupplierId();


    }
}


private bool ValidateField(Control control, string errorMessage)
{
    // Trim the input to remove leading or trailing whitespaces
    string inputValue = control.Text.Trim();

    // Check if the input is empty
    if (string.IsNullOrEmpty(inputValue))
    {
        // Display an error message using ErrorProvider
        errorProvider9.SetError(control, errorMessage);
        return false; // Field is empty or null
    }

    // Clear the error message for the control
    errorProvider9.SetError(control, string.Empty);
    return true; // Field is valid
}
```

```
        private void view_Click(object sender, EventArgs e)
        {
            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();
            sql = "select * from Supplierdetail";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            read = cmd.ExecuteReader();
            SupplierDetailsDataGridView.Rows.Clear();
            while (read.Read())
            {
                SupplierDetailsDataGridView.Rows.Add(read[0], read[1], read[2], read[3], read[4],
read[5], read[6], read[7], read[8]);
            }
            conn.Close();

        }

        private void UpdateButton_Click(object sender, EventArgs e)
        {

            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();

            string sql = "UPDATE CustomerS SET Supplier_name = @Supplier_name WHERE
Supplier_id = @Supplier_id";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
```

```
    cmd.Parameters.AddWithValue("@Supplier_id",SupplierIdTxt.Text);
    cmd.Parameters.AddWithValue("@Supplier_name",SupplierNameTxt.Text);

    MessageBox.Show("Record updated");
    cmd.ExecuteNonQuery();
    view_Click(sender, e);
    ClearTextBoxes();
}

private void ClearTextBoxes()
{
    SupplierIdTxt.Clear();
    SupplierNameTxt.Clear();
    ContactNumberTxt.Clear();
    EmailIdTxt.Clear();
    AddressTxt.Clear();
    BankNameTxt.Clear();
    AccountNumberTxt.Clear();
    BankIFSCCodeTxt.Clear();
    BankBranchNameTxt.Clear();









}
```

```
private void DeleteButton_Click(object sender, EventArgs e)
{
   MySqlConnection conn = new MySqlConnection(str);
   conn.ConnectionString = str;
   conn.Open();

   string sql = "DELETE FROM Supplierdetail WHERE Supplier_id = @Supplier_id";
   MySqlCommand cmd = new MySqlCommand(sql, conn);
   cmd.Parameters.AddWithValue("@Supplier_id", SupplierIdTxt.Text);

   MessageBox.Show("Record deleted");
   cmd.ExecuteNonQuery();
   view_Click(sender, e);

}

private void SupplierIdTxt_TextChanged(object sender, EventArgs e)
{

}
private void SupplierID_Load(object sender, EventArgs e)
{
   /*     string initialsupplierID = GenerateSupplierID();
        SupplierIdTxt.Text = initialsupplierID;*/
}



private void SupplierNameTxt_Leave(object sender, EventArgs e)
{
```

```
            // Trim the input to remove leading or trailing whitespaces
            string inputName = SupplierNameTxt.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputName))
            {
                errorProvider1.SetError(SupplierNameTxt, "Name cannot be empty");
                return;
            }

            // Use a regular expression to allow only letters and spaces
            string namePattern = "^[a-zA-Z]+( [a-zA-Z]+)*$";
            Regex nameRegex = new Regex(namePattern);
            bool isValidName = nameRegex.IsMatch(inputName);

            if (!isValidName)
            {
                errorProvider1.SetError(SupplierNameTxt, "Please Enter a Valid Name (only
letters with spaces allowed)");
            }
            else
            {
                errorProvider1.SetError(SupplierNameTxt, string.Empty);
            }
        }

        private void ContactNumberTxt_Leave(object sender, EventArgs e)
        {
            string inputContact = ContactNumberTxt.Text.Trim();
            // Check if the input is empty
            if (string.IsNullOrEmpty(inputContact))
            {
                errorProvider2.SetError(ContactNumberTxt, "Mobile Number cannot be empty");
                return;
            }

            // Use a verbatim string (@) to avoid escaping characters
            Regex ex = new Regex(@"^\+?[0-9\s-]+$");
            bool isValid = ex.IsMatch(inputContact);
            string digitsOnly = new string(inputContact.Where(char.IsDigit).ToArray());



            if (!isValid || digitsOnly.Length != 10)
            {
                errorProvider2.SetError(ContactNumberTxt, "Please Enter a Valid Mobile
Number");
            }
            else
            {
                errorProvider2.SetError(ContactNumberTxt, string.Empty);
            }
        }

        private void EmailIdTxt_Leave(object sender, EventArgs e)
        {
            // Trim the input to remove leading or trailing whitespaces
            string inputEmail = EmailIdTxt.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputEmail))
            {
                errorProvider3.SetError(EmailIdTxt, "Email cannot be empty");
                return;
            }

            // Use a regular expression for email validation
            string emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
            Regex emailRegex = new Regex(emailPattern);
            bool isValidEmail = emailRegex.IsMatch(inputEmail);

            if (!isValidEmail)
            {
                errorProvider3.SetError(EmailIdTxt, "Please Enter a Valid Email Address");

            }
            else
            {
                errorProvider3.SetError(EmailIdTxt, string.Empty);

            }
        }



        private void AddressTxt_Leave(object sender, EventArgs e)
        {
            string inputAddress = AddressTxt.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputAddress))
            {
```

```
                // Set the error message for the control
                errorProvider4.SetError(AddressTxt, "Address cannot be empty");
                return;
            }

            // Use a regular expression to allow only letters, numbers, and limited spaces
            string addressPattern = "^[a-zA-Z0-9]+( [a-zA-Z0-9]+)*$";
            Regex addressRegex = new Regex(addressPattern);
            bool isValidAddress = addressRegex.IsMatch(inputAddress);

            if (!isValidAddress)
            {
                // Set the error message for the control
                errorProvider4.SetError(AddressTxt, "Please Enter a Valid Address (only letters,
numbers, and limited spaces allowed)");
            }
            else
            {
                // Clear the error message for the control
                errorProvider4.SetError(AddressTxt, string.Empty);
            }

        }

        private void BankNameTxt_Leave(object sender, EventArgs e)
        {
            // Clear any previous error message
            errorProvider5.SetError(BankNameTxt, string.Empty);

            // Check if the bank name is empty
            if (string.IsNullOrWhiteSpace(BankNameTxt.Text))
            {
                errorProvider5.SetError(BankNameTxt, "Bank name cannot be empty");
                return; // No need to check further
            }

            // Check if the bank name contains only letters, digits, and spaces
            string pattern = "^[a-zA-Z0-9 ]+$";
            Regex regex = new Regex(pattern);
            if (!regex.IsMatch(BankNameTxt.Text))
            {
                errorProvider5.SetError(BankNameTxt, "Bank name can only contain letters,
digits, and spaces");
                return; // No need to check further
            }
            else
            {
                errorProvider5.SetError(BankNameTxt, string.Empty);

            }
        }

        private void AccountNumberTxt_Leave(object sender, EventArgs e)
        {

            // Clear any previous error message
            errorProvider6.SetError(AccountNumberTxt, string.Empty);

            // Trim the input to remove leading or trailing whitespaces
            string inputAccountNumber = AccountNumberTxt.Text.Trim();

            // Check if the bank account number is empty
            if (string.IsNullOrWhiteSpace(inputAccountNumber))
            {
                // Display an error message using ErrorProvider
                errorProvider6.SetError(AccountNumberTxt, "Bank account number cannot be
empty");
                return;
            }

            // Check if the bank account number contains only digits and limited spaces
            string pattern = "^[0-9]+$";
            Regex regex = new Regex(pattern);
            if (!regex.IsMatch(inputAccountNumber))
            {
                // Display an error message using ErrorProvider
                errorProvider6.SetError(AccountNumberTxt, "Bank account number can only
contain digits");
                return;
            }
            else
            {
                errorProvider6.SetError(AccountNumberTxt, string.Empty);

            }

        }

        private void BankIFSCCodeTxt_Leave(object sender, EventArgs e)
        {
            errorProvider7.SetError(BankIFSCCodeTxt, string.Empty);

            // Get the input IFSC code and trim it to remove leading or trailing whitespaces
```

```
    string inputIfsc = BankIFSCCodeTxt.Text.Trim();

    // Check if the IFSC code is empty
    if (string.IsNullOrWhiteSpace(inputIfsc))
    {
        // Display an error message using ErrorProvider
        errorProvider7.SetError(BankIFSCCodeTxt, "IFSC code cannot be empty");
        return;
    }

    // Check if the IFSC code matches the pattern of a valid IFSC code
    string IfscPattern = @"^[A-Z]{4}0[A-Z0-9]{6}$";
    Regex regex = new Regex(IfscPattern);
    if (!regex.IsMatch(inputIfsc))
    {
        // Display an error message using ErrorProvider
        errorProvider7.SetError(BankIFSCCodeTxt, "Invalid IFSC code");
        return;

    }
    else
    {
        errorProvider7.SetError(BankIFSCCodeTxt, string.Empty);

    }
}

private void BankBranchNameTxt_Leave(object sender, EventArgs e)
{
    errorProvider8.SetError(BankBranchNameTxt, string.Empty);
```

```
            // Get the input bank branch name and trim it to remove leading or trailing
whitespaces
            string inputBranchName = BankBranchNameTxt.Text.Trim();

            // Check if the bank branch name is empty
            if (string.IsNullOrWhiteSpace(inputBranchName))
            {
                // Display an error message using ErrorProvider
                errorProvider8.SetError(BankBranchNameTxt, "Branch name cannot be empty");
                return;
            }

            // Check if the bank branch name contains only letters, digits, and spaces
            string pattern = "^[a-zA-Z0-9 ]+$";
            Regex regex = new Regex(pattern);
            if (!regex.IsMatch(inputBranchName))
            {
                // Display an error message using ErrorProvider
                errorProvider8.SetError(BankBranchNameTxt, "Branch name can only contain
letters, digits, and spaces");
                return;
            }
            else
            {
                errorProvider8.SetError(BankBranchNameTxt, string.Empty);

            }
        }

        private void SupplierNameTxt_Validating(object sender, CancelEventArgs e)
        {
            // Trim the input to remove leading or trailing whitespaces
            string inputName = SupplierNameTxt.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputName))
            {
                errorProvider1.SetError(SupplierNameTxt, "Name cannot be empty");
                e.Cancel = true; // Prevent focus from moving to the next control
                return;
            }

            // Use a regular expression to allow only letters and spaces
            string namePattern = "^[a-zA-Z]+( [a-zA-Z]+)*$";
            Regex nameRegex = new Regex(namePattern);
            bool isValidName = nameRegex.IsMatch(inputName);

            if (!isValidName)
            {
                errorProvider1.SetError(SupplierNameTxt, "Please Enter a Valid Name (only
letters with spaces allowed)");
                e.Cancel = true; // Prevent focus from moving to the next control
            }
            else
            {
                errorProvider1.SetError(SupplierNameTxt, string.Empty);
            }
        }

        private void ContactNumberTxt_Validating(object sender, CancelEventArgs e)
        {
            string inputContact = ContactNumberTxt.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputContact))
            {
                errorProvider2.SetError(ContactNumberTxt, "Mobile Number cannot be empty");
                e.Cancel = true; // Prevent focus from moving to the next control
                return;
            }

            // Use a verbatim string (@) to avoid escaping characters
            Regex ex = new Regex(@"^\+?[0-9\s-]+$");
            bool isValid = ex.IsMatch(inputContact);
            string digitsOnly = new string(inputContact.Where(char.IsDigit).ToArray());



            if (!isValid || digitsOnly.Length != 10)
            {
                errorProvider2.SetError(ContactNumberTxt, "Please Enter a Valid Mobile
Number");
                e.Cancel = true; // Prevent focus from moving to the next control
            }
            else
            {
                errorProvider2.SetError(ContactNumberTxt, string.Empty);
            }
        }

        private void EmailIdTxt_Validating(object sender, CancelEventArgs e)
        {
            // Trim the input to remove leading or trailing whitespaces
```

```
    string inputEmail = EmailIdTxt.Text.Trim();

    // Check if the input is empty
    if (string.IsNullOrEmpty(inputEmail))
    {
        errorProvider3.SetError(EmailIdTxt, "Email cannot be empty");
        e.Cancel = true; // Cancel the event to prevent the focus change
        return;
    }

    // Use a regular expression for email validation
    string emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
    Regex emailRegex = new Regex(emailPattern);
    bool isValidEmail = emailRegex.IsMatch(inputEmail);

    if (!isValidEmail)
    {
        errorProvider3.SetError(EmailIdTxt, "Please Enter a Valid Email Address");
        e.Cancel = true; // Cancel the event to prevent the focus change
    }
    else
    {
        errorProvider3.SetError(EmailIdTxt, string.Empty);
    }
}

private void AddressTxt_Validating(object sender, CancelEventArgs e)
{
    string inputAddress = AddressTxt.Text.Trim();
```

```
            // Check if the input is empty
            if (string.IsNullOrEmpty(inputAddress))
            {
                // Set the error message for the control
                errorProvider4.SetError(AddressTxt, "Address cannot be empty");
                e.Cancel = true; // Cancel the event to prevent the focus change
                return;
            }

            // Use a regular expression to allow only letters, numbers, and limited spaces
            string addressPattern = "^[a-zA-Z0-9]+( [a-zA-Z0-9]+)*$";
            Regex addressRegex = new Regex(addressPattern);
            bool isValidAddress = addressRegex.IsMatch(inputAddress);

            if (!isValidAddress)
            {
                // Set the error message for the control
                errorProvider4.SetError(AddressTxt, "Please Enter a Valid Address (only letters,
numbers, and limited spaces allowed)");
                e.Cancel = true; // Cancel the event to prevent the focus change
            }
            else
            {
                // Clear the error message for the control
                errorProvider4.SetError(AddressTxt, string.Empty);
            }
        }

        private void BankNameTxt_Validating(object sender, CancelEventArgs e)
        {
            // Clear any previous error message
            errorProvider5.SetError(BankNameTxt, string.Empty);

            // Check if the bank name is empty
            if (string.IsNullOrWhiteSpace(BankNameTxt.Text))
            {
                errorProvider5.SetError(BankNameTxt, "Bank name cannot be empty");
                e.Cancel = true; // Prevents the focus from changing
                return; // No need to check further
            }

            // Check if the bank name contains only letters, digits, and spaces
            string pattern = "^[a-zA-Z0-9 ]+$";
            Regex regex = new Regex(pattern);
            if (!regex.IsMatch(BankNameTxt.Text))
            {
                errorProvider5.SetError(BankNameTxt, "Bank name can only contain letters,
digits, and spaces");
                e.Cancel = true; // Prevents the focus from changing
                return; // No need to check further
            }
            else
            {
                errorProvider5.SetError(BankNameTxt, string.Empty);
            }
        }

        private void AccountNumberTxt_Validating(object sender, CancelEventArgs e)
        {
            // Clear any previous error message
            errorProvider6.SetError(AccountNumberTxt, string.Empty);
            // Trim the input to remove leading or trailing whitespaces
            string inputAccountNumber = AccountNumberTxt.Text.Trim();

            // Check if the bank account number is empty
            if (string.IsNullOrWhiteSpace(inputAccountNumber))
            {
                // Display an error message using ErrorProvider
                errorProvider6.SetError(AccountNumberTxt, "Bank account number cannot be
empty");
                e.Cancel = true; // Prevents the focus from changing
                return;
            }

            // Check if the bank account number contains only digits and limited spaces
            string pattern = "^[0-9]+$";
            Regex regex = new Regex(pattern);
            if (!regex.IsMatch(inputAccountNumber))
            {
                // Display an error message using ErrorProvider
                errorProvider6.SetError(AccountNumberTxt, "Bank account number can only
contain digits");
                e.Cancel = true; // Prevents the focus from changing
                return;
            }
            else
            {
                errorProvider6.SetError(AccountNumberTxt, string.Empty);
            }
        }

        private void BankIFSCCodeTxt_Validating(object sender, CancelEventArgs e)
```

```
{
    errorProvider7.SetError(BankIFSCCodeTxt, string.Empty);

    // Get the input IFSC code and trim it to remove leading or trailing whitespaces
    string inputIfsc = BankIFSCCodeTxt.Text.Trim();

    // Check if the IFSC code is empty
    if (string.IsNullOrWhiteSpace(inputIfsc))
    {
        // Display an error message using ErrorProvider
        errorProvider7.SetError(BankIFSCCodeTxt, "IFSC code cannot be empty");
        e.Cancel = true; // Prevents the focus from changing
        return;
    }

    // Check if the IFSC code matches the pattern of a valid IFSC code
    string IfscPattern = @"^[A-Z]{4}0[A-Z0-9]{6}$";
    Regex regex = new Regex(IfscPattern);
    if (!regex.IsMatch(inputIfsc))
    {
        // Display an error message using ErrorProvider
        errorProvider7.SetError(BankIFSCCodeTxt, "Invalid IFSC code");
        e.Cancel = true; // Prevents the focus from changing
        return;
    }
    else
    {
        errorProvider7.SetError(BankIFSCCodeTxt, string.Empty);
    }
}
```

```
        private void BankBranchNameTxt_Validating(object sender, CancelEventArgs e)
        {
            errorProvider8.SetError(BankBranchNameTxt, string.Empty);

            // Get the input bank branch name and trim it to remove leading or trailing
whitespaces
            string inputBranchName = BankBranchNameTxt.Text.Trim();

            // Check if the bank branch name is empty
            if (string.IsNullOrWhiteSpace(inputBranchName))
            {
                // Display an error message using ErrorProvider
                errorProvider8.SetError(BankBranchNameTxt, "Branch name cannot be empty");
                e.Cancel = true; // Prevents the focus from changing
                return;
            }

            // Check if the bank branch name contains only letters, digits, and spaces
            string pattern = "^[a-zA-Z0-9 ]+$";
            Regex regex = new Regex(pattern);
            if (!regex.IsMatch(inputBranchName))
            {
                // Display an error message using ErrorProvider
                errorProvider8.SetError(BankBranchNameTxt, "Branch name can only contain
letters, digits, and spaces");
                e.Cancel = true; // Prevents the focus from changing
                return;
            }
            else
            {
```

```
        errorProvider8.SetError(BankBranchNameTxt, string.Empty);
    }
}



private void label2_Click(object sender, EventArgs e)
{

}

private void orderManagementToolStripMenuItem_Click(object sender, EventArgs e)
{
    orderdetail form = new orderdetail();
    form.ShowDialog();
}


private void supplierDetailToolStripMenuItem_Click(object sender, EventArgs e)
{
    SupplierDetails form = new SupplierDetails();
    form.ShowDialog();
}

private void supplierProductToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    SupplierProduct form = new SupplierProduct();
    form.ShowDialog();
}
```

```
        private void reportToolStripMenuItem_Click(object sender, EventArgs e)
        {
            report Form = new report();
            Form.ShowDialog();
        }

        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About Form = new About();
            Form.ShowDialog();
        }

        private void exitToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            ExitApplication();
        }
        private void ExitApplication()
        {
            DialogResult result = MessageBox.Show("Are you sure you want to exit?", "Exit
    Application", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                Application.Exit();
            }

        }

        private void customerManagementToolStripMenuItem_Click(object sender, EventArgs
    e)
```

```
        {
            CustomerDetail form = new CustomerDetail();
            form.ShowDialog();
        }

        private void SupplierNameTxt_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

SUPPLIER PRODUCT

```
using MySql.Data.MySqlClient;
using Mysqlx.Crud;
using System.Drawing.Printing;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Printing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
```

```
namespace Betterbebutter15
{
   public partial class SupplierProduct : Form
   {

       private int nextProductId = 1;
       private string str = "server=localhost; uid=root; pwd=Aastha@1978;
database=betterbebutter15";
       public SupplierProduct()
       {
          InitializeComponent();
          string initialProductId = GenerateProductId();
          ProductIdTxt.Text = initialProductId;
          QuantityTxt.TextChanged += quantity_TextChanged;
          PriceTxt.TextChanged += PriceTxt_TextChanged;
       }


       private void label5_Click(object sender, EventArgs e)
       {

       }

       private void AddButton_Click(object sender, EventArgs e)
       {
          if (!ValidateSupplierId(SupplierIdTxt.Text))
          {
             MessageBox.Show("Invalid Supplier ID. Please enter a valid Supplier ID.");
             return;
          }
```

```
            if (!ValidateFields()) return;
            {
                bool isValidQuantity = ValidateField(QuantityTxt, "Quantity cannot be empty.");
                bool isValidPrice = ValidateField(PriceTxt, "Price cannot be empty.");

                // If all fields are valid, proceed with adding the record
                if (isValidQuantity && isValidPrice)
                {
                    double quantity;
                    double price;

                    // Check if quantity and price are valid doubles
                    if (!double.TryParse(QuantityTxt.Text.Trim(), out quantity) ||
!double.TryParse(PriceTxt.Text.Trim(), out price))
                    {
                        MessageBox.Show("Invalid input. Please enter valid numbers for Quantity
and Price.");
                        return;
                    }

                    // Check if quantity and price are greater than zero
                    if (quantity <= 0 || price <= 0)
                    {
                        MessageBox.Show("Quantity and price must be greater than zero.");
                        return;
                    }

                    // Calculate total
                    double total = quantity * price;
```

```
TotalTxt.Text = total.ToString();




MySqlConnection conn = new MySqlConnection(str);
try
{
    conn.Open();
    string sql = "INSERT INTO Supplierproduct(Supplier_id, Product_id,
Product_category, Subcategory, Quantity, Price, Total, Date) " +
                 "VALUES(@Supplier_id, @Product_id, @Product_category,
@Subcategory, @Quantity, @Price, @Total, @Date)";
    MySqlCommand cmd = new MySqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@Supplier_id", SupplierIdTxt.Text);
    cmd.Parameters.AddWithValue("@Product_id", ProductIdTxt.Text);
    cmd.Parameters.AddWithValue("@Product_category", categoryTxt.Text);
    cmd.Parameters.AddWithValue("@Subcategory", SubcategoryTxt.Text);
    cmd.Parameters.AddWithValue("@Quantity", QuantityTxt.Text);
    cmd.Parameters.AddWithValue("@Price", PriceTxt.Text);
    cmd.Parameters.AddWithValue("@Total", TotalTxt.Text);
    cmd.Parameters.AddWithValue("@Date", DateTxt.Text);
    cmd.ExecuteNonQuery();
    MessageBox.Show("Record added");
    ClearFields();
    ProductIdTxt.Text = GenerateProductId();
}

catch (Exception ex)
{
```

```
                    MessageBox.Show("An error occurred while adding the record: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    conn.Close();
                }



            }


        }
    }

    private bool ValidateSupplierId(string supplierId)
    {
        using (MySqlConnection conn = new MySqlConnection(str))
        {
            conn.Open();
            string sql = "SELECT COUNT(*) FROM SupplierDetails WHERE Supplier_id =
@Supplier_id";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@Supplier_id", supplierId);
            int count = Convert.ToInt32(cmd.ExecuteScalar());
            return count > 0;
        }
    }




    private void SupplierProduct_Load(object sender, EventArgs e)
```

```
{
    // Generate the initial unique IDs for the supplier and product

    string initialProductId = GenerateProductId();

    // Set the initial IDs in the textboxes

    ProductIdTxt.Text = initialProductId;
}




private string GenerateProductId()
{
    MySqlConnection conn = new MySqlConnection(str);
    conn.Open();

    string sql = "SELECT MAX(Product_id) FROM Supplierproduct";
    MySqlCommand cmd = new MySqlCommand(sql, conn);
    object result = cmd.ExecuteScalar();

    conn.Close();

    if (result != null && result != DBNull.Value)
    {
        nextProductId = int.Parse(result.ToString().Substring(1)) + 1;
    }

    string sequentialNumber = nextProductId.ToString("D3");
    string productId = "P" + sequentialNumber;
    return productId;
}



private bool ValidateFields()
{

    {
        if (string.IsNullOrWhiteSpace(SupplierIdTxt.Text) ||
        string.IsNullOrWhiteSpace(ProductIdTxt.Text) ||
        string.IsNullOrWhiteSpace(categoryTxt.Text) ||
        string.IsNullOrWhiteSpace(SubcategoryTxt.Text) ||
        string.IsNullOrWhiteSpace(QuantityTxt.Text) ||
        string.IsNullOrWhiteSpace(PriceTxt.Text) ||
        string.IsNullOrWhiteSpace(TotalTxt.Text) ||
        string.IsNullOrWhiteSpace(DateTxt.Text))
        {
            MessageBox.Show("All fields are required.");
            return false;
        }
    }
    if (!double.TryParse(QuantityTxt.Text, out _) || !double.TryParse(PriceTxt.Text, out
_))
    {
        MessageBox.Show("Quantity and Price must be valid numbers.");
        return false;
    }

    return true;
}
```

```
private void ClearFields()
{
    SupplierIdTxt.Clear();
    ProductIdTxt.Clear();
    categoryTxt.Items.Clear();
    SubcategoryTxt.Items.Clear();
    QuantityTxt.Clear();
    PriceTxt.Clear();
    TotalTxt.Clear();
    DateTxt.Clear();
}



private void UpdateButton_Click(object sender, EventArgs e)
{

}

private void VIEW_Click(object sender, EventArgs e)
{
    using (MySqlConnection conn = new MySqlConnection(str))
    {
        conn.Open();
        string sql = "SELECT * FROM Supplierproduct";
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        MySqlDataReader read = cmd.ExecuteReader();
        SupplierProductDataGridView.Rows.Clear();
        while (read.Read())
```

```
                {
                    SupplierProductDataGridView.Rows.Add(read[0], read[1], read[2], read[3],
read[4], read[5], read[6], read[7]);
                }
                read.Close();

            }
        }

        private void DeleteButton_Click(object sender, EventArgs e)
        {
            MySqlConnection conn = new MySqlConnection(str);
            conn.ConnectionString = str;
            conn.Open();

            string sql = "DELETE FROM Supplierproduct WHERE Supplier_id =
@Supplier_id";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@Supplier_id", SupplierIdTxt.Text);

            MessageBox.Show("Record deleted");
            cmd.ExecuteNonQuery();
            VIEW_Click(sender, e);
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void quantity_TextChanged(object sender, EventArgs e)
```

```
{
   UpdateTotal();
}

private void PriceTxt_TextChanged(object sender, EventArgs e)
{
   UpdateTotal();
}

private void UpdateTotal()
{
   bool isValidQuantity = ValidateField(QuantityTxt, "Quantity cannot be empty.");
   bool isValidPrice = ValidateField(PriceTxt, "Price cannot be empty.");

   if (isValidQuantity && isValidPrice)
   {
      double quantity = double.Parse(QuantityTxt.Text.Trim());
      double price = double.Parse(PriceTxt.Text.Trim());

      double total = quantity * price;

      TotalTxt.Text = total.ToString();
   }
   else
   {
      TotalTxt.Clear();
   }
}

private void quantity_Validating(object sender, CancelEventArgs e)
```

```
        {
            string inputValue = QuantityTxt.Text.Trim();

            // Check if the input is empty
            if (string.IsNullOrEmpty(inputValue))
            {
                // Display an error message using ErrorProvider
                errorProvider1.SetError(QuantityTxt, "Quantity cannot be empty.");
                e.Cancel = true; // Cancel the event to prevent focus change
            }
            else
            {
                // Check if the input is a valid number
                if (!double.TryParse(inputValue, out double result))
                {
                    // Display an error message using ErrorProvider
                    errorProvider1.SetError(QuantityTxt, "Invalid input. Please enter a valid
number.");
                    e.Cancel = true; // Cancel the event to prevent focus change
                }
                else
                {
                    // Clear the error message for the control
                    errorProvider1.SetError(QuantityTxt, string.Empty);
                }
            }
        }

        private void PriceTxt_Validating(object sender, CancelEventArgs e)
        {
            ValidateField(PriceTxt, "Price cannot be empty.");
```

```
}


private bool ValidateField(Control control, string errorMessage)
{
   string inputValue = control.Text.Trim();

   if (string.IsNullOrEmpty(inputValue))
   {
      errorProvider1.SetError(control, errorMessage);
      //e.Cancel = true;
      return false;
   }

   if (!double.TryParse(inputValue, out double result))
   {
      errorProvider1.SetError(control, "Invalid input. Please enter a valid number.");
      return false;
   }

   errorProvider1.SetError(control, string.Empty);
   return true;
}


private void DateTxt_Validating(object sender, CancelEventArgs e)
{
   // Trim the input to remove leading or trailing whitespaces
   string inputValue = DateTxt.Text.Trim();
```

```
            // Check if the input is empty
            if (string.IsNullOrEmpty(inputValue))
            {
                // Display an error message using ErrorProvider
                errorProvider2.SetError(DateTxt, "Date cannot be empty.");
                e.Cancel = true; // Cancel the event to prevent focus change
            }
            else
            {
                // Check if the input is a valid date in the format 'yyyy/mm/dd'
                if (!DateTime.TryParseExact(inputValue, "yyyy/MM/dd",
CultureInfo.InvariantCulture, DateTimeStyles.None, out _))
                {
                    // Display an error message using ErrorProvider
                    errorProvider2.SetError(DateTxt, "Invalid date format. Please enter a date in the
format 'yyyy/mm/dd'.");
                    e.Cancel = true; // Cancel the event to prevent focus change
                }
                else
                {
                    // Clear the error message for the control
                    errorProvider2.SetError(DateTxt, string.Empty);
                }
            }
        }

        private void label2_Click(object sender, EventArgs e)
        {


        }
```

```
private void ProductIdTxt_TextChanged(object sender, EventArgs e)
{

}

/* private void orderDetailsToolStripMenuItem_Click(object sender, EventArgs e)
  {

  }*/

private void SupplierIdTxt_TextChanged(object sender, EventArgs e)
{

}

private void menuStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
{

}

private void SupplierProduct_Load_1(object sender, EventArgs e)
{

}
```

```
}

private void supplierDetailToolStripMenuItem_Click(object sender, EventArgs e)
{
    SupplierDetails form = new SupplierDetails();
    form.ShowDialog();
}

private void supplierProductToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    SupplierProduct form = new SupplierProduct();
    form.ShowDialog();
}



private void reportToolStripMenuItem_Click(object sender, EventArgs e)
{
    report Form = new report();
    Form.ShowDialog();
}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    About Form = new About();
    Form.ShowDialog();
}

private void exitToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    ExitApplication();
}
private void ExitApplication()
```

```
        {



            DialogResult result = MessageBox.Show("Are you sure you want to exit?", "Exit
Application", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                Application.Exit();
            }

        }

        private void orderManagementToolStripMenuItem_Click(object sender, EventArgs e)
        {
            orderdetail form = new orderdetail();
            form.ShowDialog();
        }

        private void customerManagementToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            CustomerDetail form = new CustomerDetail();
            form.ShowDialog();
        }

        private void button1_Click(object sender, EventArgs e)
        {

            PrintDocument printDocument = new PrintDocument();
            printDocument.PrintPage += new
PrintPageEventHandler(PrintDocument_PrintPage);
```

```
    try
    {
        PrintDialog printDialog = new PrintDialog();
        printDialog.Document = printDocument;

        if (printDialog.ShowDialog() == DialogResult.OK)
        {
            printDocument.Print();
            MessageBox.Show("Invoice printed successfully.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error printing invoice: {ex.Message}");
    }
}
private void PrintDocument_PrintPage(object sender, PrintPageEventArgs e)
{
    // Define the font, brush, and string format for drawing text
    Font font = new Font("Arial", 12);
    SolidBrush brush = new SolidBrush(Color.Black);
    StringFormat stringFormat = new StringFormat();

    // Define the invoice details to print
    string invoiceDetails =
      $"Invoice Details:\n" +
                  $"Supplier ID: {SupplierIdTxt.Text}\n" +
                  $"Product ID: {ProductIdTxt.Text}\n" +
                  $"Product Category: {categoryTxt.Text}\n" +
                    $"Subcategory: {SubcategoryTxt.Text}\n" +
                    $"Quantity: {QuantityTxt.Text}\n" +
                    $"Price: {PriceTxt.Text}\n" +
                    $"Total: {TotalTxt.Text}\n" +
                    $"Date: {DateTxt.Text}\n";
            // Define the rectangle to draw the invoice details
            RectangleF rectangle = new RectangleF(100, 100, 400, 300); // Adjust the position
and size as needed

            // Draw the invoice details on the print document
            e.Graphics.DrawString(invoiceDetails, font, brush, rectangle, stringFormat);
        }



        private void Search_Click(object sender, EventArgs e)
        {
            string searchText = searchtxt.Text.Trim();

            if (string.IsNullOrEmpty(searchText))
            {
                MessageBox.Show("Please enter a search term.", "Search Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            string connectionString =
"server=localhost;uid=root;pwd=Aastha@1978;database=betterbebutter15";
            using (MySqlConnection conn = new MySqlConnection(connectionString))
            {
                try
                {
```

```
                conn.Open();
                string query = "SELECT * FROM Customers WHERE CustomerName LIKE
@searchText";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@searchText", "%" + searchText + "%");

                MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                adapter.Fill(dt);

                if (dt.Rows.Count > 0)
                {
                    SupplierProductDataGridView.DataSource = dt;
                }
                else
                {
                    MessageBox.Show("No records found.", "Search Result",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    SupplierProductDataGridView.DataSource = null; // Clear the DataGridView
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error searching data: " + ex.Message, "Search Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    private void categoryTxt_SelectedIndexChanged(object sender, EventArgs e)
    {
```

```
SubcategoryTxt.Items.Clear();
if (categoryTxt.SelectedItem != null)
{
    // Get the selected category from CategoryComboBox
    string selectedCategory = categoryTxt.SelectedItem.ToString();

    // Populate SubcategoryTxt based on the selected categoryTxt
    switch (categoryTxt.SelectedItem.ToString())
    {
        case "Flour and Baking Ingredients":
            SubcategoryTxt.Items.Add("All-purpose Flour");
            SubcategoryTxt.Items.Add("Cake Flour");
            SubcategoryTxt.Items.Add("Bread Flour");
            SubcategoryTxt.Items.Add("Whole Wheat Flour");
            SubcategoryTxt.Items.Add("Gluten-free Flour");
            SubcategoryTxt.Items.Add("Sugar");
            SubcategoryTxt.Items.Add("Brown Sugar");
            SubcategoryTxt.Items.Add("Powdered Sugar");
            SubcategoryTxt.Items.Add("Baking Powder");
            SubcategoryTxt.Items.Add("Baking Soda");
            SubcategoryTxt.Items.Add("Yeast");
            SubcategoryTxt.Items.Add("Cocoa Powder");
            SubcategoryTxt.Items.Add("Cornstarch");
            break;
        case "Flavorings and Extracts":
            SubcategoryTxt.Items.Add("Vanilla Extract");
            SubcategoryTxt.Items.Add("Almond Extract");
            SubcategoryTxt.Items.Add("Lemon Extract");
            SubcategoryTxt.Items.Add("Peppermint Extract");
            SubcategoryTxt.Items.Add("Orange Extract");
    SubcategoryTxt.Items.Add("Food Coloring");
    SubcategoryTxt.Items.Add("Maple Flavoring");
    SubcategoryTxt.Items.Add("Coconut Extract");
    break;
case "Decorations and Toppings":
    SubcategoryTxt.Items.Add("Sprinkles");
    SubcategoryTxt.Items.Add("Edible Glitters");
    SubcategoryTxt.Items.Add("Chocolate Chips");
    SubcategoryTxt.Items.Add("Candied Fruit");
    SubcategoryTxt.Items.Add("Nuts and Seeds");
    SubcategoryTxt.Items.Add("Fondant");
    SubcategoryTxt.Items.Add("Whipped Cream");
    SubcategoryTxt.Items.Add("Icing");
    break;
case "Packaging and Display":
    SubcategoryTxt.Items.Add("Cake Boxes");
    SubcategoryTxt.Items.Add("Cupcake Liners");
    SubcategoryTxt.Items.Add("Cake Boards");
    SubcategoryTxt.Items.Add("Display Stands");
    SubcategoryTxt.Items.Add("Pastry Bags");
    SubcategoryTxt.Items.Add("Decorative Ribbons");
    SubcategoryTxt.Items.Add("Tissue Paper");
    SubcategoryTxt.Items.Add("Cake Dummies");
    break;
case "Preservatives and Stabilizers":
    SubcategoryTxt.Items.Add("Preservatives");
    SubcategoryTxt.Items.Add("Stabilizers");
    SubcategoryTxt.Items.Add("Emulsifiers");
    SubcategoryTxt.Items.Add("Thickeners");
    SubcategoryTxt.Items.Add("Acids");
```

```
                SubcategoryTxt.Items.Add("Antioxidants");
                SubcategoryTxt.Items.Add("Gums");
                SubcategoryTxt.Items.Add("Mold Inhibitors");
                break;
            default:
                break;
        }
    }
}

private void categoryTxt_Validating(object sender, CancelEventArgs e)
{
    if (string.IsNullOrWhiteSpace(categoryTxt.Text))
    {
        errorProvider1.SetError(categoryTxt, "Category cannot be empty");
        e.Cancel = true;
    }
    else
    {
        errorProvider1.SetError(categoryTxt, string.Empty);
    }
}

private void SubcategoryTxt_Validating(object sender, CancelEventArgs e)
{
    if (string.IsNullOrWhiteSpace(SubcategoryTxt.Text))
    {
        errorProvider1.SetError(SubcategoryTxt, "Sub-Category cannot be empty");
        e.Cancel = true;
    }
```

```
            else
            {
                errorProvider1.SetError(SubcategoryTxt, string.Empty);
            }
        }
    }
}
```

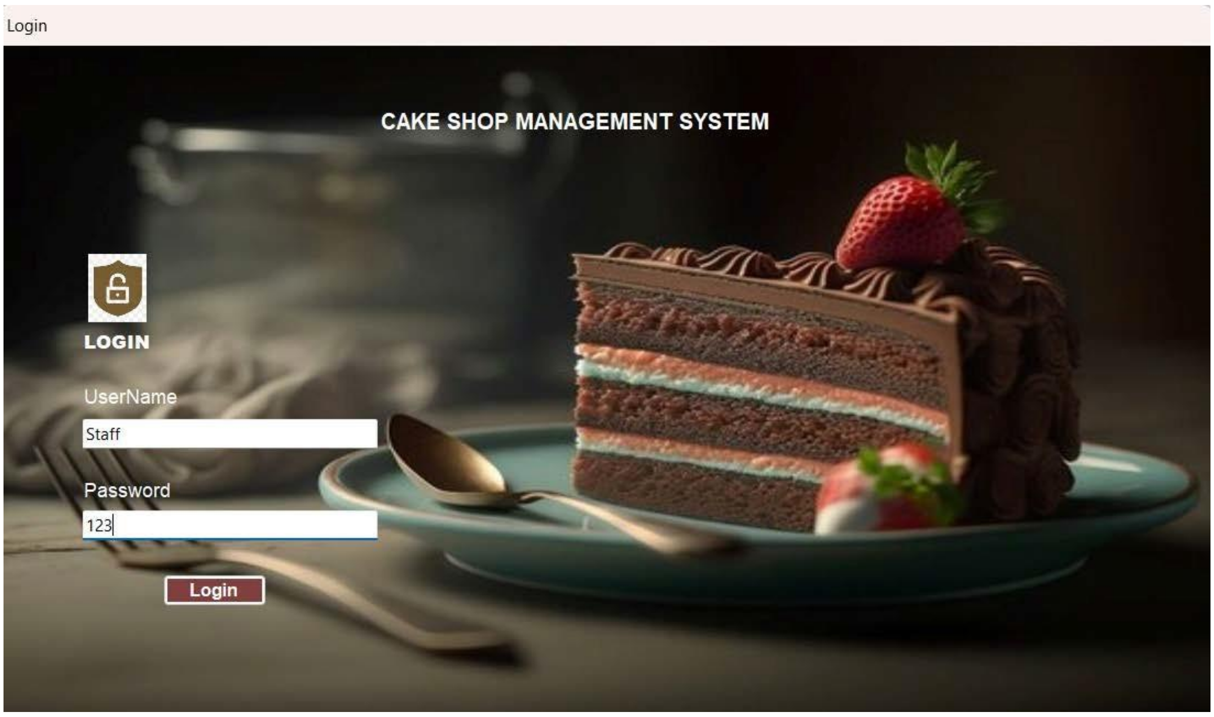
ABOUT

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Betterbebutter15
{
    public partial class About : Form
    {
        public About()
        {
            InitializeComponent();
        }
```

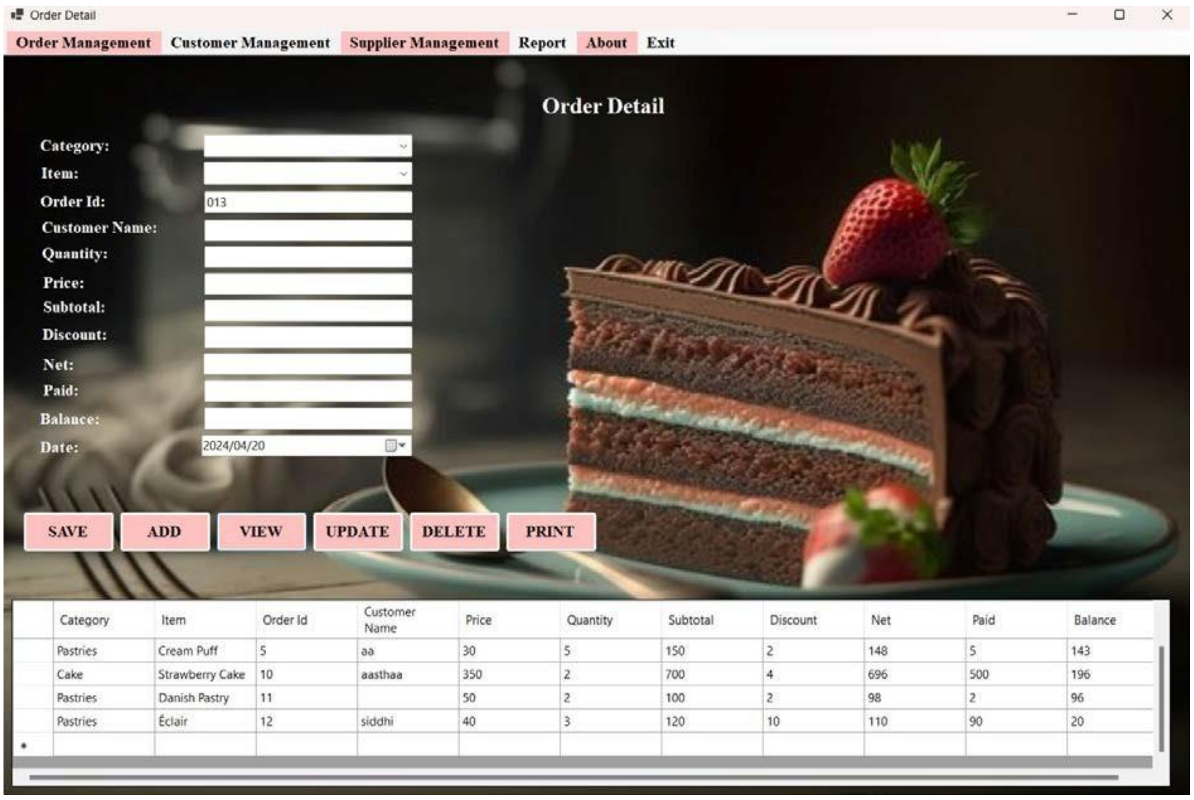
```
private void supplierDetailToolStripMenuItem_Click(object sender, EventArgs e)
{
    SupplierDetails form = new SupplierDetails();
    form.ShowDialog();
}

private void supplierProductToolStripMenuItem_Click(object sender, EventArgs e)
{
    SupplierProduct form = new SupplierProduct();
    form.ShowDialog();
}


private void reportToolStripMenuItem_Click(object sender, EventArgs e)
{
    report Form = new report();
    Form.ShowDialog();
}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    About Form = new About();
    Form.ShowDialog();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
```

```
            ExitApplication();
        }
        private void ExitApplication()
        {



            DialogResult result = MessageBox.Show("Are you sure you want to exit?", "Exit
Application", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                Application.Exit();
            }

        }

        private void orderManagementToolStripMenuItem_Click(object sender, EventArgs e)
        {
            orderdetail form = new orderdetail();
            form.ShowDialog();
        }

        private void customerManagementToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            CustomerDetail form = new CustomerDetail();
            form.ShowDialog();
        }
    }
}
```

TEST CASE



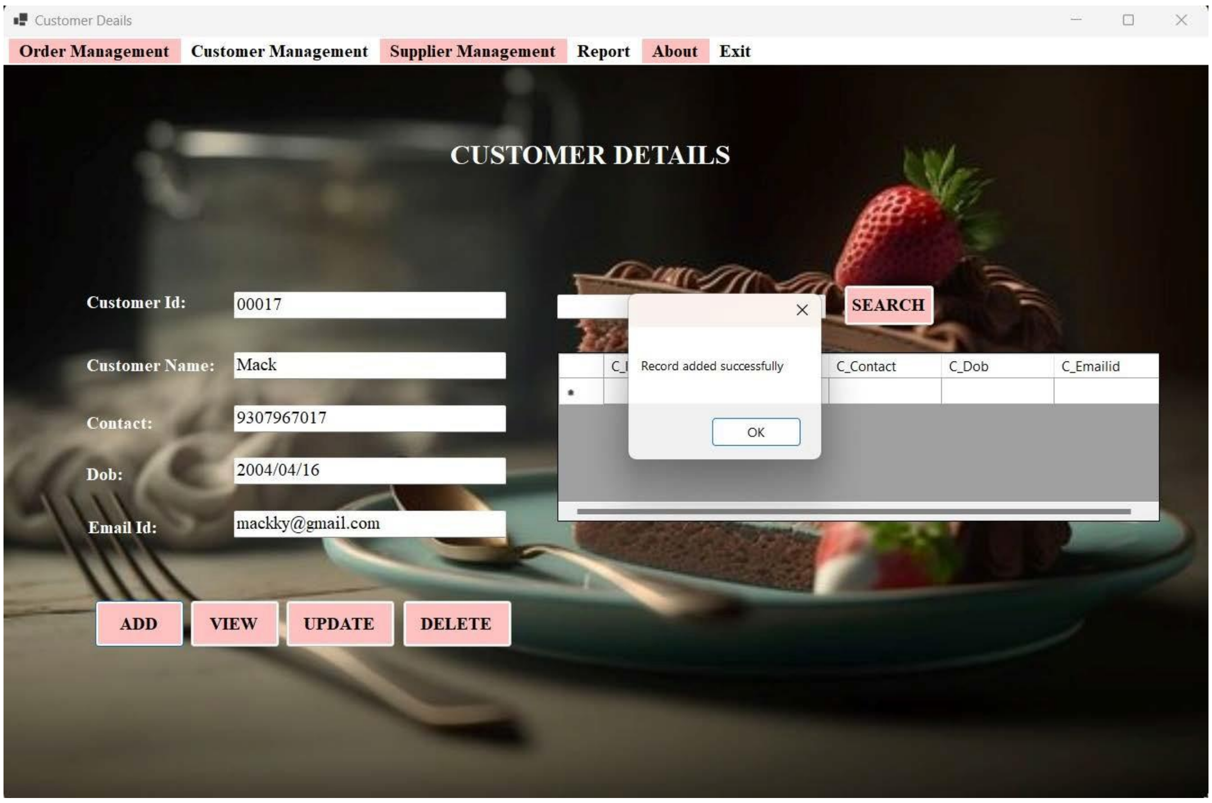
UserName: Staff
Password: 123



Order Detail

Category:	
Item:	
Order Id:	013
Customer Name:	
Quantity:	
Price:	
Subtotal:	
Discount:	
Net:	
Paid:	
Balance:	
Date:	2024/04/20

	Category	Item	Order Id	Customer Name	Price	Quantity	Subtotal	Discount	Net	Paid	Balance
	Pastries	Cream Puff	5	aa	30	5	150	2	148	5	143
	Cake	Strawberry Cake	10	aasthaa	350	2	700	4	696	500	196
	Pastries	Danish Pastry	11		50	2	100	2	98	2	96
	Pastries	Éclair	12	siddhi	40	3	120	10	110	90	20
*											

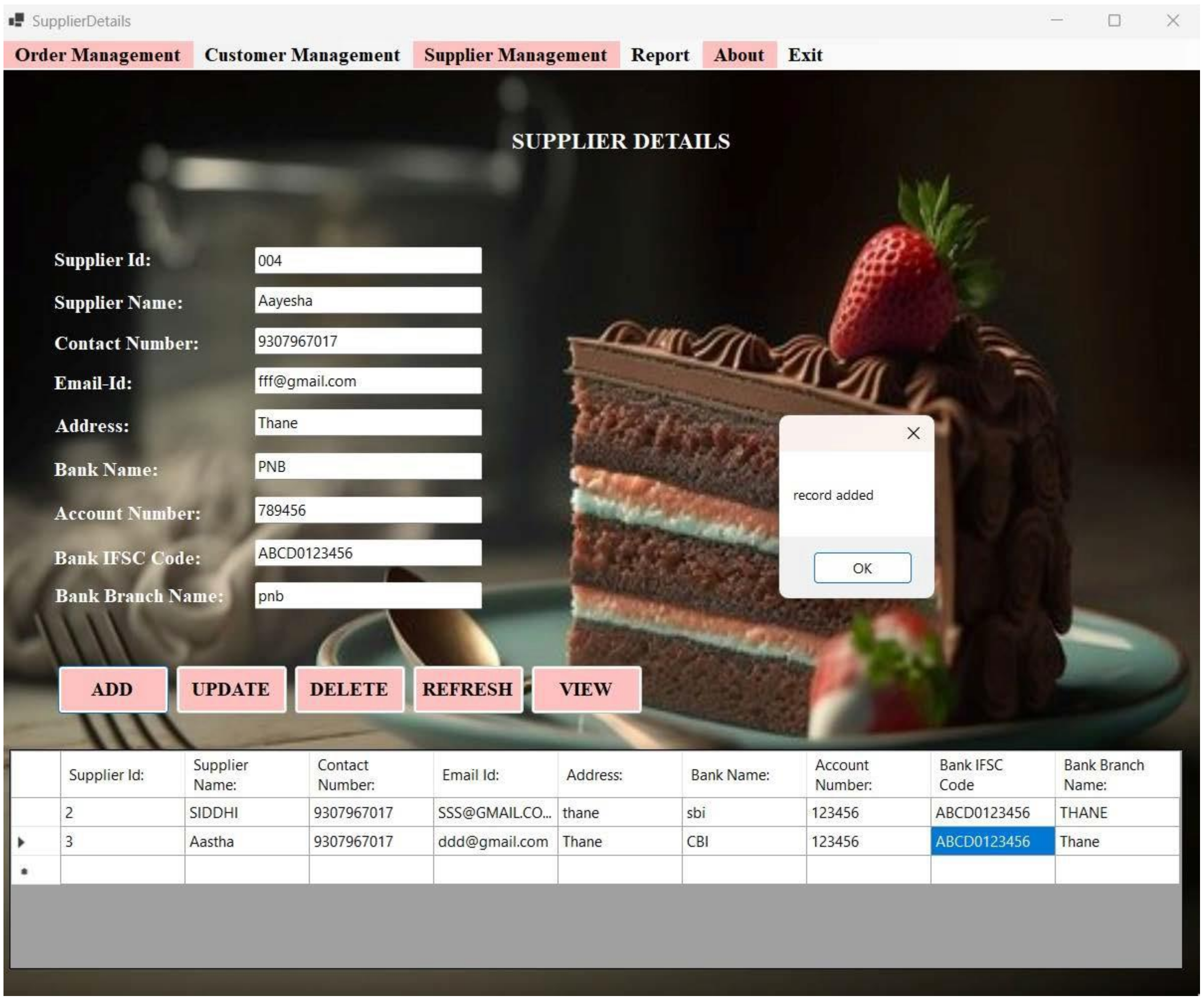


CUSTOMER DETAILS

Customer Id:	00017
Customer Name:	Mack
Contact:	9307967017
Dob:	2004/04/16
Email Id:	mackky@gmail.com

Record added successfully

C_I	C_Contact	C_Dob	C_Emailid



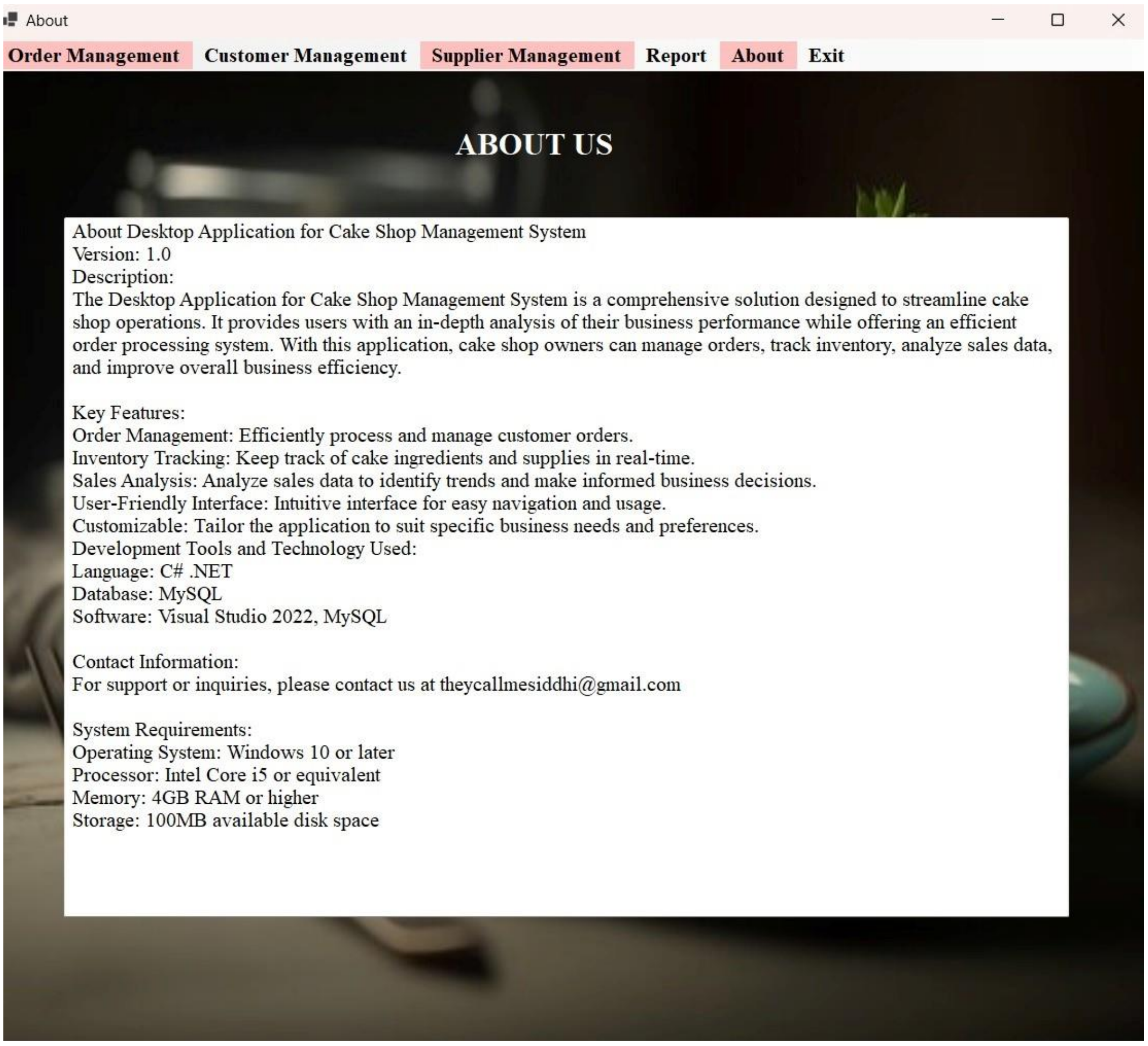
SUPPLIER DETAILS

Supplier Id:	004
Supplier Name:	Aayesha
Contact Number:	9307967017
Email-Id:	fff@gmail.com
Address:	Thane
Bank Name:	PNB
Account Number:	789456
Bank IFSC Code:	ABCD0123456
Bank Branch Name:	pnb

record added

Supplier Id:	Supplier Name:	Contact Number:	Email Id:	Address:	Bank Name:	Account Number:	Bank IFSC Code	Bank Branch Name:
2	SIDDHI	9307967017	SSS@GMAIL.CO...	thane	sbi	123456	ABCD0123456	THANE
3	Aastha	9307967017	ddd@gmail.com	Thane	CBI	123456	ABCD0123456	Thane

SYSTEM REQUIREMENTS



ABOUT US

About Desktop Application for Cake Shop Management System
Version: 1.0
Description:
The Desktop Application for Cake Shop Management System is a comprehensive solution designed to streamline cake shop operations. It provides users with an in-depth analysis of their business performance while offering an efficient order processing system. With this application, cake shop owners can manage orders, track inventory, analyze sales data, and improve overall business efficiency.

Key Features:
Order Management: Efficiently process and manage customer orders.
Inventory Tracking: Keep track of cake ingredients and supplies in real-time.
Sales Analysis: Analyze sales data to identify trends and make informed business decisions.
User-Friendly Interface: Intuitive interface for easy navigation and usage.
Customizable: Tailor the application to suit specific business needs and preferences.
Development Tools and Technology Used:
Language: C# .NET
Database: MySQL
Software: Visual Studio 2022, MySQL

Contact Information:
For support or inquiries, please contact us at theycallmesiddhi@gmail.com

System Requirements:
Operating System: Windows 10 or later
Processor: Intel Core i5 or equivalent
Memory: 4GB RAM or higher
Storage: 100MB available disk space

CONCLUSION

It is concluded that the application works well and satisfies the requirements of the end-users. Thorough testing of the application has been done to ensure the quality and it is user friendly.

The owner can easily understand how the whole system is implemented by going through the documentation. The system is tested, implemented and the performance is found to be satisfactory. All necessary output is generated as needed.

The Cake Shop Management system has a facility to generate bills and only the authorized user can login in system and view details of customers and orders. This application Provides facility for adding customer details (for reference), adding product details and it automatically calculates amount and generate bill.

It also provides Automation of the entire system improves the efficiency.

The system has adequate scope for modification in Future if it is necessary. Thus, the project is completed successfully.

SCOPE FOR FUTURE ENHANCEMENT

There is scope for future development of this project. The world of computer fields is not static; it is always subject to be dynamic. The technology which is famous today becomes outdated the very next day. To keep abreast of technical improvements, the system may be further refined.

Future Modifications:

- We will add courses and workshops on cake baking.
- In future if business grows, we will add more branches and more varieties of products in our application.
- We will try to make this system online.

BIBLIOGRAPHY

www.google.com

www.youtube.com

https://youtu.be/qjddNvxKPpg

https://youtu.be/YhAwNITpnno

https://youtu.be/1EpYqtSlOr8

https://youtu.be/B2KZ96ja9FA