VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL TEST PROJECT**

**SOFTWARE ENGINEERING**

**Supplement Facts Ordering System**

*Instruction*: **Lecturer PHAM THAI KY TRUNG**

*Student*: **HO GIA KHIEM – 520K0341**

Class: **20K50301**

Enrollment Course: **24**

**HO CHI MINH CITY, 2022**

VIETNAM GENEFAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL TEST PROJECT**

**SOFTWARE ENGINEERING**

**Supplement Facts Ordering System**

Instructor: **Lecturer** **PHAM THAI KY TRUNG**

Student: **HO GIA KHIEM**

Class: **20K50301**

Enrollment Course: **24**

**HO CHI MINH CITY, 2022**

ACKNOWLEDGEMENT

I would like to thank the lecturer for providing necessary documents and materials for this project.

SUMMARY

An order management system is a computer software system used in a number of industries for order entry and processing.

Orders can be received from businesses, consumers, or a mix of both, depending on the products. Offers and pricing may be done via catalogs, websites, or broadcast network advertisements.

An integrated order management system may encompass these modules:

- Product information (descriptions, attributes, locations, quantities)

- Inventory available to promise (ATP) and sourcing

- Vendors, purchasing and receiving

- Marketing (catalogs, promotions, pricing)

- Customers and prospects

- Order entry and customer service (including returns and refunds)

- Financial processing (credit cards, billing, payment on account)

- Order processing (selection, printing, picking, packing, shipping)

Order management requires multiple steps in a sequential process like capture, validation, fraud check, payment authorization, sourcing, backorder management, pick, pack, ship and associated customer communications. Order management systems usually have workflow capabilities to manage this process. This time, we will demonstrate this with a supplement facts ordering system.

TABLE OF CONTENTS

*Pages*

Acknowledgement .........................................................................................................................3

Summary ........................................................................................................................................4

List of figures and tables ................................................................................................................7

***Contents***

**1. Introduction ...............................................................................................................................8**

1.1. Purpose and Scope ........................................................................................................8

1.2. Product Overview .........................................................................................................8

1.3. Structure of the Document ............................................................................................8

1.4. Terms, Acronyms and Abbreviations ...........................................................................9

**2. Project Management Plan .........................................................................................................9**

2.1. Project Organization .....................................................................................................9

2.2. Used Lifecycle Model ...................................................................................................9

2.3. Risk Analysis ..............................................................................................................10

2.4. Hardware and Software Resource Requirements ........................................................10

2.5. Deliverables and Schedule ..........................................................................................10

2.6. Monitoring, Reporting and Controlling Mechanisms .................................................10

2.7. Professional Standards ................................................................................................11

2.8. Evidence that all the artifacts have been placed under the configuration management ............................................................................................................................................11

2.9. Impact of the project on individuals and organizations ...............................................11

**3. Requirement Specifications ....................................................................................................11**

3.1. Stakeholders for the system ........................................................................................11

3.2. Use case model ...........................................................................................................12

3.2.1. Graphical use case model .............................................................................12

3.2.2. Textual Description for each use case ..........................................................12

3.3. Functional requirements .............................................................................................13

3.4. Non-functional requirements ......................................................................................13

**4. Architecture .............................................................................................................................14**

4.1. Used architectural styles .............................................................................................14

4.2. Architectural model ....................................................................................................14

4.3. Technology, software and hardware used ...................................................................14

4.4. Rationale for the architectural style and model ...........................................................14

**5. Design .......................................................................................................................................15**

5.1. Database design ..........................................................................................................15

5.2. Static model (class diagrams) ......................................................................................15

5.3. Dynamic model (sequence diagrams) .........................................................................16

5.4. Rationale for the detailed design model ......................................................................16

5.5. Traceability from requirements to detailed design model ...........................................16

**6. Test Plan ...................................................................................................................................17**

6.1. Requirements/specification-based system level test cases ..........................................17

6.2. Traceability of test cases to use-cases .........................................................................17

6.3. Techniques used for the test generation ......................................................................17

6.4. Assessment of the goodness of the test suite ...............................................................17

**7. Demonstration .........................................................................................................................17**

7.1. Database .....................................................................................................................17

7.2. Source code .................................................................................................................18

7.3. Testing ........................................................................................................................47

**LIST OF FIGURES AND TABLES**

*Pages*

**FIGURES**

**Figure 1.** The structure of the document ........................................................................................9

**Figure 2.** The waterfall lifecycle model for the supplement facts ordering system .....................10

**Figure 3.** The use case model for the supplement facts ordering system .....................................12

**Figure 4.** The architectural model for the supplement facts ordering system ..............................14

**Figure 5.** The database design for the supplement facts ordering system ....................................15

**Figure 6.** The class diagram for the supplement facts ordering system .......................................16

**Figure 7.** The sequence diagram for the supplement facts ordering system ................................16

**TABLES**

**Table 1.** Use case description for the supplement facts ordering system .....................................13

**Table 2.** Rationale for the architectural style ...............................................................................14

**Table 3.** Rationale for the architectural model .............................................................................14

**1. Introduction**

**1.1. Purpose and Scope**

The main purpose of the supplement facts ordering system is to provide customers a way to place an order of supplement fact products online.

The proposed system is a software solution for users to easily add and take supplement fact orders.

**1.2. Product Overview**

**(a) *Capabilities***

These are the capabilities of a supplement facts ordering system:

- Allow customers to place their own orders.

- Help companies communicate with their customers easily over the Internet.

**(b) *Scenarios for using the product***

**Goal 1.** The system software has a user-friendly user interface.

**Scenario 1.1.** A particular user can easily read the text on the display.

**Scenario 1.2.** The software is controllable and usable.

**Scenario 1.3.** Product choices are presented in the form of buttons, which contain texts as well as little pictures illustrating the choice for better understanding.

**Goal 2.** The system software takes order from the customer as their choice.

**Scenario 2.1.** Users select their desired supplement fact products.

**Goal 3.** The system software calculates and displays the bill based on the placed order.

**Scenario 3.1.** The software calculates the bill based on the quantity of the products multiplied by their unit price topped up by the shipping fee (if any).

**Scenario 3.2.** The user is given the option to either pay for the order or revise the order.

**Goal 4.** The system software handles the payment for the user-defined order.

**Scenario 4.1.** The software will ask users to enter their cash in the box if the user decides to pay with cash.

**Scenario 4.2.** The software verifies the cash amount and gives refund if any after deducting the amount.

**Scenario 4.3.** The user decides to pay with their credit/debit card. Therefore, the software will ask the user to provide their credit/debit card information.

**Scenario 4.4.** The software verifies the credit card and charges the amount of the bill to the credit/debit card.

**Scenario 4.5.** The software will display the receipt containing a token number, details of the order, bill and the payment method.

**Scenario 4.6.** The software communicates the order to the company through the internal ordering system.

**Goal 5.** The system software offers the choice to change the menu products to the product manager.

**Scenario 5.1.** The product manager decides to add/delete a product from the menu.

**Scenario 5.2.** The product manager wants to put festive offers on some items because of which there is a change in the price of some products that are declared by the company themselves.

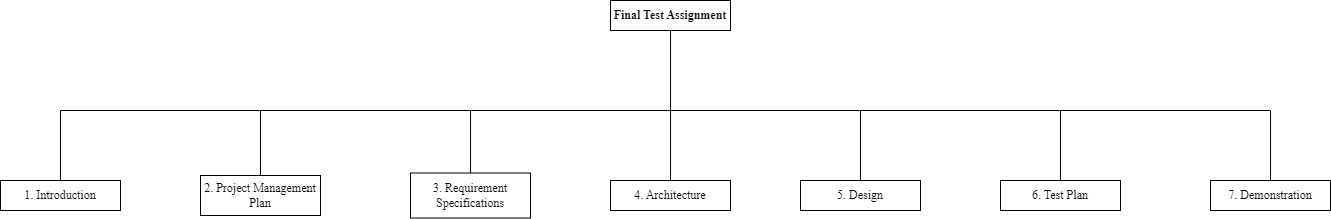
**Scenario 5.3.** The product manager notices that some supplement fact products are out of stock. Consequently, the product manager updates the menu so that those products will temporarily display the following message: “Out of Stock” below the product image to the user.

**Goal 6.** The system software is resistant to active/rigorous handling.

**Scenario 6.1.** The customer can be a minor or hacker who might want to get into the system to change it. Therefore, it should provide security for these areas.

**Scenario 6.2.** The system software will be used by different users and might experience some rough hand every now and then.

**1.3. Structure of the Document**



**Figure 1.** The structure of the document

**1.4. Terms, Acronyms and Abbreviations**

There are no terms, acronyms or abbreviations available for this document.

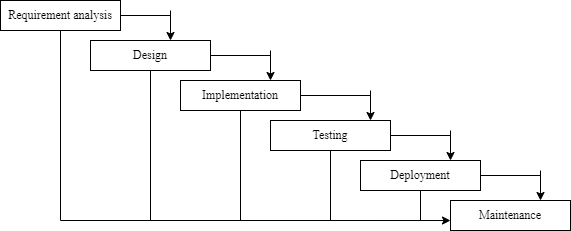
**2. Project Management Plan**

**2.1. Project Organization**

This project is done individually.

**2.2. Used Lifecycle Model**

The used lifecycle model is the waterfall model.



**Figure 2.** The waterfall lifecycle model for the supplement facts ordering system

**2.3. Risk Analysis**

Getting started with the supplement facts ordering system isn’t just as easy as taking orders and driving them to people’s houses. Many of the choices companies need to make come with risks that need to be mitigated. The way that companies address these risks can make a huge difference on how profitable the delivery services will be. Therefore, the supplement fact products ordering system risk analysis is:

- Customer privacy & data

- Product safety

- Product quality

- Database management and operation

**2.4. Hardware and Software Resource Requirements**

This software can be used and managed on a computer. For the software resource requirements, the latest compatible version of the computer’s operating system is not specified, but strongly recommended.

**2.5. Deliverables and Schedule**

Consumers will choose their products in the system software. There is a choice for scheduled order at the checkout page. The customer should mention the date and time for the delivery of products. Now, for instance, a customer ensures that the supplement fact products should delivered to his/her house on the date and time which is mentioned earlier. Therefore, the system software will display a notification about the order ahead of time. It is shown in the top of the system software’s dashboard page. At this time, the company will prepare the supplement fact products as per the order. In the meantime, the system software will allow the user to track his/her orders and will notify them if there is anything happens to the orders or their ordered products have arrived.

**2.6. Monitoring, Reporting and Controlling Mechanisms**

**Monitoring Mechanisms:**

- Customer’s personal information and payment gathering and progress monitoring.

**Reporting Mechanisms:**

- Information about customers’ placed orders.

**Controlling Mechanisms:**

- Process the customers’ orders.

**2.7. Professional Standards**

Professional standards define the nature and the quality of the software. The professional standards for this software are as follows:

- Efficiency and order processing.

- Completion of the customers’ orders.

**2.8. Evidence that all the artifacts have been placed under the configuration management**

The configuration management for this software is:

- Performance measurement

- Customer’s order analysis

- Processing of placed orders

- Data control

- Resources management

- Impact analysis

- Roles & responsibilities

The development of this software will be placed under the configuration management.

**2.9. Impact of the project on individuals and organizations**

These are the impact of the supplement facts ordering system software on individuals and organizations:

- The ordering process is easier.

- Efficient customer and order management.

- Monitor the expenses incurred in real time.

- Free and low-cost marketing.

- Better customers’ data analysis.

- Individuals and organizations can promote their supplement fact products to reach more customers.

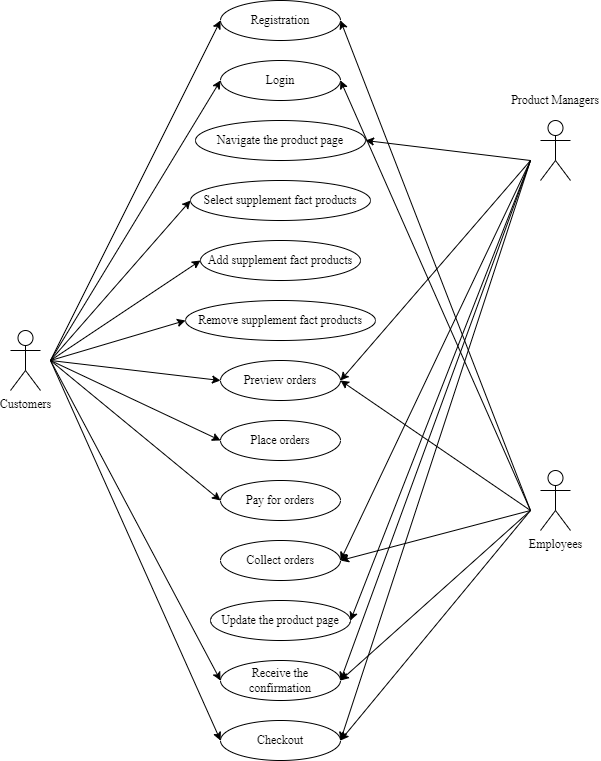
**3. Requirement Specifications**

**3.1. Stakeholders for the system**

There are no requirements for stakeholders for the system, as this project will be done individually.

**3.2. Use case model**

**3.2.1. Graphical use case model**



**Figure 3.** The use case model for the supplement facts ordering system

**3.2.2. Textual Description for each use case**

|  |  |
| --- | --- |
| **Use case** | **Description** |
| Actors | Customers, product managers, employees |
| Registration | Customers will create an account to shop supplement fact products, while employees will create an account to collect and manage customers’ information and orders. |
| Login | Both customers and employees will login to their accounts to respectively start shopping and managing information from customers. |
| Navigate the product page | The product managers will control the product page. |
| Select supplement fact products | While browsing the product page, customers will select the supplement fact product that they want to buy. |
| Add supplement fact products | After customers have chosen the supplement fact product that they want to buy, they can choose to buy it by adding it to their shopping cart. |
| Remove supplement fact products | While checking the shopping cart, if customers found that there are unwanted products inside, they can choose to remove them. |
| Preview orders | After shopping, customers can check the shopping cart to preview their orders. This allows customers to make sure that their orders only contain products they want to buy. For product managers and employees, the software will display orders in preview mode so that they can collect and process them. |
| Place orders | After previewing orders, customers will confirm their orders by placing them. |
| Pay for orders | After placing orders, customers will then be asked to enter their payment method information. Accepted payment methods can be cash, credit/debit card and digital wallet. |
| Collect orders | Product managers and employees will start collecting and processing customers’ orders after the orders have been submitted to the company. |
| Update the product page | While controlling the product page, product managers can make changes to it, for example, if they detected that some products are out of stock, sales are announced by the company, more products are released by the company or some products will no longer be produced and available. |
| Receive the confirmation | Customers will submit their orders to employees to confirm, verify and process them. |
| Checkout | Customers will confirm their orders, payment method information with the employees to complete the purchase. |

**Table 1.** Use case description for the supplement facts ordering system

**3.3. Functional requirements**

- The system software must have functions that allows users to view products and place orders. Note: There must be 7 – 10 available products from the company themselves.

- The ordered products will be delivered to the customer’s house.

- The system software must have a user-friendly user interface.

**3.4. Non-functional requirements**

- The system software will process the orders.

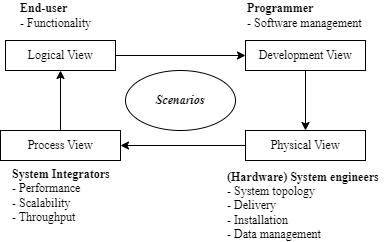
- The system software has the management dashboard that allows the company to monitor and process orders, as well as customers’ personal information

**4. Architecture**

**4.1. Used architectural styles**

The used architectural styles are layered style and object-oriented style.

**4.2. Architectural model**



**Figure 4.** The architectural model for the supplement facts ordering system

**4.3. Technology, software and hardware used**

- The C# programming language is used to develop this software.

- This software is developed and operated on a computer.

**4.4. Rationale for the architectural style and model**

**Rationale for the architectural style:**

|  |  |
| --- | --- |
| **Layered** | **Object-oriented** |
| The layered structure fashion is one of the maximum not unusual place architectural styles. The concept at the back of Layered Architecture is that modules or additives with comparable functionalities are prepared into horizontal layers. As a result, every layer plays a selected position inside the software. | Object-orientated architectural fashion maps the software program to real-global items for making it extra understandable. It is simple to preserve and improves the fine of the device because of software program reuse. This structure affords reusability through polymorphism and abstraction. |

**Table 2.** Rationale for the architectural style

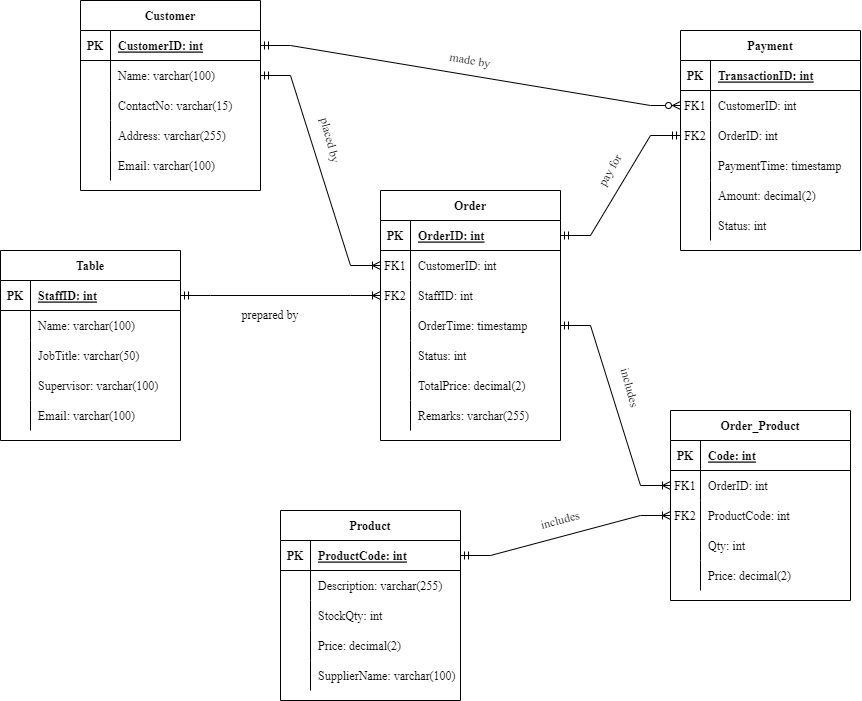
**Rationale for the architectural model:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Logical View** | **Development View** | **Process View** | **Physical View** |
| **End user:** Functionality | **Programmer:** Software management | **System Integrators:** Performance, scalability, throughput | **(Hardware) System engineers:** System topology, delivery, installation, data management |

**Table 3.** Rationale for the architectural model

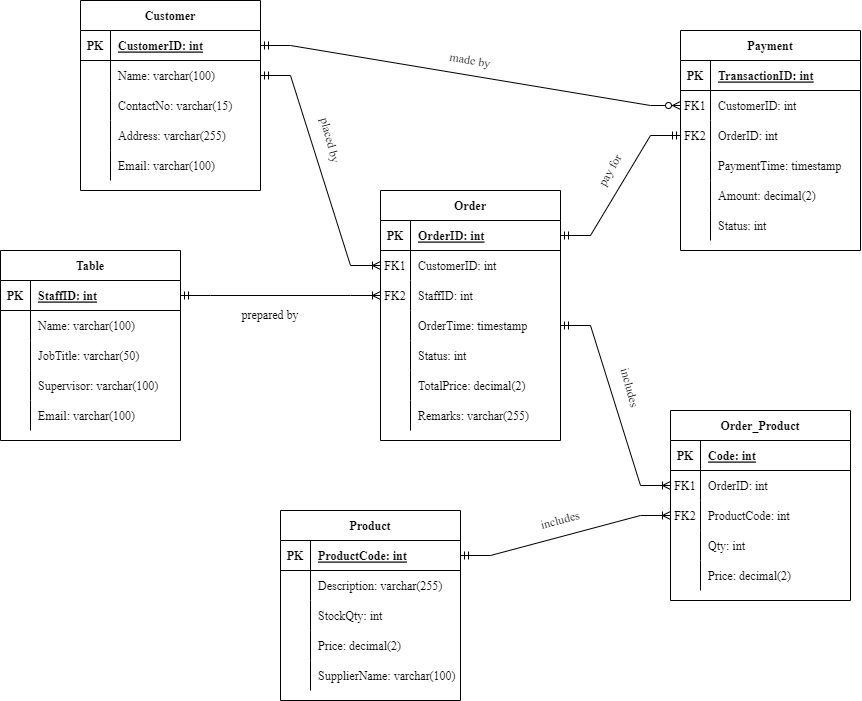
**5. Design**

**5.1. Database design**



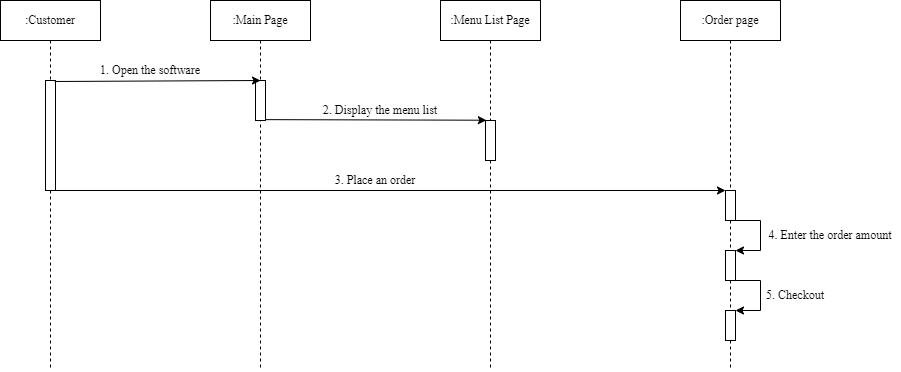
**Figure 5.** The database design for the supplement facts ordering system

**5.2. Static model (class diagrams)**



**Figure 6.** The class diagram for the supplement facts ordering system

**5.3. Dynamic model (sequence diagrams)**



**Figure 7.** The sequence diagram for the supplement facts ordering system

**5.4. Rationale for the detailed design model**

The detailed design model is made to ensure a good experience by customers.

**5.5. Traceability from requirements to detailed design model**

The detailed design model is decided based on the requirements of the software.

**6. Test Plan**

**6.1. Requirements/specification-based system level test cases**

**Test case description:**

The objective of the test case is to place an order to test the software.

**Prerequisites:**

**1.** An internet connection.

**2.** The personal information (such as name, date of birth, age) and payment methods (credit/debit card, cash).

**Test steps:**

**1.** Browse the product page.

**2.** Click “Add to cart” to add a product to the online shopping cart.

**3.** Navigate to the shopping cart.

**4.** Click “Checkout”.

**5.** Enter the payment method information and the personal information.

**Action result:**

**1.** If the entered personal information and the entered payment method information is valid, then the software will notify that the order has been placed and will be processed.

**2.** If there is any invalid information detected, then nothing will happen, and error messages will be displayed under the text boxes.

**Status:**

**Pass** (if all entered checkout information is valid).

**Fail** (if there is invalid information in the checkout information).

**6.2. Traceability of test cases to use cases**

**Test scenario:**

Place an order of supplement fact products.

**Test case ID:** TC\_neworder\_01.

**6.3. Techniques used for test case generation**

The used technique for test case generation is goal-oriented technique. The program is tested, given that the objective is defined.

**6.4. Assessment of the goodness of the test suite**

The goodness of the test suite can be determined by testing the software. If the results are good as expected, then the test suite is already good.

**7. Demonstration**

**7.1. Database**

create database FinalProject

use FinalProject

create table Items (

    ItemID char(10) primary key,

    ItemName char(50),

    Size char(10)

)

create table Agents (

    AgentID char(10) primary key,

    AgentName char(50),

    Add\_ress char(50)

)

create table Orders (

    OrderID char(10) primary key,

    OrderDate char(15),

    AgentID char(10)

    foreign key (AgentID) references Agents(AgentID)

)

create table OrderDetails (

    ID char(10) primary key,

    OrderID char(10),

    ItemID char(10),

    Quantity int,

    UnitAmount int

    foreign key (OrderID) references Orders(OrderID),

    foreign key (ItemID) references Items(ItemID)

)

**7.2. Source code** (in C#)

**Product Details:**

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;

using System.Data.SqlClient;

namespace FinalProject

{

public partial class Food\_Details : MetroFramework.Forms.MetroForm

{

public Food\_Details()

{

InitializeComponent();

}

private void Panel3\_Paint(object sender, PaintEventArgs e)

{

}

private void BtnShowAll\_Click(object sender, EventArgs e)

{

}

private void DgvFoodDetails\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void BtnCancel\_Click(object sender, EventArgs e)

{

txtFoodID.Text = null;

txtFoodName.Text = null;

}

private void Button1\_Click(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button2\_Click(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Button4\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button3\_Click(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Button6\_Click(object sender, EventArgs e)

{

Manage\_Food manage\_Food = new Manage\_Food();

manage\_Food.Show();

this.Hide();

}

private void Button5\_Click(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void Button9\_Click(object sender, EventArgs e)

{

this.Hide();

}

string constring = @"Data Source=LAPTOP-5S178792; Initial Calatog = FinalProject; Integrated Security = True;";

private void BtnShowAll\_Click\_1(object sender, EventArgs e)

{

string qry = "SELECT \* FROM Food ";

SqlDataAdapter DA = new SqlDataAdapter(qry, constring);

DataSet DS = new DataSet();

DA.Fill(DS, "Food");

dgvFoodDetails.DataSource = DS.Tables["Food"];

}

private void BtnSearch\_Click(object sender, EventArgs e)

{

int foodid = int.Parse(txtFoodID.Text);

string foodname = txtFoodName.Text;

string qry = "SELECT \* FROM Food where foodID = '"+foodid+"' or Name = '"+foodname+"'";

SqlDataAdapter DA = new SqlDataAdapter(qry, constring);

DataSet DS = new DataSet();

DA.Fill(DS, "Food");

dgvFoodDetails.DataSource = DS.Tables["Food"];

}

private void Button7\_Click(object sender, EventArgs e)

{

ManageOffers manageOffers = new ManageOffers();

manageOffers.Show();

this.Hide();

}

private void Button8\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void MetroButton1\_Click(object sender, EventArgs e)

{

Admin\_Dashboard admin\_Dashboard = new Admin\_Dashboard();

admin\_Dashboard.Show();

this.Hide();

}

}

}

**Manage Users:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FinalProject

{

public partial class Manage\_Users : MetroFramework.Forms.MetroForm

{

public Manage\_Users()

{

InitializeComponent();

}

private void Manage\_Users\_Load(object sender, EventArgs e)

{

}

private void BtnUpdate\_Click(object sender, EventArgs e)

{

}

private void Button1\_Click(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button2\_Click(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Button4\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button3\_Click(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Button6\_Click(object sender, EventArgs e)

{

Manage\_Food manage\_Food = new Manage\_Food();

manage\_Food.Show();

this.Hide();

}

private void Button5\_Click(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void Button7\_Click(object sender, EventArgs e)

{

}

private void Button8\_Click(object sender, EventArgs e)

{

}

private void Button9\_Click(object sender, EventArgs e)

{

this.Hide();

}

SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-5S178792; Initial Calatog = FinalProject; Integrated Security = True;");

private void BtnUpdate\_Click\_1(object sender, EventArgs e)

{

string nic = txtNIC.Text,

name = txtName.Text,

username = txtUname.Text,

email = txtEmail.Text,

password = txtPassword.Text,

confirmPassword = txtConfirmPassword.Text;

string address = txtAddress.Text;

int phoneNumber = int.Parse(txtPhoneNumber.Text);

if (txtNIC.Text == "" && txtPhoneNumber.Text == "")

{

MessageBox.Show("The NIC or phone number is missing.");

}

else

{

string qry = "Update userdetails Set Name='" + name + "' Uname = '"+username+"' Email = '"+email+"' Password = '"+password+"' ConfirmPassword = '"+confirmPassword+"' Address = '"+address+"' PhoneNumber = '"+phoneNumber+"' where NIC ='" + nic + "' or PhoneNumber='" + phoneNumber + "'";

SqlCommand cmd = new SqlCommand(qry,con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been updated.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

}

private void BtnDelete\_Click(object sender, EventArgs e)

{

string nic = txtNIC.Text;

string qry = "Delete \* from userdetails Where NIC = '" + nic+"' ";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been deleted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

private void BtnCancel\_Click(object sender, EventArgs e)

{

txtNIC.Text = null;

txtName.Text = null;

txtUname.Text = null;

txtEmail.Text = null;

txtPassword.Text = null;

txtConfirmPassword.Text = null;

txtConfirmPassword.Text = null;

txtAge.Text = null;

txtAddress.Text = null;

txtPhoneNumber = null;

}

private void MetroButton1\_Click(object sender, EventArgs e)

{

Admin\_Dashboard admin\_Dashboard = new Admin\_Dashboard();

admin\_Dashboard.Show();

this.Hide();

}

}

}

**Manage Products:**

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;

using System.Data.SqlClient;

namespace FinalProject

{

public partial class Manage\_Food : MetroFramework.Forms.MetroForm

{

public Manage\_Food()

{

InitializeComponent();

}

private void Panel3\_Paint(object sender, PaintEventArgs e)

{

}

private void BtnUserDetails\_Click(object sender, EventArgs e)

{

}

private void BtnAddFood\_Click(object sender, EventArgs e)

{

}

private void BtnManageFood\_Click(object sender, EventArgs e)

{

}

SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-5S178792; Initial Calatog = FinalProject; Integrated Security = True;");

private void BtnUpdate\_Click(object sender, EventArgs e)

{

}

private void BtnDelete\_Click(object sender, EventArgs e)

{

}

private void BtnCancel\_Click(object sender, EventArgs e)

{

txtFoodID.Text = null;

txtFoodName.Text = null;

txtPrice.Text = null;

}

private void BtnCancel\_Click\_1(object sender, EventArgs e)

{

txtFoodID.Text = null;

txtFoodName.Text = null;

txtPrice.Text = null;

}

private void Button1\_Click(object sender, EventArgs e)

{

}

private void Button2\_Click(object sender, EventArgs e)

{

}

private void Button4\_Click(object sender, EventArgs e)

{

}

private void Button3\_Click(object sender, EventArgs e)

{

}

private void Button6\_Click(object sender, EventArgs e)

{

}

private void Button5\_Click(object sender, EventArgs e)

{

}

private void Button5\_Click\_1(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void Button1\_Click\_1(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button2\_Click\_1(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Button4\_Click\_1(object sender, EventArgs e)

{

this.Hide();

}

private void Button3\_Click\_1(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Button6\_Click\_1(object sender, EventArgs e)

{

Manage\_Food manage\_Food = new Manage\_Food();

manage\_Food.Show();

this.Hide();

}

private void BtnUpdate\_Click\_1(object sender, EventArgs e)

{

int foodID = int.Parse(txtFoodID.Text);

string name = txtFoodName.Text;

double price = double.Parse(txtPrice.Text);

if (txtFoodName.Text == "" || txtFoodID.Text == null && !(txtPrice.Text == ""))

{

string qry = "Update Food Set price='" + price + "' where foodID='" + foodID + "' or Name='" + name + "'";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been inserted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

else

{

MessageBox.Show("Please fill in the required fields.");

}

}

private void BtnDelete\_Click\_1(object sender, EventArgs e)

{

if (txtFoodID.Text == "")

{

MessageBox.Show("The product ID is missing.");

}

else

{

int foodID = int.Parse(txtFoodID.Text);

string qry = "Delete from Food where FoodID = '" + foodID + "' ";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been deleted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

}

private void Button8\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button7\_Click(object sender, EventArgs e)

{

ManageOffers manageOffers = new ManageOffers();

manageOffers.Show();

this.Hide();

}

private void MetroButton1\_Click(object sender, EventArgs e)

{

Admin\_Dashboard admin\_Dashboard = new Admin\_Dashboard();

admin\_Dashboard.Show();

this.Hide();

}

private void Button9\_Click(object sender, EventArgs e)

{

this.Hide();

}

}

}

**Manage Offers:**

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;

using System.Data.SqlClient;

namespace FinalProject

{

public partial class ManageOffers : MetroFramework.Forms.MetroForm

{

public ManageOffers()

{

InitializeComponent();

}

private void ManageOffers\_Load(object sender, EventArgs e)

{

}

private void Button1\_Click(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button2\_Click(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Button4\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button3\_Click(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Button6\_Click(object sender, EventArgs e)

{

Manage\_Food manage\_Food = new Manage\_Food();

manage\_Food.Show();

this.Hide();

}

private void Button5\_Click(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void Button7\_Click(object sender, EventArgs e)

{

}

private void Button8\_Click(object sender, EventArgs e)

{

this.Hide();

}

SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-5S178792; Initial Calatog = FinalProject; Integrated Security = True;");

private void Button10\_Click(object sender, EventArgs e)

{

int offerID = int.Parse(txtOfferID.Text);

string offerName = txtOfferName.Text;

double price = Double.Parse(txtPrice.Text);

if (txtOfferID.Text == "" || txtOfferName.Text == "" || txtPrice.Text == "")

{

MessageBox.Show("Please fill in the required fields.");

}

else

{

string qry = "Insert Into offers values('" + offerID + "','" + offerName + "','" + price + "')";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been inserted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

finally

{

con.Close();

}

}

}

private void BtnUpdate\_Click(object sender, EventArgs e)

{

int offerID = int.Parse(txtOfferID.Text);

string offerName = txtOfferName.Text;

double price = Double.Parse(txtPrice.Text);

if (txtOfferID.Text == "" || txtOfferName.Text == "" || txtPrice.Text == "")

{

MessageBox.Show("Please fill in the required fields.");

}

else

{

string qry = "Update offers Set price = '"+price+"' Where OfferID = '"+offerID+"' or OfferName = '"+offerName+"' ";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been inserted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

}

private void BtnDelete\_Click(object sender, EventArgs e)

{

int offerID = int.Parse(txtOfferID.Text);

string offerName = txtOfferName.Text;

if (txtOfferID.Text == "" || txtOfferName.Text == "" )

{

MessageBox.Show("Please fill in the required fields.");

}

else

{

string qry = "Delete \* from offers Where OfferID = '"+offerID+"' or OfferName = '"+offerName+"' ";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been inserted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

}

private void BtnCancel\_Click(object sender, EventArgs e)

{

txtOfferID.Text = null;

txtOfferName.Text = null;

txtPrice = null;

}

private void Panel2\_Paint(object sender, PaintEventArgs e)

{

}

private void MetroButton1\_Click(object sender, EventArgs e)

{

Admin\_Dashboard admin\_Dashboard = new Admin\_Dashboard();

admin\_Dashboard.Show();

this.Hide();

}

private void Button9\_Click(object sender, EventArgs e)

{

this.Hide();

}

}

}

**Add Products:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FinalProject

{

public partial class Add\_food : MetroFramework.Forms.MetroForm

{

public Add\_food()

{

InitializeComponent();

}

private void BtnAdd\_Click(object sender, EventArgs e)

{

}

private void BtnManageFood\_Click(object sender, EventArgs e)

{

Manage\_Food mngFood = new Manage\_Food();

mngFood.Show();

this.Hide();

}

private void Add\_food\_Load(object sender, EventArgs e)

{

}

private void BtnAdd\_Click\_1(object sender, EventArgs e)

{

int foodID = int.Parse(txtFoodID.Text);

string name = txtFoodName.Text;

double price = double.Parse(txtPrice.Text);

if (txtFoodID.Text == "" || txtFoodName.Text == "" || txtPrice.Text == "")

{

MessageBox.Show("Fillout required fields");

}

else

{

SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-5S178792; Initial Calatog = FinalProject; Integrated Security = True;");

string qry = "Insert Into Food values('" + foodID + "','" + name + "','" + price + "')";

SqlCommand cmd = new SqlCommand(qry, con);

try

{

con.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("The data has been inserted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

}

private void BtnCancel\_Click(object sender, EventArgs e)

{

txtFoodID.Text = null;

txtFoodName.Text = null;

txtPrice.Text = null;

}

private void Button1\_Click(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button2\_Click(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Btnuserdetails\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Btnaddfood\_Click(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Btnmanagefood\_Click\_1(object sender, EventArgs e)

{

Manage\_Food mngFood = new Manage\_Food();

mngFood.Show();

this.Hide();

}

private void Btnfooddetails\_Click(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void Btnmanageoffers\_Click(object sender, EventArgs e)

{

ManageOffers manageOffers = new ManageOffers();

manageOffers.Show();

this.Hide();

}

private void Btnlogout\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Btnofferdetails\_Click(object sender, EventArgs e)

{

this.Show();

}

private void Panel2\_Paint(object sender, PaintEventArgs e)

{

}

private void MetroButton1\_Click(object sender, EventArgs e)

{

Admin\_Dashboard admin\_Dashboard = new Admin\_Dashboard();

admin\_Dashboard.Show();

this.Hide();

}

}

}

**Add Users:**

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;

using System.Data.SqlClient;

namespace FinalProject

{

public partial class AddUsers : MetroFramework.Forms.MetroForm

{

string ValueOfCombo;

public AddUsers()

{

InitializeComponent();

}

private void BtnAdd\_Click(object sender, EventArgs e)

{

string nic = txtNIC.Text,

name = txtName.Text,

username = txtUname.Text,

email = txtEmail.Text,

password = txtPassword.Text,

confirmPassword = txtConfirmPassword.Text;

int age = int.Parse(txtAge.Text);

string address = txtAddress.Text;

int phoneNumber = int.Parse(txtPhoneNumber.Text);

if (txtNIC.Text == "" || txtName.Text == "" || txtUname.Text == "" || txtPassword.Text == "" || txtConfirmPassword.Text == "" || txtAddress.Text == "" || txtPhoneNumber.Text == "" || ValueOfCombo == null)

{

MessageBox.Show("Please fill in the required fields.");

}

else

{

if (txtPassword.Text == txtConfirmPassword.Text)

{

string qry = "Insert Into userdetails Values('" + nic+"','"+name+"','"+username+"','"+email+"','"+password+"','"+confirmPassword+"','"+age+"','"+address+"','"+phoneNumber+ "','" + ValueOfCombo + "')";

try

{

MessageBox.Show("The data has been inserted.");

}

catch (SqlException sqlex)

{

MessageBox.Show("" + sqlex);

}

}

else

{

MessageBox.Show("The passwords don't match.");

}

}

}

private void Button2\_Click(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Button1\_Click(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button4\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button3\_Click(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Button6\_Click(object sender, EventArgs e)

{

Manage\_Food manage\_Food = new Manage\_Food();

manage\_Food.Show();

this.Hide();

}

private void Button5\_Click(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void Button7\_Click(object sender, EventArgs e)

{

ManageOffers manageOffers = new ManageOffers();

manageOffers.Show();

this.Hide();

}

private void Button8\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button9\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void BtnCancel\_Click(object sender, EventArgs e)

{

txtNIC.Text = null;

txtName.Text = null;

txtUname.Text = null;

txtEmail.Text = null;

txtPassword.Text = null;

txtConfirmPassword.Text = null;

txtAge.Text = null;

txtAddress.Text = null;

txtPhoneNumber.Text = null;

}

private void MetroButton1\_Click(object sender, EventArgs e)

{

Admin\_Dashboard admin\_Dashboard = new Admin\_Dashboard();

admin\_Dashboard.Show();

this.Hide();

}

private void CmbType\_SelectedIndexChanged(object sender, EventArgs e)

{

ValueOfCombo = cmbType.Text;

}

}

}

**Admin Dashboard:**

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 FinalProject

{

public partial class Admin\_Dashboard : MetroFramework.Forms.MetroForm

{

public Admin\_Dashboard()

{

InitializeComponent();

}

private void BtnAddUser\_Click(object sender, EventArgs e)

{

AddUsers addUser = new AddUsers();

addUser.Show();

this.Hide();

}

private void Button1\_Click(object sender, EventArgs e)

{

Manage\_Users manage\_Users = new Manage\_Users();

manage\_Users.Show();

this.Hide();

}

private void Button2\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void Button5\_Click(object sender, EventArgs e)

{

Add\_food add\_Food = new Add\_food();

add\_Food.Show();

this.Hide();

}

private void Button4\_Click(object sender, EventArgs e)

{

Manage\_Food manage\_Food = new Manage\_Food();

manage\_Food.Show();

this.Hide();

}

private void BtnFoodDetails\_Click(object sender, EventArgs e)

{

Food\_Details food\_Details = new Food\_Details();

food\_Details.Show();

this.Hide();

}

private void BtnManageOffers\_Click(object sender, EventArgs e)

{

ManageOffers manageOffers = new ManageOffers();

manageOffers.Show();

this.Hide();

}

private void MetroUserControl1\_Load(object sender, EventArgs e)

{

}

private void BtnOfferDetails\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void BtnLogout\_Click(object sender, EventArgs e)

{

this.Hide();

}

}

}

**Cashier:**

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;

using System.Data.SqlClient;

namespace FinalProject

{

public partial class Cashier : MetroFramework.Forms.MetroForm

{

DataTable dt;

public Cashier()

{

InitializeComponent();

dt = new DataTable("OrderTable");

dt.Columns.Add("ID");

dt.Columns.Add("Name");

dt.Columns.Add("Quantity");

dt.Columns.Add("Price");

}

private void Cashier\_Load(object sender, EventArgs e)

{

timer1.Start();

lblDate.Text = DateTime.Now.ToLongDateString();

lblTime.Text = DateTime.Now.ToLongTimeString();

}

private void Timer1\_Tick(object sender, EventArgs e)

{

lblTime.Text = DateTime.Now.ToLongTimeString();

timer1.Start();

}

private void MetroTile1\_Click(object sender, EventArgs e)

{

}

string constring = @"Data Source=LAPTOP-5S178792; Initial Calatog = FinalProject; Integrated Security = True;";

private void MetroButton1\_Click(object sender, EventArgs e)

{

string qry = "SELECT \* FROM Food ";

SqlDataAdapter DA = new SqlDataAdapter(qry, constring);

DataSet DS = new DataSet();

DA.Fill(DS, "Food");

dgvFoodGrid.DataSource = DS.Tables["Food"];

}

private void MetroButton3\_Click(object sender, EventArgs e)

{

string foodid = txtFoodCodeC.Text;

string foodname = txtFoodNameC.Text;

string qry = "SELECT \* FROM Food where foodID = '" + foodid + "' or Name = '" + foodname + "'";

SqlDataAdapter DA = new SqlDataAdapter(qry, constring);

DataSet DS = new DataSet();

DA.Fill(DS, "Food");

dgvFoodGrid.DataSource = DS.Tables["Food"];

}

private void BtnAdd\_Click(object sender, EventArgs e)

{

try

{

DataRow rw = dt.NewRow();

string itemid = txtItemCode.Text;

int Quantity = int.Parse(txtQuantity.Text);

string qry = "SELECT \* FROM Food where foodID = '" + itemid + "'";

SqlDataAdapter DA = new SqlDataAdapter(qry, constring);

DataSet DS = new DataSet();

DA.Fill(DS, "Food");

rw["ID"] = itemid;

rw["Quantity"] = Quantity.ToString();

rw["Name"] = DS.Tables["Food"].Rows[0]["Name"];

rw["Price"] = (Quantity \* int.Parse(DS.Tables["Food"].Rows[0]["Price"].ToString())).ToString();

dt.Rows.Add(rw);

dgvBasket1.DataSource = dt;

txtQuantity.Text = null;

txtItemCode.Text = null;

}

catch(Exception ex)

{

MessageBox.Show("" + ex);

}

}

private void TxtItemCode\_TextChanged(object sender, EventArgs e)

{

}

private void MetroTile2\_Click(object sender, EventArgs e)

{

string itemid = txtItemCode.Text;

DataRow rw = dt.NewRow();

dt.AcceptChanges();

{

foreach (DataRow row in dt.Rows)

{

string id = row[0].ToString();

if (id == itemid )

{

row.Delete();

}

}

}

dt.AcceptChanges();

}

private void Label4\_Click(object sender, EventArgs e)

{

}

private void TxtFoodNameC\_TextChanged(object sender, EventArgs e)

{

}

private void MetroButton4\_Click(object sender, EventArgs e)

{

string qry = "SELECT \* FROM offers ";

SqlDataAdapter DA = new SqlDataAdapter(qry, constring);

DataSet DS = new DataSet();

DA.Fill(DS, "offers");

dgvOffers.DataSource = DS.Tables["offers"];

}

private void MetroTile1\_Click\_1(object sender, EventArgs e)

{

try

{

DataRow rw1 = dt.NewRow();

string offerid = txtOfferId.Text;

int Quantity = 1;

string qry = "SELECT \* FROM offers where OfferId = '" + offerid + "'";

SqlDataAdapter DA1 = new SqlDataAdapter(qry, constring);

DataSet DS1 = new DataSet();

DA1.Fill(DS1, "offers");

rw1["ID"] = offerid;

rw1["Quantity"] = Quantity;

rw1["Name"] = DS1.Tables["offers"].Rows[0]["OfferName"];

rw1["Price"] = (Quantity \* int.Parse(DS1.Tables["offers"].Rows[0]["Price"].ToString())).ToString();

dt.Rows.Add(rw1);

dgvBasket1.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show("" + ex);

}

txtOfferId.Text = null;

}

private void MetroTile4\_Click(object sender, EventArgs e)

{

if(!(txtPayment.Text == ""))

{

double discount = double.Parse(txtDiscount.Text);

double payment = double.Parse(txtPayment.Text);

dgvBasket1.AllowUserToAddRows = false;

lblSubTotal.Text = "0";

for (int i = 0; i < dgvBasket1.Rows.Count; i++)

{

lblSubTotal.Text = Convert.ToString(double.Parse(lblSubTotal.Text) + double.Parse(dgvBasket1.Rows[i].Cells[3].Value.ToString()));

}

lblDiscount.Text = discount.ToString();

lblBalance.Text = (payment - double.Parse(lblSubTotal.Text)).ToString();

lblTotal.Text = (double.Parse(lblSubTotal.Text) - discount).ToString();

}

else

{

MessageBox.Show("Please enter the payment information.");

}

}

private void MetroTile5\_Click(object sender, EventArgs e)

{

lblBalance.Text = "0.00";

lblDiscount.Text = "0.00";

lblSubTotal.Text = "0.00";

lblTotal.Text = "0.00";

txtPayment.Text = null;

txtDiscount.Text = "0";

dgvBasket1.DataSource = null;

dgvBasket1.Rows.Clear();

dgvBasket1.Refresh();

}

private void DgvOffers\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void MetroTile3\_Click(object sender, EventArgs e)

{

string offerid = txtOfferId.Text;

DataRow rw = dt.NewRow();

dt.AcceptChanges();

{

foreach (DataRow row in dt.Rows)

{

string id = row[0].ToString();

if (id == offerid)

{

row.Delete();

}

}

}

dt.AcceptChanges();

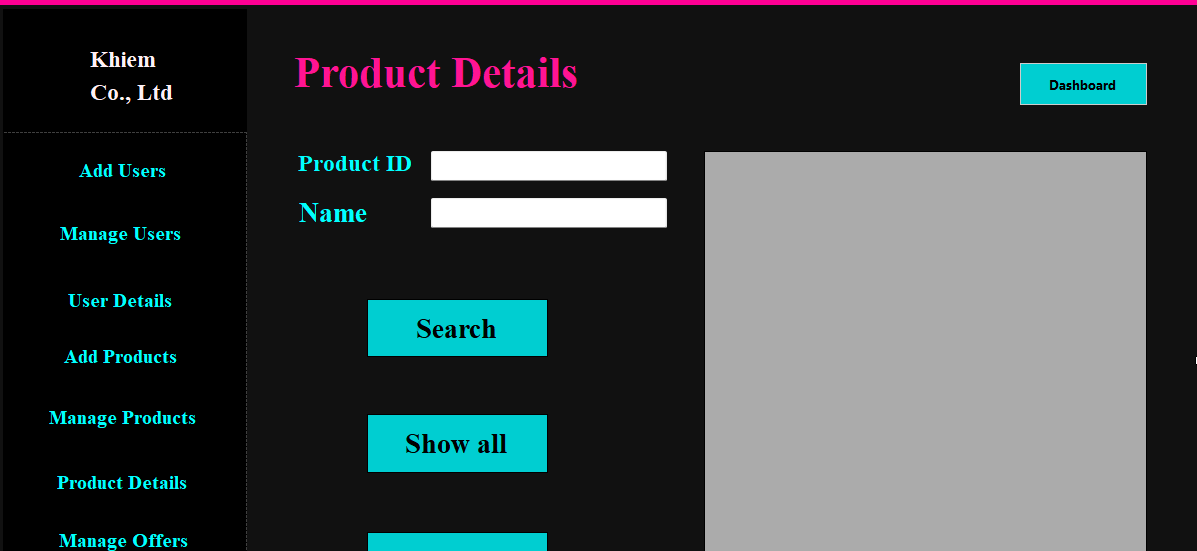
}

}

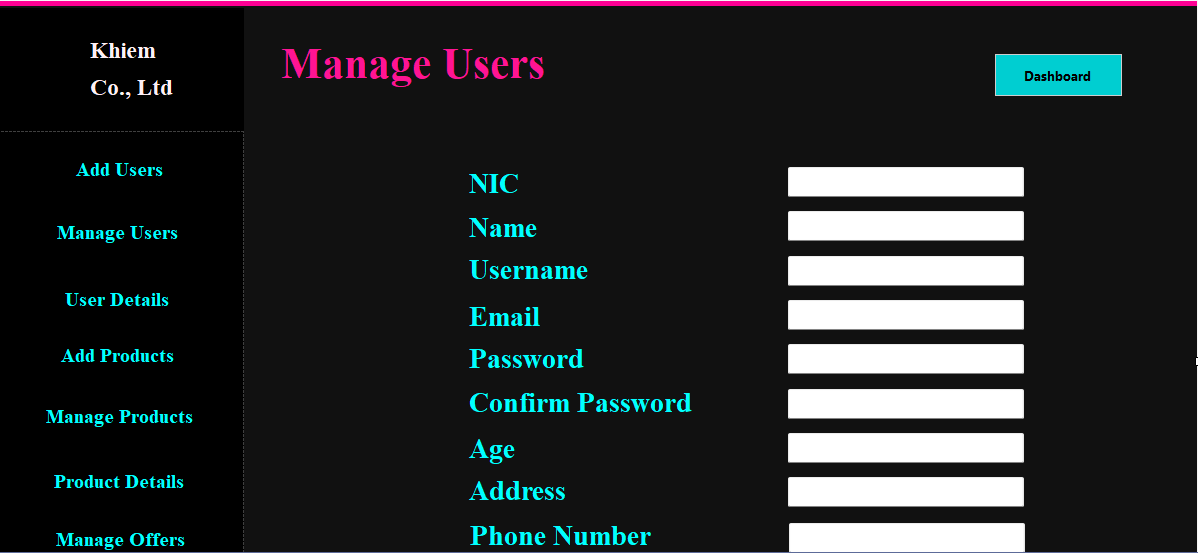
}

**7.3. Testing**

**Product Details:**



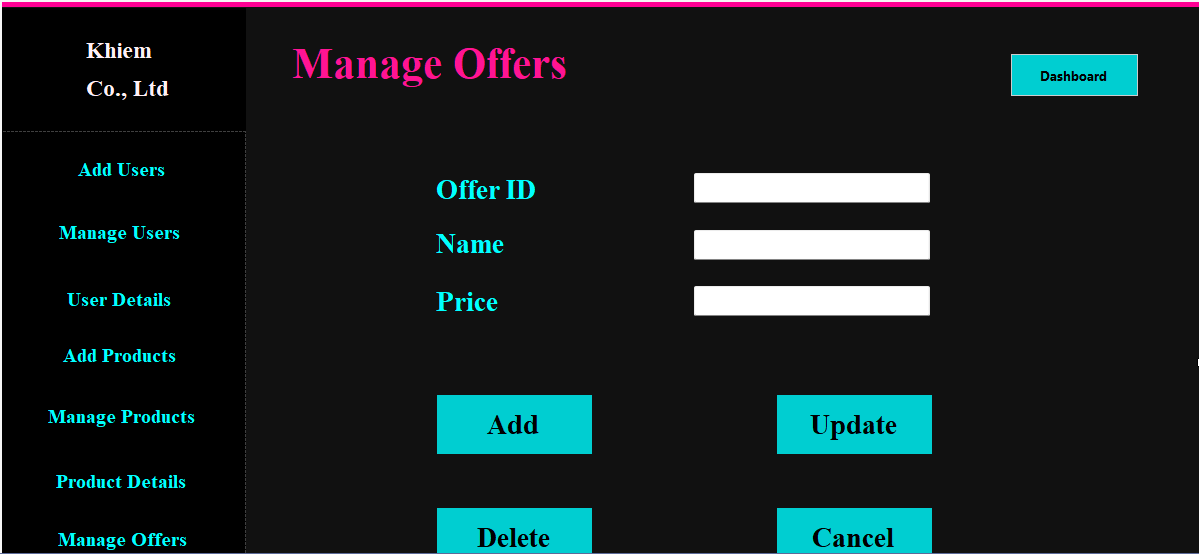
**Manage Users:**



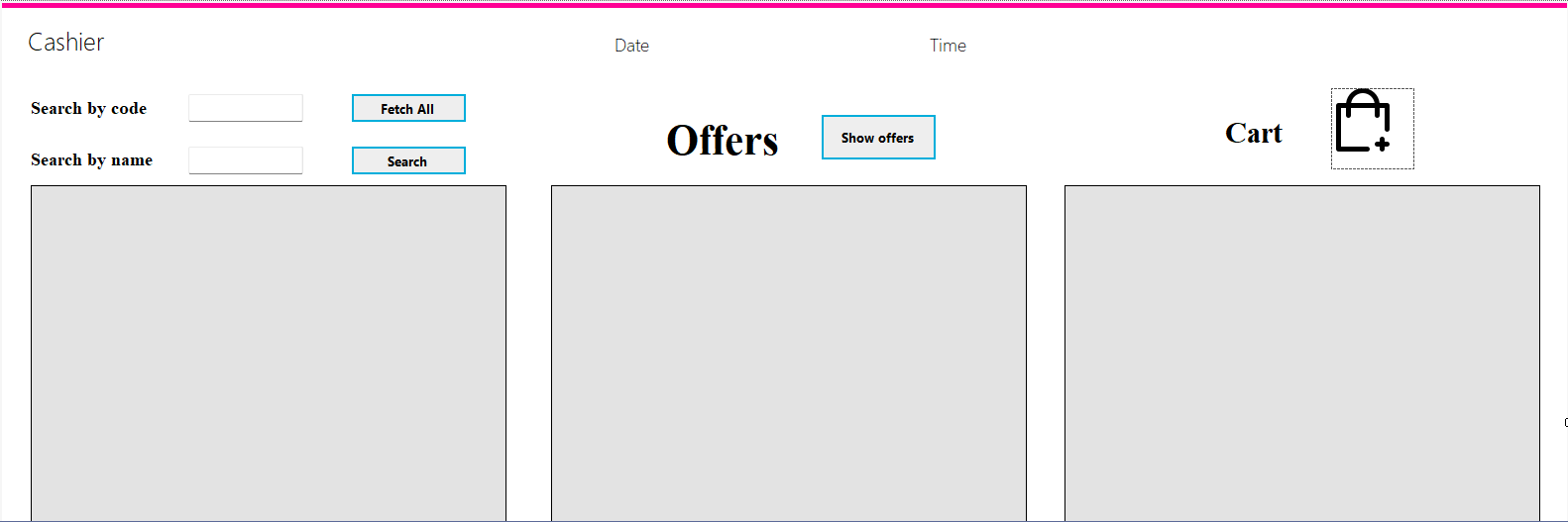
**Manage Products:**



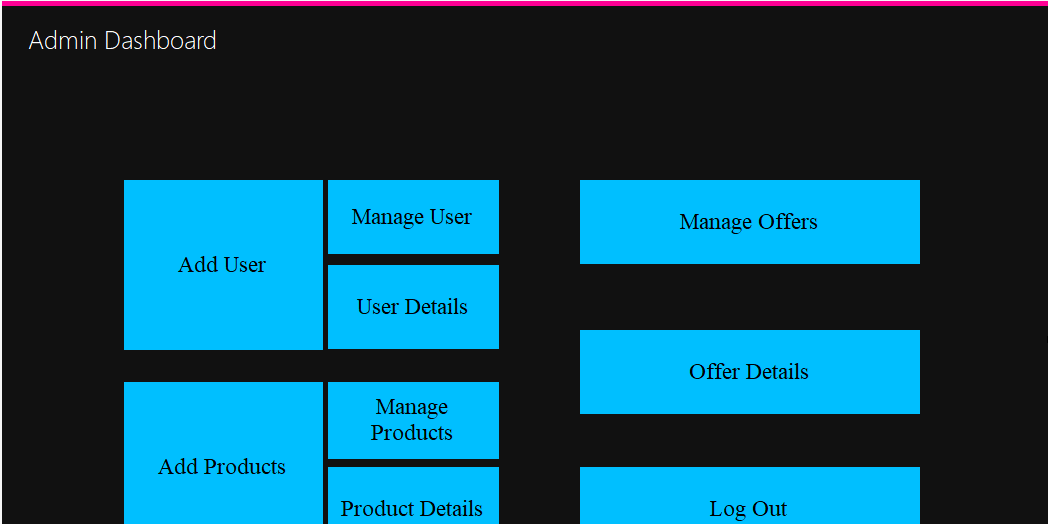
**Manage Offers:**



**Cashier:**



**Admin Dashboard:**



**Add Users:**

