ITE 1942 – ICT PROJECT

PROJECT REPORT

Level 01

Café Management System

Submitted by:

Kuruppu M.P.

E2248469

Bachelor of Information Technology (External Degree)
Faculty of Information Technology
University of Moratuwa

Table of Contents

1. Introduction

This chapter introduces the problem that the cafe management system project aims to address, providing a comprehensive overview of the various factors necessitating the development of a computer-based solution. As the demand for cafes continues to grow alongside increasing competition in the food and beverage industry, the need for efficient, streamlined management systems becomes ever more critical. This chapter delves into the background and motivation behind the project, detailing the specific problems inherent in traditional cafe management methods, and highlighting the shortcomings and inefficiencies that these methods perpetuate. Additionally, the chapter outlines the aims and objectives of the project, offering a clear vision of what the proposed cafe management system intends to achieve. The chapter concludes with a summary that bridges to the subsequent sections, which will further elaborate on the technical and functional requirements of the system.

The management of a cafe involves a multitude of processes, from order taking and inventory management to customer service and staff coordination. Traditionally, many of these processes have been handled manually, which, while feasible on a small scale, becomes increasingly problematic as the business grows. Manual processes are not only time-consuming but also prone to human error, which can lead to inaccuracies in orders, inventory discrepancies, and inconsistent customer service. In an industry where customer satisfaction and operational efficiency are paramount, these issues can significantly impact a cafe's success and profitability.

The motivation for developing a computer-based cafe management system stems from the need to overcome these challenges and provide a robust, integrated solution that enhances both operational efficiency and customer experience. By leveraging modern software development techniques and technologies, the proposed system aims to automate and streamline various aspects of cafe management, thereby reducing the reliance on manual processes and minimizing the associated risks.

I. Background & Motivation

I.I. Identifying Needs or Problems

The cafe management system is designed to tackle the multifaceted challenges faced by cafes and similar establishments in managing their day-to-day operations. The primary needs or problems to be addressed include:

- Inefficiencies in Order Management:
  Manual order taking often results in inaccuracies, delayed service, and customer dissatisfaction. There is a need for a more efficient and accurate method of managing orders to ensure timely and correct service delivery.

- Challenges in Inventory Management:
  Tracking inventory manually is error-prone and inefficient. This can lead to overstocking or understocking, both of which have financial implications. An automated system can provide real-time inventory tracking and management.

- User Management Issues:
  Managing staff and customer data manually makes it difficult to maintain accurate records and track performance. A systematic approach to user management can enhance operational oversight and improve customer relationship management.

- Overall Operational Inefficiencies:
  The cumulative effect of these manual processes is a general lack of efficiency that can increase operational costs and reduce profitability.

I.II. Impact of the Problem

The impact of these problems on relevant users, which include cafe owners, managers, staff, and customers, is significant:

- Cafe Owners and Managers: Inefficiencies and inaccuracies in management processes can lead to increased operational costs, reduced profitability, and challenges in scaling the business. Owners and managers need a reliable system to streamline operations and improve decision-making.

- Staff: Manual processes can be time-consuming and lead to frustration among staff. An efficient system can help streamline their tasks, reduce errors, and improve job satisfaction.

- Customers: The ultimate impact of inefficient cafe management is felt by the customers. Delays, incorrect orders, and inconsistent service can lead to customer dissatisfaction and loss of business. A streamlined system can enhance the customer experience, ensuring faster service and accurate order fulfillment.

I.III. Existing Solutions

Several existing solutions attempt to address these challenges, including:

POS Systems: Many cafes use Point of Sale (POS) systems to streamline transactions and manage orders. These systems can integrate various functions like billing, inventory management, and customer relationship management. However, POS systems can be expensive and may not offer all the necessary features required for comprehensive cafe management.

Inventory Management Software: Standalone inventory management software solutions help in tracking stock levels, managing suppliers, and forecasting demand. However, they often need to be integrated with other systems for a complete management solution, which can be cumbersome and costly.

Custom-Built Applications: Some cafes develop custom applications tailored to their specific needs. While these can provide a perfect fit for the cafe's requirements, developing and maintaining custom applications can be expensive and resource intensive.

Despite the availability of these solutions, there remains a gap in the market for an integrated, cost-effective, and user-friendly cafe management system specifically designed to address the unique challenges faced by cafes.

II. Problem in Brief

II.I. Detailed Problem Statement

Traditional cafe management methods involve several manual processes that are inefficient and prone to errors. The key issues include:

Order Management: The process of taking orders manually is susceptible to mistakes such as incorrect order entries and miscommunication between staff, leading to customer dissatisfaction.

Inventory Tracking: Manual inventory management is prone to human errors, resulting in stock discrepancies, overstocking, or running out of essential items, which affects sales and customer satisfaction.

User Management: Without a systematic user management system, tracking staff performance, managing schedules, and maintaining customer relationships becomes challenging, impacting overall efficiency.

Operational Inefficiencies: These manual processes cumulatively result in significant operational inefficiencies, increasing the workload for staff, leading to higher operational costs, and reducing overall profitability.

II.II. Brief Solution Overview

The solution to these problems is a comprehensive cafe management application developed using C#. This application will automate and streamline various aspects of cafe management, including order processing, inventory management, and user management. By reducing the reliance on manual processes, the application will minimize errors, improve operational efficiency, and enhance customer satisfaction.

III. Aims & Objectives

III.I. Aim

The aim of this project is to develop a user-friendly and intuitive cafe management application that addresses the shortcomings of traditional methods and provides a modern, efficient solution for cafe owners and managers.

III.II. Objectives

To achieve this aim, the project will accomplish the following objectives:

1. **Critical Analysis of System Requirements**: Conduct a thorough analysis of the specific requirements for the cafe management system, ensuring it meets the needs of cafe owners, managers, staff, and customers.

2. **Design and Develop a System for Solving the Problem**: Create a comprehensive application with features including:

- A user-friendly login form connected to a database for user authentication, allowing both guests and registered users to access the system.

- An order placement system for guests without the need for authentication, providing a seamless ordering experience.

- Advanced features for registered users, including order forms, user management, and item management functionalities.

- Functionality to add, edit, delete, and view user accounts and items, allowing for efficient management of cafe resources.

- Detailed order summaries, including date, seller, order number, and total amount, providing insights into sales performance and customer preferences.

- Filtering options for items based on categories (Food or Beverage), facilitating easy navigation and selection.

- Printing functionality for order summaries, enhancing operational efficiency and compliance with record-keeping and reporting requirements.

3. Evaluation of the Proposed System: Assess the effectiveness and efficiency of the developed application through rigorous testing and user feedback, ensuring it meets the desired performance criteria and user satisfaction levels.

IV. Summary

This chapter provided an introduction to the cafe management system project, covering the background and motivation for developing a computer-based solution, detailing the specific problems inherent in traditional cafe management methods, and outlining the project's aims and objectives. The development of a comprehensive cafe management application aims to streamline operations, reduce errors, and enhance both operational efficiency and customer satisfaction. The next chapter will delve into the system requirements, providing a detailed analysis of the functional and non-functional requirements necessary to build the proposed cafe management system.

2. Related Work

In this chapter, we delve into the challenges associated with managing a cafe, exploring the intricacies of the problem and examining existing solutions in detail. By analyzing other systems and solutions, we aim to understand how they address the same issues and identify gaps that our proposed cafe management system seeks to fill. This comprehensive review of related work provides a foundation for developing a robust and efficient system tailored to the specific needs of cafe management.

I. Challenges in Cafe Management

I.I. Operational Inefficiencies

One of the primary challenges in cafe management is operational inefficiency, which stems from several factors:

Manual Processes: Many cafes still rely on manual processes for order taking, inventory tracking, and staff management. These processes are time-consuming and prone to human error, leading to inaccuracies and delays.

Lack of Integration: Different aspects of cafe operations, such as order management, inventory control, and user management, are often handled by separate systems or manually, resulting in a lack of integration and coordination.

Scalability Issues: As a cafe grows, the volume of orders and inventory increases, making manual processes even more cumbersome and inefficient.

I.II. Customer Experience

Customer satisfaction is critical to the success of a cafe. Challenges affecting customer experience include:

Order Accuracy: Manual order taking increases the likelihood of mistakes, resulting in incorrect orders and dissatisfied customers.

Service Speed: Inefficient processes can lead to longer wait times, negatively impacting the customer experience.

Personalization: Without a systematic way to track customer preferences and history, providing personalized service becomes difficult.

I.III. Inventory Management

Effective inventory management is crucial for minimizing costs and ensuring availability of items. Challenges in this area include:

Stock Discrepancies: Manual inventory tracking is prone to errors, leading to discrepancies between actual stock and recorded inventory.

Overstocking and Understocking: Without accurate tracking, cafes may overstock, leading to wastage, or understocking, resulting in missed sales opportunities.

Supplier Coordination: Managing relationships with multiple suppliers and ensuring timely restocking can be complex without an integrated system.

I.IV. Staff Management

Managing staff efficiently is essential for smooth operations. Challenges include:

- Scheduling: Creating and managing staff schedules manually can lead to conflicts and inefficiencies.
- Performance Tracking: Without a systematic way to track staff performance, identifying areas for improvement and rewarding high performers becomes challenging.
- Training: Ensuring consistent training and adherence to standards is difficult without a standardized system.

II. Existing Solutions

II.I Point of Sale (POS) Systems

Many cafes use POS systems to manage transactions and streamline some aspects of their operations. Key features of POS systems include:

1. Transaction Processing: POS systems handle billing and payments efficiently, reducing wait times and errors at the point of sale.
2. Basic Inventory Management: Some POS systems offer basic inventory tracking features, helping cafes manage stock levels.
3. Customer Relationship Management: Advanced POS systems include CRM features, allowing cafes to track customer preferences and purchase history.

However, POS systems often have limitations:

Cost: High-quality POS systems can be expensive, making them less accessible for small cafes.

Limited Integration: Many POS systems do not offer comprehensive integration with other aspects of cafe management, such as staff scheduling and detailed inventory tracking.

II.II. Inventory Management Software

Standalone inventory management software solutions help cafes track stock levels and manage suppliers. Features include:

- Real-Time Tracking: These systems provide real-time updates on stock levels, reducing the risk of overstocking or understocking.
- Supplier Management: Inventory management software often includes tools for managing supplier relationships and orders.

Despite their benefits, these systems have drawbacks:

1. Integration Challenges: Standalone inventory management software may not integrate seamlessly with other systems, requiring manual data entry and reconciliation.
2. Cost: Advanced inventory management solutions can be costly, particularly for smaller establishments.

II.III. Custom-Built Applications

Some cafes develop custom applications tailored to their specific needs. Advantages of custom-built applications include:

1. Tailored Features: Custom applications can be designed to meet the exact requirements of the cafe, providing a perfect fit for their operations.
2. Flexibility: Custom solutions offer flexibility in terms of features and scalability.

However, custom-built applications also have significant disadvantages:

-Development Costs: Developing a custom application can be expensive and time-consuming, requiring significant investment.

- Maintenance and Updates: Ongoing maintenance and updates can be costly and require specialized technical expertise.

II.IV Comprehensive Management Systems

There are comprehensive management systems designed specifically for cafes that integrate various aspects of operations, including order management, inventory control, and staff management. Examples include:

-Square for Restaurants: This system offers a range of features, including order management, inventory tracking, and staff scheduling. It also integrates with various payment systems and provides detailed reporting.

- Toast: Toast is a cloud-based system that provides tools for order management, inventory control, and customer relationship management. It is known for its user-friendly interface and robust feature set.

While comprehensive management systems offer numerous advantages, they also have limitations:

- Cost: These systems can be prohibitively expensive for smaller cafes.

- Complexity: The wide range of features can be overwhelming, requiring training and adjustment periods for staff.

III. Comparative Analysis of Existing Solutions

To better understand the strengths and weaknesses of existing solutions, a comparative analysis is essential. The table below provides a summary of key features and limitations of the various solutions discussed:

Solution	Key Features	Limitations
POS Systems	Transaction processing, basic inventory management, CRM	High cost, limited integration
Inventory Management Software	Real-time tracking, supplier management	Integration challenges, high cost
Custom-Built Applications	Tailored features, flexibility	High development costs, maintenance requirements
Comprehensive Management Systems	Integrated operations, detailed reporting, staff management	High cost, complexity

IV. Gaps and Opportunities

Despite the availability of various solutions, significant gaps remain in the market for a cost-effective, integrated cafe management system. Key opportunities for improvement include:

1. Affordability: Developing a system that offers robust features at a lower cost, making it accessible to small and medium-sized cafes.
2. Integration: Providing seamless integration of order management, inventory tracking, and staff management in a single platform.
3. User-Friendliness: Designing an intuitive interface that requires minimal training and can be easily adopted by staff.
4. Customization: Offering customizable features that can be tailored to the specific needs of different cafes.

V. Summary

This chapter provided a detailed overview of the challenges associated with cafe management and examined existing solutions in the market. While current systems offer various benefits, they also have limitations that leave room for improvement. The analysis highlighted the need for an affordable, integrated, and user-friendly cafe management system that can effectively address the inefficiencies of traditional methods. The next chapter will outline the specific requirements for developing such a system, building on the insights gained from the comparative analysis of existing solutions.

3. Systems Analysis

This chapter covers the analysis of all functional and non-functional requirements essential for the development of the cafe management system. Each process within the scope of the project is analyzed in detail to ensure that all identified requirements are clearly presented. No requirements within the project's scope are overlooked. The sections and subsections are structured to provide a comprehensive overview of the system requirements.

I. System Requirements

This section presents all the requirements addressed in the project in detail, encompassing hardware, software, functional, and non-functional requirements.

I.I. Hardware Requirements

The hardware requirements for the cafe management system include:

- Servers: Necessary for hosting the application and managing the database.
- Workstations: Provide the interface for user interaction.
- Printers: Required for generating reports and order summaries.

I.II. Software Requirements

The software requirements for the system include:

- Operating System: Compatible environments such as Windows.
- Database Management System: MySQL or PostgreSQL for storing and retrieving data.
- Development Frameworks and Programming Languages: C# and .NET are utilized for building the application.

II. Functional Requirements

This section presents all functional requirements addressed in the project in detail.

II.I. Login Functionality

The login functionality enables users to securely authenticate themselves to access the system. Features include:

- Username/password authentication.
- View as Guest login options.

II.II. Order Management

Order management functionalities enable users to create, view, modify, and delete orders for drinks and beverages. This includes:

- Adding items to orders.
- Specifying quantities.
- Managing order status.
- Generating order summaries.

II.III. User Management

User management functionalities allow administrators to manage user accounts. This includes:

- Adding new users.
- Editing existing user information.
- Disabling or deleting user accounts.
- Assigning user roles and permissions.

II.IV. Item Management

Item management functionalities enable users to manage the menu items available in the cafe. This includes:

- Adding new items.
- Editing item details such as name, description, and price.
- Categorizing items.
- Removing discontinued items from the menu.

II.V. Order Summary

The order summary functionality provides users with detailed summaries of each order. This includes:

- Date and time of the order.
- Items ordered.
- Total amount due.
- Customer information.
- Features for printing or exporting order summaries.

II.VI. Filtering Functionality

The filtering functionality allows users to filter and sort items based on predefined criteria. This includes Category (food or beverage). The user can choose either food or beverages to be displayed in the items list depending on his/her need.

III. Non-Functional Requirements

This section presents all non-functional requirements addressed in the project in detail.

III.I. Security

The security requirements ensure the confidentiality, integrity, and availability of data. This includes:

- User authentication and authorization mechanisms.
- Access controls.

III.II. User Interface

The user interface requirements describe the design principles and usability guidelines for the system's graphical user interface (GUI). This includes:

- Layout.
- Navigation.
- Consistency.
- Responsiveness.
- Accessibility to accommodate users with diverse needs and preferences.

III.III. Printing Functionality

The printing functionality requirements specify the system’s ability to generate and print various documents, reports, and summaries. This includes:

- Support for different printing formats.

- Compatibility with standard printers.

- Options for customizing print settings and layouts.

III.IV Summary

This chapter provided a comprehensive analysis of the system requirements for the cafe management system. It detailed the hardware, software, functional, and non-functional requirements essential for the system's development. By addressing all identified requirements, this analysis ensures that the system will meet the needs of cafe owners, managers, and staff. The next chapter will focus on the design and implementation phases, building on the requirements outlined in this chapter.

5. System Design

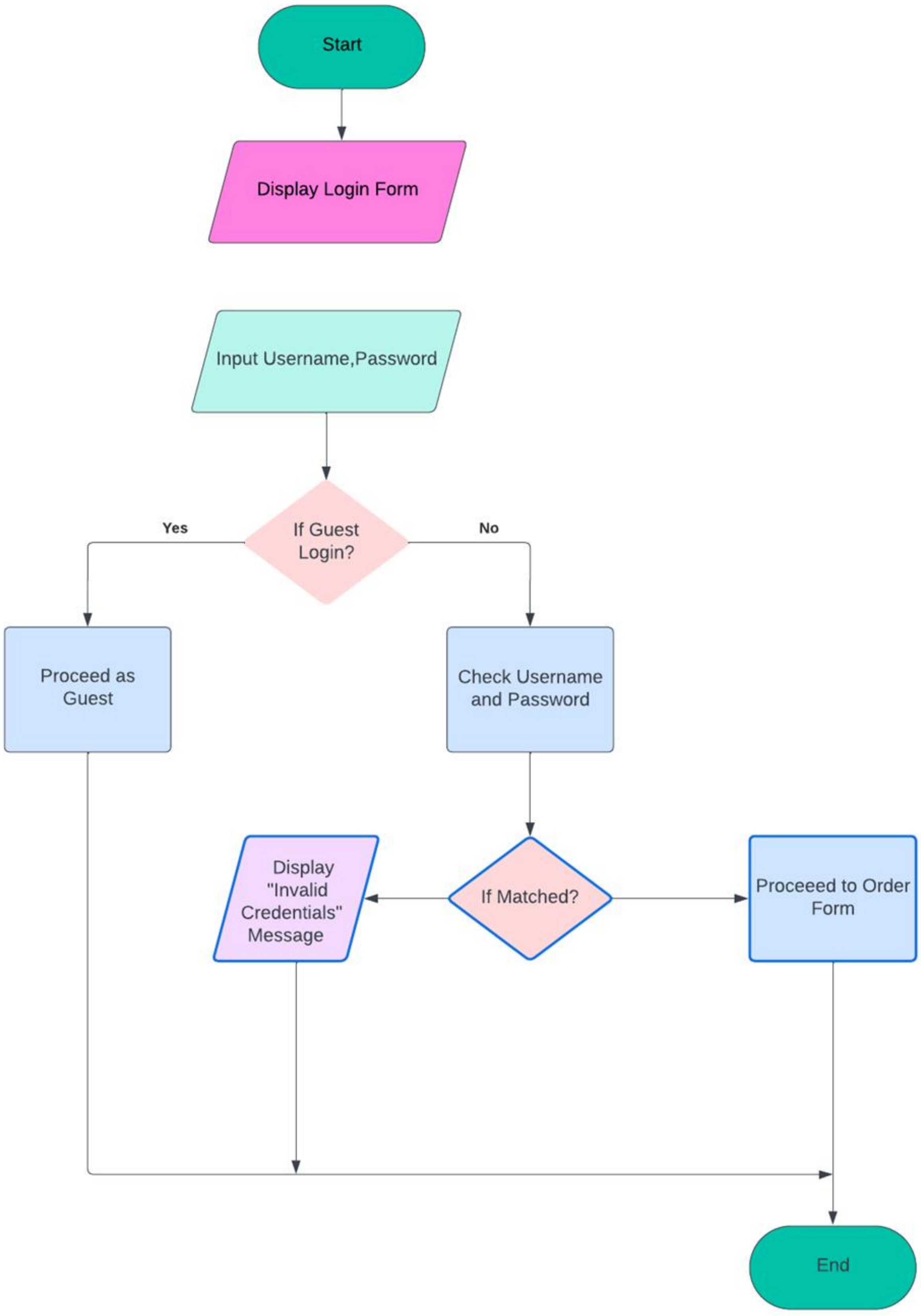
The design of the cafe management system solution is detailed in this chapter. Each functional requirement is modeled using pseudocodes and flowcharts, clearly illustrating their inputs and outputs. Complex diagrams are utilized to ensure comprehensive representation of the system's functionality. The following subsections provide a structured overview of the design process.

I. Pseudocode for Login Form

```
Begin
  Display Login Form
  Input Username, Password
  If User clicks Login Button then
    Check Username and Password in Database
    If Username and Password Match then
      Proceed to User Order Form
    Else
      Display "Wrong Username or Password"
    Endif
  Else If User clicks View as Guest Button then
    Proceed to Guest Order Form
  Endif
End
```

In this pseudocode:

- The user inputs their username and password.
- If the user clicks the "Login" button, the system checks the credentials against the database.
- If the credentials match, the user is directed to the User Order Form.
- If the user clicks the "View as Guest" button, they are directed to the Guest Order Form without authentication.
- If the credentials do not match, an invalid credentials message is displayed.


Start
Display Login Form
Input Username,Password
If Guest Login?
Yes
No
Proceed as Guest
Check Username and Password
Display "Invalid Credentials" Message
If Matched?
Proceeed to Order Form
End

II. Pseudocode for UserOrder Form

```
Begin
    Display Order Form
    OnRefreshButtonClick:
        RefreshItemsTable() // Function to refresh Items Table with both food and beverages
    OnDropdownMenuSelectionChange:
        If SelectedOption is Food:
            Display FoodItems in ItemsTable
        Else If SelectedOption is Beverages:
            Display BeverageItems in ItemsTable
        Else:
            Display AllItems in ItemsTable
    While UserInteracts:
        Display Items Table with Food and Beverage Menu
        Select Item from Item Table
        Input Quantity in QuantityBox
        OnAddToCartButtonClick:
            If Item and Quantity are valid:
                AddItemToCart(Item, Item Category, Unit Price, Total Price)
            Else:
                Display Invalid Item or Quantity Message

            End While
    If Cart is not Empty:
        Display Cart Summary
        Input OrderNumber
        OnPlaceOrderButtonClick:
            If OrderNumber is provided:
                    AddOrderToDatabase(OrderNumber)
```

```
                        Display "Order Successfully Placed"
                        Else:
                        Display OrderNumber Required Message
            Else:
                Display Empty Cart Message
            OnViewOrdersButtonClick:
                    Display View Orders Form
            OnItemsButtonClick:
                ItemsForm will appear
            OnItemsButtonClick:
                UsersForm will appear

        End
```

The Order Form interface is designed to facilitate a seamless ordering process for users. Upon opening the form, users are presented with options to interact with the items available in the cafe's menu. A refresh button is provided to update the Items Table dynamically, ensuring it displays the latest offerings of both food and beverages.

Users can select items from a dropdown menu that categorizes items into "Food" and "Beverages". Depending on their selection, the Items Table updates accordingly to show the chosen category, enhancing user convenience by allowing focused browsing of menu items.

After selecting an item, users input the quantity they wish to order into a designated Quantity input box. Error handling mechanisms are in place to validate user inputs, ensuring that only valid selections (both item and quantity) are accepted. If there's an error, such as not selecting an item or entering an invalid quantity, the system prompts the user with an appropriate error message to correct their input.

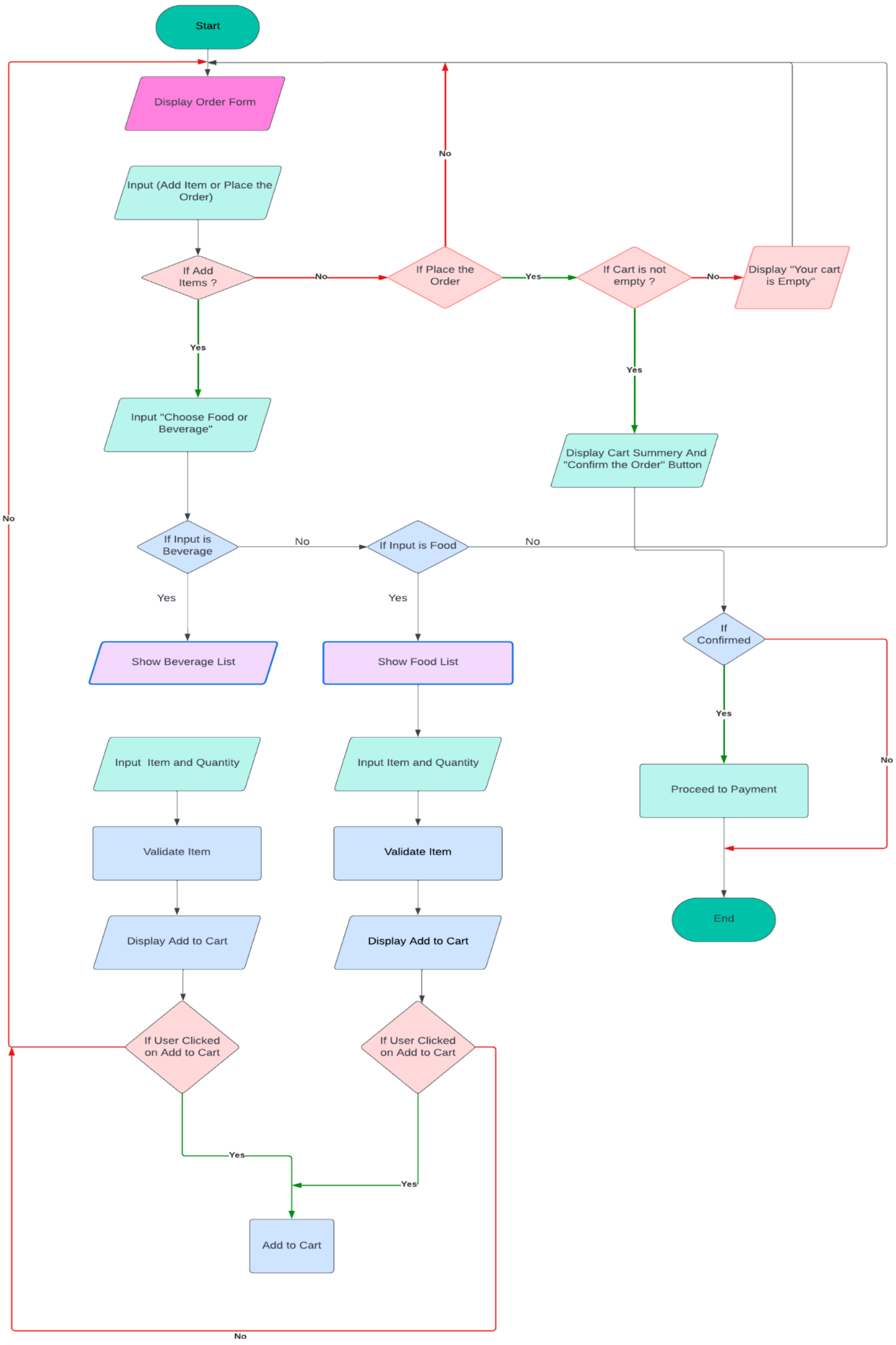
Once the desired items and quantities are chosen, users proceed to add them to their shopping cart by clicking the "Add to Cart" button. This action triggers the addition of the selected items with their respective quantities into the cart storage within the system.

To finalize their order, users are prompted to enter an order number before clicking the "Place Order" button. This step ensures that each order is uniquely identified and logged in the system's database for future reference and management. If no order number is provided, the system alerts the user to enter one before proceeding.

Upon successfully placing the order, a confirmation message "Order Successfully Placed" is displayed to acknowledge the completion of the transaction. This feedback reassures users that their order has been successfully recorded and processed.

Additionally, users have the option to view their past orders or continue shopping after completing a transaction. This feature supports user engagement and convenience by allowing easy access to order history and seamless continuation of their browsing or ordering experience.

Overall, the Order Form is designed with user-friendly functionalities and robust error handling to ensure a smooth and efficient ordering process in the cafe management system, enhancing user satisfaction and operational efficiency.


Start
Display Order Form
Input (Add Item or Place the Order)
If Add Items ?
No
If Place the Order
No
Yes
If Cart is not empty ?
No
Display "Your cart is Empty"
Yes
Yes
Input "Choose Food or Beverage"
Display Cart Summery And "Confirm the Order" Button
No
If Input is Beverage
No
If Input is Food
No
Yes
Yes
Show Beverage List
Show Food List
If Confirmed
Yes
No
Input Item and Quantity
Input Item and Quantity
Proceed to Payment
Validate Item
Validate Item
End
Display Add to Cart
Display Add to Cart
If User Clicked on Add to Cart
If User Clicked on Add to Cart
Yes
Yes
Add to Cart
No

III. Pseudocode for GuestOrder Form

```
Begin
        Display Order Form
        OnRefreshButtonClick:
                RefreshItemsTable() // Function to refresh Items Table with both food and
beverages
         OnDropdownMenuSelectionChange:
If SelectedOption is Food:
        Display FoodItems in ItemsTable
Else If SelectedOption is Beverages:
Display BeverageItems in ItemsTable
Else:
Display AllItems in ItemsTable
        While UserInteracts:
                Display Items Table with Food and Beverage Menu
                 Select Item from Item Table
                 Input Quantity in QuantityBox
                 OnAddToCartButtonClick:
                         If Item and Quantity are valid:
                                 AddItemToCart(Item, Item Category, Unit Price, Total Price)
                         Else:
                                 Display Invalid Item or Quantity Message

                        End While
        If Cart is not Empty:
                Display Cart Summary
                 Input OrderNumber
                 OnPlaceOrderButtonClick:
                         If OrderNumber is provided:
                                         AddOrderToDatabase(OrderNumber)
```

```
                  Display "Order Successfully Placed"
                  Else:
                  Display OrderNumber Required Message
   Else:
         Display Empty Cart Message
End
```

The function is the same as the userorder form. But the view orders, Access buttons for Items and User Forms are not there as they are accessible by only the authenticated users.

IV. Pseudocode for Items Form

```
Begin
  Display Item Form with Items Table
  While User Interacts:
    If User Clicks on Add Button:
       // User has input new item details directly into text boxes,
      // User Has Entered Item Details (Item Number, Item Name, Category, Price)
      OnAddButtonClick:
        Validate Item Details
        If Valid:
          Add Item to Items Table
          Display Success Message: "Item Added Successfully"
        Else:
          Display Error Message: "Invalid Item Details"

    Else If User Clicks on Edit Button:
      // User selects an item from the Items Table to edit
```

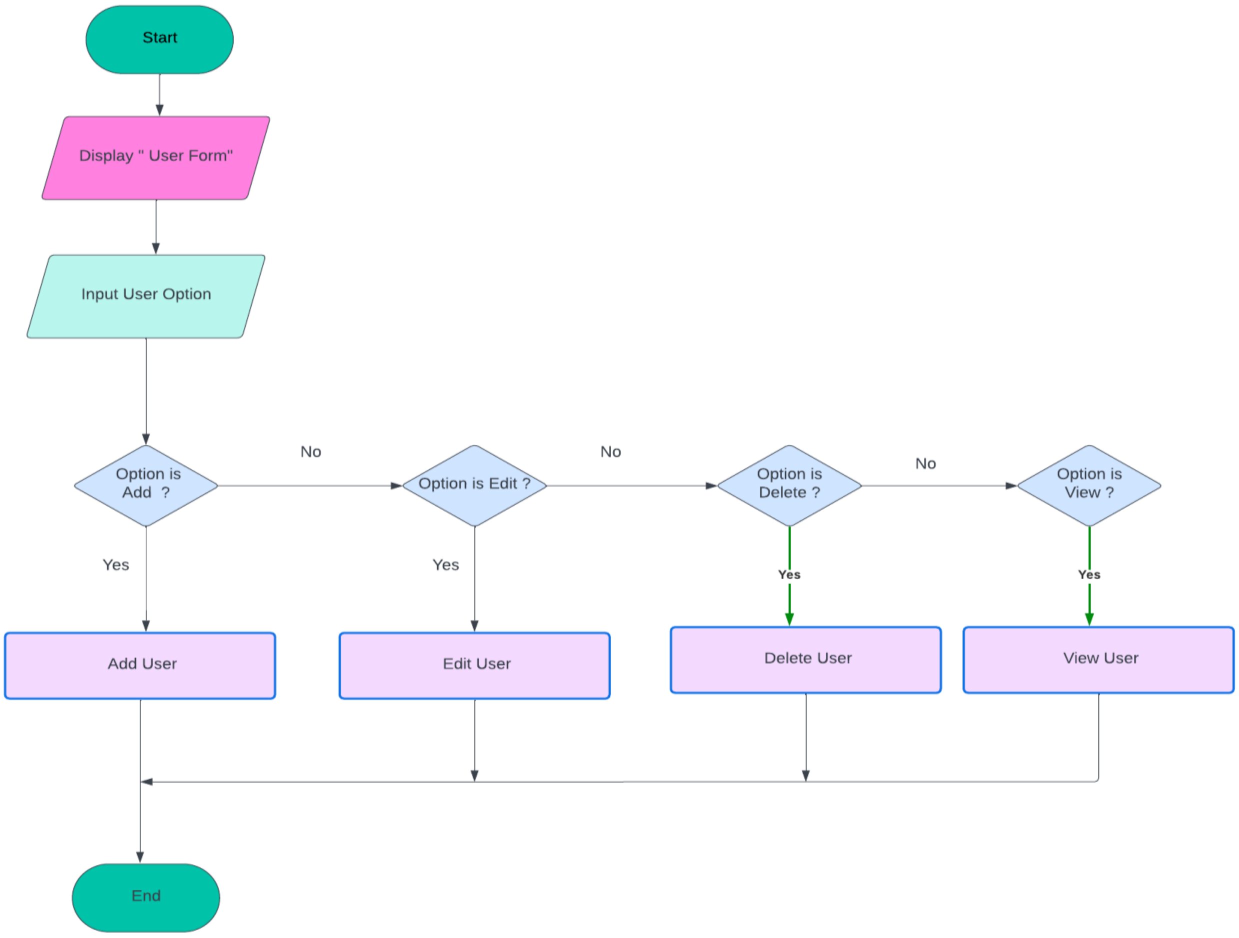
```
            User Selects Item from Items Table
            User Modifies Item Details (Item Number, Item Name, Category, Price)
            OnEditButtonClick:
                Validate Edited Item Details
                If Valid:
                    Update Item in Items Table
                    Display Success Message: "Item Updated Successfully"
                Else:
                    Display Error Message: "Invalid Item Details"
        Else If User Clicks on Delete Button:
            // User selects an item from the Items Table to delete
            User Selects Item from Items Table
            OnDeleteButtonClick:
                Confirm Item Deletion
                Remove Item from Items Table
                Display Success Message: "Item Deleted Successfully"
        Else:
            // Display Items Table with current item details
            Display Items Table with Current Item Details
    End While
End
```

The Item Form pseudocode allows users to manage items effectively within the cafe management system. It begins by displaying an interface containing an Items Table populated with existing items. Users interact with the form through Add, Edit, and Delete buttons:

Add Button: Users input new item details directly into text boxes (Item Number, Item Name, Category, Price). Upon validation, the item is added to the Items Table with a success message displayed. If details are invalid, an error message is shown.

Edit Button: Users select an item from the Items Table to modify. They update item details directly in the text boxes. The system validates changes and updates the Items Table accordingly, displaying a success message for successful updates or an error message for invalid edits.

Delete Button: Users select an item from the Items Table for deletion. The system prompts for confirmation before removing the item from the Items Table. A success message confirms deletion.

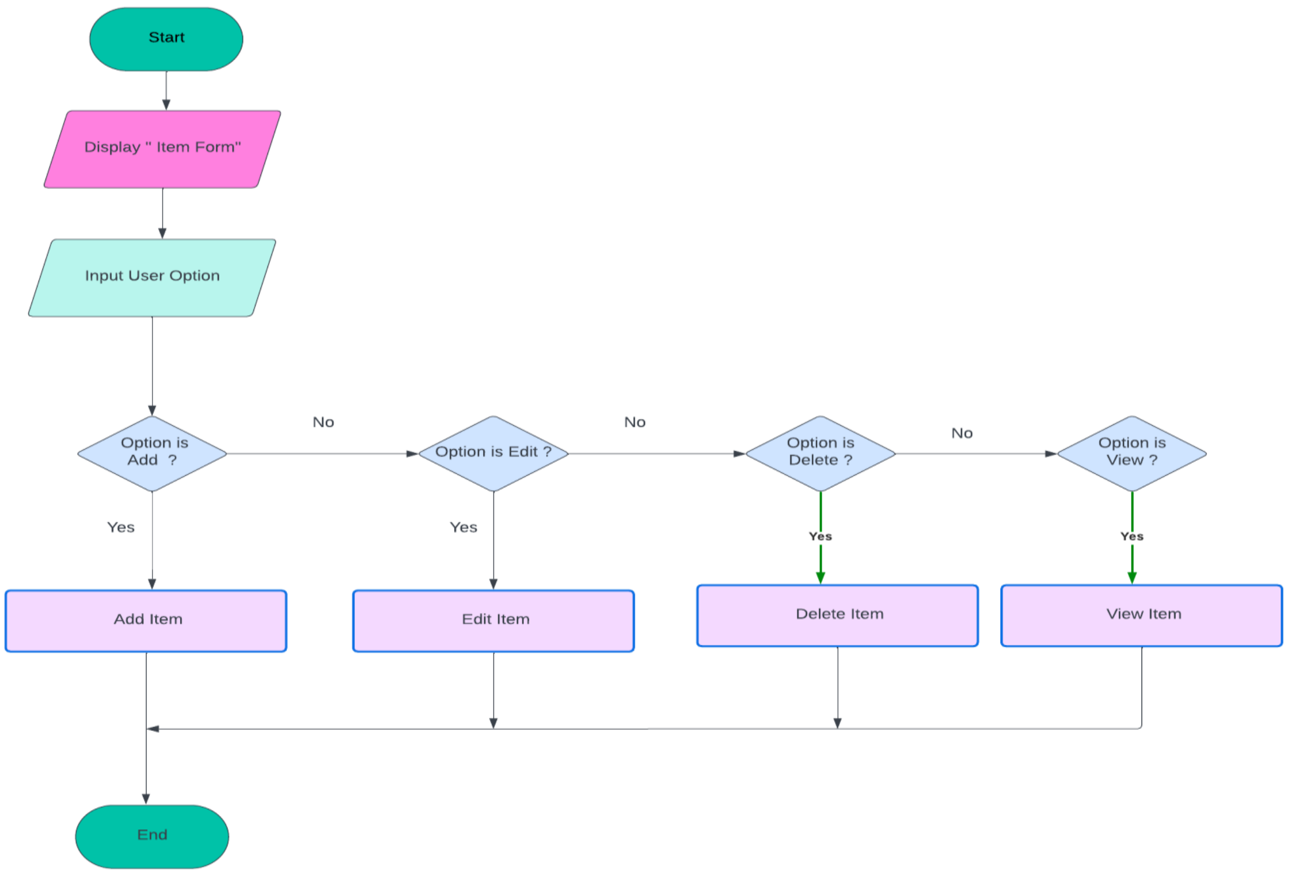


IV. Pseudocode for User Form

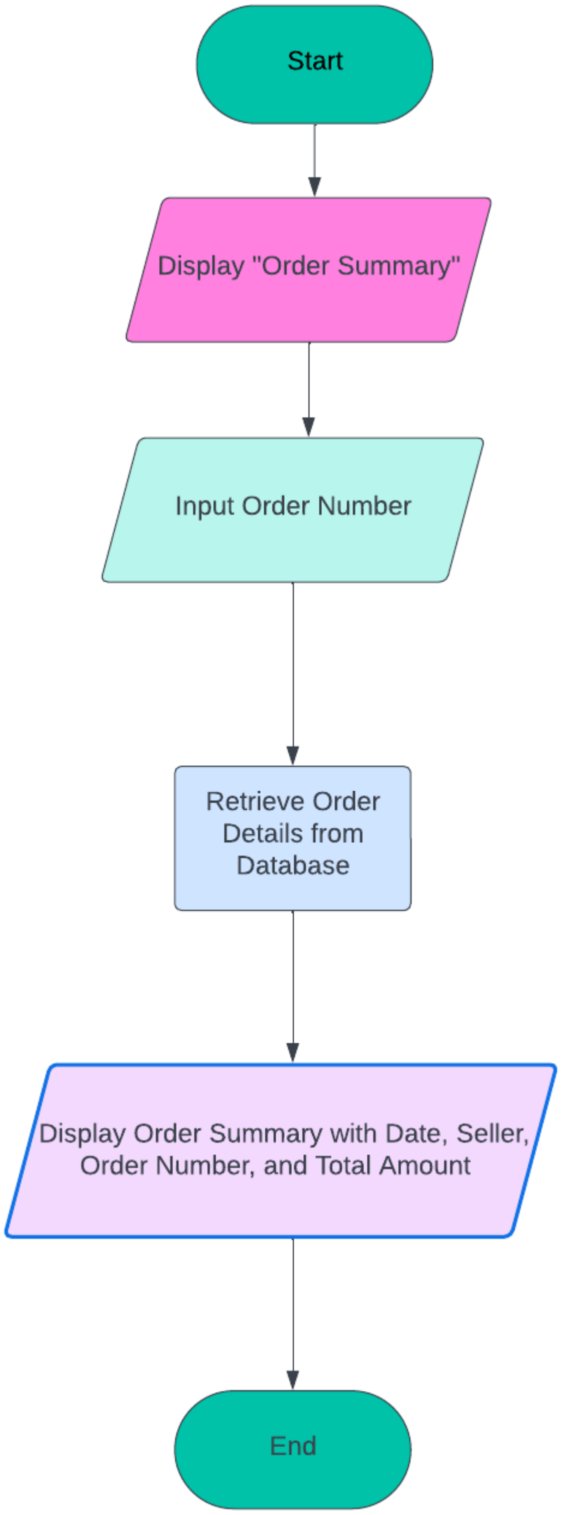
```
Begin
    Display User Form with Users Table
    While User Interacts:
        If User Clicks on Add Button:
            // User inputs new user details directly into text boxes (Username, Mobile Number,
Password)
            OnAddButtonClick:
                Validate User Details
                If Valid:
                    Add User to Users Table
                    Display Success Message: "User Added Successfully"
                Else:
                    Display Error Message: "Invalid User Details"

        Else If User Clicks on Edit Button:
            // User selects a user from the Users Table to edit
            User Selects User from Users Table
            User Modifies User Details (Username, Mobile Number, Password)
            OnEditButtonClick:
                Validate Edited User Details
                If Valid:
                    Update User in Users Table
                    Display Success Message: "User Updated Successfully"
                Else:
                    Display Error Message: "Invalid User Details"

        Else If User Clicks on Delete Button:
            // User selects a user from the Users Table to delete
            User Selects User from Users Table
            OnDeleteButtonClick:
                Confirm User Deletion
                Remove User from Users Table
                Display Success Message: "User Deleted Successfully"

        Else:
            // Display Users Table with current user details
            Display Users Table with Current User Details

    End While
End
```

The User Form pseudocode facilitates user management within the cafe management system. It presents an interface with a Users Table containing existing user details. Users interact with the form through Add, Edit, and Delete buttons:

- **Add Button**: Users input new user details directly into text boxes (Username, Mobile Number, Password). Upon validation, the user is added to the Users Table with a success message displayed. If details are invalid, an error message is shown.
- **Edit Button**: Users select a user from the Users Table to modify. They update user details directly in the text boxes. The system validates changes and updates the Users Table, accordingly, displaying a success message for successful updates or an error message for invalid edits.
- **Delete Button**: Users select a user from the Users Table for deletion. The system prompts for confirmation before removing the user from the Users Table. A success message confirms deletion.



IV. Pseudocode for Order Summary


Start
Display "Order Summary"
Input Order Number
Retrieve Order Details from Database
Display Order Summary with Date, Seller, Order Number, and Total Amount
End

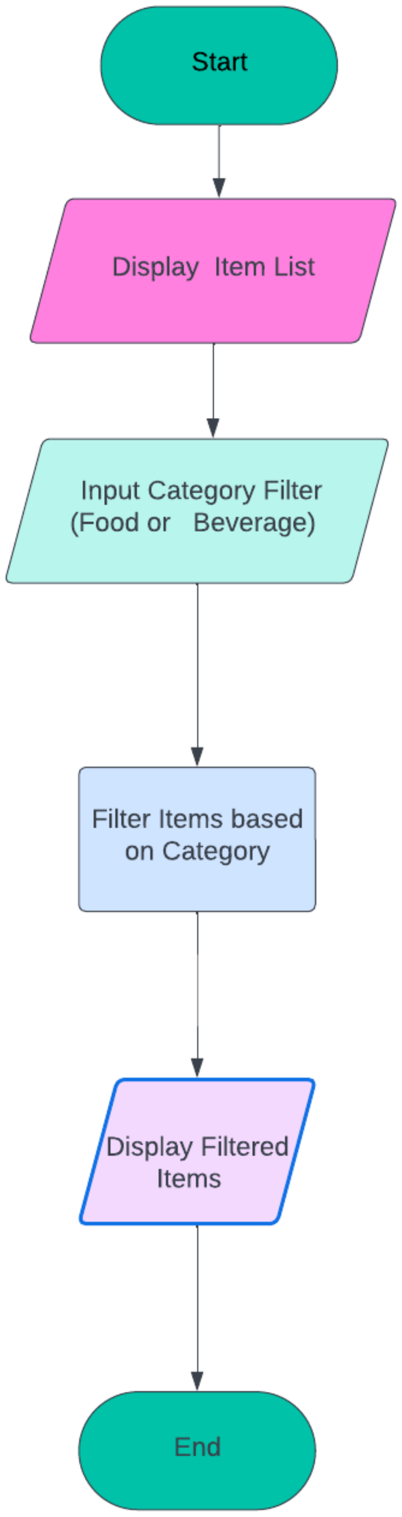
```
Begin
    Display Order Summary
    Input Order Number
    Retrieve Order Details from Database
    Display Order Summary with Date, Seller, Order Number, and Total Amount
End
```

The pseudocode provided outlines the systematic process for presenting an Order Summary within the cafe management system. It begins by initiating the display of the Order Summary interface, prompting users to input a specific Order Number. This input triggers the system to retrieve the corresponding order details from the database, including critical information such as the Date of the order, the identifying Seller, the Order Number for reference, and the Total Amount spent. Once retrieved, these details are then displayed comprehensively on the interface, ensuring clarity and completeness for users reviewing their order history or current transactions. This process concludes upon successfully presenting the Order Summary, thereby facilitating efficient order management and enhancing user interaction within the system.

V. Pseudocode for Filtering Items

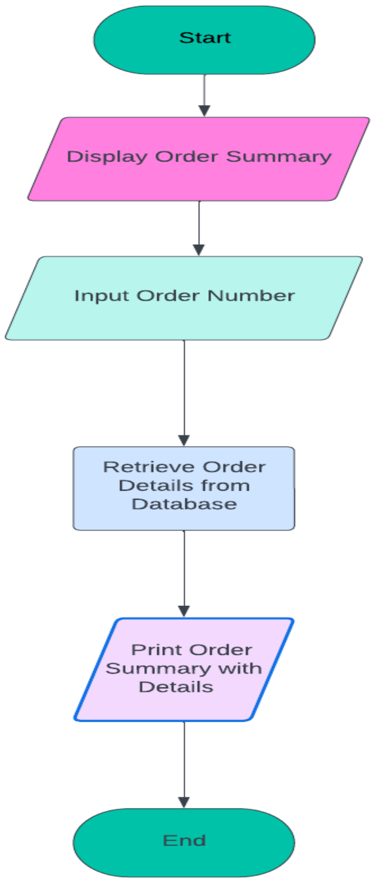
```
Begin
    Display Item List
    Input Category Filter (Food or Beverage)
    Filter Items based on Category
    Display Filtered Items
End
```

The pseudocode segment initiates by displaying an item list interface within the cafe management system. Users are prompted to input a category filter, such as "Food" or "Beverage", to refine the displayed items. Upon receiving the category input, the system filters and displays only those items that belong to the specified category. This functionality allows users to quickly and efficiently browse through items based on their categorization, enhancing usability and streamlining the selection process within the system.


Start
Display Item List
Input Category Filter (Food or Beverage)
Filter Items based on Category
Display Filtered Items
End

V. Pseudocode for Printing Order Summary

```
Begin
        Display Order Summary
        Input Order Number
        Retrieve Order Details from Database
        Print Order Summary with Date, Seller, Order Number, and Total Amount
End
```


Start
Display Order Summary
Input Order Number
Retrieve Order Details from Database
Print Order Summary with Details
End

The process begins by displaying the order summary interface, prompting the user to input an order number. Once the order number is entered, the system retrieves the corresponding order details from the database. After fetching the necessary information such as the date of the order, seller's details, order number, and total amount, the system prints a comprehensive order summary. This streamlined approach ensures that users can quickly access and review essential order information, facilitating efficient management of orders within the cafe management system.

5. System Implementation

The system implementation phase marks the transition from planning and design to the actual construction of the cafe management system. This section provides a detailed account of how the designed solution is executed and integrated into a functional software application. It encompasses the development and deployment of various system components, including user interfaces, backend functionalities, and database interactions.

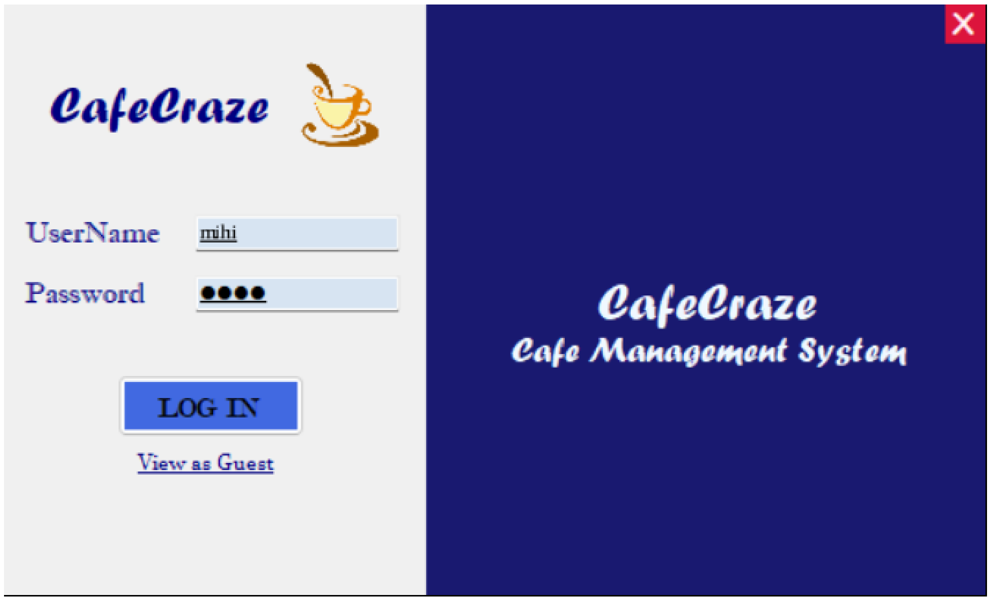
In this phase, emphasis is placed on demonstrating how the system meets the specified requirements outlined in earlier project phases. Key aspects covered include the development environment, programming languages and frameworks used, user interface designs, and integration with backend systems such as databases.

Throughout this section, selected excerpts of code snippets and user interface mockups will be presented to illustrate the implementation process. Detailed descriptions of major processes and their corresponding implementations will be provided, highlighting the functionality and usability enhancements aimed at improving cafe operations and management efficiency.

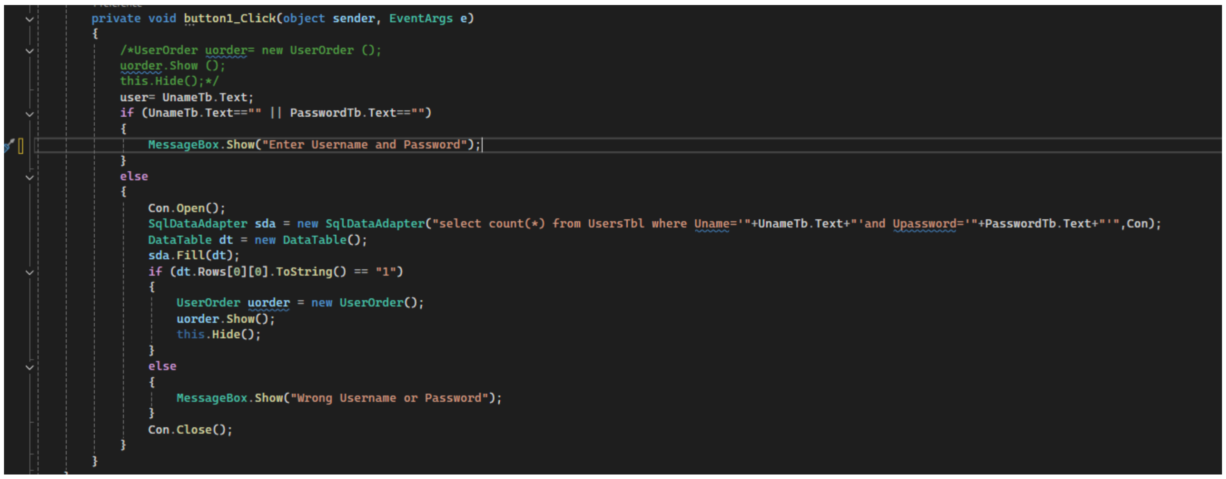
The section also includes relevant inputs and outputs generated during the implementation, showcasing the system's capability to handle user interactions, data processing, and information retrieval effectively. Additional detailed codes and technical specifications will be available in the appendix for comprehensive reference.

Overall, the system implementation section serves to demonstrate how theoretical concepts and design principles are translated into a robust and functional cafe management system, ready address operational challenges and enhance user experience effectively.

Login Form

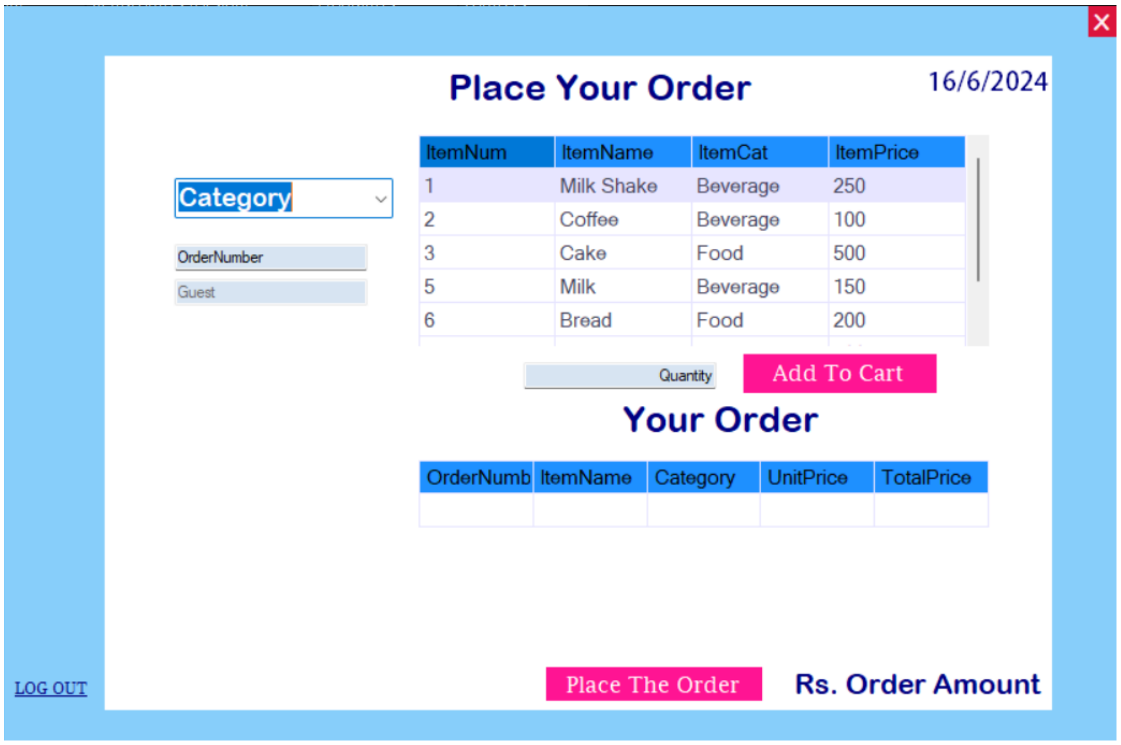


Login Page 1



```
private void button1_Click(object sender, EventArgs e)
{
    /*UserOrder uorder= new UserOrder ();
    uorder.Show ();
    this.Hide();*/
    user= UnameTb.Text;
    if (UnameTb.Text=="" || PasswordTb.Text=="")
    {
        MessageBox.Show("Enter Username and Password");
    }
    else
    {
        Con.Open();
        SqlDataAdapter sda = new SqlDataAdapter("select count(*) from UsersTbl where Uname='"+UnameTb.Text+"'and Upassword='"+PasswordTb.Text+"'",Con);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        if (dt.Rows[0][0].ToString() == "1")
        {
            UserOrder uorder = new UserOrder();
            uorder.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Wrong Username or Password");
        }
        Con.Close();
    }
}
```

Code Snippet for Login Button


Place Your Order

16/6/2024

Category

OrderNumber

Guest

ItemNum	ItemName	ItemCat	ItemPrice
1	Milk Shake	Beverage	250
2	Coffee	Beverage	100
3	Cake	Food	500
5	Milk	Beverage	150
6	Bread	Food	200

Quantity

Add To Cart

Your Order

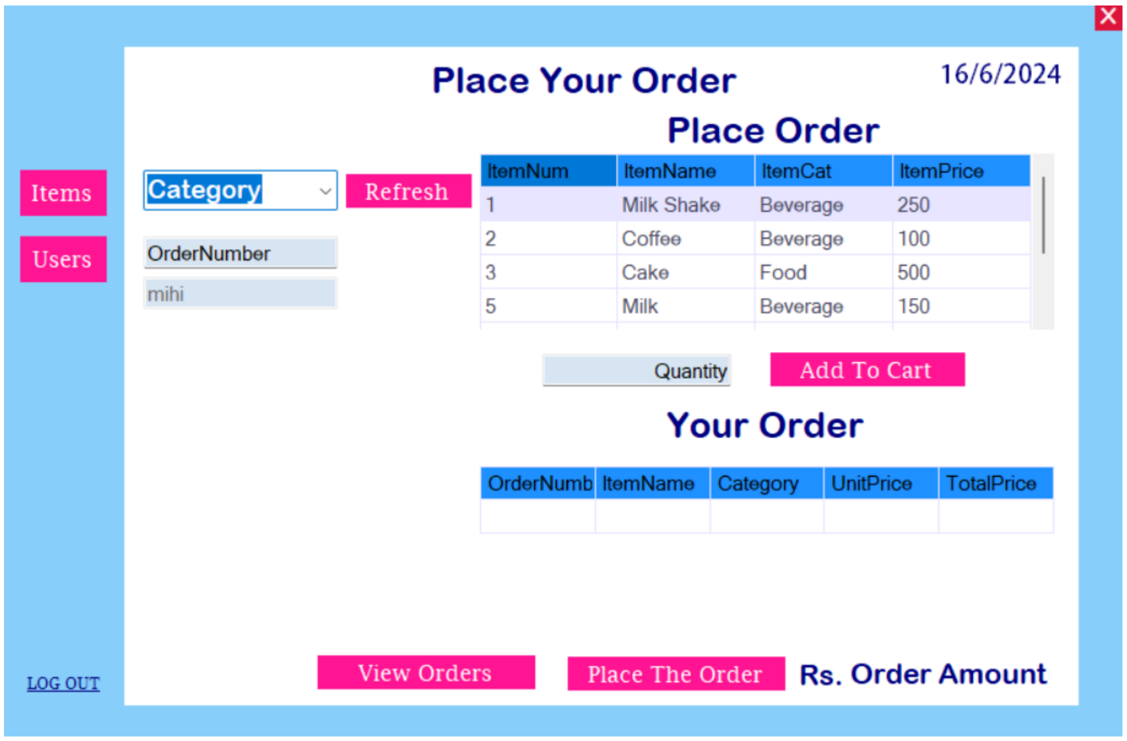
OrderNumb	ItemName	Category	UnitPrice	TotalPrice

LOG OUT

Place The Order

Rs. Order Amount

Guest Order Form 1


Place Your Order

16/6/2024

Place Order

Items

Users

Category

Refresh

OrderNumber

mihi

ItemNum	ItemName	ItemCat	ItemPrice
1	Milk Shake	Beverage	250
2	Coffee	Beverage	100
3	Cake	Food	500
5	Milk	Beverage	150

Quantity

Add To Cart

Your Order

OrderNumb	ItemName	Category	UnitPrice	TotalPrice

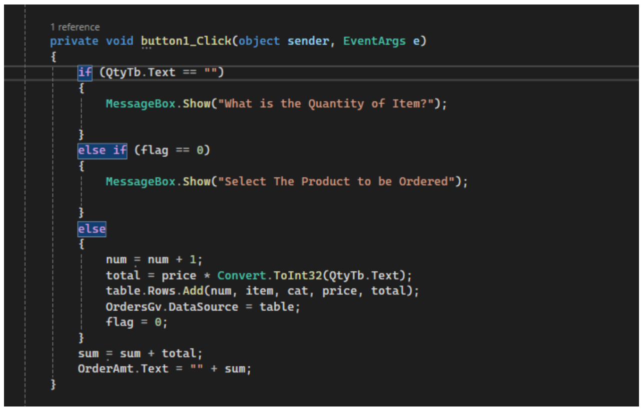
LOG OUT

View Orders

Place The Order

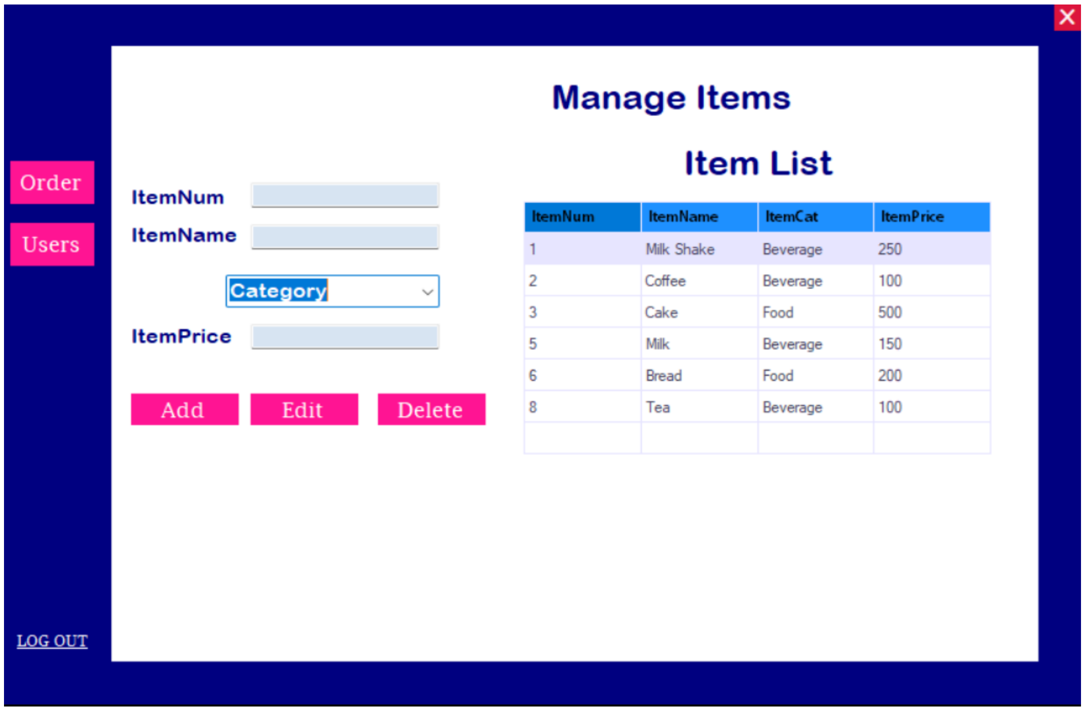
Rs. Order Amount

User Order Form



```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (QtyTb.Text == "")
    {
        MessageBox.Show("What is the Quantity of Item?");

    }
    else if (flag == 0)
    {
        MessageBox.Show("Select The Product to be Ordered");

    }
    else
    {
        num = num + 1;
        total = price * Convert.ToInt32(QtyTb.Text);
        table.Rows.Add(num, item, cat, price, total);
        OrdersGv.DataSource = table;
        flag = 0;
    }
    sum = sum + total;
    OrderAmt.Text = "" + sum;
}
```

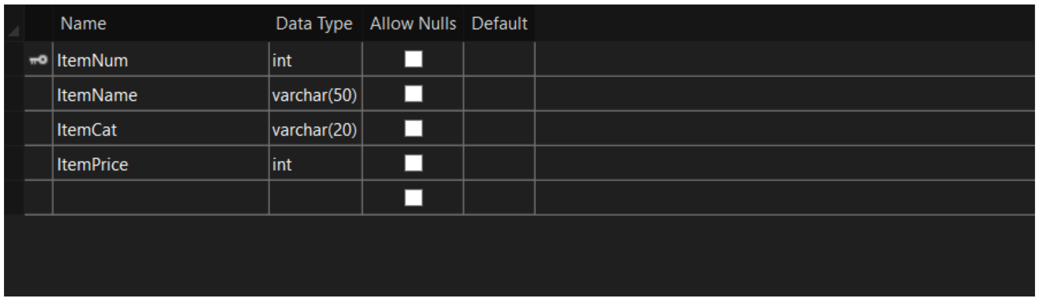
Code Snippet for Add to Cart Button



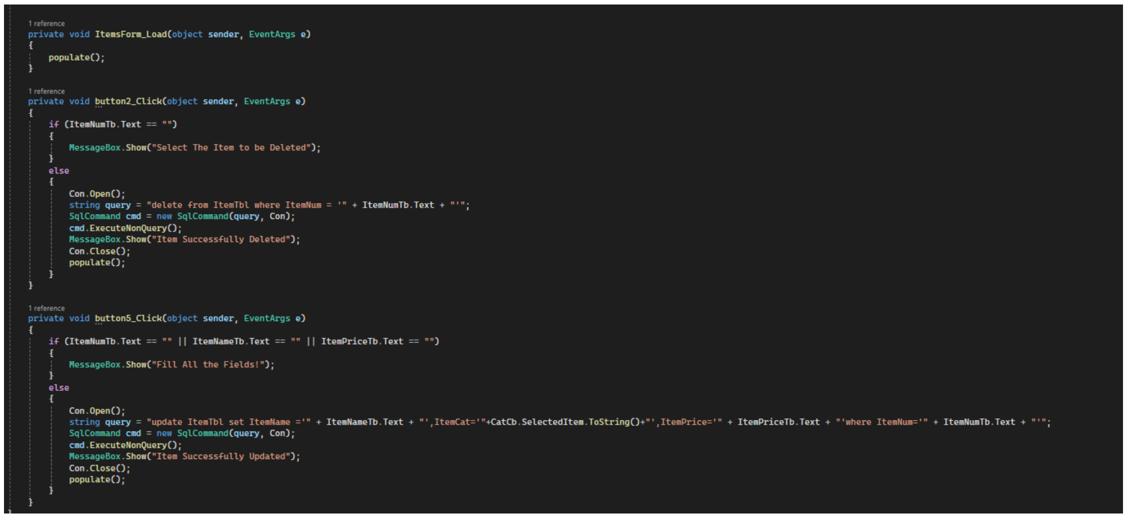
Item List

ItemNum	ItemName	ItemCat	ItemPrice
1	Milk Shake	Beverage	250
2	Coffee	Beverage	100
3	Cake	Food	500
5	Milk	Beverage	150
6	Bread	Food	200
8	Tea	Beverage	100

Item Form 1

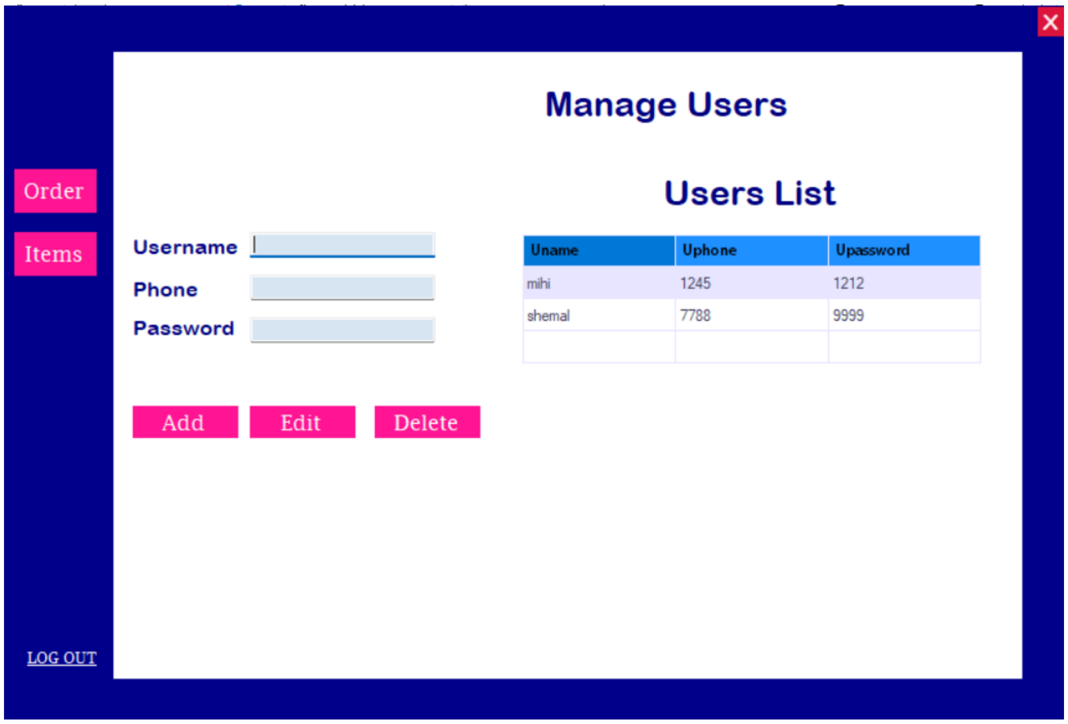


Name	Data Type	Allow Nulls	Default
ItemNum	int	☐	
ItemName	varchar(50)	☐	
ItemCat	varchar(20)	☐	
ItemPrice	int	☐	
		☐	



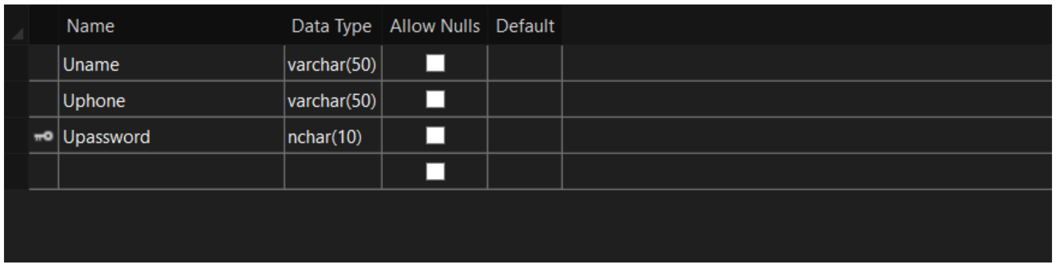
```
1 reference
private void ItemsForm_Load(object sender, EventArgs e)
{
    populate();
}

1 reference
private void button2_Click(object sender, EventArgs e)
{
    if (ItemNumTb.Text == "")
    {
        MessageBox.Show("Select The Item to be Deleted");
    }
    else
    {
        Con.Open();
        string query = "delete from ItemTbl where ItemNum = '" + ItemNumTb.Text + "'";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Item Successfully Deleted");
        Con.Close();
        populate();
    }
}

1 reference
private void button5_Click(object sender, EventArgs e)
{
    if (ItemNumTb.Text == "" || ItemNameTb.Text == "" || ItemPriceTb.Text == "")
    {
        MessageBox.Show("Fill All the Fields!");
    }
    else
    {
        Con.Open();
        string query = "update ItemTbl set ItemName ='" + ItemNameTb.Text + "',ItemCat='"+CatCb.SelectedItem.ToString()+"',ItemPrice='" + ItemPriceTb.Text + "'where ItemNum='" + ItemNumTb.Text + "'";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Item Successfully Updated");
        Con.Close();
        populate();
    }
}
```

Code Snippet For Edit and Delete Buttons

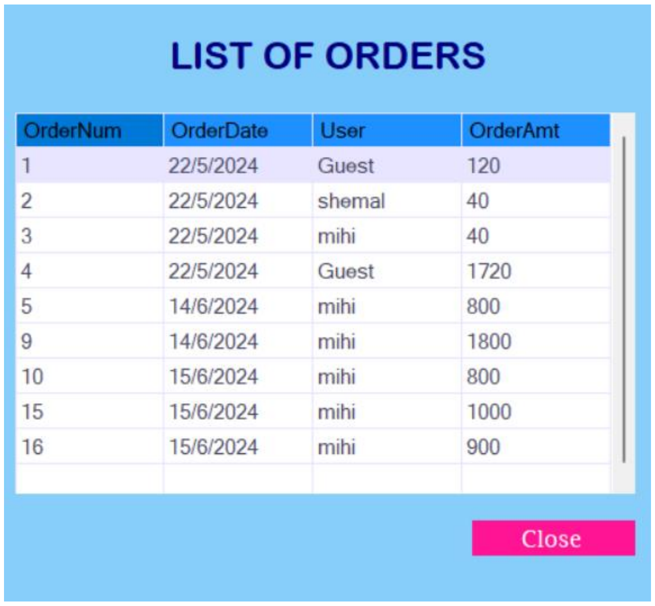


Uname	Uphone	Upassword
mihi	1245	1212
shemal	7788	9999

Users Form



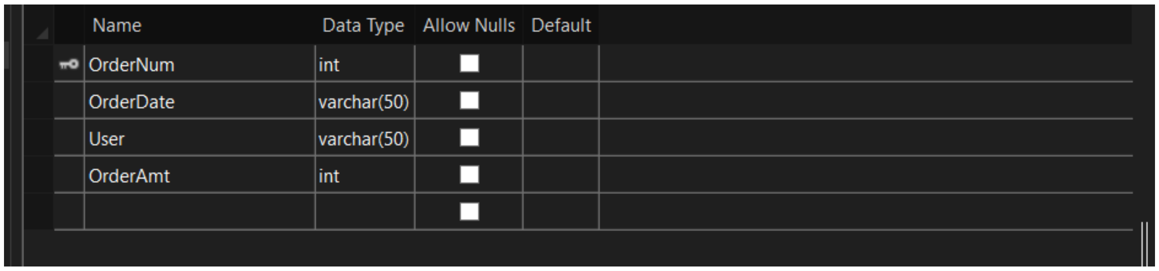
Name	Data Type	Allow Nulls	Default
Uname	varchar(50)	☐	
Uphone	varchar(50)	☐	
Upassword	nchar(10)	☐	
		☐	



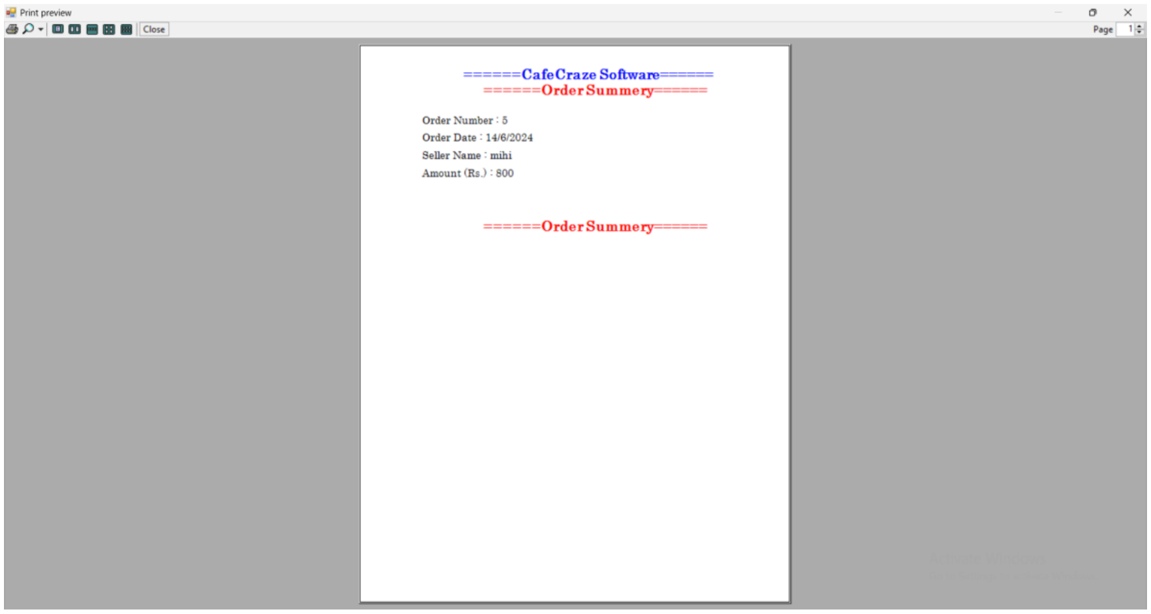
LIST OF ORDERS

OrderNum	OrderDate	User	OrderAmt
1	22/5/2024	Guest	120
2	22/5/2024	shemal	40
3	22/5/2024	mihi	40
4	22/5/2024	Guest	1720
5	14/6/2024	mihi	800
9	14/6/2024	mihi	1800
10	15/6/2024	mihi	800
15	15/6/2024	mihi	1000
16	15/6/2024	mihi	900

List of Orders



Name	Data Type	Allow Nulls	Default
OrderNum	int	☐	
OrderDate	varchar(50)	☐	
User	varchar(50)	☐	
OrderAmt	int	☐	
		☐	

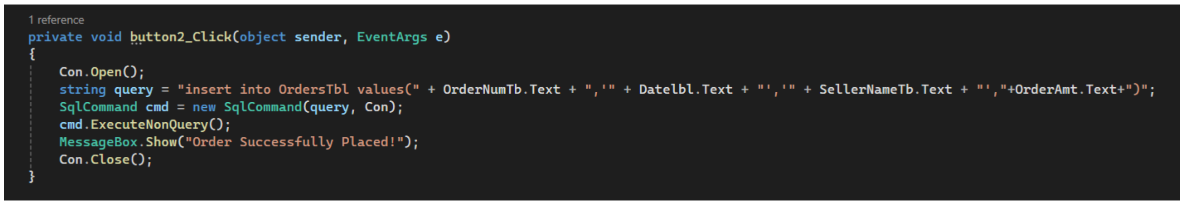


======CafeCraze Software======
======Order Summery======

Order Number : 5
Order Date : 14/6/2024
Seller Name : mihi
Amount (Rs.) : 800

======Order Summery======

Print Order Summery



```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    Con.Open();
    string query = "insert into OrdersTbl values(" + OrderNumTb.Text + ",'" + Datelbl.Text + "','" + SellerNameTb.Text + "',"+OrderAmt.Text+")";
    SqlCommand cmd = new SqlCommand(query, Con);
    cmd.ExecuteNonQuery();
    MessageBox.Show("Order Successfully Placed!");
    Con.Close();
}
```

Code Snippet For Place the Order

6. Appendix

I. Login Form

```
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 CafeCraze__Cafe_Management_System
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        SqlConnection Con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mihir\Documents\Cafedb.m
df;Integrated Security=True;Connect Timeout=30");


        private void label4_Click(object sender, EventArgs e)
        {
            this.Hide();
            GuestOrder guest=new GuestOrder ();
            guest.Show ();
        }

        private void label6_Click(object sender, EventArgs e)
        {

        }

        private void label7_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        public static string user;
        private void button1_Click(object sender, EventArgs e)
        {
            /*UserOrder uorder= new UserOrder ();
            uorder.Show ();
            this.Hide();*/
            user= UnameTb.Text;
            if (UnameTb.Text=="" || PasswordTb.Text=="")
            {
                MessageBox.Show("Enter Username and Password");

            }
            else
            {
                Con.Open();
                SqlDataAdapter sda = new SqlDataAdapter("select count(*) from
UsersTbl where Uname='"+UnameTb.Text+"'and
Upassword='"+PasswordTb.Text+"'",Con);
                DataTable dt = new DataTable();
                sda.Fill(dt);
                if (dt.Rows[0][0].ToString() == "1")
                {
                    UserOrder uorder = new UserOrder();
                    uorder.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Wrong Username or Password");
                }
                Con.Close();
            }
        }
    }
}
```

II. User Order Form

```
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 CafeCraze__Cafe_Management_System
{
    public partial class UserOrder : Form
    {
        public UserOrder()
        {
            InitializeComponent();
        }
        SqlConnection Con = new SqlConnection(@"Data 
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mihir\Documents\Cafedb.m
df;Integrated Security=True;Connect Timeout=30");
        void populate()
        {
            Con.Open();
            string query = "select * from ItemTbl";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            ItemsGV.DataSource = ds.Tables[0];

            Con.Close();

        }
        void filterbycategory()
        {
            Con.Open();
            string query = "select * from ItemTbl where Itemcat='"+ 
categorycb.SelectedItem.ToString() + "'";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            ItemsGV.DataSource = ds.Tables[0];

            Con.Close();

        }
        int num = 0;
        int price,total;
        string item, cat;



        private void label4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 login = new Form1();
            login.Show();
        }

        private void label7_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            ItemsForm Item = new ItemsForm();
            Item.Show();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.Hide();
            UsersForm user = new UsersForm();
            user.Show();
        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }



        private void label5_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (QtyTb.Text == "")
            {
                MessageBox.Show("What is the Quantity of Item?");

            }
            else if (flag == 0)
            {
                MessageBox.Show("Select The Product to be Ordered");

            }
            else
            {
                num = num + 1;
                total = price * Convert.ToInt32(QtyTb.Text);
                table.Rows.Add(num, item, cat, price, total);
                OrdersGv.DataSource = table;
                flag = 0;
            }
            sum = sum + total;
            OrderAmt.Text = "" + sum;
        }

        private void UsersGV_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void ItemsGV_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            item = ItemsGV.SelectedRows[0].Cells[1].Value.ToString();
            cat = ItemsGV.SelectedRows[0].Cells[2].Value.ToString();
            price =
Convert.ToInt32(ItemsGV.SelectedRows[0].Cells[3].Value.ToString());
            flag = 1;
        }

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

        DataTable table = new DataTable();

        private void comboBox1_SelectionChangeCommitted(object sender, EventArgs
e)
        {
            filterbycategory();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            populate();
        }

        private void OrdersGv_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Con.Open();
            string query = "insert into OrdersTbl values(" + OrderNum.Text +
",'" + Datelbl.Text + "','" + SellerName.Text + "'," + OrderAmt.Text + ")";
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Order Successfully Placed!");
            Con.Close();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            ViewOrders view =new ViewOrders();
            view.Show();
        }

        int flag = 0;
        int sum = 0;
        private void UserOrder_Load(object sender, EventArgs e)
        {
            populate();
            table.Columns.Add("OrderNumber", typeof(int));
            table.Columns.Add("ItemName", typeof(string));
            table.Columns.Add("Category", typeof(string));
            table.Columns.Add("UnitPrice", typeof(int));
            table.Columns.Add("TotalPrice", typeof(int));
            OrdersGv.DataSource = table;
            Datelbl.Text = DateTime.Today.Day.ToString() + "/" +
DateTime.Today.Month.ToString() + "/" + DateTime.Today.Year.ToString();
            SellerName.Text = Form1.user;
        }



    }
```

```
}
```

III. Guest Order Form

```
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;
using System.Web.UI.WebControls;

namespace CafeCraze__Cafe_Management_System
{
    public partial class GuestOrder : Form
    {
        public GuestOrder()
        {
            InitializeComponent();
        }
        SqlConnection Con = new SqlConnection(@"Data 
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mihir\Documents\Cafedb.m
df;Integrated Security=True;Connect Timeout=30");
        void populate()
        {
            Con.Open();
            string query = "select * from ItemTbl";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            ItemsGV.DataSource = ds.Tables[0];

            Con.Close();

        }
        void filterbycategory()
        {
            Con.Open();
            string query = "select * from ItemTbl where Itemcat='" + 
categorycb.SelectedItem.ToString() + "'";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            ItemsGV.DataSource = ds.Tables[0];

            Con.Close();
```

```
}
private void label7_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}
int num = 0;
int price, qty, total;
string item, cat;
DataTable table = new DataTable();
int flag = 0;
int sum = 0;
private void button1_Click(object sender, EventArgs e)
{
    if (QtyTb.Text == "")
    {
        MessageBox.Show("What is the Quantity of Item?");

    }
    else if (flag == 0)
    {
        MessageBox.Show("Select The Product to be Ordered");

    }
    else
    {
        num = num + 1;
        total = price * Convert.ToInt32(QtyTb.Text);
        table.Rows.Add(num, item, cat, price, total);
        OrdersGv.DataSource = table;
        flag = 0;
    }
    sum = sum + total;
    OrderAmt.Text = "" + sum;
}

private void label3_Click(object sender, EventArgs e)
{

}

private void panel1_Paint(object sender, PaintEventArgs e)
{

}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{

}

private void textBox2_TextChanged(object sender, EventArgs e)
{
        }

        private void GuestOrder_Load(object sender, EventArgs e)
        {
            populate();
            table.Columns.Add("OrderNumber", typeof(int));
            table.Columns.Add("ItemName", typeof(string));
            table.Columns.Add("Category", typeof(string));
            table.Columns.Add("UnitPrice", typeof(int));
            table.Columns.Add("TotalPrice", typeof(int));
            OrdersGv.DataSource = table;
            Datelbl.Text=DateTime.Today.Day.ToString()+"/"+
DateTime.Today.Month.ToString()+"/"+DateTime.Today.Year.ToString();
        }

        private void ItemsGV_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Con.Open();
            string query = "insert into OrdersTbl values(" + OrderNumTb.Text +
",'" + Datelbl.Text + "','" + SellerNameTb.Text + "',"+OrderAmt.Text+")";
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Order Successfully Placed!");
            Con.Close();
        }

        private void OrdersGv_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void ItemsGV_CellContentClick_1(object sender,
DataGridViewCellEventArgs e)
        {
            item = ItemsGV.SelectedRows[0].Cells[1].Value.ToString();
            cat = ItemsGV.SelectedRows[0].Cells[2].Value.ToString();
            price =
Convert.ToInt32(ItemsGV.SelectedRows[0].Cells[3].Value.ToString());
            flag = 1;
        }

        private void label2_Click_1(object sender, EventArgs e)
        {

        }

        private void label2_Click_2(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 login=new Form1();
            login.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {

        }
        private void categorycb_SelectionChangeCommitted(object sender,
EventArgs e)
        {
            filterbycategory();
        }
    }
}
```

IV. User Form (This is used to authenticated manage users)

```
 using System;
using System.Data;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace CafeCraze__Cafe_Management_System
{
    public partial class UsersForm : Form
    {
        public UsersForm()
        {
            InitializeComponent();
        }
        SqlConnection Con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mihir\Documents\Cafedb.m
df;Integrated Security=True;Connect Timeout=30");
        void populate()
        {
            Con.Open();
            string query = "select * from UsersTbl";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            UsersGV.DataSource = ds.Tables[0];

            Con.Close();

        }
        private void button3_Click(object sender, EventArgs e)
        {
            UserOrder uorder = new UserOrder();
            uorder.Show();
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ItemsForm item = new ItemsForm();
            item.Show();
            this.Hide();
        }

        private void label4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 login = new Form1();
            login.Show();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Con.Open();
            string query = "insert into UsersTbl values('" + unameTb.Text +
"','" + uphoneTb.Text + "','" + upassTb.Text + "')";
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("User Successfully Created!");
            Con.Close();
            populate();
        }

        private void label7_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }


        private void UsersForm_Load(object sender, EventArgs e)
        {
            populate();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void UsersGV_CellContentClick(object sender, 
DataGridViewCellEventArgs e)
        {

            unameTb.Text = UsersGV.SelectedRows[0].Cells[0].Value.ToString();
            uphoneTb.Text = UsersGV.SelectedRows[0].Cells[1].Value.ToString();
            upassTb.Text = UsersGV.SelectedRows[0].Cells[2].Value.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (uphoneTb.Text == "")
            {
                MessageBox.Show("Select The User to be Deleted");
            }
            else
            {
                Con.Open();
                string query = "delete from UsersTbl where Uphone = '" + 
uphoneTb.Text + "'";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("User Successfully Deleted");
                Con.Close();
                populate();
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (uphoneTb.Text == "" || upassTb.Text == "" || unameTb.Text == "")
            {
                MessageBox.Show("Fill All the Fields!");
            }
            else
            {
                Con.Open();
                string query = "update UsersTbl set Uname ='" + unameTb.Text + 
"',Upassword='" + upassTb.Text + "'where Uphone='" + uphoneTb.Text + "'";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("User Successfully Updated");
                Con.Close();
                populate();
            }

        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {
        }
    }
}
```

V. Item Form (This is used to authenticated manage items)

```
 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;
using System.Data.SqlClient;
namespace CafeCraze__Cafe_Management_System
{
    public partial class ItemsForm : Form
    {
        public ItemsForm()
        {
            InitializeComponent();
        }
        SqlConnection Con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mihir\Documents\Cafedb.m
df;Integrated Security=True;Connect Timeout=30");
        void populate()
        {
            Con.Open();
            string query = "select * from ItemTbl";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            ItemsGV.DataSource = ds.Tables[0];
            Con.Close();

        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            UserOrder order = new UserOrder();
            order.Show();
        }

        private void label7_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }

        private void button4_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 login = new Form1();
            login.Show();
        }

        private void UsersGV_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            ItemNumTb.Text = ItemsGV.SelectedRows[0].Cells[0].Value.ToString();
            ItemNameTb.Text = ItemsGV.SelectedRows[0].Cells[1].Value.ToString();
            CatCb.SelectedItem =
ItemsGV.SelectedRows[0].Cells[2].Value.ToString();
            ItemPriceTb.Text =
ItemsGV.SelectedRows[0].Cells[3].Value.ToString();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (ItemNameTb.Text == "" || ItemNumTb.Text == "" ||
ItemPriceTb.Text == "")
            {
                MessageBox.Show("Fill All the Data!");
            }
            else
            {

                Con.Open();
                string query = "insert into ItemTbl values(" + ItemNumTb.Text +
",'" +ItemNameTb.Text + "','" + CatCb.SelectedItem.ToString() + "'," +
ItemPriceTb.Text + ")";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Item Successfully Created!");
                Con.Close();
                populate();
            }
        }

        private void ItemsForm_Load(object sender, EventArgs e)
        {
            populate();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (ItemNumTb.Text == "")
            {
                MessageBox.Show("Select The Item to be Deleted");
            }
            else
            {
                Con.Open();
                string query = "delete from ItemTbl where ItemNum = '" +
ItemNumTb.Text + "'";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Item Successfully Deleted");
                Con.Close();
                populate();
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (ItemNumTb.Text == "" || ItemNameTb.Text == "" ||
ItemPriceTb.Text == "")
            {
                MessageBox.Show("Fill All the Fields!");
            }
            else
            {
                Con.Open();
                string query = "update ItemTbl set ItemName ='" +
ItemNameTb.Text + "',ItemCat='"+CatCb.SelectedItem.ToString()+"',ItemPrice='" +
ItemPriceTb.Text + "'where ItemNum='" + ItemNumTb.Text + "'";
                SqlCommand cmd = new SqlCommand(query, Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Item Successfully Updated");
                Con.Close();
                populate();
            }
        }
    }
}
```

VI. View Order Form (This is used to view order items)

```
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;
using System.Data.SqlClient;

namespace CafeCraze__Cafe_Management_System
{
    public partial class ViewOrders : Form
    {
        public ViewOrders()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

        SqlConnection Con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mihir\Documents\Cafedb.m
df;Integrated Security=True;Connect Timeout=30");
        void populate()
        {
            Con.Open();
            string query = "select * from OrdersTbl";
            SqlDataAdapter sda = new SqlDataAdapter(query, Con);
            SqlCommandBuilder builder = new SqlCommandBuilder(sda);
            var ds = new DataSet();
            sda.Fill(ds);
            OrdersGV.DataSource = ds.Tables[0];

            Con.Close();

        }
        private void ViewOrders_Load(object sender, EventArgs e)
        {
            populate();
        }

        private void OrdersGV_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if(printPreviewDialog1.ShowDialog() == DialogResult.OK)
            {
                printDocument1.Print();
            }
        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            e.Graphics.DrawString("======CafeCraze Software======", new
Font("Century", 20, FontStyle.Bold), Brushes.Blue, new Point(200,40));
            e.Graphics.DrawString("======Order Summery======",new
Font("Century",20, FontStyle.Bold),Brushes.Red,new Point(240,70));
            e.Graphics.DrawString("Order Number : " +
OrdersGV.SelectedRows[0].Cells[0].Value.ToString(), new Font("Century", 15,
FontStyle.Regular), Brushes.Black, new Point(120, 135));
            e.Graphics.DrawString("Order Date : " +
OrdersGV.SelectedRows[0].Cells[1].Value.ToString(), new Font("Century", 15,
FontStyle.Regular), Brushes.Black, new Point(120, 170));
            e.Graphics.DrawString("Seller Name : " +
OrdersGV.SelectedRows[0].Cells[2].Value.ToString(), new Font("Century", 15,
FontStyle.Regular), Brushes.Black, new Point(120, 205));
            e.Graphics.DrawString("Amount (Rs.) : " +
OrdersGV.SelectedRows[0].Cells[3].Value.ToString(), new Font("Century", 15,
FontStyle.Regular), Brushes.Black, new Point(120, 240));
            e.Graphics.DrawString("======Order Summery======", new
Font("Century", 20, FontStyle.Bold), Brushes.Red, new Point(240, 340));
        }
    }
}
```

6. References

[1] W. M. Sari, F. Fitria, E. Suharto, Y. Ikhwani, W. Wagino, N. Alamsyah, and A. P. Windarto. "Improving the Quality of Management with the Concept of Decision Support Systems in Determining Factors for Choosing a Cafe based on Consumers". In Proceedings of the 1st Bukittinggi International Conference on Education, Journal of Physics: Conference Series, Volume 1471, October 17-18, 2019, West Sumatera, Indonesia, 2020, pp. 012009.

[2] 1 S. Macha. Management System for a Restaurant. *Governors State University, Fall 2022, pp. 1-120.*

[3] Fundamentals of Computer Programming With C#, ISBN 978-954-400-773-7. s.l. : Svetlin Nakov & Co.

[4] Abhi. "Coffee Shop Management System C#". Internet: https://www.scribd.com/document/435834741/Coffee-shop-management-system-c, Nov. 19, 2019 [June 14,2024.]

[5] SQL Server, .NET and C# Video Tutorial. "LINQ Tutorial". Internet: https://csharp-video-tutorials.blogspot.com/2014/07/linq-tutorial.html,[June13,2024.]

[6] Diwale, S. S. (2020). "Coffee Shop Management: A Project Report". ProjectReportProject ReportProjectReport. Dnyandeep College of Science and Commerce, Khed, Maharashtra, India. Internet: https://www.dnyandeepcollege.org/NAAC/Project%20Documentation.pdf

[7] Goodison, J. (2022, June 29). *How I would learn to code (If I could start over)* [Video]. YouTube. https://www.youtube.com/watch?v=GhQdlIFylQ8