AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Dept. of Computer Science
Faculty of Science and Technology

CSC2210: OBJECT ORIENTED PROGRAMMING 2

Fall 2024-2025

Section: [D]

Group No: 05

Project Report On

*Project Name [**Restaurant Point of Sale]***

Supervised By

Md. Hasibul Hasan

Submitted By:

Name	ID
1. Tahmid Bin Morshed	**22-49439-3**
2. Redwanul Haque Mazumder	**22-48481-3**
3. Sabeha Aktar Emu	**22-49471-3**

Obtained Marks for CO2 and CO3 (Description given in the following page)

Assessment Criteria	Not Attended/ Incorrect (0)	Inadequate (1-2)	Average (3)	Good (4)	Excellent (5)

Evaluation Criteria (CO2)	Total =	Evaluation Criteria (CO3)	Total =
Requirement fulfillment		Organization of the application	
Validation		Representation and Integration of Database	
Verification		Graphical User Interface	

CO2: Display and verify the mean of a real-life Project using the concepts of C# Graphical User Interface based environment with database integration to depict a desktop-based application.					
Assessment Criteria	**Not Attended/ Incorrect (0)**	**Inadequate (1-2)**	**Average (3)**	**Good (4)**	**Excellent (5)**
Evaluation Criteria	**Evaluation Definition**				
Requirement fulfillment	Fails to demonstrate any understanding of real-life scenario-based project development or functional requirement identification. There is no attempt to depict a project or identify functional requirements accurately.	Demonstrates limited understanding of real-life scenario-based project development and functional requirement identification. The project depicted lacks coherence or relevance to real-life scenarios, and functional requirements are inaccurately identified or insufficiently described.	Presents a basic depiction of a real-life scenario-based project and identifies some functional requirements. However, the project lacks depth or complexity, and some functional requirements may be vaguely defined or missing key details.	Effectively demonstrates a realistic scenario-based project and accurately identifies most functional requirements. The project is well-developed with appropriate complexity, and functional requirements are clearly articulated with relevant details.	Exhibits an exceptional understanding of real-life scenario-based project development and accurately identifies all functional requirements. The project is meticulously developed with thorough attention to detail, reflecting a comprehensive understanding of Object-Oriented Programming project development activities.
Validation	Fails to demonstrate any understanding or implementation of validation forms in their system. There is no attempt to deal with data validation, and validation requirements are completely ignored or incorrectly applied.	Demonstrates limited understanding of validation forms and data validation techniques. While some attempt may be made to implement validation, it is incomplete or poorly executed, leading to inadequate handling of data validation.	Shows a basic understanding of validation forms and data validation techniques. They attempt to implement validation, but some aspects may be missing or incorrectly implemented, resulting in partial or inconsistent handling of data validation.	Effectively demonstrates the use of validation forms and implements data validation techniques. Validation is mostly accurate and comprehensive, ensuring the proper handling of data input and verification in the system.	Exhibits an exceptional understanding and implementation of validation forms and data validation techniques. Validation is meticulously implemented with thorough attention to detail, ensuring robust data validation procedures and contributing to the overall reliability and integrity of the system.
Verification	Fails to demonstrate any attempt to verify the system data or functional requirements. There is no evidence of understanding or implementation of verification processes, and	Demonstrates limited understanding of verification processes and data flow in the system. Verification attempts are incomplete or inaccurate, and there is	Shows a basic understanding of verification processes and attempts to verify system data. However, verification efforts may be inconsistent or lack thoroughness,	Identifies and verifies system data, ensuring proper functional requirements are met. Verification efforts are mostly accurate and thorough, with attention to ensuring data integrity and	Exhibits an exceptional understanding of verification processes and meticulously verifies system data. Verification efforts are comprehensive and precise, with a keen focus on

	data flow is not considered.	insufficient consideration given to ensuring data integrity and functionality.	and there may be gaps in ensuring proper functional requirements and data flow.	appropriate data flow within the system.	ensuring all functional requirements are met and maintaining proper data flow throughout the system.

CO3: Prepare and Explain a real life desktop based application synthesizing several component of C# along with development tools to adhere the given requirements.

Assessment Criteria	Not Attended/ Incorrect (0)	Inadequate (1-2)	Average (3)	Good (4)	Excellent (5)
Evaluation Criteria	**Evaluation Definition**				
Organization of the application	Fails to identify any suitable real time application or requirements for project development activities related to OOP.	Limited understanding about the project scopes and scenarios or identification of functional requirements.	Lacks depth or relevance to OOP project development activities and may contain inaccuracies. Real-life scenarios are mentioned, but the discussion lacks depth or clarity.	Consider and integrate the idea of several core aspects of the project along with relevance to real-life scenarios. Demonstrating a solid understanding of the application presentation.	Generalize and exhibits an exceptional understanding of project preparation according to a to real-life scenarios. Also contains proper and insightful identification of the system which is comprehensive and precise.
Representation and Integration of Database	Fails to identify and present any understanding or implementation of database. Also failed to integrate the data with the project itself.	Limited understanding of the database concepts or their proper way of using in a real time project. While some attempt may be made to implement but it is incomplete or poorly executed, leading to inadequate design.	Lacks depth or relevance to database integration with the application. Shows a basic understanding but some aspects may be missing or incorrectly implemented, resulting in partial or inconsistency. May lack proper normalization.	Integrate the database with the forms properly and implements it with proper validation which is mostly accurate and comprehensive, ensuring the proper handling of data input and verification along with general normalization.	Exhibits an exceptional understanding and implementation of database ensuring attention to detail, and robust data manipulation procedures and contributing to the overall clarity.
Graphical User Interface	Fails to present or prepare GUI based application interfaces. There is no evidence of creating or integrating such things according to their usefulness.	Limited understanding of graphical user interfaces. Lack of design knowledge. Very poor attempt to make such things which are currently obsolete or can't be identified as coherent.	Shows a basic understanding of creating user interfaces. Most of them are interconnected but maybe some of them lack it. However, most of it can be described as user friendly.	Effectively identifies and meet the consider the simplicity. Design related works are mostly accurate and taken proper attention to ensuring a user-friendly coherent system.	Exhibits an exceptional work design following a high standard of simple and elegant work. Several controls and mechanism has been organized in a preferred way according to the coherent usage .

Table of Contents: **Page no.**

Introduction

The Restaurant Point of Sale (POS) System is a cutting-edge software solution designed to enhance and simplify the operational and transactional workflows of a restaurant. This system focuses on providing an intuitive interface for efficient order management, payment processing, and administrative control. It is tailored for two types of users – Admin and Staff – ensuring a role-based approach to restaurant operations.

The admin role includes powerful features for managing the restaurant's menu, overseeing user accounts, and generating sales reports to track performance and revenue. Meanwhile, Staff users are provided with tools to take customer orders quickly and process payments with ease, ensuring a seamless dining experience for customers.

Developed using Windows Forms (C#) and a robust SQL database backend, this system integrates essential Point of Sale features such as role-based authentication, menu management, real-time reporting, and payment processing. These capabilities make it an indispensable tool for restaurants looking to modernize operations, increase efficiency, and improve customer satisfaction.

The system serves as a centralized platform to address the typical challenges faced by restaurants, such as slow order processing, data inaccuracies, and disorganized menu management. By providing streamlined and efficient processes, the Restaurant POS System delivers a comprehensive solution for managing daily operations and achieving business goals effectively.

User Story

Actors:

1. **Admin**: An admin user manages high-level operations, including menu management and sales analysis. Admins also analyze payment reports to track financial health.
2. **Staff**: A staff user handles daily operations like creating orders, adding items to orders, and processing payments.

Entities and Relationships:

1. **Admins/Staff inherit from Users**:
    a. All users (both Admins and Staff) share common properties like UserID, Username, and Password. The system ensures secure login and access control based on the user's role.
2. **Orders and OrderItems**:
    a. Each order contains multiple items. For example, a customer ordering food will generate an order (identified by OrderID) that links to one or more OrderItems. Each item in the order has attributes like ItemName, Price, and Quantity.
3. **Payments and Orders**:
    a. Each payment is tied to an order. For dine-in, a payment can be linked to a single order (one-to-one), while for large table orders or split payments, multiple payments might be linked to the same order (one-to-many).

4. **MenuItems and MenuCategories**:

    a. Menu categories (e.g., Food, Drinks, Desserts) contain multiple menu items. Each item in the menu (e.g., Pasta, Coffee, Ice Cream) belongs to a specific category, which simplifies organization and lookup.
5. **Staff and Orders**:
    a. Staff members create and manage orders. They are responsible for taking customer requests and entering them into the system. This ensures that every order is tied to a specific staff member.

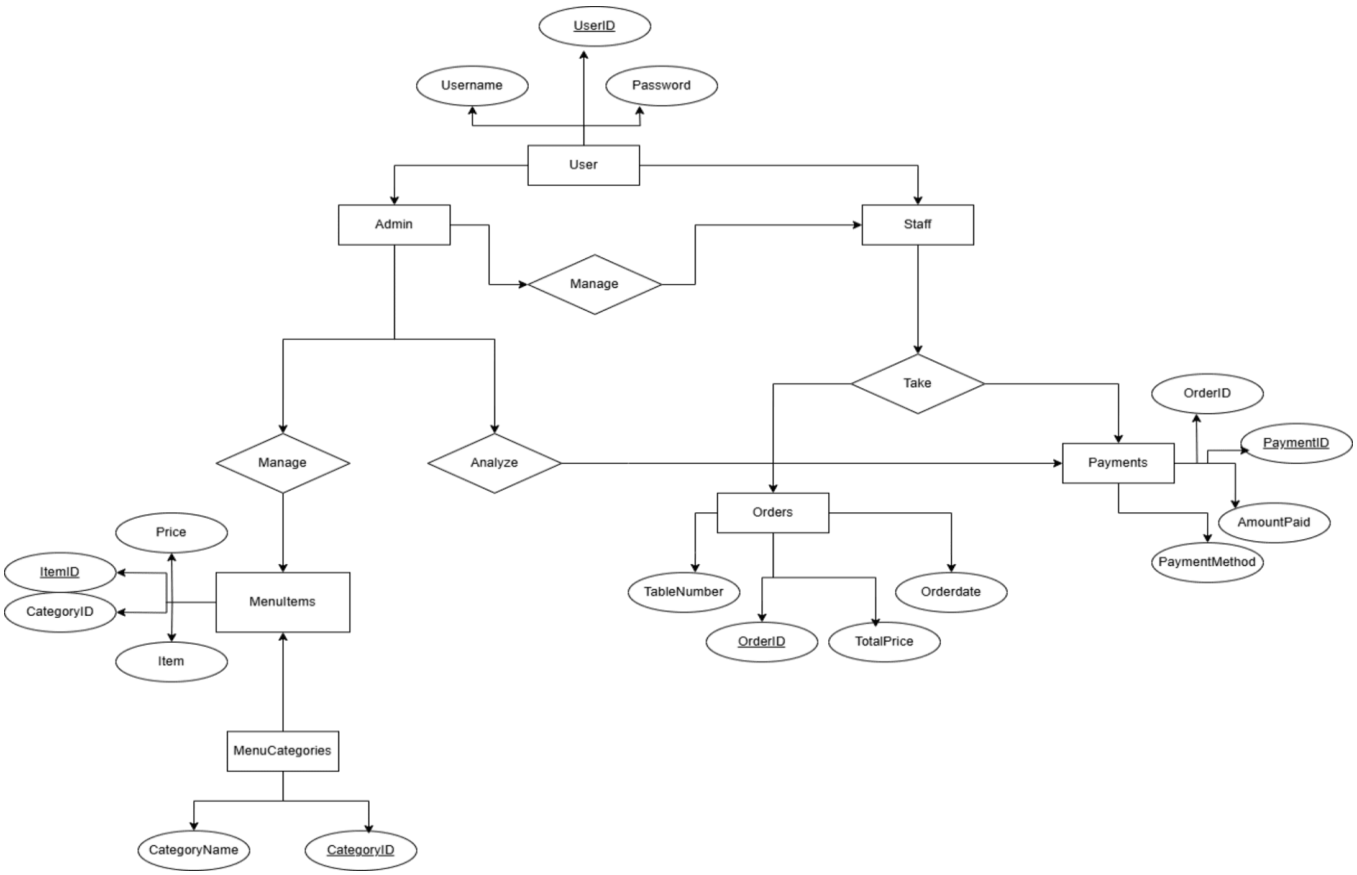
System Flow and User Stories:

1. **Admin Managing Menu**:
    a. *As an Admin*, I can create new menu categories (e.g., Drinks) and add items (e.g., Latte, Cappuccino) under these categories.
    b. I can edit or remove items from the menu as per customer demand or availability.
2. **Admin Analyzing Payments**:
    a. *As an Admin*, I can view payment reports to understand trends, revenue, and payment methods. This helps in decision-making for promotions and operations.
3. **Staff Creating Orders**:
    a. *As a Staff*, I can create an order for a table, add multiple items to it, and calculate the total cost.
    b. I can also split or merge orders if requested by the customer.
4. **Staff Processing Payments**:
    a. *As a Staff*, I can process payments for each order. Payments can be handled in various methods (e.g., cash, credit card, mobile payments) and are linked to the corresponding order for record-keeping.
5. **Tracking Table Usage**:
    a. *As a System*, I can track which table is associated with each order, including its status (e.g., reserved, occupied). This allows seamless table management.

Example Case:

1. A customer walks into a restaurant and takes a seat at a table.
2. A staff member logs into the system and creates an order for the table.
3. The staff member selects items from the menu and adds them to the order.
4. Once the customer finishes dining, the staff member processes the payment and marks the table as free.
5. At the end of the day, an Admin views the sales report and checks revenue generated and payment trends.

ER Diagram



SQL Queries

User selection query: "select * from Users where UserID = '" + this.txtBoxID.Text + "' and Password = '" + this.txtBoxPass.Text + "'"

Menu Populate query: "SELECT
MenuCategories.CategoryID,
MenuCategories.CategoryName,
MenuItems.Item,
MenuItems.Price,
MenuItems.ItemID
FROM
MenuCategories
INNER JOIN
MenuItems
ON
MenuCategories.CategoryID = MenuItems.CategoryID"

Item delete : "delete from MenuItems where ItemID = '" + itemID + "';"

Menu Update : "UPDATE MenuItems
SET Item = '" + this.txtItem.Text + @"',
Price = " + this.txtPrice.Text + @",
CategoryID = '" + this.txtCategoryID.Text + @"'

WHERE ItemID = '" + this.txtItemID.Text + "';"

Insert Item : “INSERT INTO MenuItems VALUES ('" + this.txtItemID.Text + "','" + this.txtItem.Text + "','" + this.txtCategoryID.Text + "', '" + this.txtPrice.Text + "');"

Auto Search by Item name :

```
"SELECT
    MenuCategories.CategoryID,
    MenuCategories.CategoryName,
    MenuItems.Item,
    MenuItems.Price,
    MenuItems.ItemID
FROM
    MenuCategories
INNER JOIN
    MenuItems
ON
    MenuCategories.CategoryID = MenuItems.CategoryID
WHERE
    MenuItems.Item LIKE '" + this.txtSearch.Text + "%';"
```

Populate User Management: select * from Users ORDER BY UserID;

Search Query:

SELECT * FROM Users WHERE Username LIKE 'search_text%';

Check for Existing User Query

SELECT * FROM Users WHERE UserID = 'specific_user_id';

Update User Query

UPDATE Users
SET Username = 'new_username', Password = 'new_password', Role = 'new_role' WHERE UserID = 'specific_user_id';

Insert User Query

INSERT INTO Users VALUES ('new_user_id', 'new_username', 'new_password', 'new_role');

Delete User Query

DELETE FROM Users WHERE UserID = 'specific_user_id';

Revenue Query

SELECT SUM(AmountPaid) AS Revenue FROM Sales;

Sales Data Query

```
SELECT PaymentDate, SUM(AmountPaid) AS TotalAmount

FROM Sales

GROUP BY PaymentDate

ORDER BY PaymentDate;
```

Total Orders Query

```
SELECT COUNT(*) AS TotalOrders FROM Sales;
```

Payment Methods Query

```
SELECT PaymentMethod, COUNT(*) AS UsageCount
FROM Sales

GROUP BY PaymentMethod

ORDER BY UsageCount DESC;
```

Orders Trend Query

```
SELECT OrderDate, COUNT(*) AS TotalOrders

FROM Orders

GROUP BY OrderDate

ORDER BY OrderDate;
```

Tables Query

```
SELECT TableID, TableNumber

FROM Tables;
```

Menu Items Query

```
SELECT Item, Price

FROM MenuItems

WHERE CategoryID = (

   SELECT CategoryID

   FROM MenuCategories

   WHERE CategoryName = 'CategoryName'

);
```

Insert Order Query

```
INSERT INTO Orders (TableNumber, TotalPrice, OrderDate)
OUTPUT INSERTED.OrderID
VALUES (TableNumber, TotalPrice, GETDATE());
```

Insert Order Items Query

```
INSERT INTO OrderItems (OrderID, ItemName, Price, Quantity)
VALUES (OrderID, 'ItemName', Price, Quantity);
```

Load Sales Data Query

```
SELECT PaymentDate, SUM(AmountPaid) AS TotalAmount
                    FROM Sales
                    GROUP BY PaymentDate
                    ORDER BY PaymentDate;
```

Update total orders Query

```
SELECT COUNT(*) AS TotalOrders FROM Sales
```

Load Payment Methods Data Query

```
SELECT PaymentMethod, COUNT(*) AS UsageCount
                     FROM Sales
                     GROUP BY PaymentMethod
                     ORDER BY UsageCount DESC;
```

Load Orders Trend Query

```
SELECT OrderDate, COUNT(*) AS TotalOrders
                    FROM Orders
                    GROUP BY OrderDate
                    ORDER BY OrderDate;
```

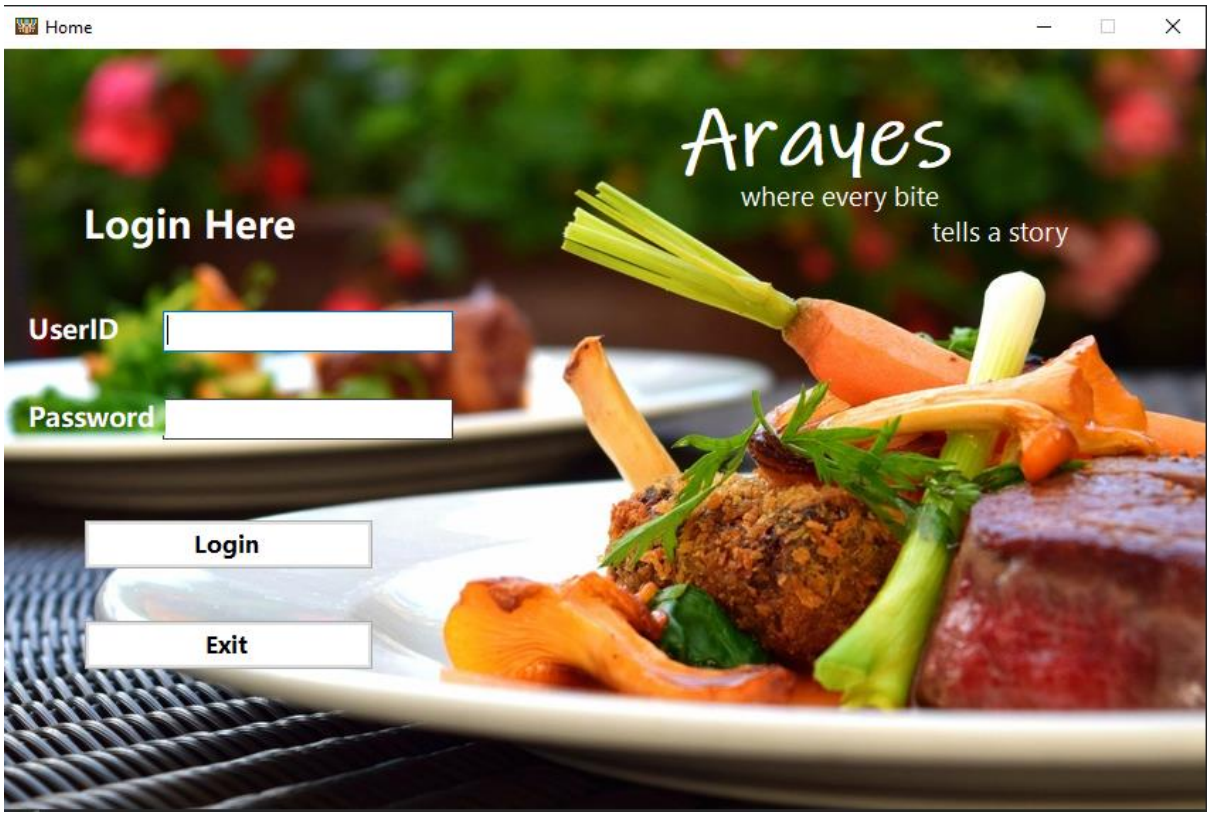


Fig 1: Login page

Admin Point Of View:

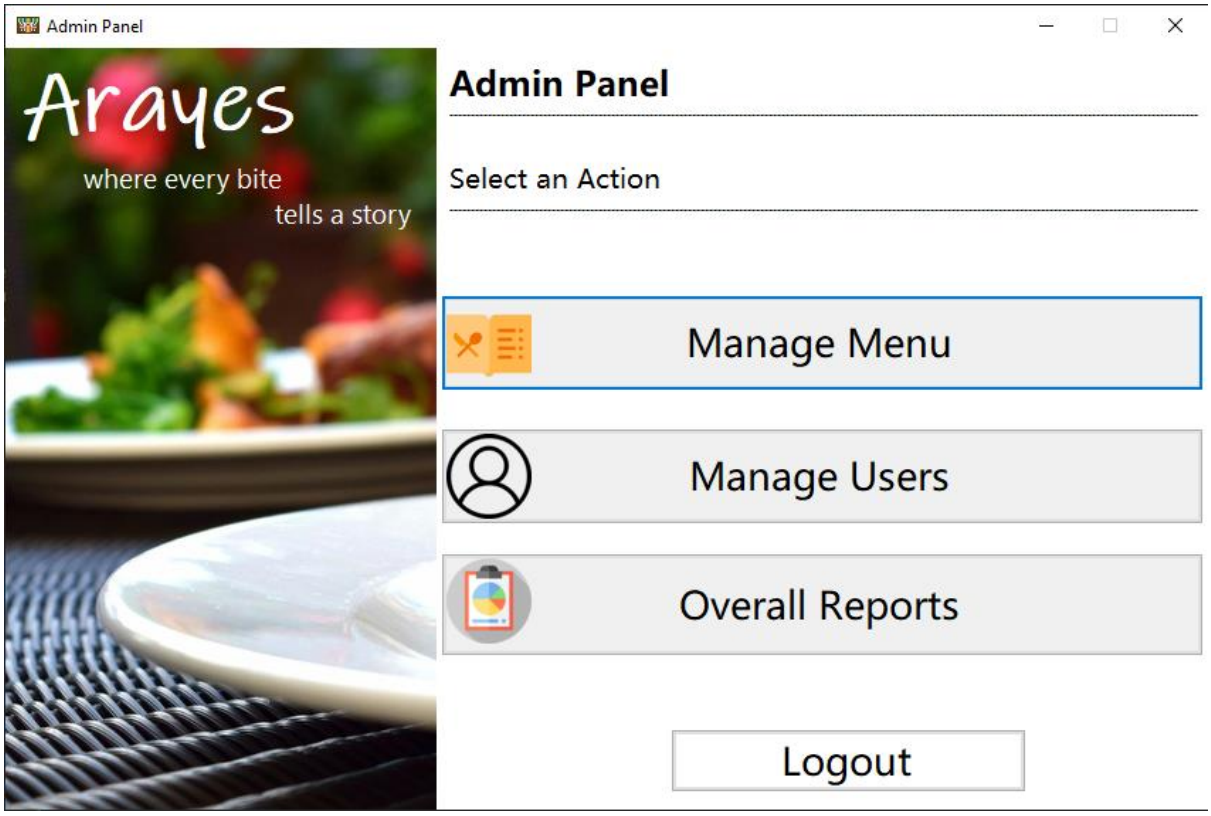


Fig 2.1 : Admin Panel

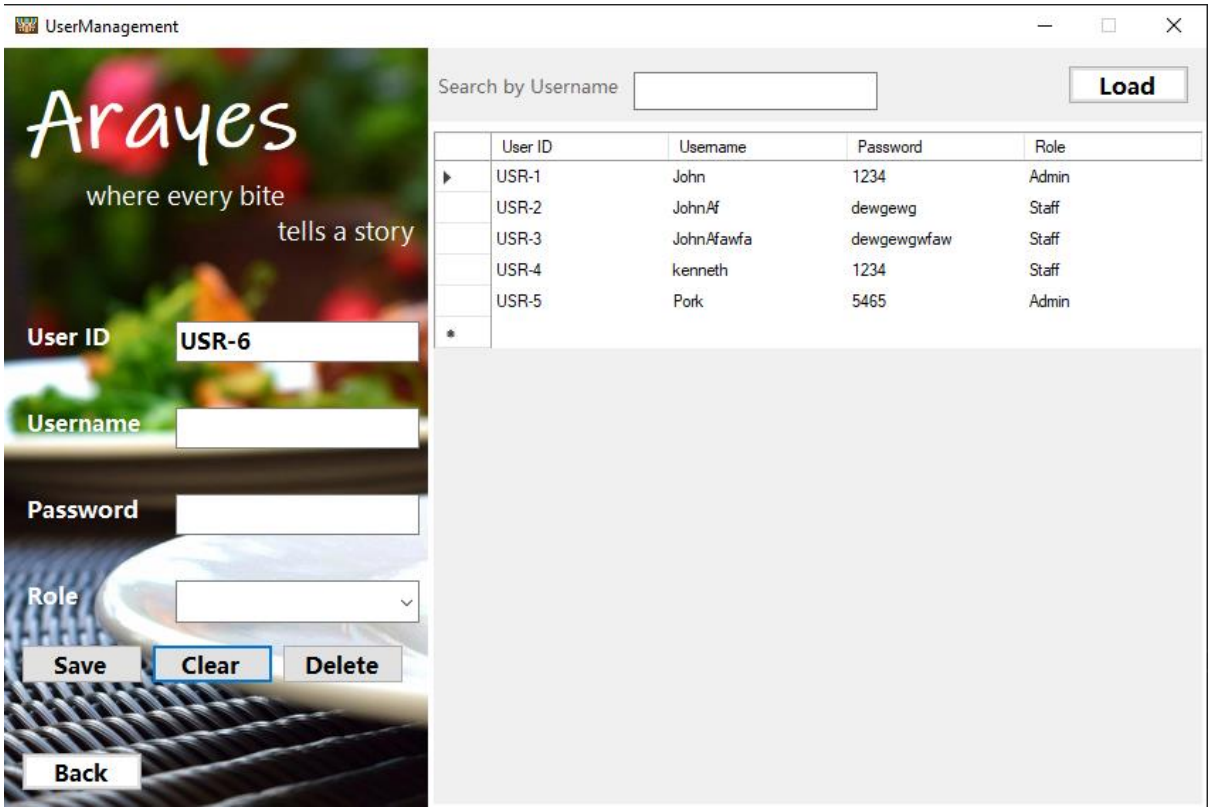


Fig 2.2 : User-Management Window

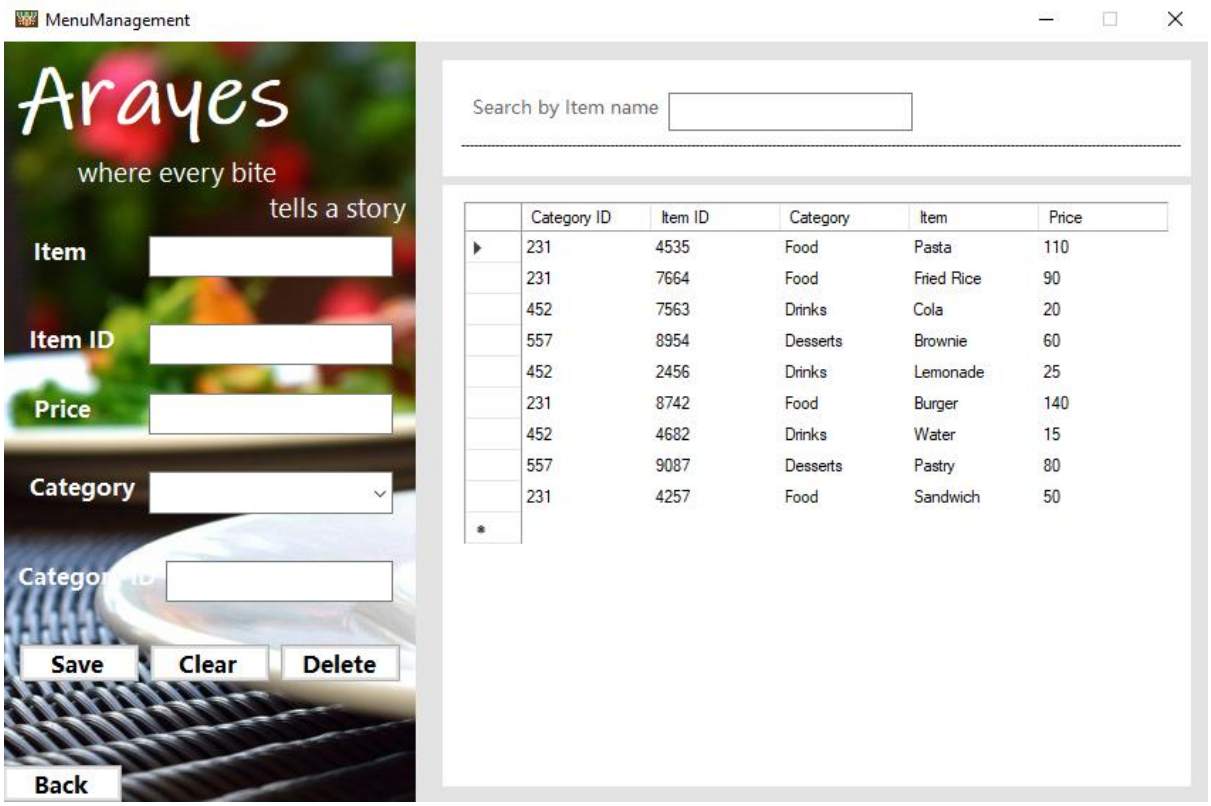


Fig 2.3: Menu-Management Window

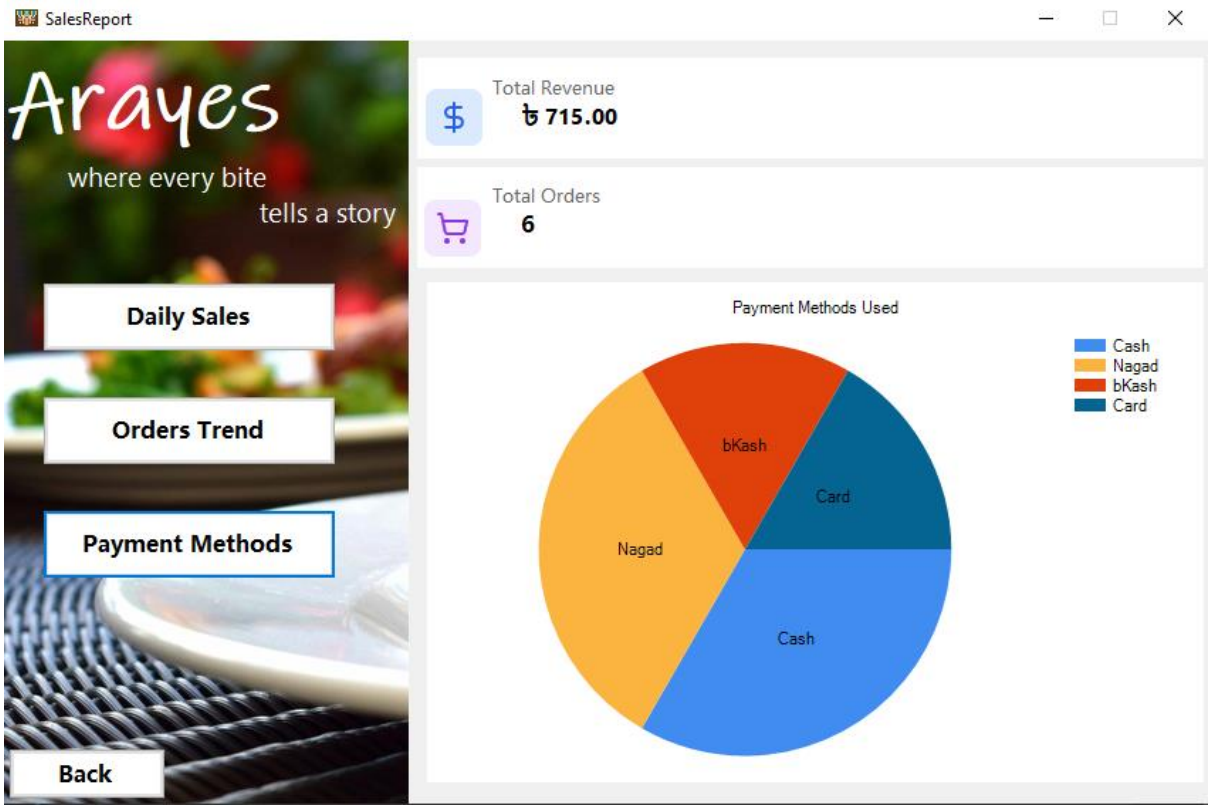


Fig 2.4 : Sales Report Window

Staff Point Of View:

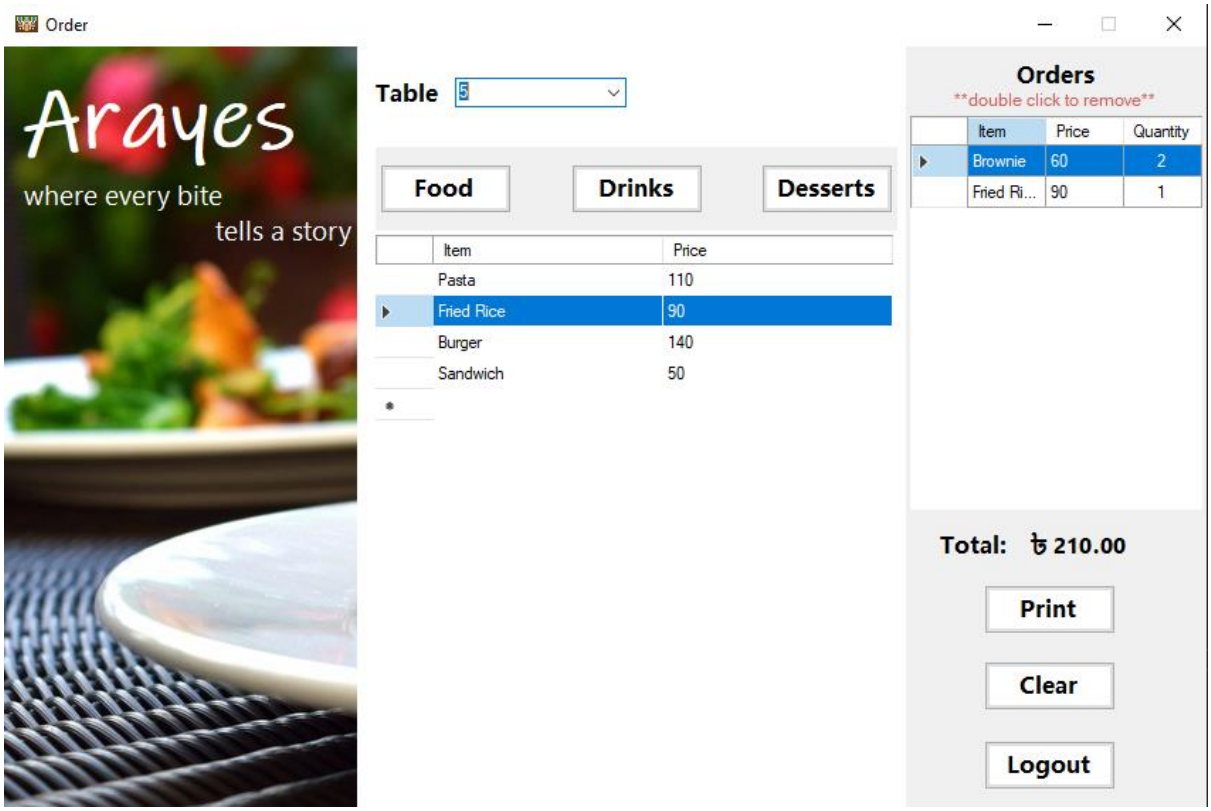


Fig 3.1 : Take Order Window for Staff

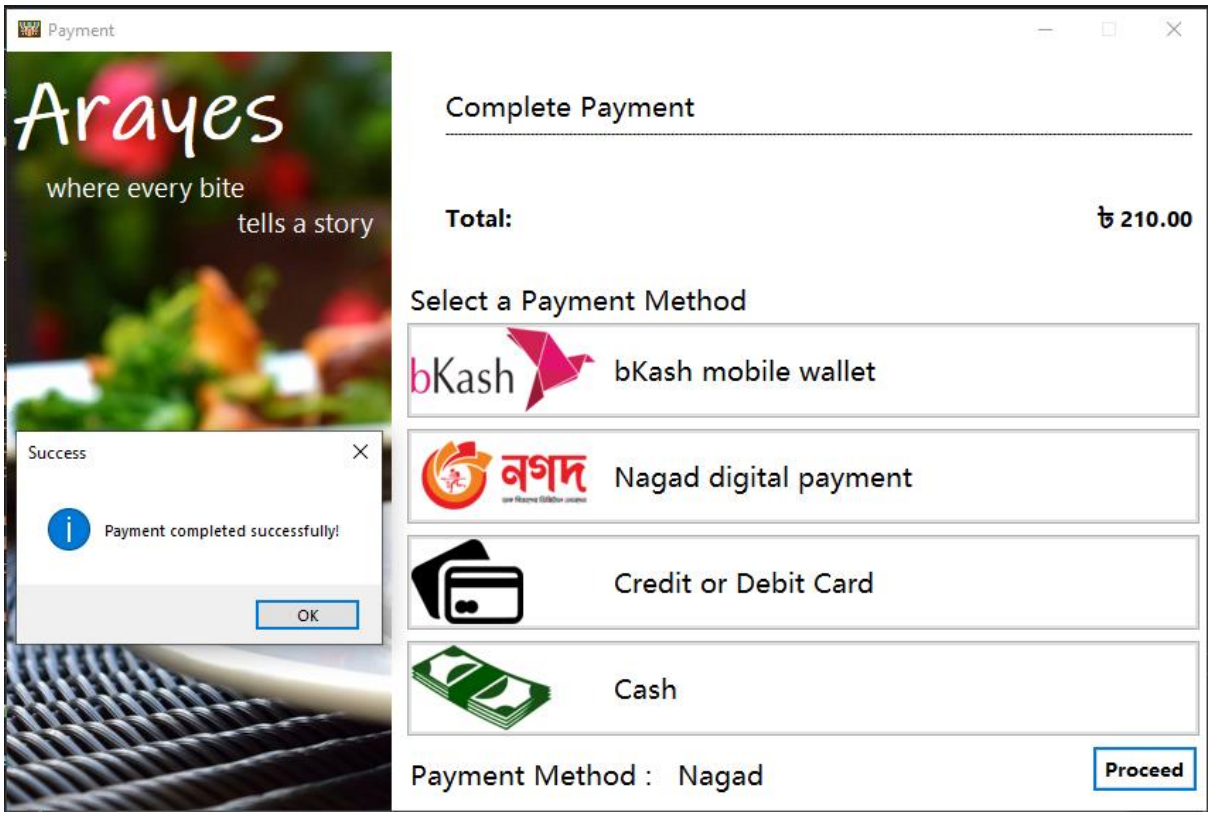


Fig 3.2 : Complete Payment

References

1. https://www.w3schools.com/sql (SQL queries)
2. https://claude.ai/ (For getting visual design suggestions for the GUI panels)
3. https://youtube.com/ (Guides on creating charts on Forms)