AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Dept. of Computer Science
Faculty of Science and Technology

CSC2210: OBJECT ORIENTED PROGRAMMING 2

Fall 2025-2026

Section: [G]

Group No: 09

Project Report On

Project name [Farm Invoice Manager]

Supervised By

Md. Hasibul Hasan

Submitted By: MAHMMOD, MD. SHAFI

Name	ID
TASNIM, JARIN	**23-50749-1**
MAHMMOD, MD. SHAFI	**23-54122-3**
RISHAD, NAJIAT ISLAM	**23-55218-3**

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	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	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	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	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

	ignored or incorrectly applied.	inadequate handling of data validation.	in partial or inconsistent handling of data validation.	in the system.	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 data flow is not considered.	Demonstrates limited understanding of verification processes and data flow in the system. Verification attempts are incomplete or inaccurate, and there is insufficient consideration given to ensuring data integrity and functionality.	Shows a basic understanding of verification processes and attempts to verify system data. However, verification efforts may be inconsistent or lack thoroughness, and there may be gaps in ensuring proper functional requirements and data flow.	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 appropriate data flow within the system.	Exhibits an exceptional understanding of verification processes and meticulously verifies system data. Verification efforts are comprehensive and precise, with a keen focus on 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: Pag no

Chapter 01: Introduction

The **Farm Invoice Manager – Billing Management System** is a desktop-based application developed using **C# Windows Forms** and **SQL Server** to efficiently manage billing, sales, products, customers, and users in a farm-based business environment. The primary goal of this system is to simplify and automate the manual invoice and sales management process, reducing errors and improving productivity.This system provides a **secure login mechanism** with role-based access for **Admin and Employee**, ensuring proper data security and controlled access. Admin users can manage products, users, customers, memberships, and generate sales reports, while employees can perform sales operations and view necessary information based on their permissions.The application supports complete **product management**, **customer management**, **invoice generation**, **sales reporting**, and **printable invoices**. Features such as **password recovery**, **username and password reset**, **premium membership handling**, **date-wise sales reports**, and **graphical sales analysis** enhance usability and decision-making.With a user-friendly interface and structured data handling, the Farm Invoice Manager helps farm businesses maintain accurate records, track inventory, manage customers, and generate reliable financial reports. Overall, this project demonstrates effective use of object-oriented programming, database connectivity, and real-world business logic implementation in a desktop application.

Chapter 02: User Story

The **Farm Invoice Manager – Billing Management System** is designed with two essential user roles in mind: **Admin** and **Employee**. Each role has specific responsibilities and access permissions to ensure smooth, secure, and efficient management of farm billing and sales operations. Below is a brief overview of how each user interacts with the system.

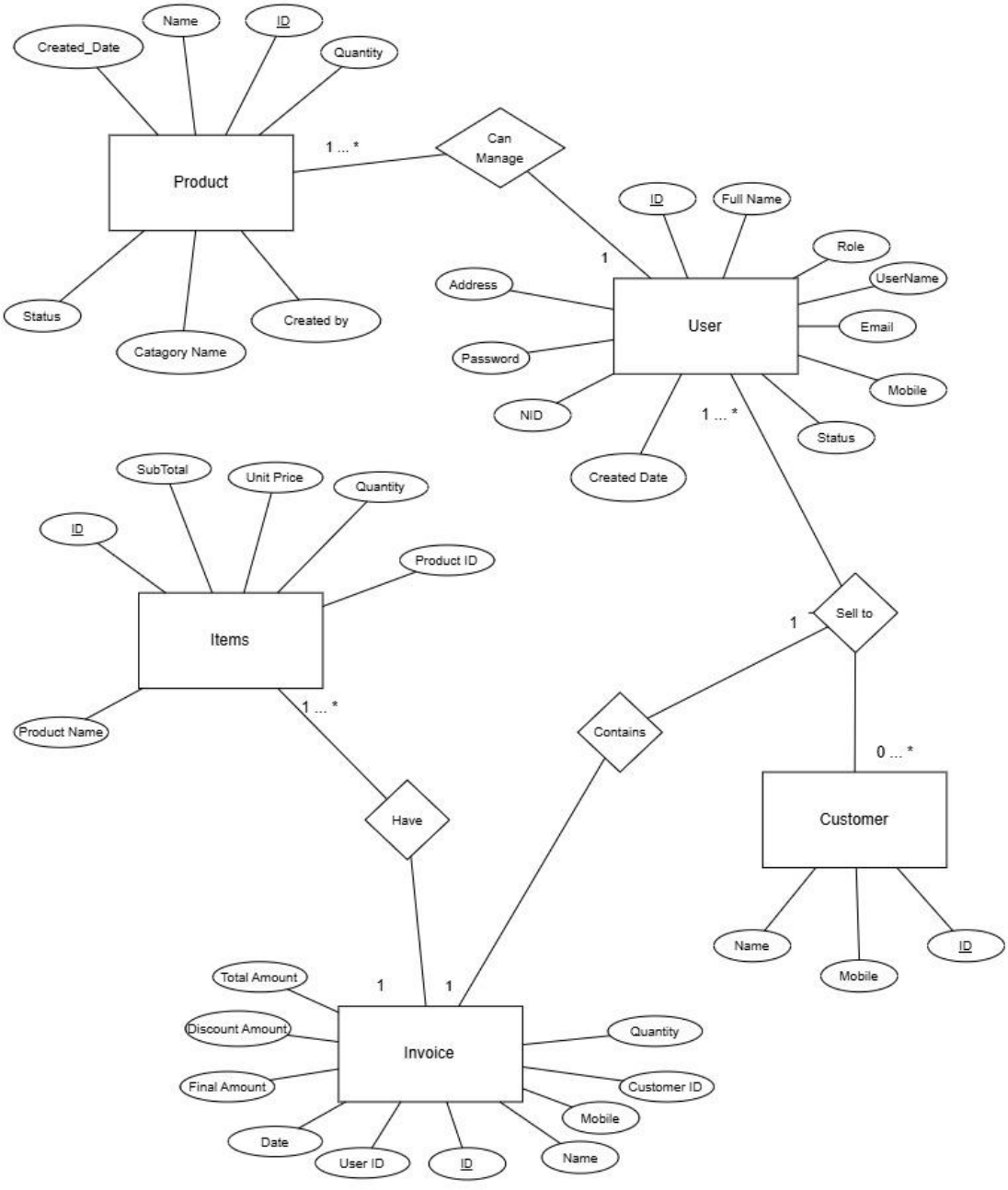
As an Admin

- Log in securely to the admin dashboard using valid credentials.
- Add, update, or remove system users and manage their personal and login information.
- Add new farm products, update product details such as name, category, stock, and unit price, and remove unavailable products.
- Manage customer information, including customer name and mobile number.
- Identify and manage premium members for special benefits or discounts.
- Monitor and manage all sales transactions and invoices.
- Generate, view, and print invoices for customers.
- View sales reports with date-wise summaries and graphical representations.
- Review and update product lists, customer records, user data, and sales information.
- Reset usernames and passwords to maintain system security.
- Log out safely to protect system data.

As an Employee

- Log in securely to the employee dashboard using authorized credentials.
- View personal information and account details.
- Add customer information during sales transactions.
- Create sales invoices by selecting products and entering quantities.
- View daily and previous sales reports as permitted.
- Print invoices for customers when required.
- Reset own username and password for security purposes.
- Log out safely after completing work.

Chapter 03a: ER Diagram



Explanation of Entities and Relationships:

UserInfo (Stores user information)

- USER_ID (Primary Key)
- ROLE (Admin / Employee)
- FULL_NAME
- USERNAME (Unique)
- EMAIL (Unique)
- MOBILE (Unique)
- PASSWORD
- NID (Unique)
- ADDRESS
- CREATED_DATE
- STATUS (Active / Inactive)

CustomerInfo (Stores customer details)

- CUSTOMER_ID (Primary Key)
- CUSTOMER_NAME
- CUSTOMER_MOBILE (Unique recommended)

ProductInfo (Stores product and inventory details)

- PRODUCT_ID (Primary Key)
- PRODUCT_NAME
- QUANTITY (Stock)
- CATEGORY_NAME
- UNITPRICE
- CREATED_DATE
- IS_AVAILABLE
- CREATED_BY_USER_ID (Foreign Key)

Invoice (Stores invoice billing details)

- INVOICE_ID (Primary Key)
- USER_ID (Foreign Key)
- CUSTOMER_ID (Foreign Key)
- CUSTOMER_NAME **(**Stored for quick display**)**
- CUSTOMER_MOBILE (Stored for quick display)
- TOTAL_AMOUNT
- DISCOUNT_AMOUNT
- FINAL_AMOUNT
- INVOICE_DATE

InvoiceItems (Stores invoice item details)

- INVOICE_ITEM_ID (Primary Key)
- INVOICE_ID (Foreign Key)
- PRODUCT_ID (Foreign Key)
- PRODUCT_NAME
- QUANTITY
- UNIT_PRICE
- SUB_TOTAL

Chapter: 03b – SQL Queries

1. Database Creation

The database is created to store all information related to users, customers, products, invoices, and sales records.

Database Name: FarmInvoiceManager

This database serves as the central storage system for the entire application.

2. Table Creation

Several tables are created to define the structure of the database. These tables represent the core entities of the Farm Invoice Manager system, including:

- **UserInfo** – stores admin and employee information
- **CustomerInfo** – stores customer details
- **ProductInfo** – stores product information
- **InvoiceDetails** – stores invoice summary data
- **InvoiceItems** – stores item-wise invoice details

3. Insert Queries (Create Operation)

Insert queries are used to add new records into the database.
Examples include:

- Adding new users (Admin / Employee)
- Adding customers
- Adding products
- Creating invoices and invoice items

4. Select Queries (Read Operation)

Select queries are used to retrieve data from the database for display and reporting purposes.
They are used to:

- Show user lists
- Display customer information
- Load product lists
- Generate invoices
- View sales reports

5. Update Queries (Update Operation)

Update queries are used to modify existing records in the database.
They are mainly used for:

- Updating user information
- Updating product details
- Resetting username or password
- Updating invoice or stock data

6. Delete Queries (Delete Operation)

Delete queries are used to remove unwanted or obsolete records from the database.
Examples include:

- Deleting users
- Deleting products
- Deleting customers
- Deleting invoices and invoice items

```
CREATE TABLE [dbo].[UserInfo] (
   [USER_ID] NVARCHAR(10) NOT NULL,
   [ROLE] NVARCHAR(10) NOT NULL,
   [FULL_NAME] NVARCHAR(50) NOT NULL,
   [USERNAME] NVARCHAR(50) NOT NULL,
   [EMAIL] NVARCHAR(50) NOT NULL,
   [MOBILE] NVARCHAR(20) NOT NULL,
   [PASSWORD] NVARCHAR(50) NOT NULL,
   [NID] NVARCHAR(20) NOT NULL,
   [ADDRESS] NVARCHAR(50) NOT NULL,
   [CREATED_DATE] DATE NOT NULL,
   [STATUS] BIT NOT NULL,
   CONSTRAINT [PK_UserInfo] PRIMARY KEY CLUSTERED ([USER_ID] ASC)
);

CREATE TABLE dbo.CustomerInfo (
   CUSTOMER_ID NVARCHAR(10) NOT NULL PRIMARY KEY,
   CUSTOMER_NAME NVARCHAR(50) NOT NULL,
   CUSTOMER_MOBILE NVARCHAR(20) NOT NULL
);

CREATE TABLE [dbo].[ProductInfo] (
   [PRODUCT_ID] NVARCHAR(10) NOT NULL PRIMARY KEY,
   [PRODUCT_NAME] NVARCHAR(50) NOT NULL,
   [QUANTITY] INT NOT NULL,
   [CATEGORY_NAME] NVARCHAR(50) NOT NULL,
   [UNITPRICE] FLOAT NOT NULL,
   [CREATED_DATE] DATE NOT NULL,
   [IS_AVAILABLE] BIT NOT NULL,
   [CREATED_BY_USER_ID] NVARCHAR(10) NOT NULL,
   FOREIGN KEY ([CREATED_BY_USER_ID]) REFERENCES [dbo].[UserInfo]([USER_ID])
);

CREATE TABLE dbo.InvoiceItems (
   INVOICE_ITEM_ID INT IDENTITY(1,1) PRIMARY KEY,
   INVOICE_ID NVARCHAR(10) NOT NULL,
   PRODUCT_ID NVARCHAR(10) NOT NULL,
   PRODUCT_NAME NVARCHAR(100) NOT NULL,
   QUANTITY INT NOT NULL,
   UNIT_PRICE DECIMAL(10,2) NOT NULL,
   SUB_TOTAL DECIMAL(10,2) NOT NULL,
   CONSTRAINT FK_InvoiceItems_ProductInfo
      FOREIGN KEY (PRODUCT_ID) REFERENCES dbo.ProductInfo(PRODUCT_ID)
```

```
);

CREATE TABLE dbo.InvoiceDetails (
    INVOICE_ID NVARCHAR(10) NOT NULL PRIMARY KEY,
    USER_ID NVARCHAR(10) NOT NULL,
    CUSTOMER_ID NVARCHAR(10) NOT NULL,
    CUSTOMER_NAME NVARCHAR(50) NOT NULL,
    CUSTOMER_MOBILE NVARCHAR(20) NOT NULL,
    TOTAL_AMOUNT DECIMAL(10,2) NOT NULL,
    DISCOUNT_AMOUNT DECIMAL(10,2) NOT NULL,
    FINAL_AMOUNT DECIMAL(10,2) NOT NULL,
    INVOICE_DATE DATE NOT NULL,
    FOREIGN KEY (USER_ID) REFERENCES dbo.UserInfo(USER_ID)
);
```

FormLogin

```
string query = $"SELECT USER_ID, FULL_NAME, ROLE, STATUS
FROM UserInfo
WHERE (USERNAME = '{username}' OR USER_ID = '{username}')
AND PASSWORD = '{password}'";
```

FormAddCustomer

```
SELECT MAX(CUSTOMER_ID) FROM CustomerInfo;

INSERT INTO CustomerInfo (CUSTOMER_ID, CUSTOMER_NAME,
CUSTOMER_MOBILE)
VALUES ('{newCustomerId}', '{this.txtCustomerName.Text}', '{this.txtMobile.Text}');
```

FormAddProduct

```
SELECT ISNULL(MAX(CAST(SUBSTRING(PRODUCT_ID,5,10) AS INT)),0) FROM
ProductInfo

INSERT INTO ProductInfo (PRODUCT_ID, PRODUCT_NAME, QUANTITY,
CATEGORY_NAME, UNITPRICE, CREATED_DATE, IS_AVAILABLE,
CREATED_BY_USER_ID)
VALUES (@ID, @NAME, @QTY, @CAT, @PRICE, @DATE, @STATUS,
@CREATED_BY_USER_ID)
```

FormAddSales

```
SELECT UNITPRICE, QUANTITY FROM ProductInfo WHERE PRODUCT_ID =
'{cmbProduct.SelectedValue}';

SELECT PRODUCT_ID, PRODUCT_NAME, UNITPRICE, QUANTITY, CREATED_DATE
FROM ProductInfo WHERE PRODUCT_ID = '{productId}';

SELECT PRODUCT_ID, PRODUCT_NAME, IS_AVAILABLE FROM ProductInfo WHERE
IS_AVAILABLE = 1;

SELECT CUSTOMER_ID, CUSTOMER_MOBILE FROM CustomerInfo;
```

SELECT CUSTOMER_NAME FROM CustomerInfo WHERE CUSTOMER_MOBILE = '{cmbNumber.Text.Trim()}';

INSERT INTO InvoiceDetails (INVOICE_ID, USER_ID, CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_MOBILE, TOTAL_AMOUNT, DISCOUNT_AMOUNT, FINAL_AMOUNT, INVOICE_DATE) VALUES (...)

INSERT INTO InvoiceItems (INVOICE_ID, PRODUCT_ID, PRODUCT_NAME, QUANTITY, UNIT_PRICE, SUB_TOTAL) VALUES (...)

UPDATE ProductInfo SET QUANTITY = QUANTITY - {soldQuantity} WHERE PRODUCT_ID = '{productID}';

FormAddUser

SELECT ISNULL(MAX(CAST(SUBSTRING(USER_ID,3,10) AS INT)),0) FROM UserInfo WHERE USER_ID LIKE '[AE]-%'

INSERT INTO dbo.UserInfo (USER_ID, ROLE, FULL_NAME, USERNAME, EMAIL, MOBILE, PASSWORD, NID, ADDRESS, CREATED_DATE, STATUS) VALUES (@USER_ID, @ROLE, @FULL_NAME, @USERNAME, @EMAIL, @MOBILE, @PASSWORD, @NID, @ADDRESS, CAST(GETDATE() AS DATE), @STATUS)

FormAdminView

SELECT * FROM UserInfo;

SELECT * FROM CustomerInfo;

DELETE FROM UserInfo WHERE USER_ID = 'userId';
SELECT * FROM UserInfo WHERE FULL_NAME LIKE '%searchText%' OR USERNAME LIKE '%searchText%' OR USER_ID LIKE '%searchText%' ORDER BY USER_ID;

SELECT * FROM UserInfo WHERE USER_ID = 'adminUserID';

DELETE FROM CustomerInfo WHERE CUSTOMER_ID = 'customerId';

SELECT * FROM CustomerInfo WHERE CUSTOMER_MOBILE LIKE '%searchText%' ORDER BY CUSTOMER_ID;

FormEditUser

SELECT *
FROM UserInfo
WHERE USER_ID = '{UserId}';

UPDATE UserInfo SET
FULL_NAME = '{fullName}',
EMAIL = '{email}',
USERNAME = '{username}',
PASSWORD = '{password}',
MOBILE = '{mobile}',
NID = '{nid}',
ADDRESS = '{address}',

```
ROLE = '{role}',
STATUS = {status}
WHERE USER_ID = '{UserId}';
```

FormEmployee

```
SELECT *
FROM UserInfo
WHERE USER_ID = '{UserID}';

SELECT *
FROM CustomerInfo;

SELECT *
FROM CustomerInfo
WHERE CUSTOMER_MOBILE LIKE '%{searchText}%'
ORDER BY CUSTOMER_ID;
```

FormFindPassword

```
SELECT PASSWORD
FROM UserInfo
WHERE USER_ID = '{userId}'
AND USERNAME = '{username}'
AND MOBILE = '{mobile}';
```

FormInvoicePrint

```
SELECT CUSTOMER_NAME, CUSTOMER_MOBILE,
TOTAL_AMOUNT, DISCOUNT_AMOUNT, FINAL_AMOUNT, INVOICE_DATE
FROM InvoiceDetails
WHERE INVOICE_ID = '{InvoiceID}';

SELECT P.PRODUCT_NAME, I.QUANTITY, I.UNIT_PRICE, I.SUB_TOTAL
FROM InvoiceItems I
INNER JOIN ProductInfo P ON I.PRODUCT_ID = P.PRODUCT_ID
WHERE I.INVOICE_ID = '{InvoiceID}';
```

FormManageProducts

```
SELECT *
FROM ProductInfo;

SELECT *
FROM ProductInfo
WHERE PRODUCT_NAME LIKE '%{searchText}%'
OR PRODUCT_ID LIKE '%{searchText}%'
ORDER BY PRODUCT_ID;

DELETE
FROM ProductInfo
WHERE PRODUCT_ID = '{productId}';
```

FormReset

```
SELECT USERNAME, PASSWORD
FROM UserInfo
WHERE USER_ID = '{UserId}';

UPDATE UserInfo
SET USERNAME = '{newUserName}', PASSWORD = '{newPassword}'
WHERE USERNAME = '{currentUsername}' AND PASSWORD = '{currentPassword}';
```

FormUpdateProduct

```
SELECT *
FROM ProductInfo
WHERE PRODUCT_ID = @ID;

UPDATE ProductInfo
SET PRODUCT_NAME = @NAME,
QUANTITY = @QTY,
CATEGORY_NAME = @CAT,
UNITPRICE = @PRICE,
CREATED_DATE = @DATE,
IS_AVAILABLE = @STATUS
WHERE PRODUCT_ID = @ID;
```

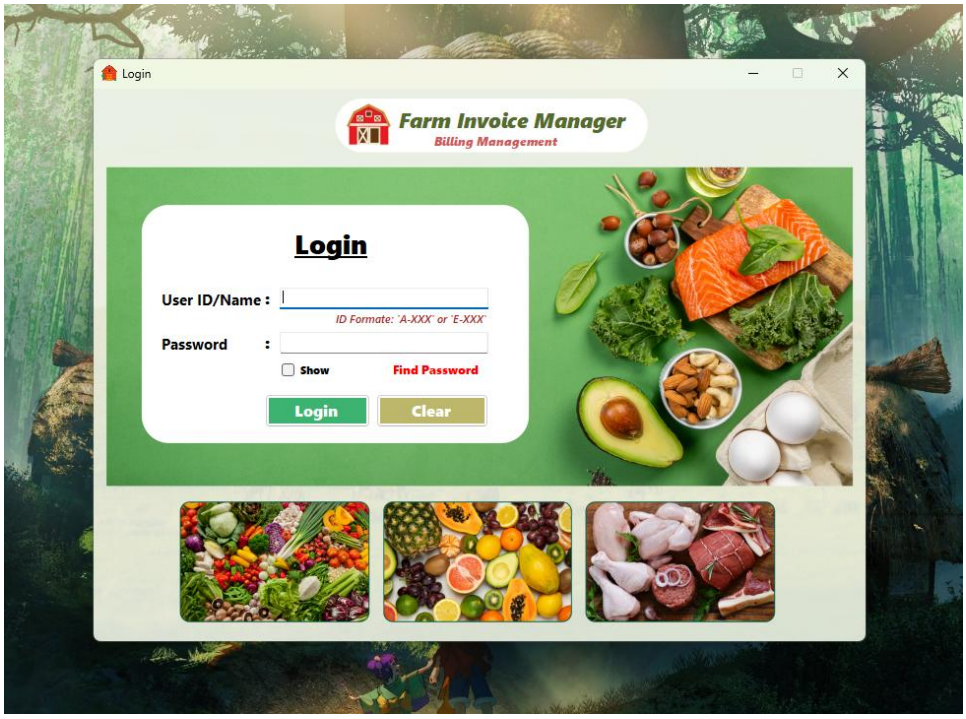
FromSalesReport

```
SELECT * FROM InvoiceDetails;

SELECT INVOICE_DATE, SUM(CAST(FINAL_AMOUNT AS float)) AS TotalSales
FROM InvoiceDetails
GROUP BY INVOICE_DATE
ORDER BY INVOICE_DATE;

SELECT INVOICE_DATE, SUM(CAST(FINAL_AMOUNT AS float)) AS TotalSales
FROM InvoiceDetails
WHERE INVOICE_DATE >= '{fromDate}'
AND INVOICE_DATE <= '{toDate}'
GROUP BY INVOICE_DATE
ORDER BY INVOICE_DATE;

SELECT *
FROM InvoiceDetails
WHERE INVOICE_DATE >= '{fromDate}'
AND INVOICE_DATE <= '{toDate}'
ORDER BY INVOICE_DATE;

SELECT PRODUCT_ID, PRODUCT_NAME, QUANTITY, UNIT_PRICE, SUB_TOTAL
FROM InvoiceItems
WHERE INVOICE_ID = '{invoiceId}';

SELECT *
FROM InvoiceDetails
WHERE CUSTOMER_MOBILE LIKE '%{mobile}%'
ORDER BY INVOICE_DATE;

DELETE FROM InvoiceItems
```

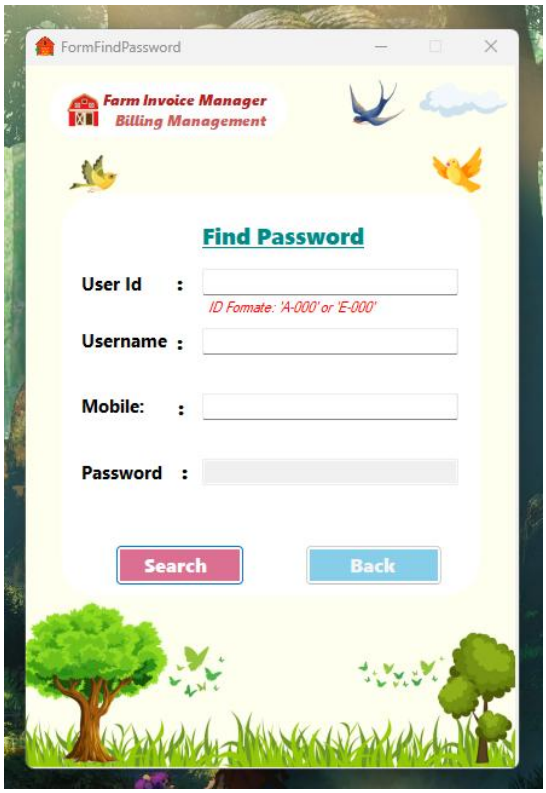
```
WHERE INVOICE_ID = '{invoiceId}';

DELETE FROM InvoiceDetails
WHERE INVOICE_ID = '{invoiceId}';
```

Chapter 04: Screenshots

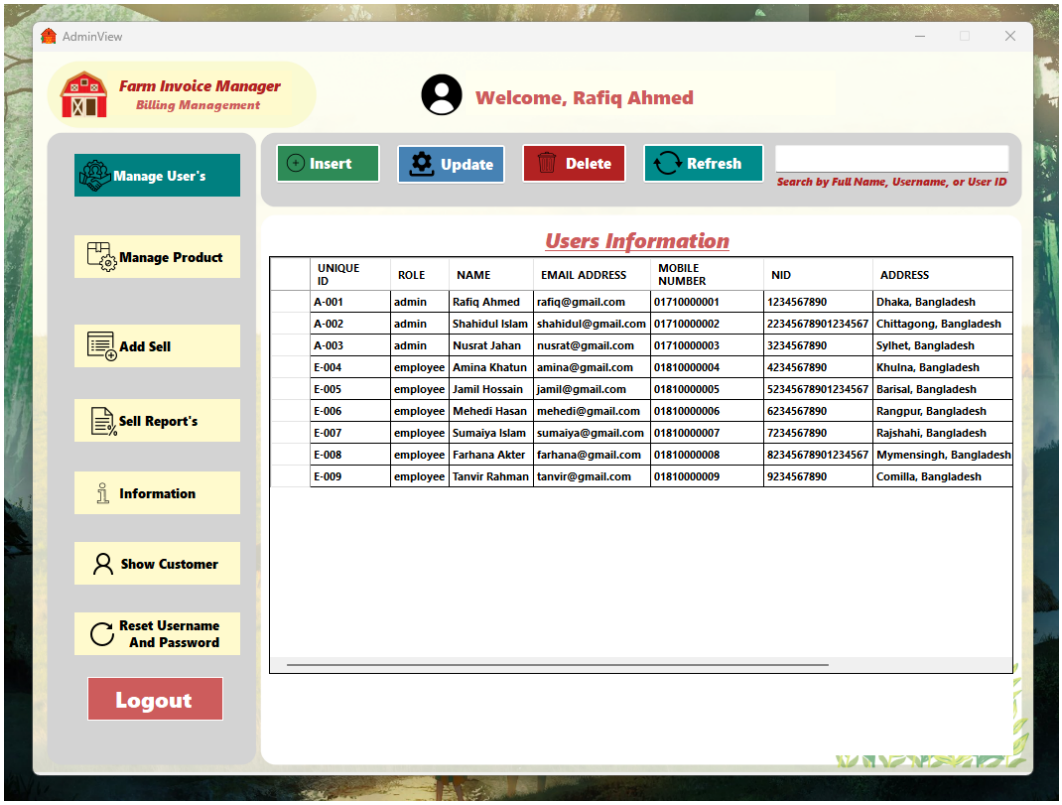
Form Name: Login Form
Purpose: User authentication (Admin / Employee)



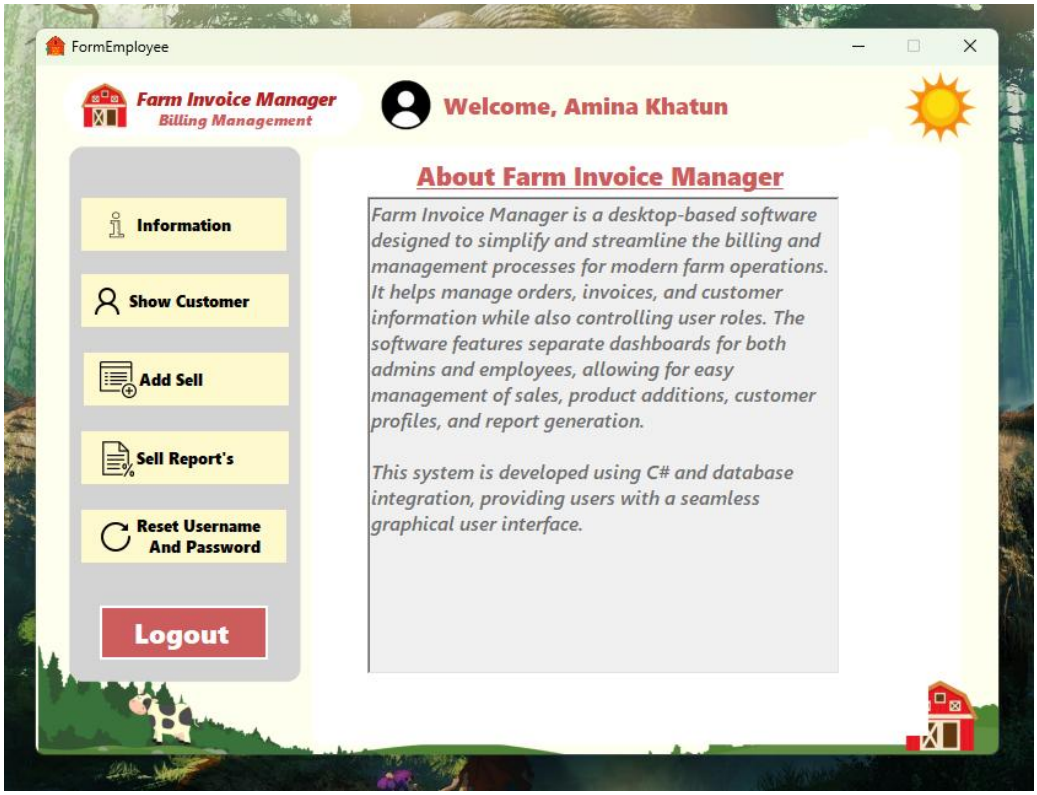
Form Name: Find Password
Purpose: Recover login credentials



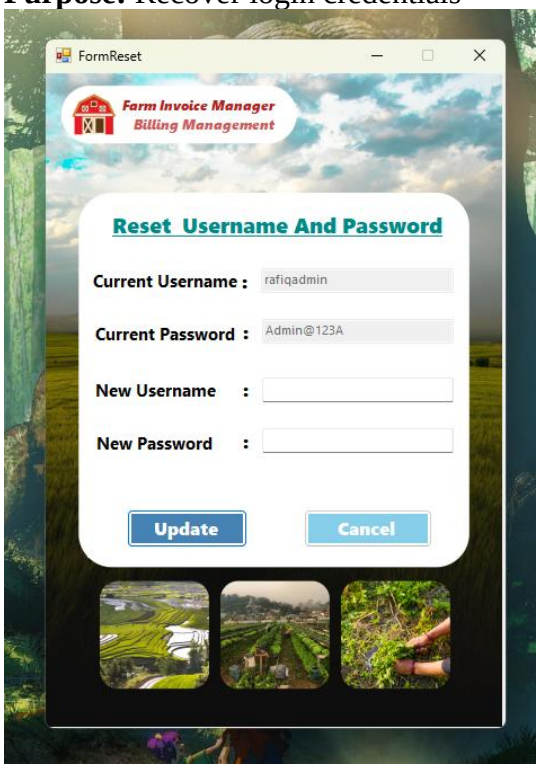
Form Name: Admin Dashboard
Purpose: Admin main control pane



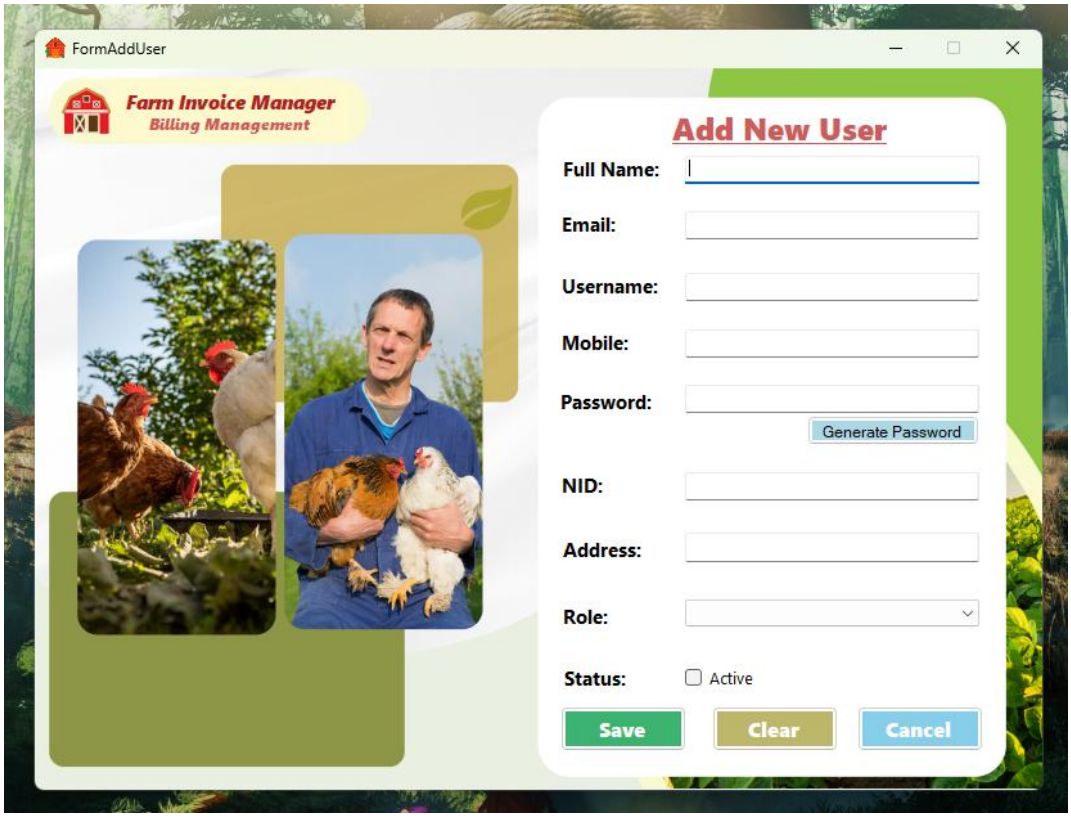
Form Name: Employee Dashboard
Purpose: Employee main control panel



Form Name: Reset Password Form
Purpose: Recover login credentials

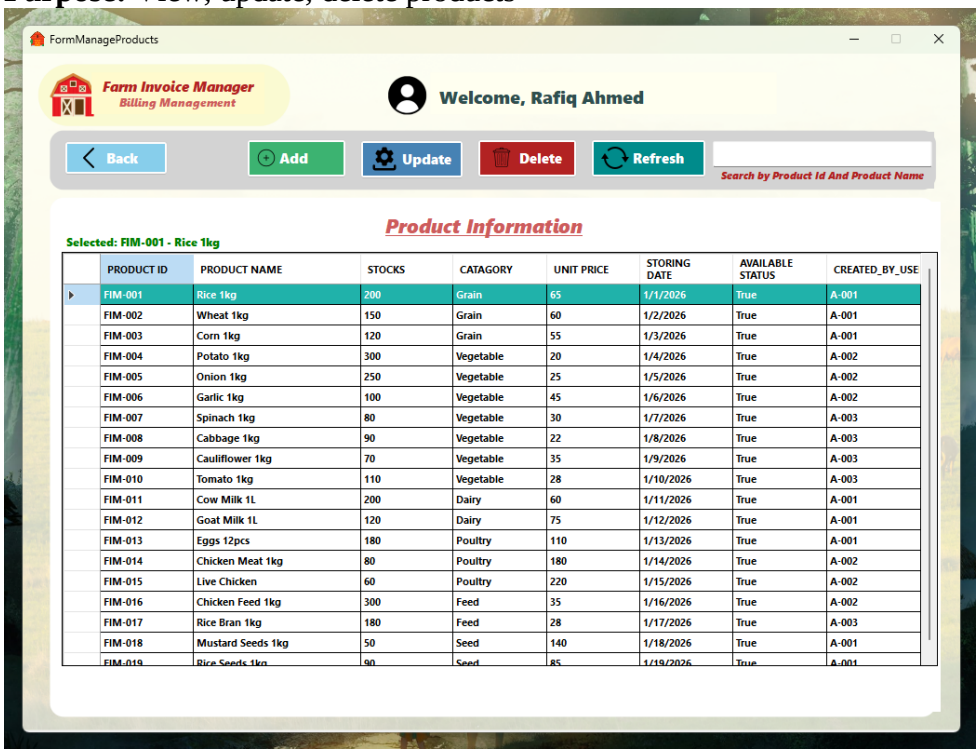


Form Name: Add New User Form
Purpose: Create admin or employee account

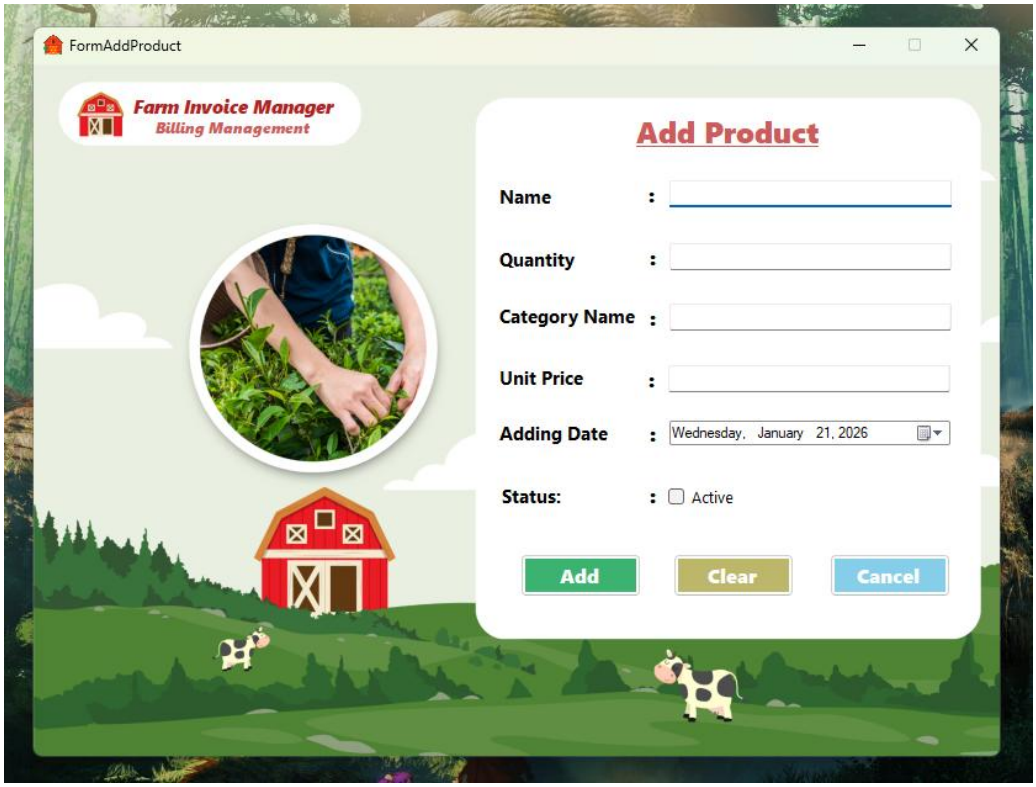


Form Name: Edit Existing User Form
Purpose: Update user information

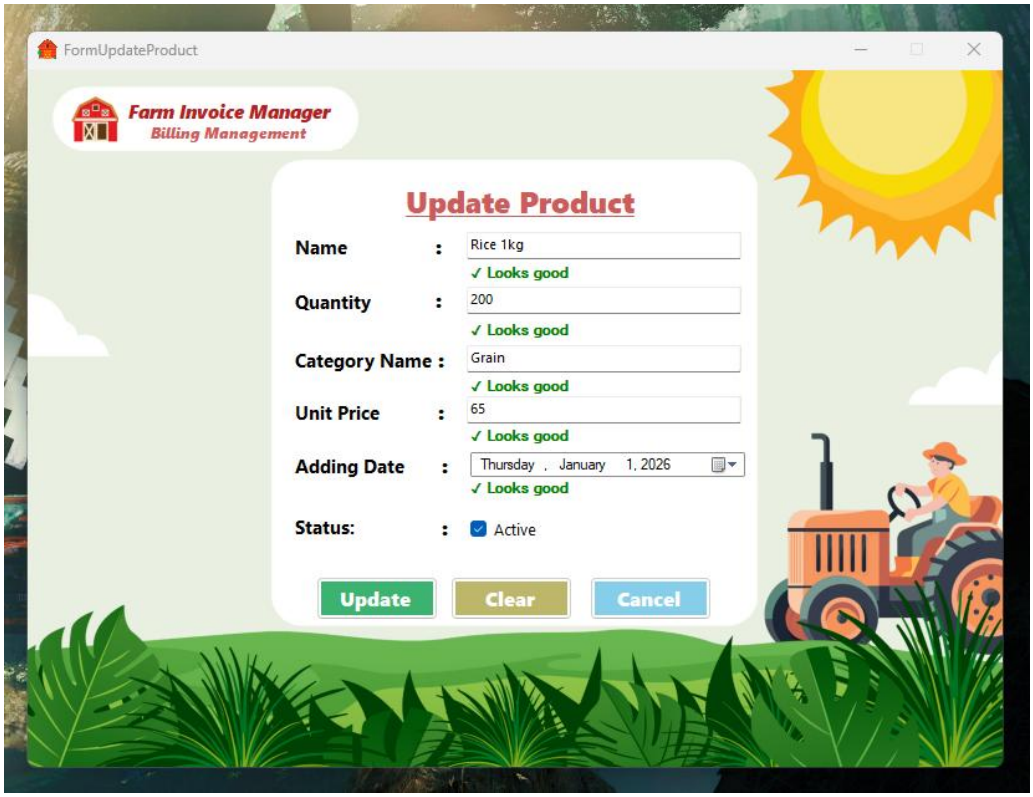
Form Name: Manage Products Form
Purpose: View, update, delete products



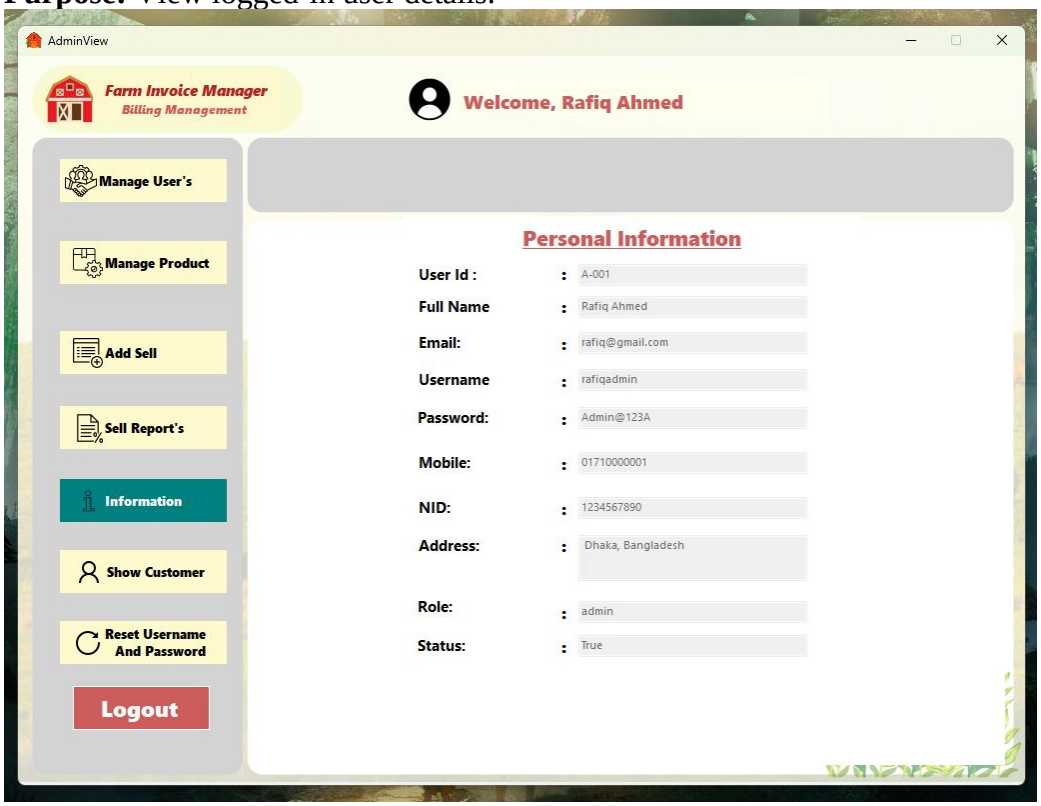
Form Name: Add Product Form
Purpose: Add new product information.



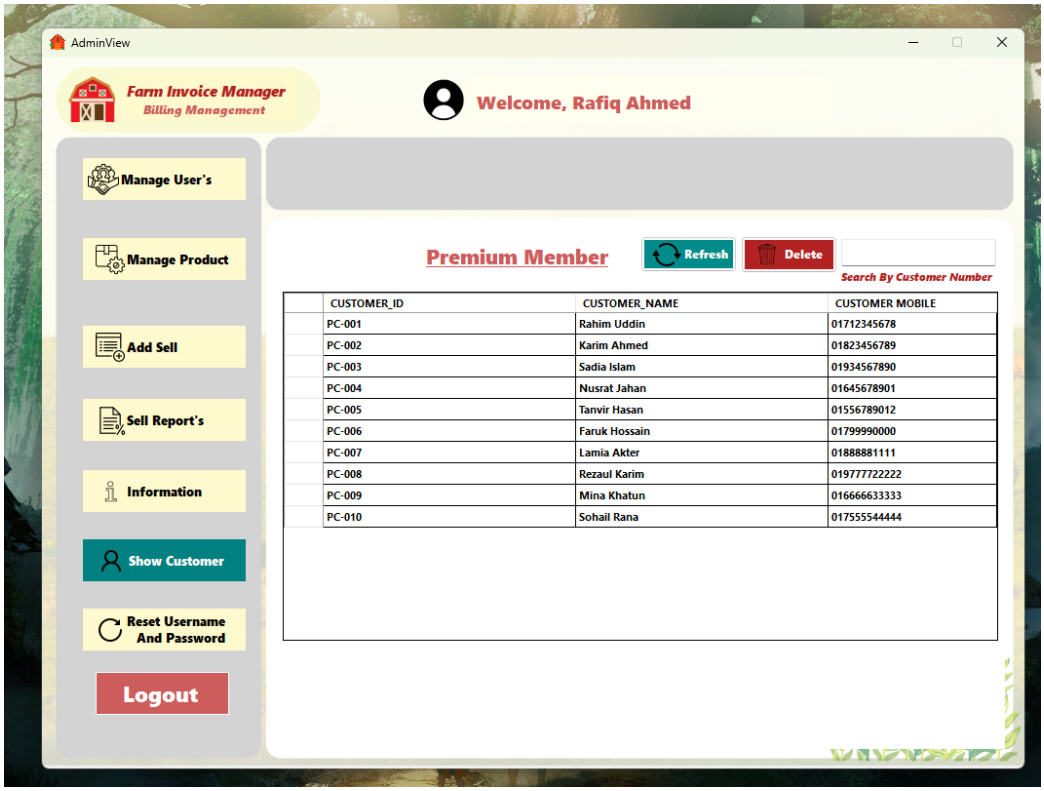
Form Name: Update Product Form
Purpose: Update existing product details.



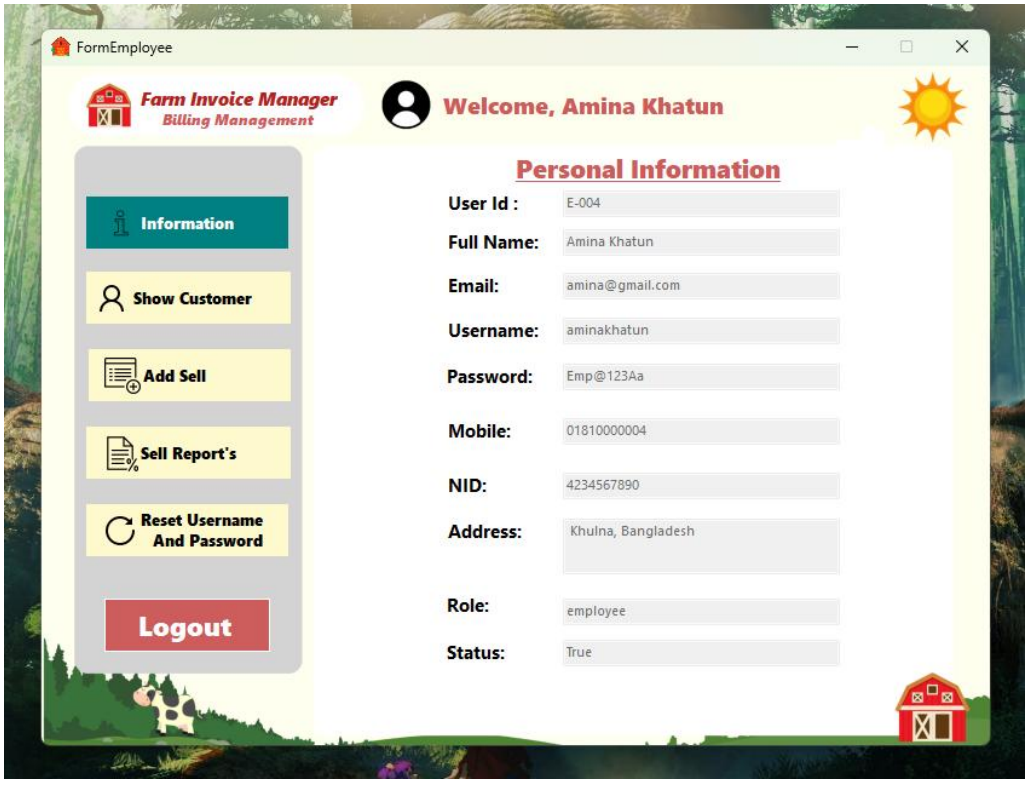
Form Name: Personal Information Form
Purpose: View logged-in user details.



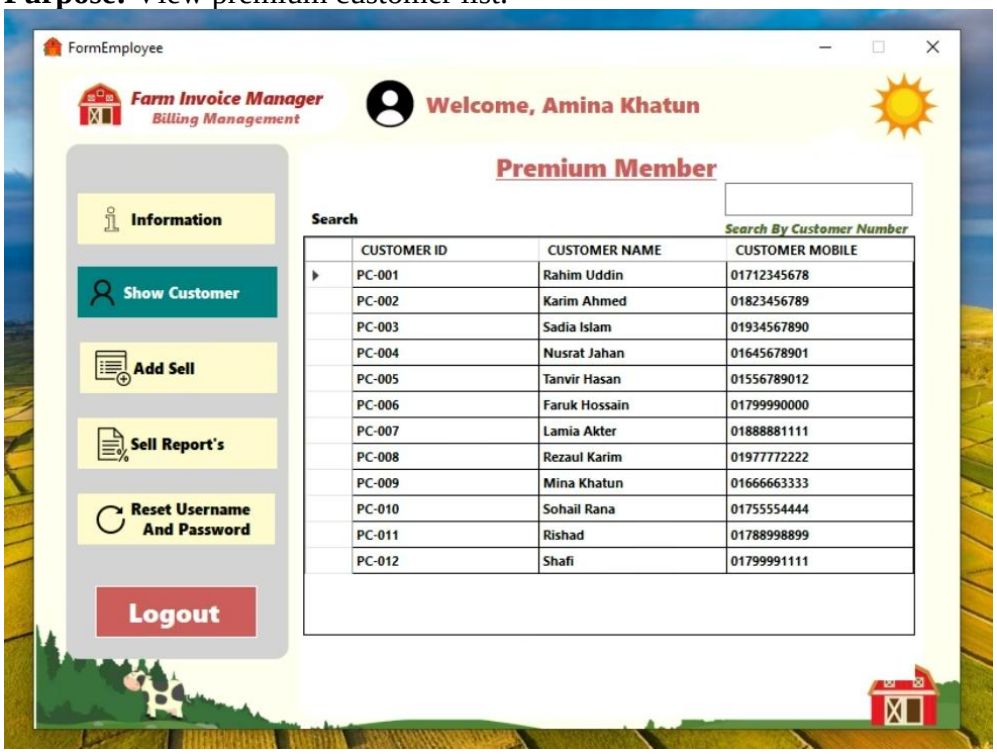
Form Name: Show Customer Form
Purpose: View premium customer list.



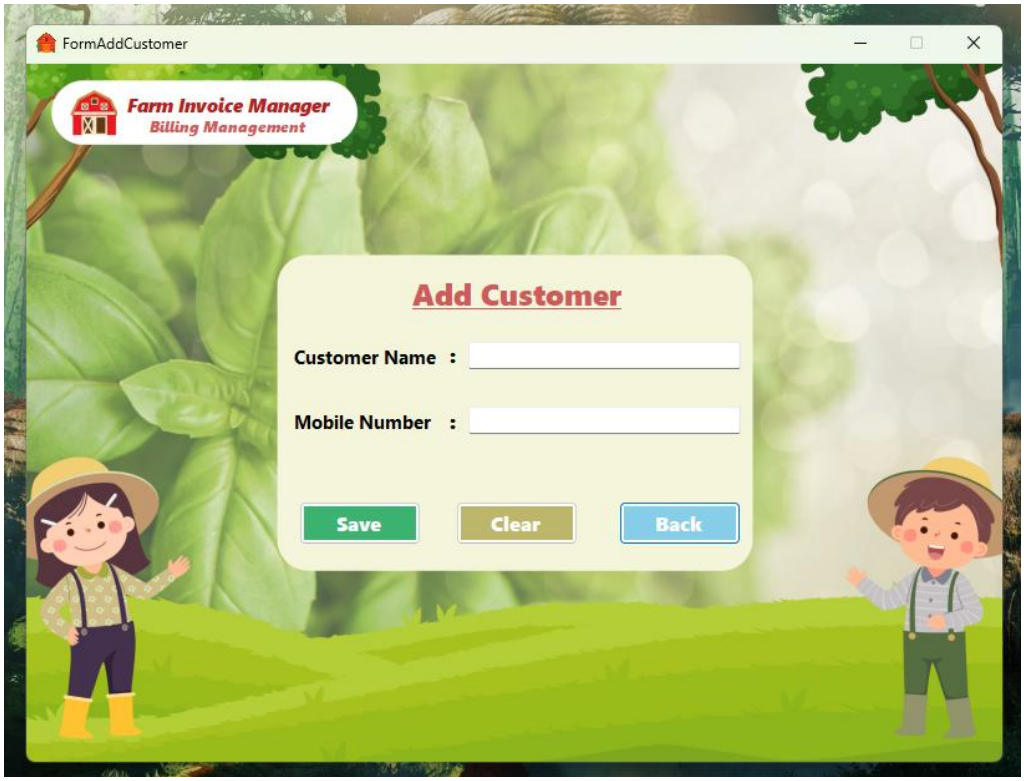
Form Name: Personal Information Form (Employee)
Purpose: View logged-in employee details.



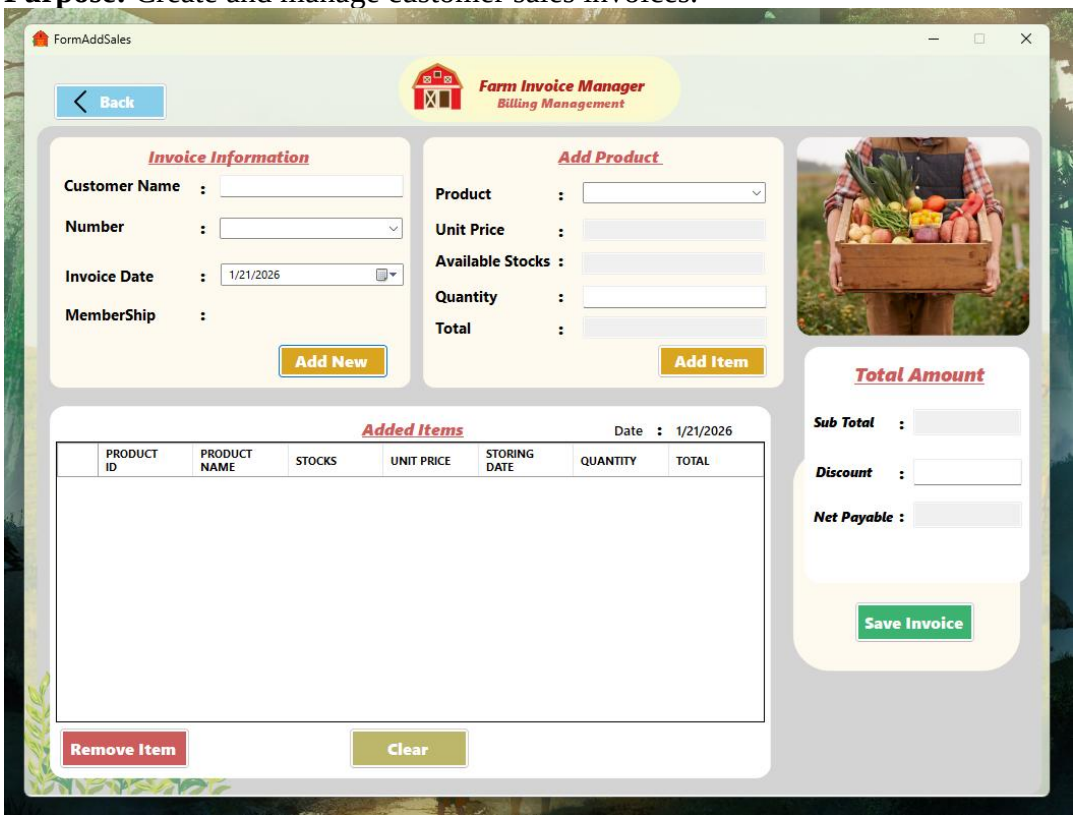
Form Name: Show Customer Form (Employee)
Purpose: View premium customer list.



Form Name: Add Customer Form
Purpose: Add new customer information



Form Name: Add Sales Form
Purpose: Create and manage customer sales invoices.



Form Name: Sales Report Form
Purpose: View and analyze date-wise sales information.