AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH

(AIUB)

Dept. of Computer Science

Faculty of Science and Technology

CSC2210: OBJECT ORIENTED PROGRAMMING 2

Spring 2024-2025

Section: R

Group No: 03

Project Report On

Project Name: ***Scholars' Nest***

Supervised By

MD. KHAIRUL ALAM MAZUMDER

Submitted By:

S.N.	Name	ID
01	NAYEM SIDDIKI	23-52361-2
02	FARHAN CHOWDHURY	23-52314-2
03	MD. SAJID ARAFAT	23-53526-3
04	HASAN SHAD SHAHRIAR	23-52529-2

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**				**Total =**
Requirement fulfillment	Properly demonstrate a real-life scenario-based project with proper functional requirement identification for the Object-Oriented Programming project development activities.				
Validation	Ensuring the ability of students' proper demonstration on validation forms in their system in terms of dealing with the data.				
Verification	Identifying if the students can verify the system data along with proper functional requirements in terms of data flow.				

Scholars' Nest

("Where Bright Minds Find Rest")

TABLE OF CONTENT

Chapter 01: Introduction

Purpose:

In Bangladesh, many student hostels are still operating using manual systems. For example, in Dhaka University student halls, students typically go to the dining room and manually collect tokens to buy food. This often leads to problems like food shortages because the hall management prepares meals based on rough approximations rather than actual demand. Similarly, in bachelor messes across Bangladesh, meal updates are done using traditional "**Talli Khata**" (manual logs), which creates confusion and complicates management.

The purpose of this project is to develop a **Hostel Management System** that will automate meal updates and student hostel management in an easy, efficient, and organized way. This system will solve the current problems of food wastage, poor planning, and manual complexities.

Objective:

The main objective of this system is to:

- Provide an easy way for students to book their meals in advance.
- Help hall/mess managers accurately estimate the food needed each day.
- Simplify student admission, payment, and meal management.
- Reduce manual errors and improve the efficiency of hostel operations.

Scope:

The system will include the following features and user roles:

- **Students:**
  - Register and log in.
  - Submit admission forms and pay admission fees.
  - Pay monthly seat rent and utility bills.
  - Deposit money into a food wallet.
  - Book meals (breakfast, lunch, dinner) for the next day.
  - View current balances, meal charges, and payment validity.
  - Access food price lists and food breakdown in daily basis.
  - Can complain
- **Admin:**
  - Only log in.
  - Manage employees (Managers) by adding, updating, deleting, searching or resetting their accounts.
  - View the list of registered students.
  - View complains and delete it.
- **Employee (Manager):**
  - Only log in.
  - Admit, remove, search, update, or reset student accounts.
  - Update the meal list and prices.

Tools Used:

- **Programming Language:** C#
- **For Database:** Microsoft SQL Server 2021 and SQL Server Management Studio
- **IDE:** Visual Studio 2022
- **Development Approach:** The system will follow Object-Oriented Programming (OOP) principles: **Encapsulation, Inheritance, Abstraction and Polymorphism etc.**

Chapter 02: User Story

Purpose:

This system is designed to solve real-life problems that students, admins, and managers face in student hostels and messes in Bangladesh. It focuses on making meal booking, payments, and hostel management simple, fast, and efficient.

User Roles:

- **Student**
- **Admin**
- **Manager (Employee)**

User Needs/Goals:

1. Student:
   - **Need:** A simple way to book meals without standing in line.
   - **Scenario:** A student wants to have lunch tomorrow. He books the meal today using the system to ensure food will be prepared for him.
   - **Need:** Track payment and meal balances easily.
   - **Scenario:** A student checks his food wallet to see if he has enough balance to book the next day's dinner.
   - **Need:** Make payments for food wallet deposits in a hassle-free way.
   - **Scenario:** At the beginning of the month, the student deposits money into his food wallet through the system.
   - **Need:** Can complin about services.
   - **Scenario:** It student faces any problem or it he find anything wrong he can complin it to the authority.

2. Admin:
   - **Need:** Manage hostel employees easily.
   - **Scenario:** The admin wants to add a new manager to handle student and meal operations. He can quickly create or update manager accounts through the system.
   - **Need:** View all student records.
   - **Scenario:** The admin checks the full list of students to verify their admission, payment, and activity status.
   - **Need:** View complains.
   - **Scenario:** When any of the student complain about anything the admin can checks the complaint list and can solve it or ignore it.

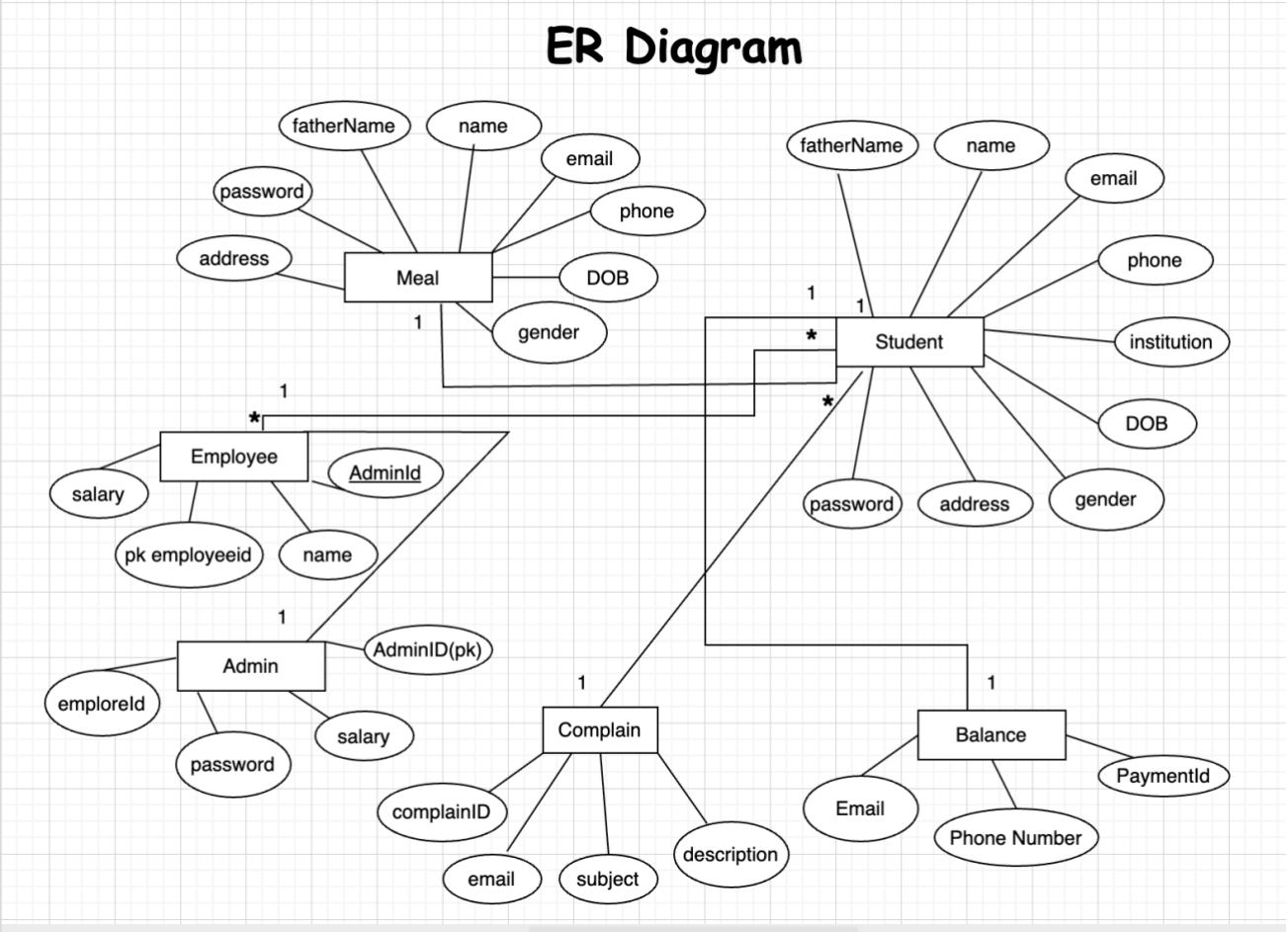
3. Manager (Employee):
   - **Need:** Admit new students or update their records smoothly.
   - **Scenario:** A new student joins the hostel. The manager can quickly create his account and set up his details in the system.
   - **Need:** Update daily meal lists and prices based on current situations.
   - **Scenario:** The price of lunch increases. The manager can immediately update the price in the system, so all students see the new rate.

Chapter 03: ER Diagram and other diagrams

Purpose: This chapter presents the database structure, showing the relationships between different entities (tables) using the Entity Relationship (ER) Diagram. It also includes system flow diagrams such as the Use Case Diagram, Activity Diagram, and Sequence Diagram to illustrate the user interactions and system processes clearly.

Entity Relationship (ER) Diagram

1. **Student**
   - Attributes: name, fatherName, email (PK), phone, institution, DOB, gender, address, password
2. **Admin**
   - Attributes: adminId (PK), name, salary, password
3. **Employee**
   - Attributes: employeeId (PK), name, salary, password, role, adminId (FK)
4. **Meal**
   - Attributes: mealId (PK), day, breakfast, breakfast_price, lunch, lunch_price, dinner, dinner_price
5. **Balance**
   - Attributes: paymentID (PK), email (FK), phone, balance
6. **Complain**
   - Attributes: complainId (PK), email (FK), subject, description



Use Case Diagram:

Purpose:

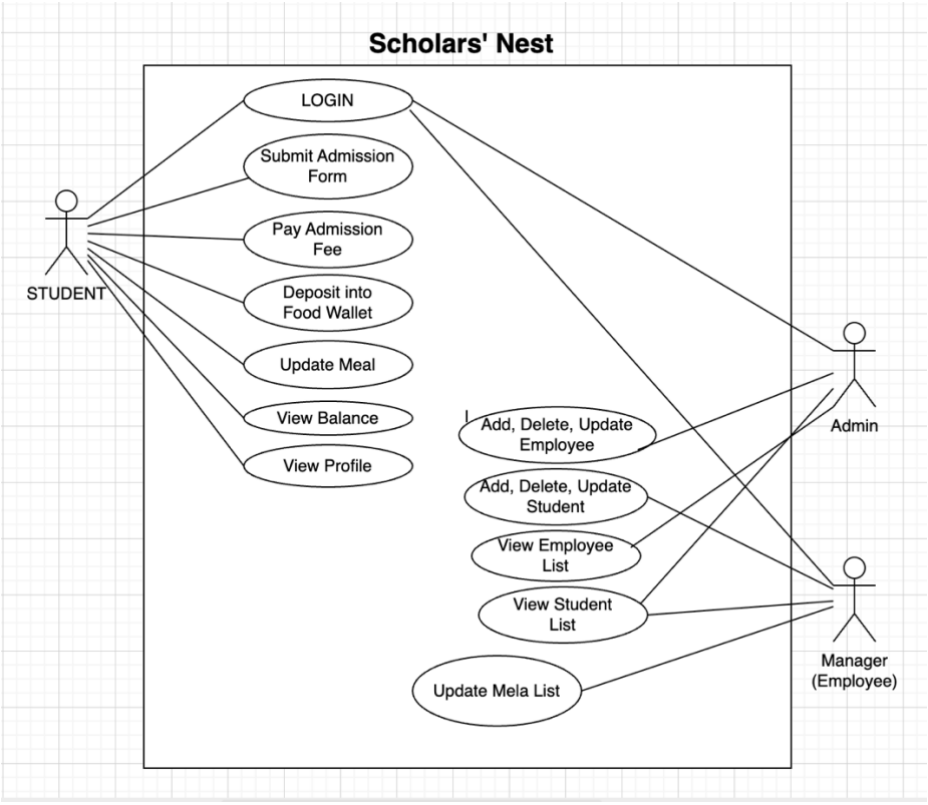
Shows the interactions between users (Actors) and the system functionalities.

Key Actors:

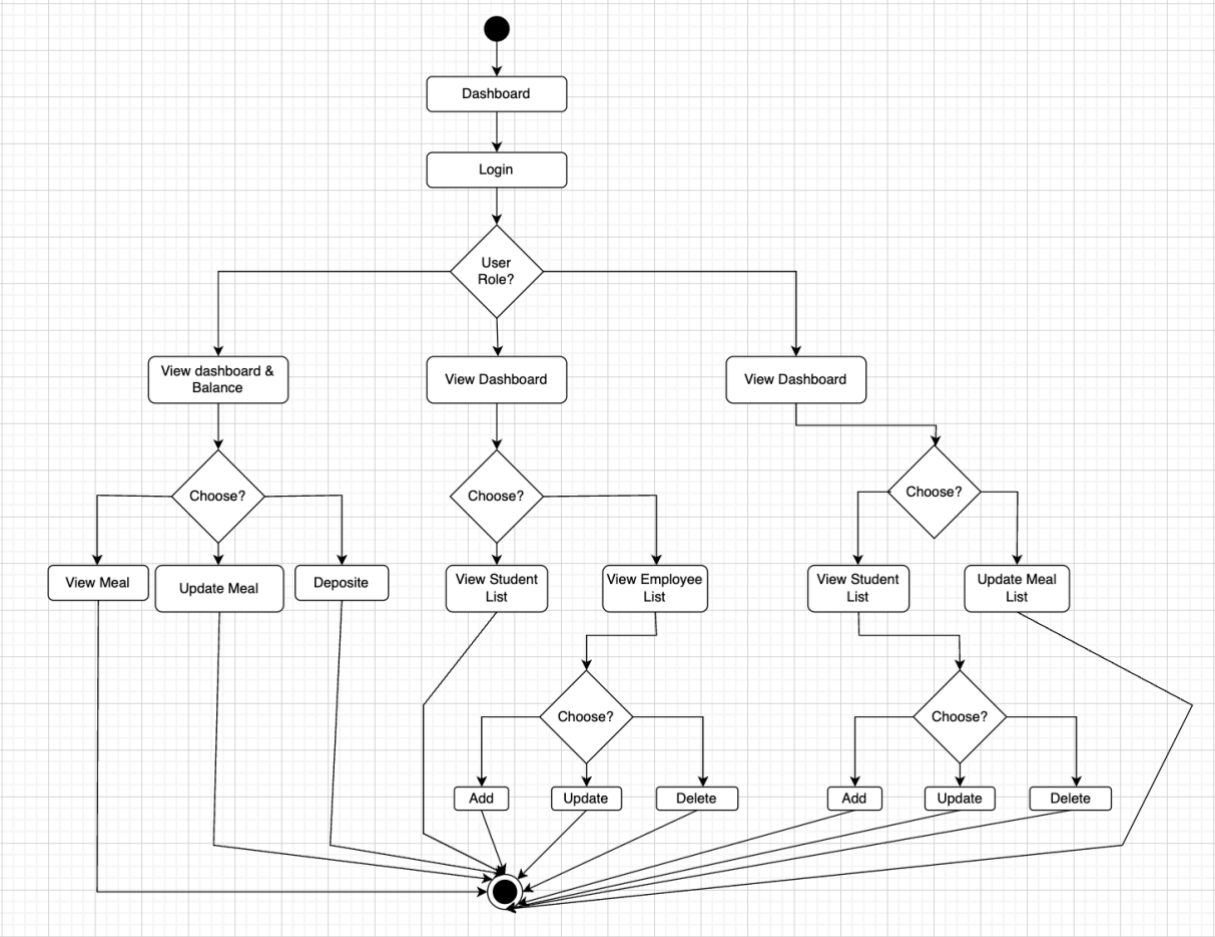
- **Student**
- **Admin**
- **Manager (Employee)**

Main Use Cases:

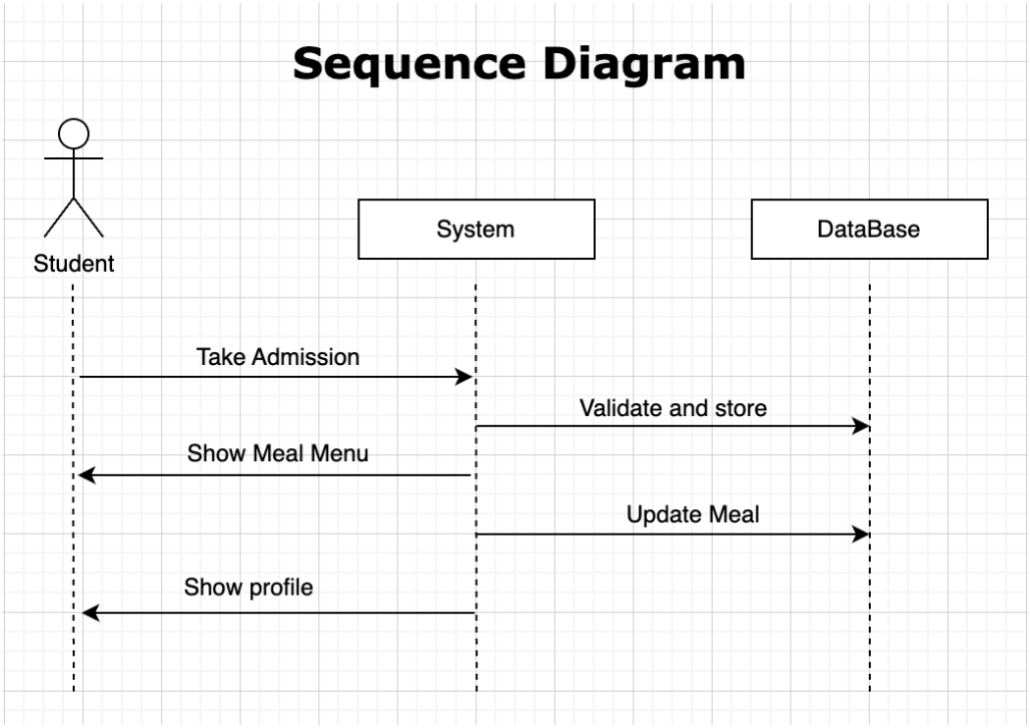
- Student: Login, Admission Form Fill Up, Update Balance, Book Meal, Check Balance.
- Admin: Manage Employees, View Student Info.
- Manager: View and Manage Student info, Update Meal List and Price.



Activity Diagram:



Sequence Diagram:



Chapter 04: SQL Queries

Purpose:

This chapter demonstrates how the **ScholarsNest** system interacts with the database using **SQL queries**. It includes:

- Table Creation Queries
- Insert Queries
- Select Queries
- Update/Delete Queries
- Search Queries

Database:

```
USE ScholarsNest;
```

Table Creation Queries

1. Student Table

```
CREATE TABLE Student (
    name VARCHAR(100),
    fatherName VARCHAR(100),
    email VARCHAR(100) PRIMARY KEY,
    phone VARCHAR(20),
    institution VARCHAR(100),
```

```
    DOB VARCHAR(20),
    gender VARCHAR(10),
    address VARCHAR(255),
    password VARCHAR(100)
);
```

2. Admin Table

```
CREATE TABLE Admin (
    adminId VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100),
    salary DECIMAL(10, 2),
    password VARCHAR(100)
);
```

3. Employee Table

```
CREATE TABLE Employee (
    employeeId VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100),
    salary VARCHAR(10),
    password VARCHAR(100),
    role VARCHAR(50)
);
```

4. Meal Table

```
CREATE TABLE Meal (
    mealId VARCHAR(10) PRIMARY KEY,
    day VARCHAR(20),
    breakfast VARCHAR(100),
    breakfast_price VARCHAR(10),
    lunch VARCHAR(100),
    lunch_price VARCHAR(10),
    dinner VARCHAR(100),
    dinner_price VARCHAR(10)
);
```

5. Complaint Table

```
CREATE TABLE Complaint (
    complainId INT IDENTITY(200,1) PRIMARY KEY,
    studentemail VARCHAR(100),
    subject VARCHAR(255),
    description VARCHAR(500),
    FOREIGN KEY (studentemail) REFERENCES Student(email)
);
```

6. AcceptedStudent Table

```
CREATE TABLE AcceptedStudent (
    serial INT IDENTITY(1,1) PRIMARY KEY,
    id AS ('23-' + RIGHT('0000' + CAST(serial AS VARCHAR(4)), 4))
PERSISTED,
    name VARCHAR(100),
    fatherName VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20),
    institution VARCHAR(100),
    DOB VARCHAR(20),
    gender VARCHAR(10),
    address VARCHAR(255),
    password VARCHAR(100)
);
```

7. RequestedStudent Table

```
CREATE TABLE RequestedStudent (
    name VARCHAR(100),
    fatherName VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(20),
    institution VARCHAR(100),
    DOB VARCHAR(20),
    gender VARCHAR(10),
    address VARCHAR(255),
    password VARCHAR(100)
);
```

8. StudentMeal Table

```
CREATE TABLE StudentMeal (
    studentEmail VARCHAR(100),
    mealDate VARCHAR(50),
    bprice VARCHAR(10),
    lprice VARCHAR(10),
    nprice VARCHAR(10),
    total VARCHAR(10),
    CONSTRAINT FK_StudentEmail FOREIGN KEY (studentEmail)
        REFERENCES Student(email)
);
```

9. Payment Table

```
CREATE TABLE Payment (
    serial INT IDENTITY(1,1) PRIMARY KEY,
    paymentId AS ('AX-' + RIGHT('00' + CAST(serial AS VARCHAR(2)),
2)) PERSISTED,
    studentEmail VARCHAR(100),
    paymentDate VARCHAR(30),
    amount VARCHAR(30),
```

```
    phone VARCHAR(30),
    paymentMethod VARCHAR(50),
    CONSTRAINT FK_Payment_StudentEmail FOREIGN KEY (studentEmail)
        REFERENCES Student(email)
);
```

Insert Queries

1. Insert Admin Data

```
INSERT INTO Admin (adminId, name, salary, password)
VALUES
('F-1', 'Farhan', 75000.00, 'farhan@123'),
('S-1', 'Sajid', 72000.00, 'sajid@123'),
('N-1', 'Nayeem', 73000.00, 'nayeem@123');
```

2. Insert Employee Data

```
INSERT INTO Employee (employeeId, name, salary, password, role)
VALUES
('E-101', 'Alice Karim', '35000.00', 'alice123', 'Manager'),
('E-102', 'Tanvir Rahman', '28000.00', 'tanvir123', 'Staff'),
('E-201', 'Nusrat Jahan', '32000.00', 'nusrat321', 'HR'),
('E-301', 'Sajid Khan', '30000.00', 'sajid789', 'Assistant'),
('E-302', 'Rumi Das', '26000.00', 'rumi456', 'Staff');
```

3. Insert Student Data (Dynamic)

```
INSERT INTO Student VALUES (@name, @fatherName, @email, @phone,
@institution, @dob, @gender, @address, @password);
```

4. Insert Meal Data

```
INSERT INTO Meal (mealId, day, breakfast, breakfast_price, lunch,
lunch_price, dinner, dinner_price)
VALUES
('M-1', 'Monday', 'Paratha & Egg', '25.00', 'Rice & Chicken Curry',
'60.00', 'Roti & Daal', '30.00'),
('M-2', 'Tuesday', 'Bread & Jam', '20.00', 'Rice & Fish Curry',
'55.00', 'Khichuri & Egg', '35.00'),
('M-3', 'Wednesday', 'Suji & Banana', '18.00', 'Polao & Chicken
Roast', '70.00', 'Roti & Mixed Veg', '28.00'),
('M-4', 'Thursday', 'Chotpoti', '22.00', 'Plain Rice & Egg Curry',
'50.00', 'Noodles & Sausage', '40.00'),
('M-5', 'Friday', 'Halwa & Puri', '30.00', 'Beef Tehari', '80.00',
'Roti & Lentils', '25.00'),
('M-6', 'Saturday', 'Panta & Fried Hilsha', '35.00', 'Vegetable
Biryani', '65.00', 'Roti & Egg Curry', '30.00'),
('M-7', 'Sunday', 'Milk Bread & Banana', '20.00', 'Chicken
Biryani', '75.00', 'Khichuri & Chicken', '40.00');
```

Select Queries

1. View All Students

```
SELECT * FROM Student;
```

2. View All Admins

```
SELECT * FROM Admin;
```

3. View All Employees

```
SELECT * FROM Employee;
```

4. View All Meals

```
SELECT * FROM Meal;
```

5. View All Complaints

```
SELECT * FROM Complaint;
```

6. View All Accepted Students

```
SELECT * FROM AcceptedStudent;
```

7. View All Requested Students

```
SELECT * FROM RequestedStudent;
```

8. View All Student Meals

```
SELECT * FROM StudentMeal;
```

9. View All Payments

```
SELECT * FROM Payment;
```

Search Queries

1. Search Student by Institution

```
SELECT name, email, password FROM Student WHERE institution =
'aiub';
```

2. Search Monday's Breakfast Menu

```
SELECT breakfast, breakfast_price FROM Meal WHERE day = 'Monday';
```

3. Search Accepted Student by Email

```
SELECT id, password FROM AcceptedStudent WHERE email =
'taimoor.aslam@example.com';
```

Update Queries

1. Update Full Student Record

```
UPDATE Student
SET name = @name, fatherName = @fatherName, phone = @phone,
institution = @institution, DOB = @dob, gender = @gender, address =
@address, password = @password
WHERE email = @email;
```

2. Update Student Password Only

```
UPDATE Student
SET password = @password
WHERE email = @email;
```

Delete Queries

1. Delete a Student by Email

```
DELETE FROM Student
WHERE email = 'clara.lee@example.com';
```

2. Delete Accepted Student by Name

```
DELETE FROM AcceptedStudent
WHERE name = 'test';
```

Chapter 05: Screenshots

Purpose: Demonstrate your working application using screenshots.

Screenshots:

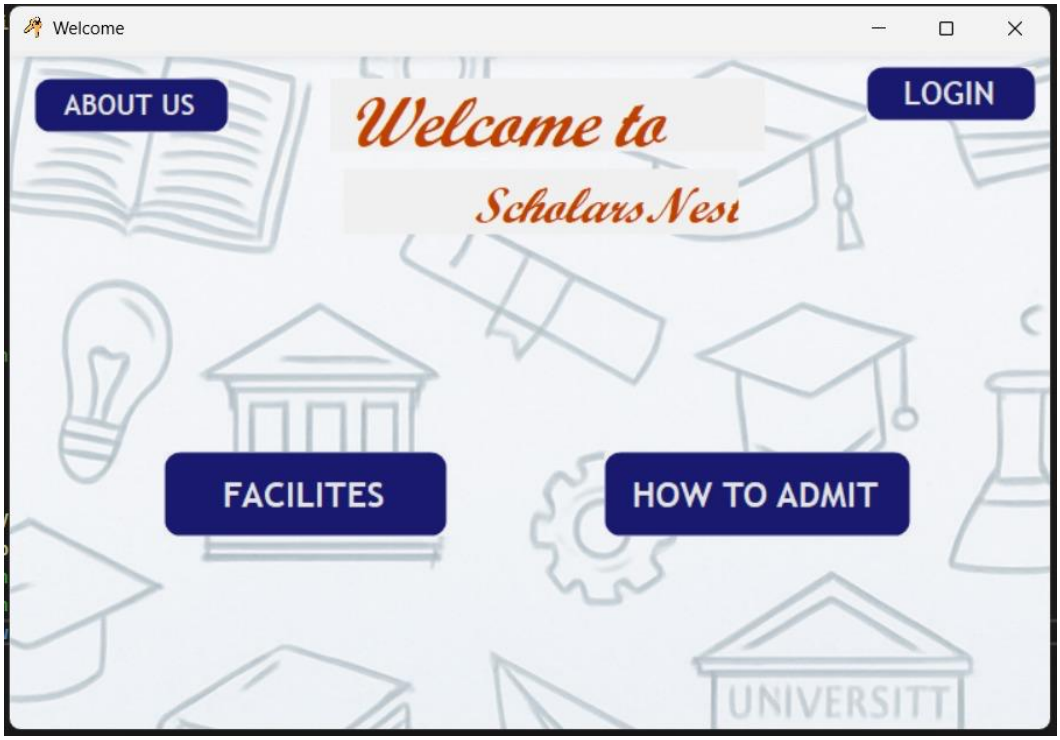


Figure 01: Welcome Page

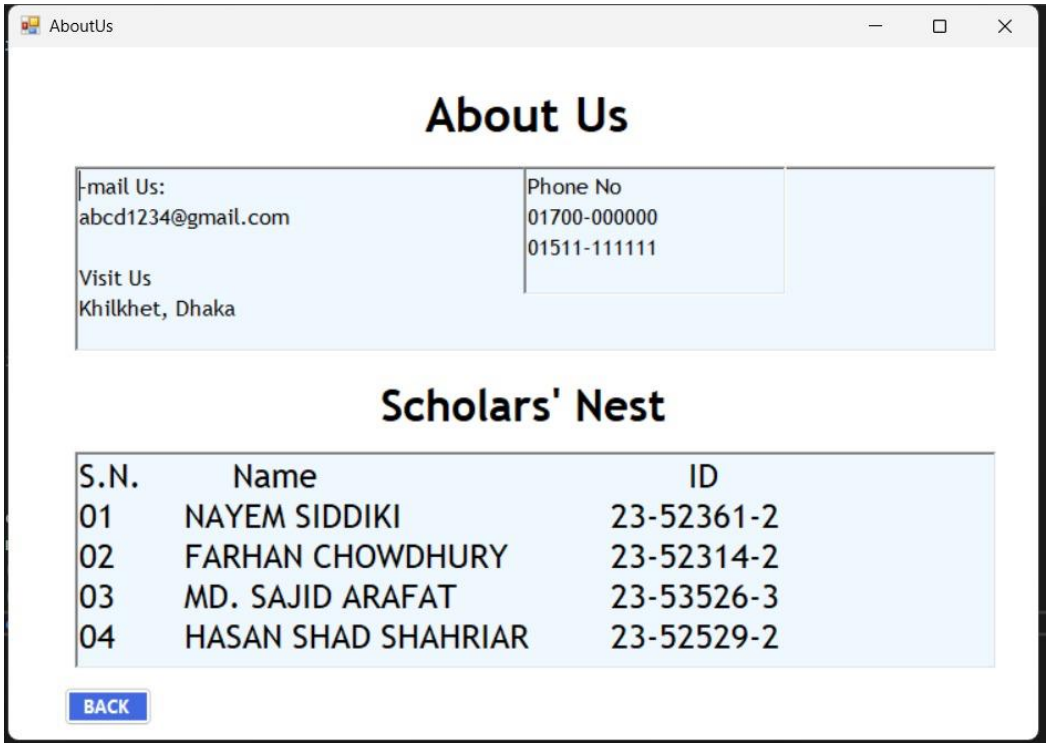


Figure 02: About Us Page

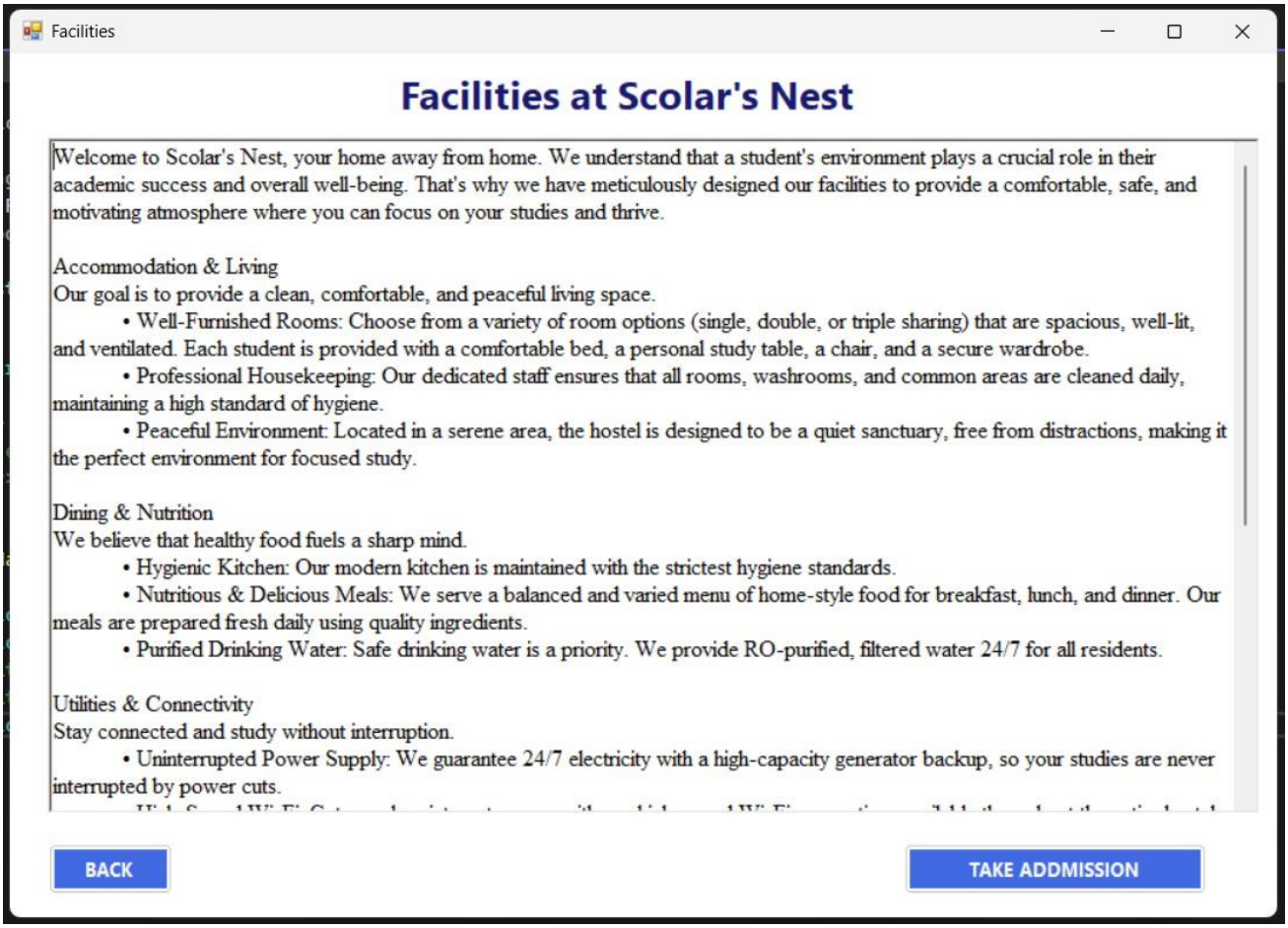


Figure 03: Facilities Page

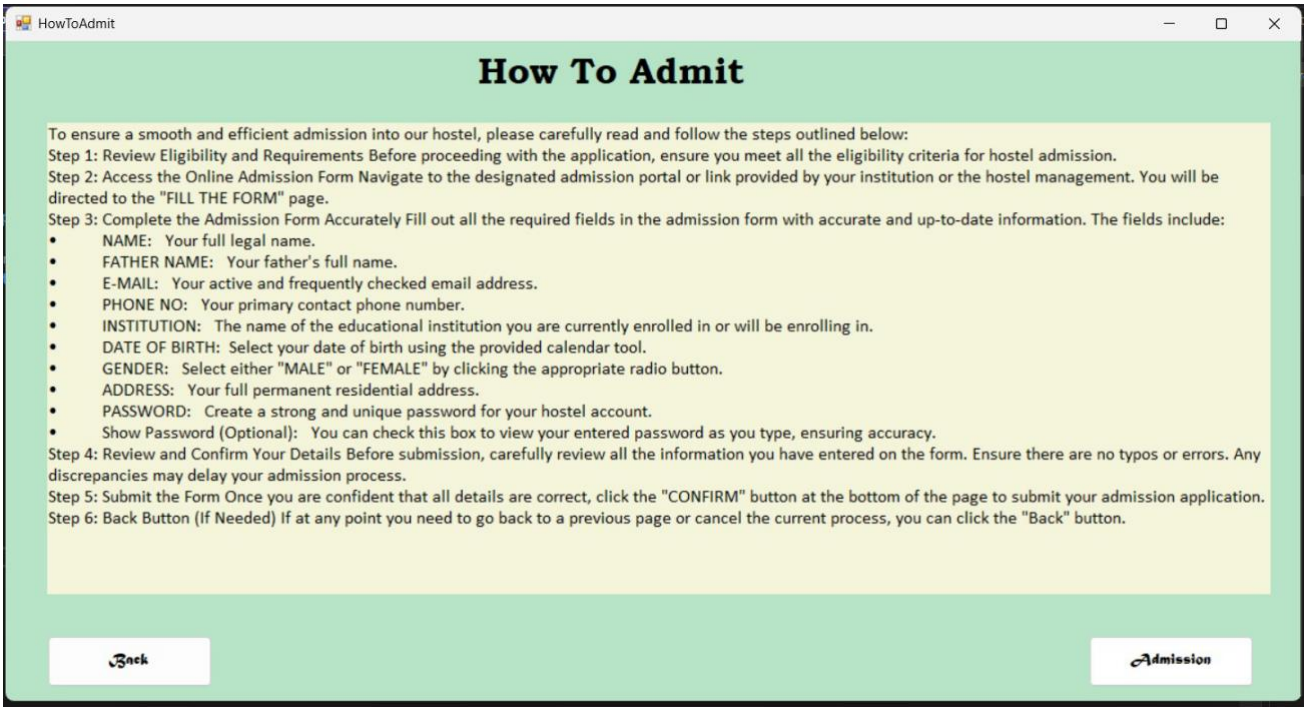


Figure 04: How To Admit Page

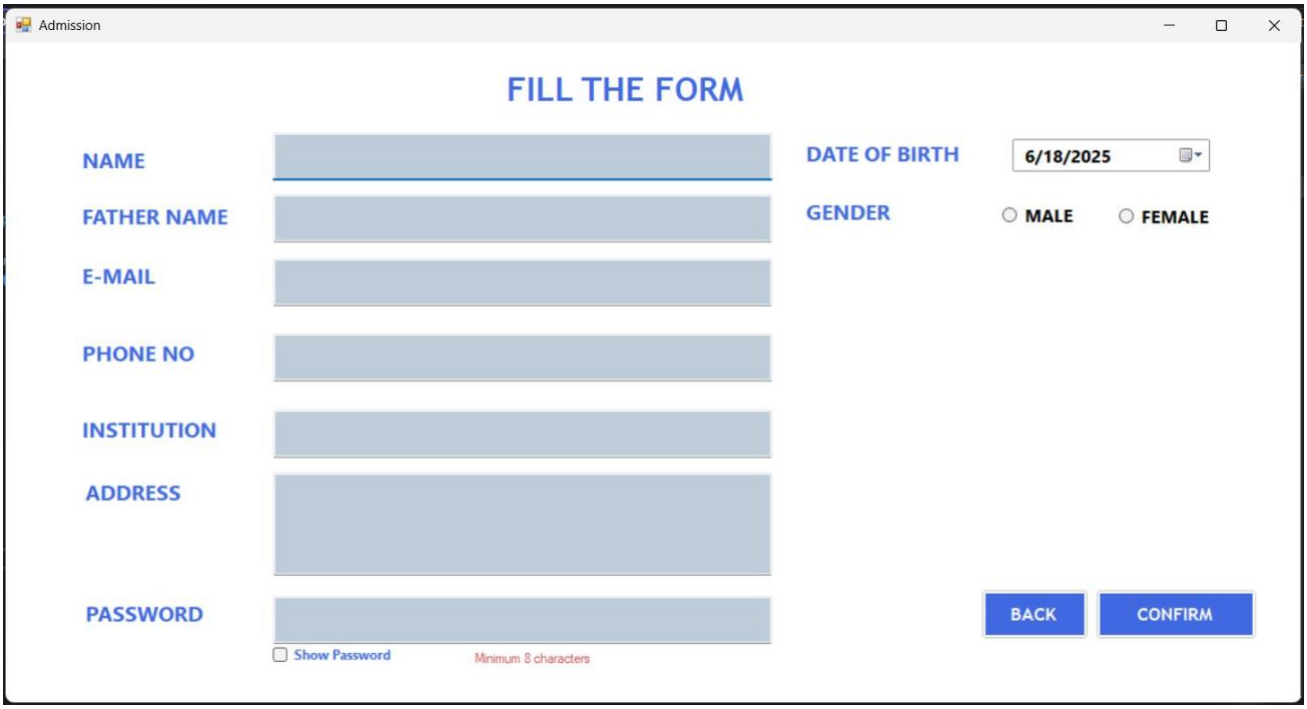


Figure 05: Admission Form Page

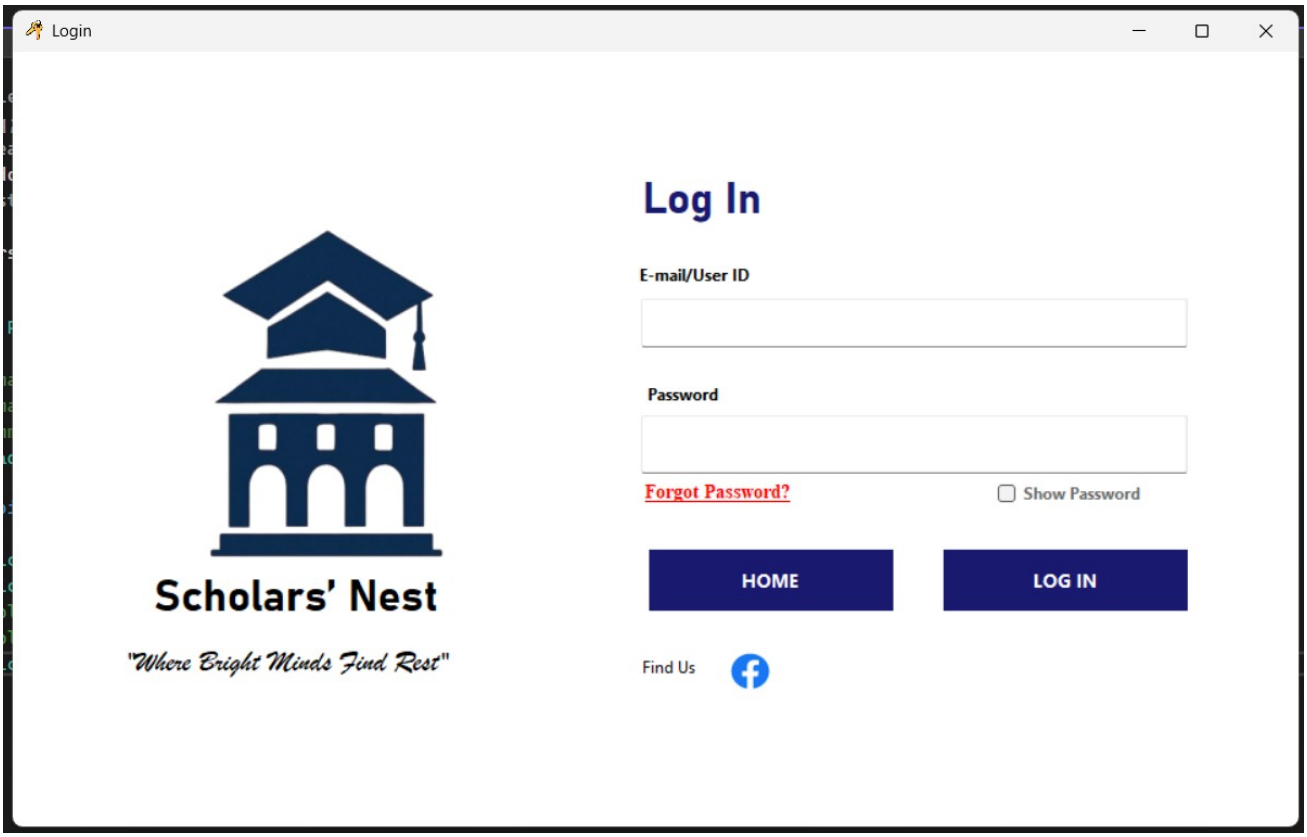


Figure 06: Log In Page

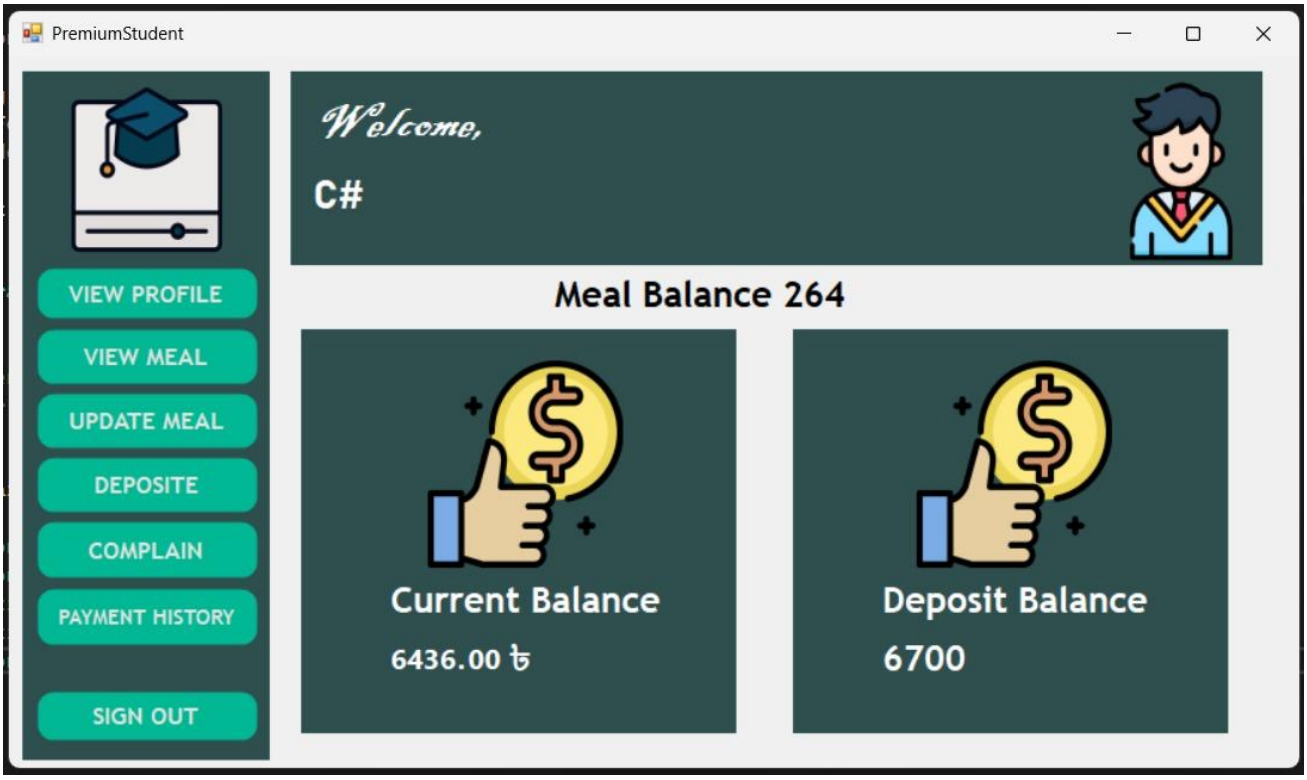


Figure 07: Student Dashboard

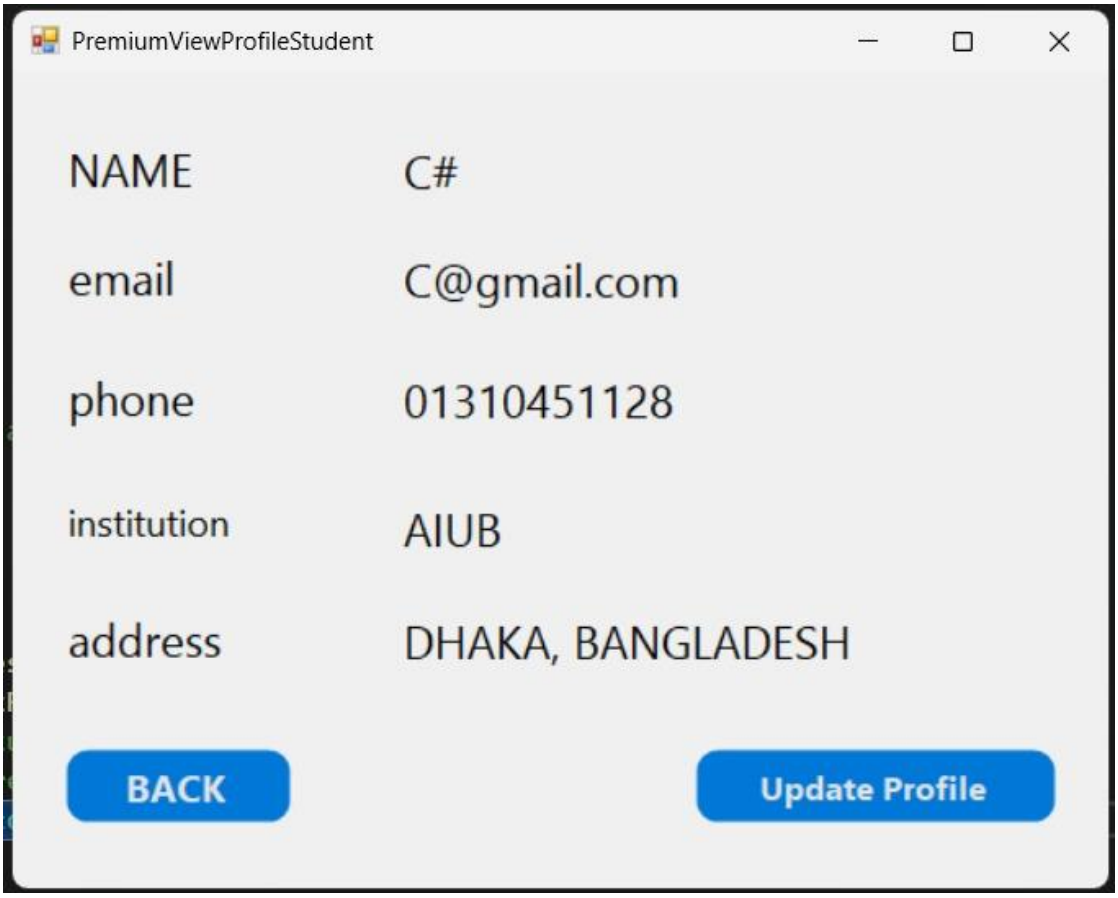


Figure 08: View Profile Page

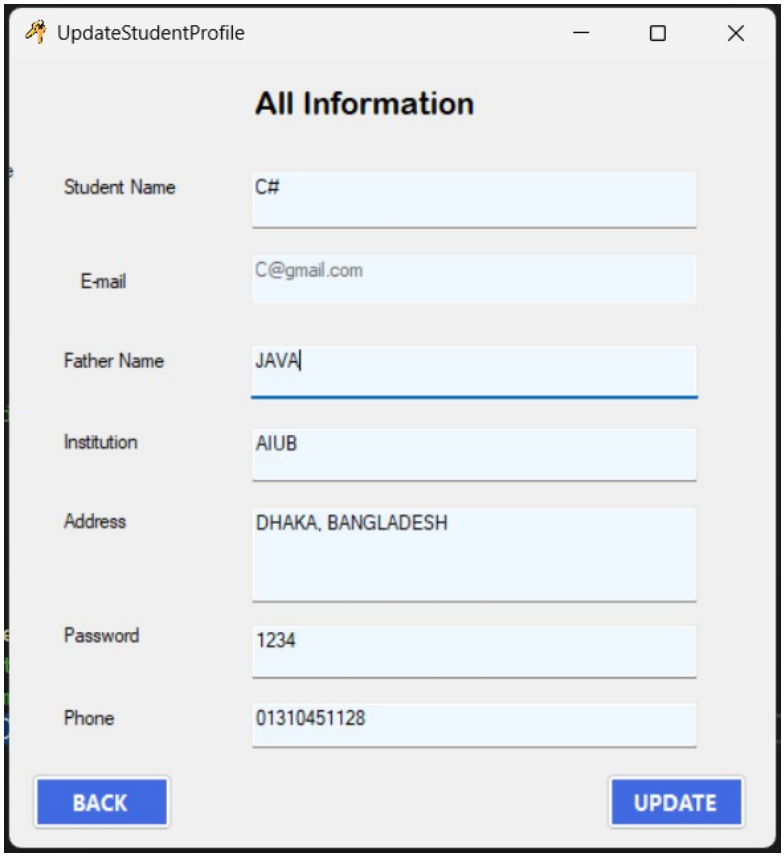


Figure 09: Update Student Profile Page

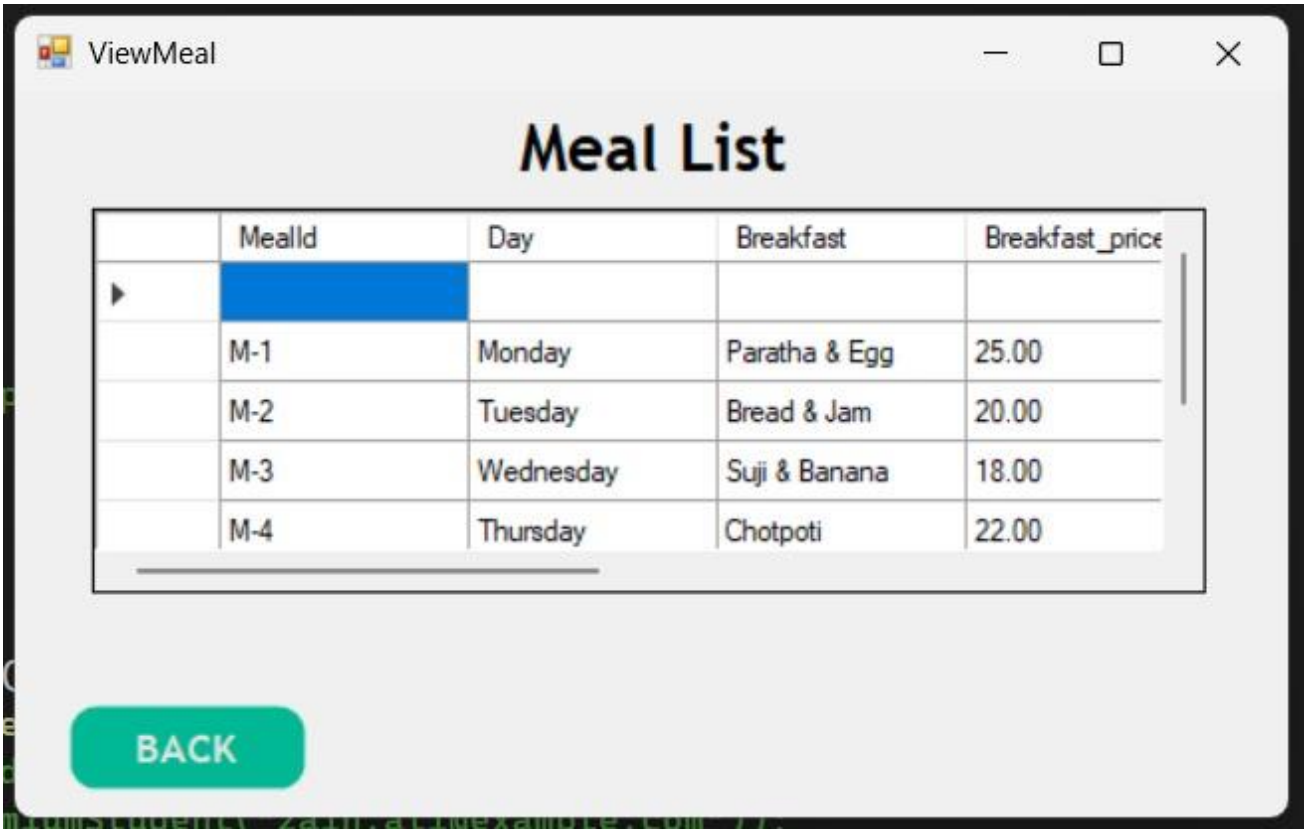


Figure 10: View Meal Page

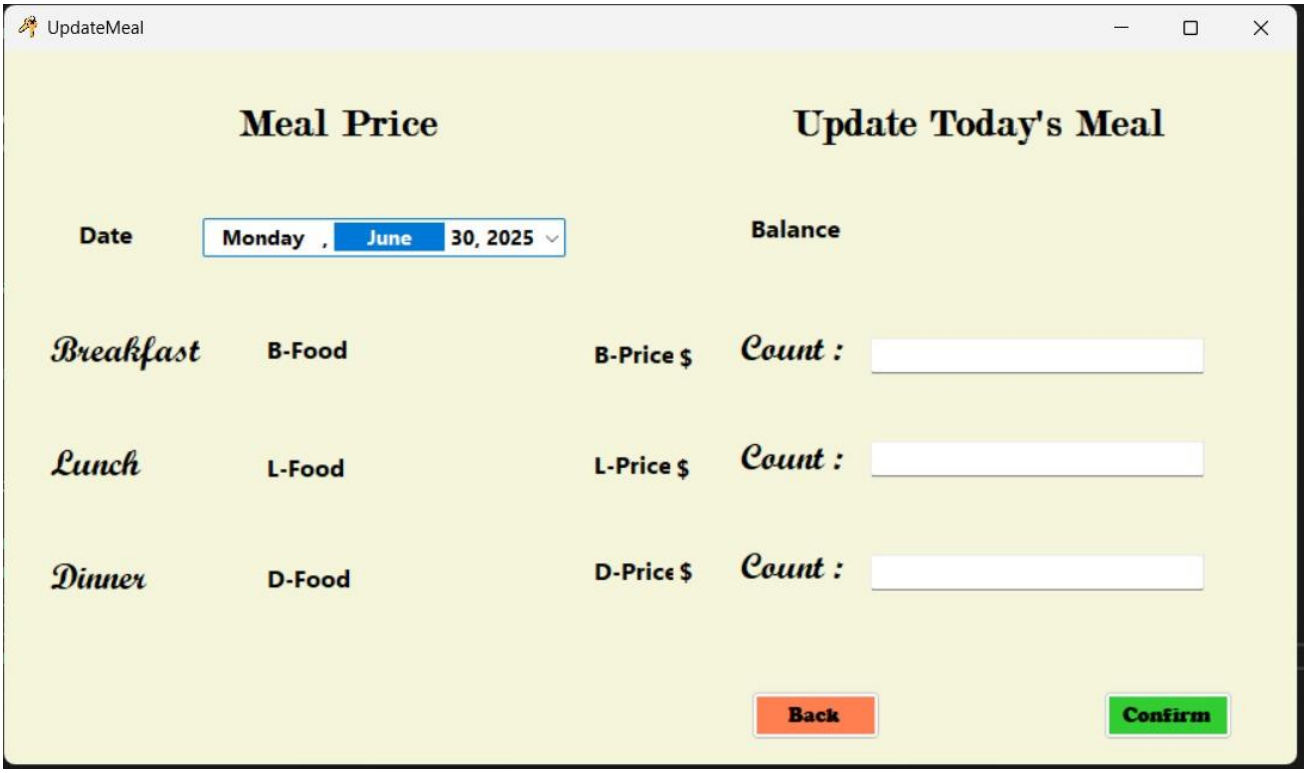


Figure 11: Update Meal Page

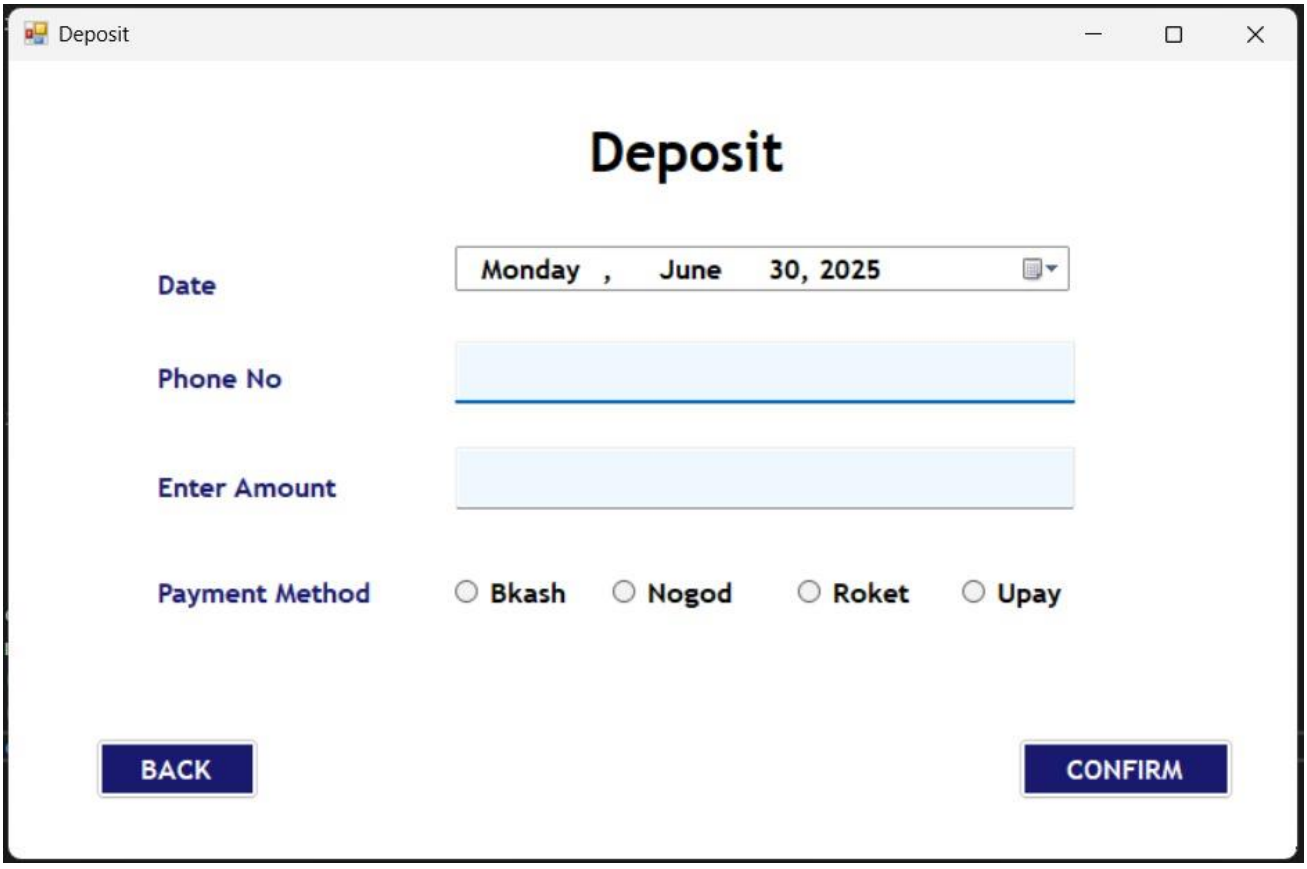


Figure 12: Deposit Page

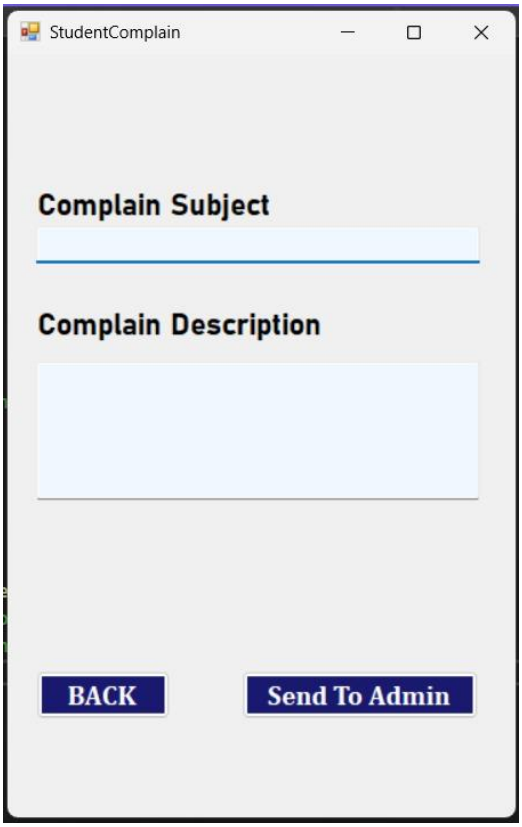


Figure 13: Student Complain Page

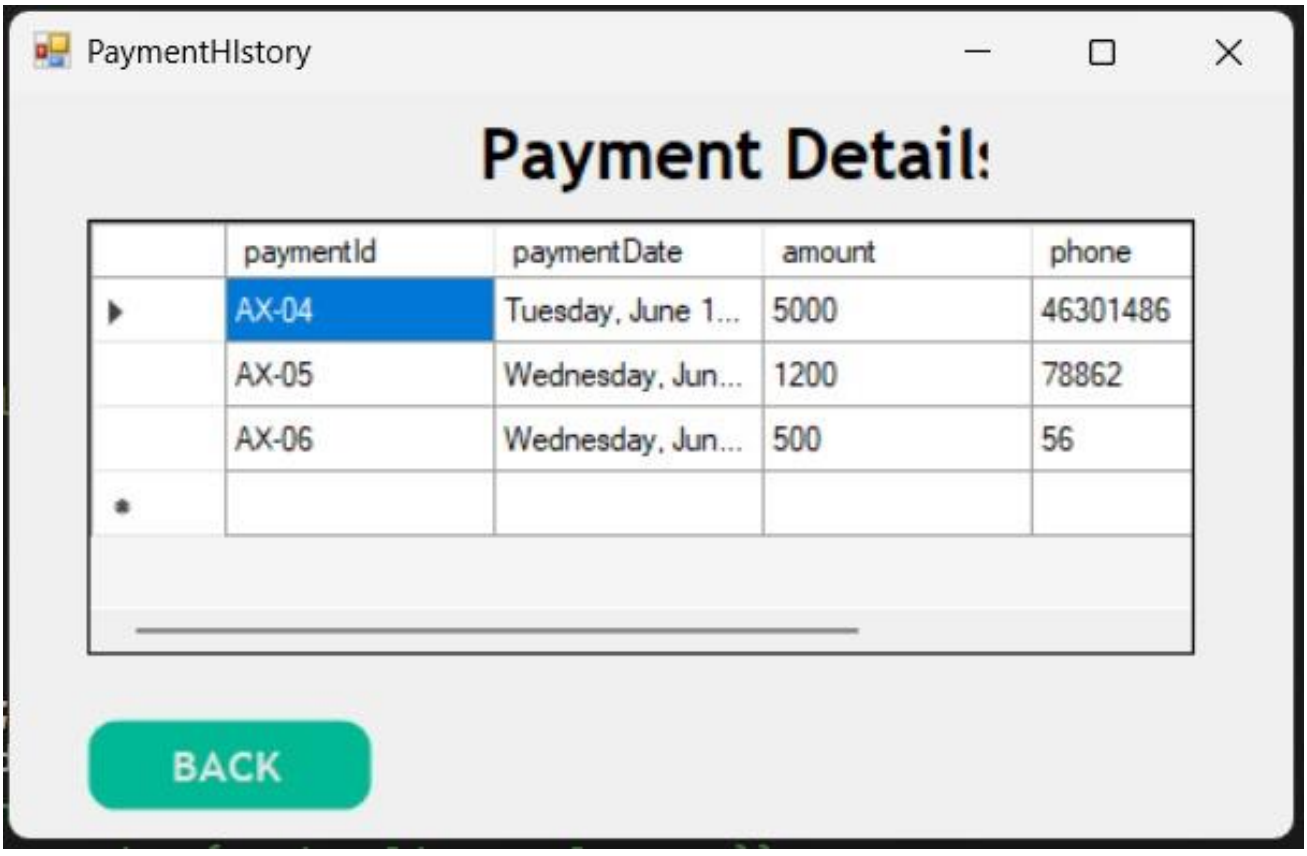


Figure 14: Payment History Page

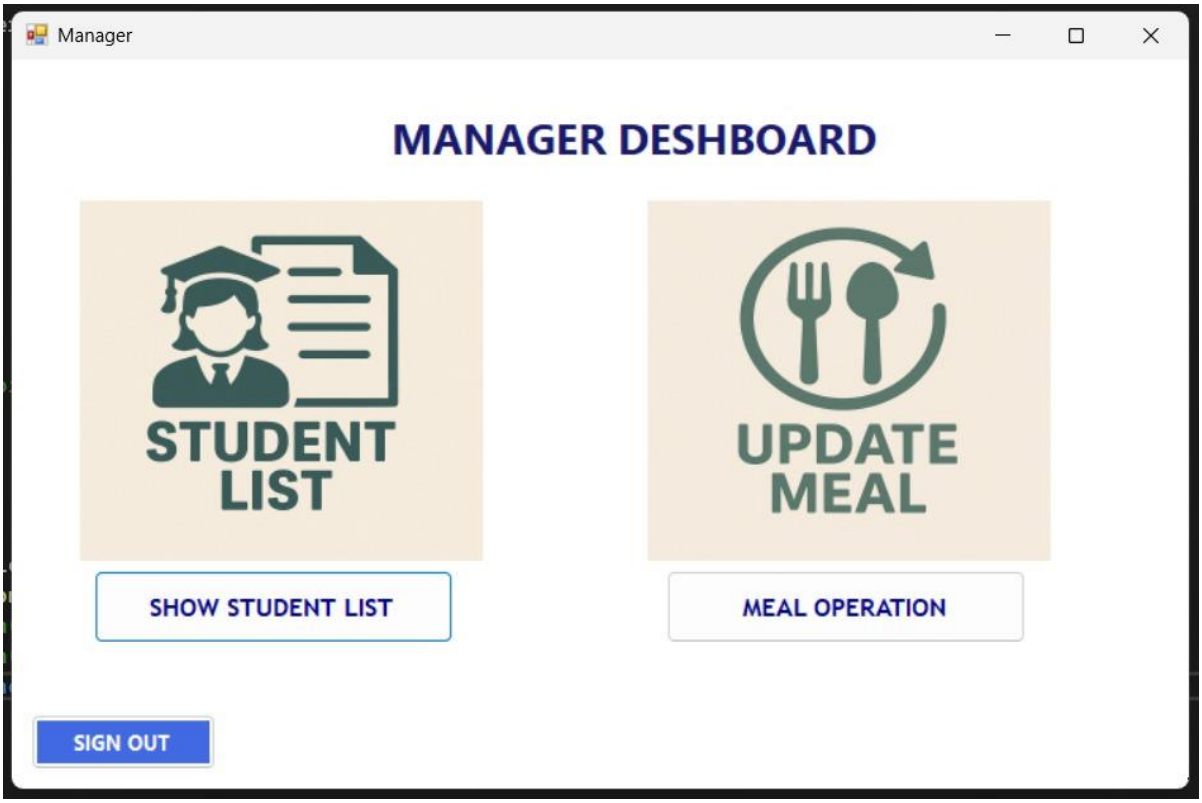


Figure 15: Manager Dashboard

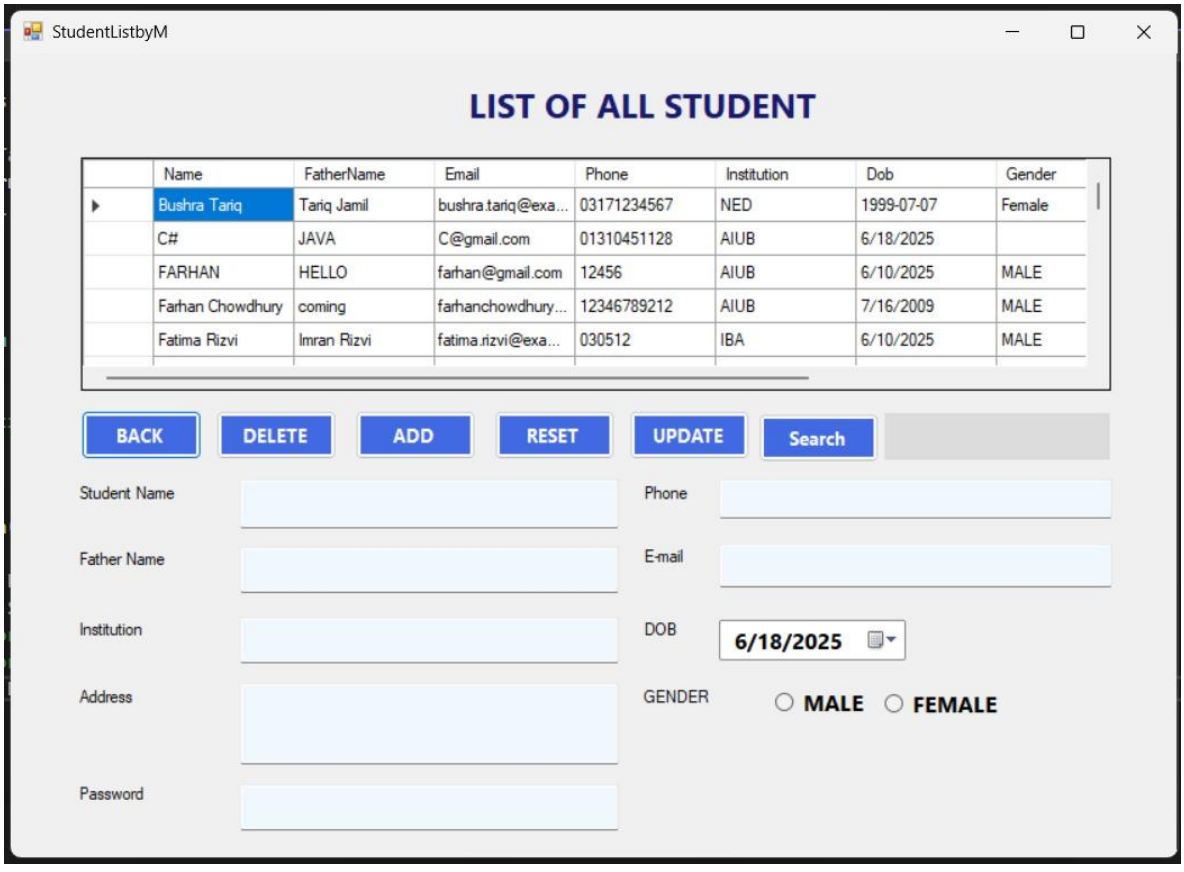


Figure 16: Student List by Manager (For Add, Delete, Update Information

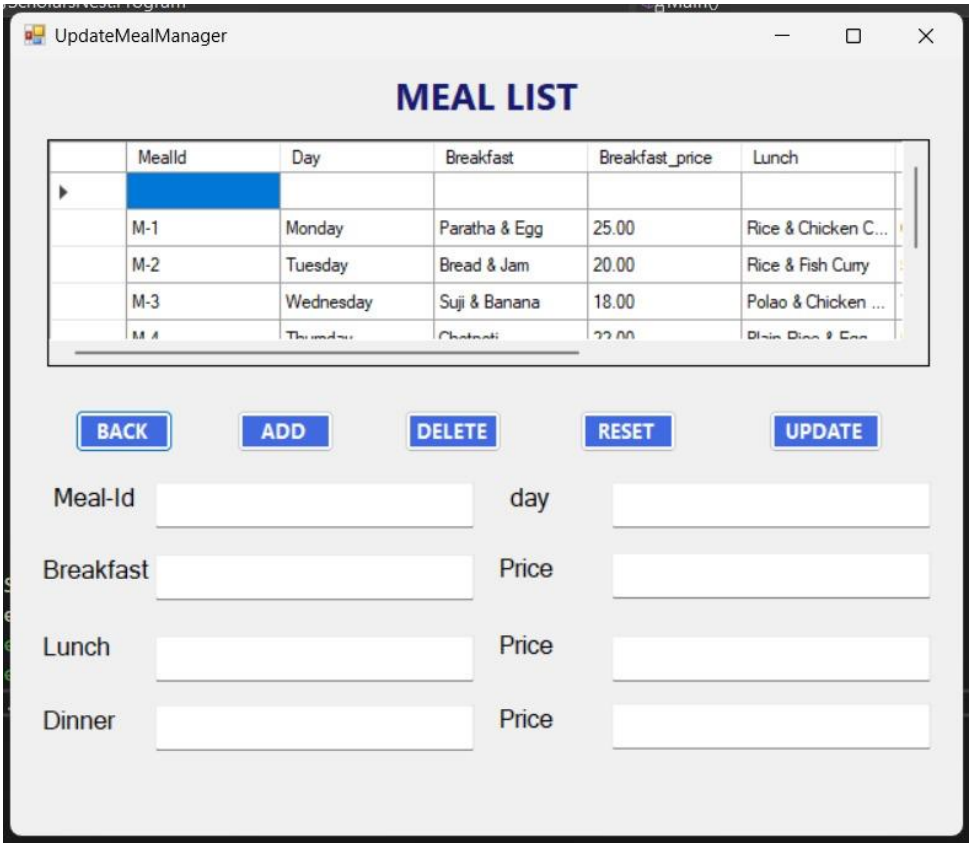


Figure 17: View and Update Meal Menu by Manager

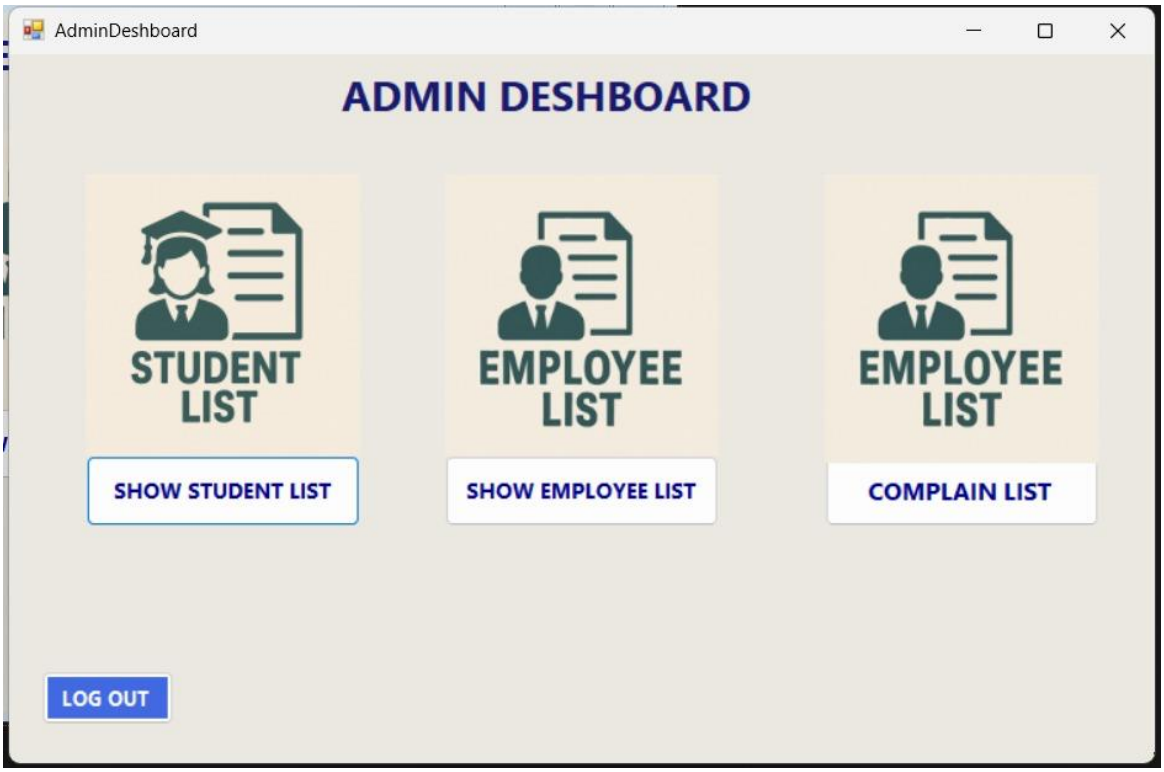


Figure 18: Admin Dashboard Page

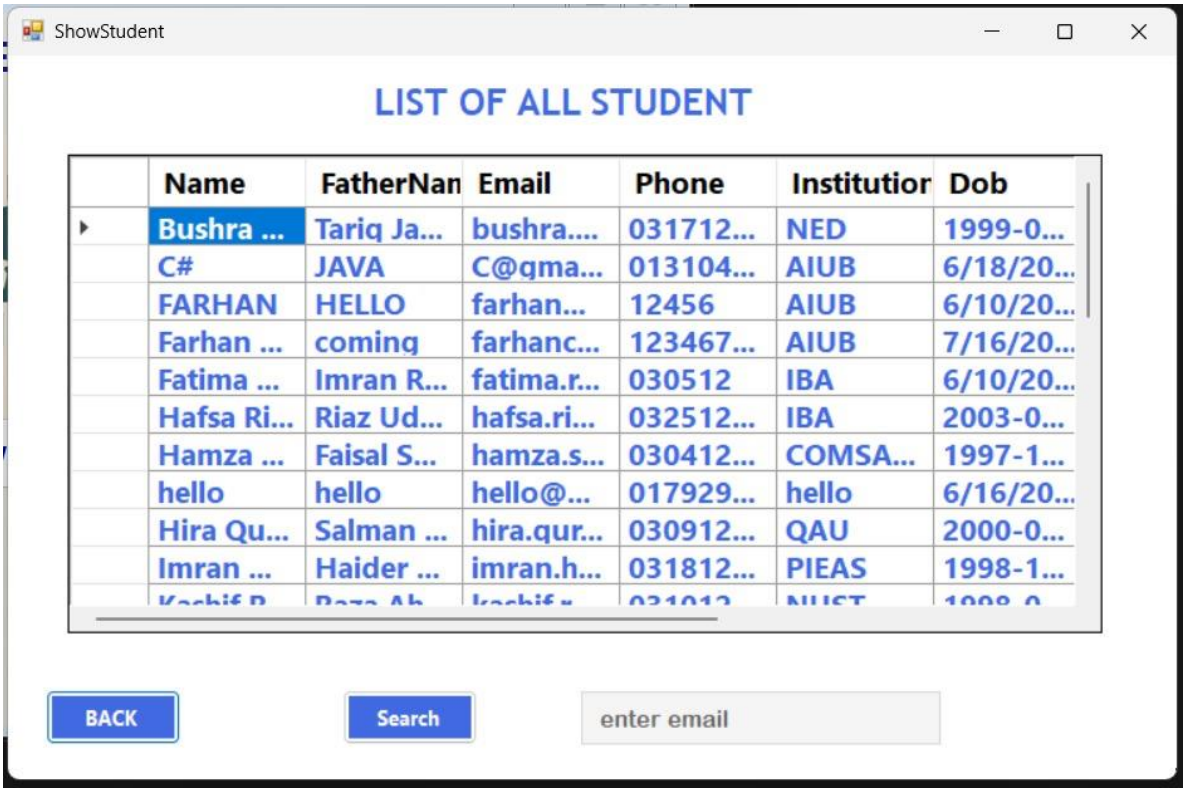


Figure 19: Student List (only) View by Admin

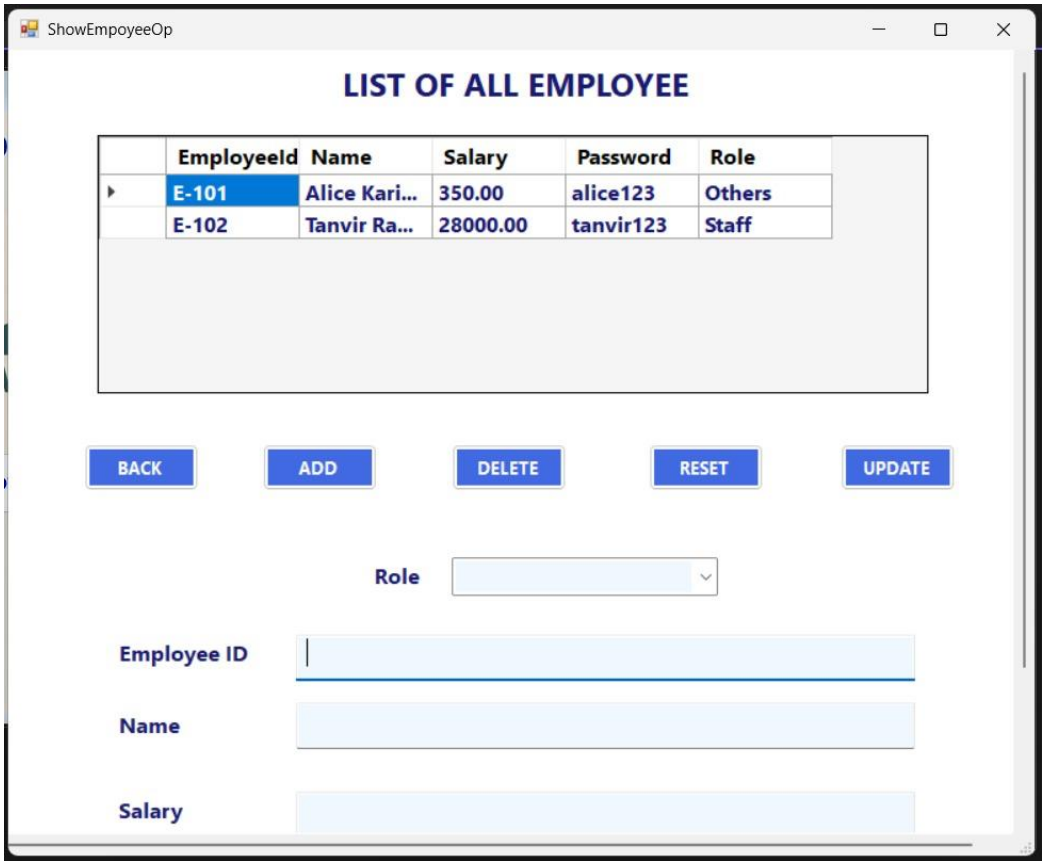


Figure 20: Password Update after Verifying E-mail

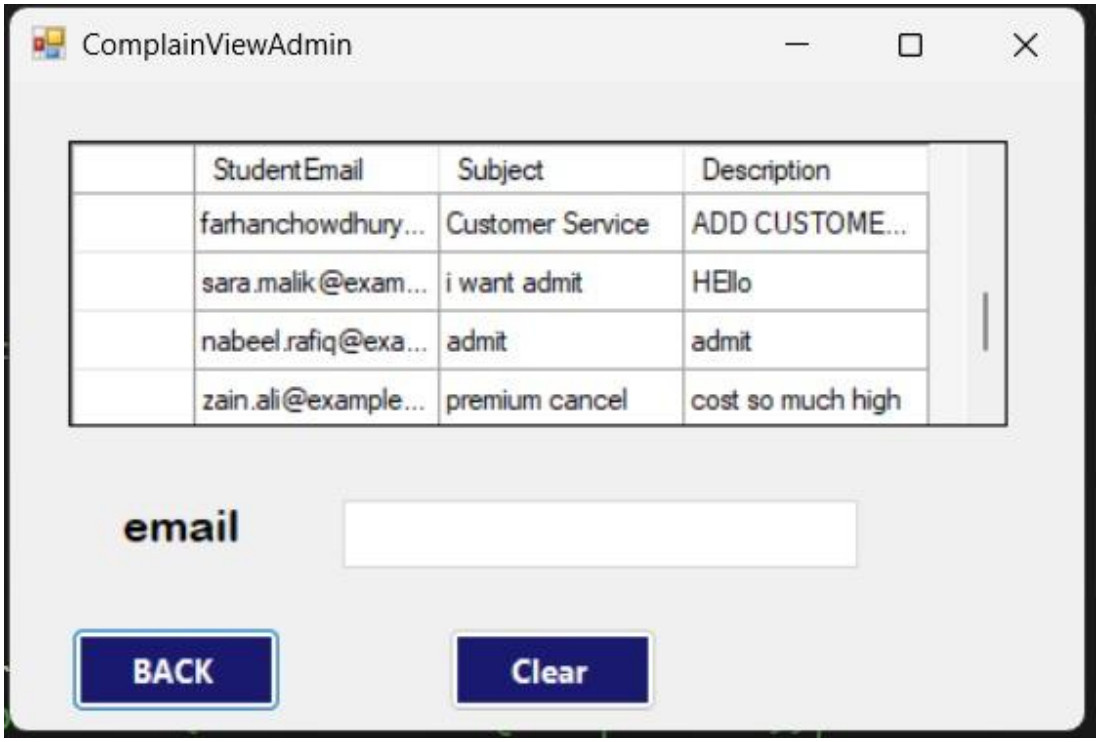


Figure 21: View Complain Page

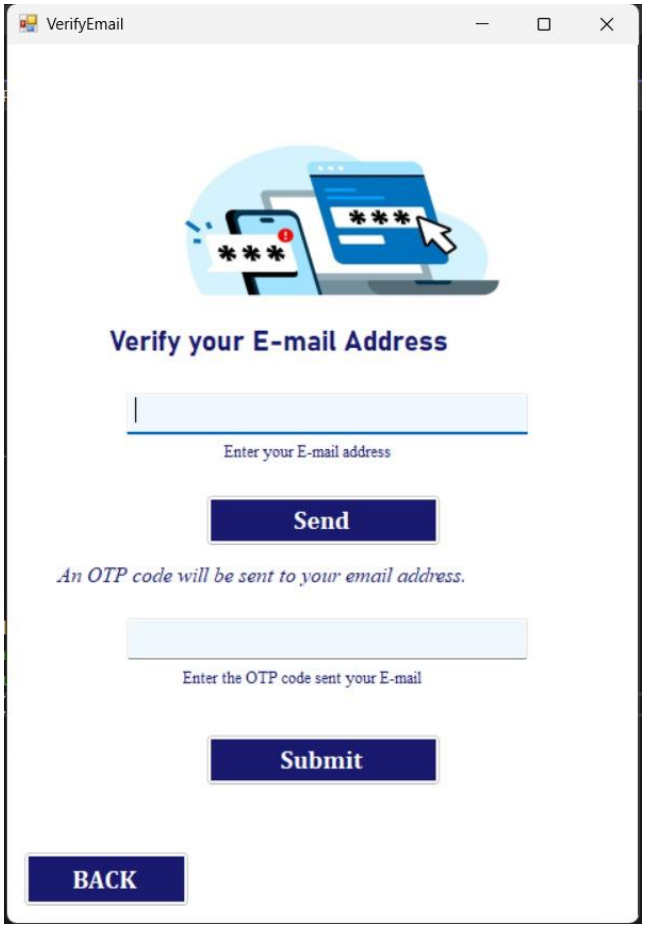


Figure 22: Verify E-mail for Forget Password Click

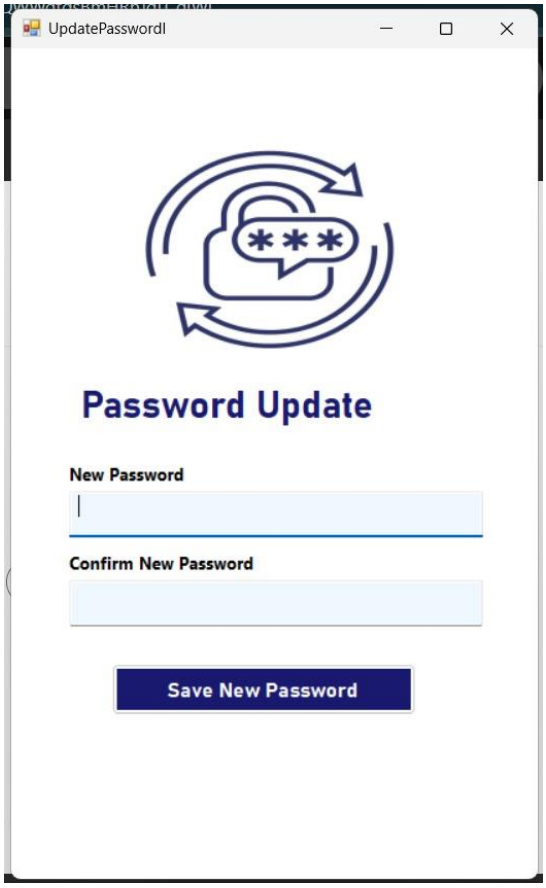


Figure 23: Password Update after Verifying E-mail

THE END