UnicomTIC Management System - Project Report

1. Project Overview (Point-Wise)

- • Key Features Implemented:
    - - Student, Lecturer, Admin, and Staff login system.
    - - Course, Subject, and Room management modules.
    - - Exam creation and Marks entry module.
    - - Attendance tracking per subject per date.
    - - Timetable assignment with subjects and rooms.
    - - Role-based access and restrictions for each user type.
    - - Use of MVC design pattern for form separation and logic.

- • Technologies Used:
    - - Language: C# (Windows Forms)
    - - Database: SQLite
    - - Pattern: MVC (Model-View-Controller)
    - - Other Tools: Visual Studio, NuGet packages (System.Data.SQLite)

- • Challenges Faced & Solutions:
    - - Challenge: Handling one-to-many relationships in SQLite.
    - Solution: Used mapping tables and foreign key logic.
    - - Challenge: Preventing duplicate attendance entries.
    - Solution: Used unique constraints and validation.
    - - Challenge: Replacing SQLiteDataAdapter with SQLiteCommand.
    - Solution: Rewrote DB logic using parameterized queries.

To Run This Project Need To RE Install Some Packages

☑ Step 1: Restore NuGet Packages

- Open the solution (.sln) file in Visual Studio, then:
    i. Go to the Tools menu → NuGet Package Manager → **Package Manager Console**
    ii. In the console, run:

- `Update-Package -reinstall`

☑ Step 2: Clean & Rebuild the Project In Visual Studio:

- Go to Build → Clean Solution
- Then Build → Rebuild Solution

☑ Step 5: Run the App

- Press F5 to run the application.

2. Code Samples (Screenshots)

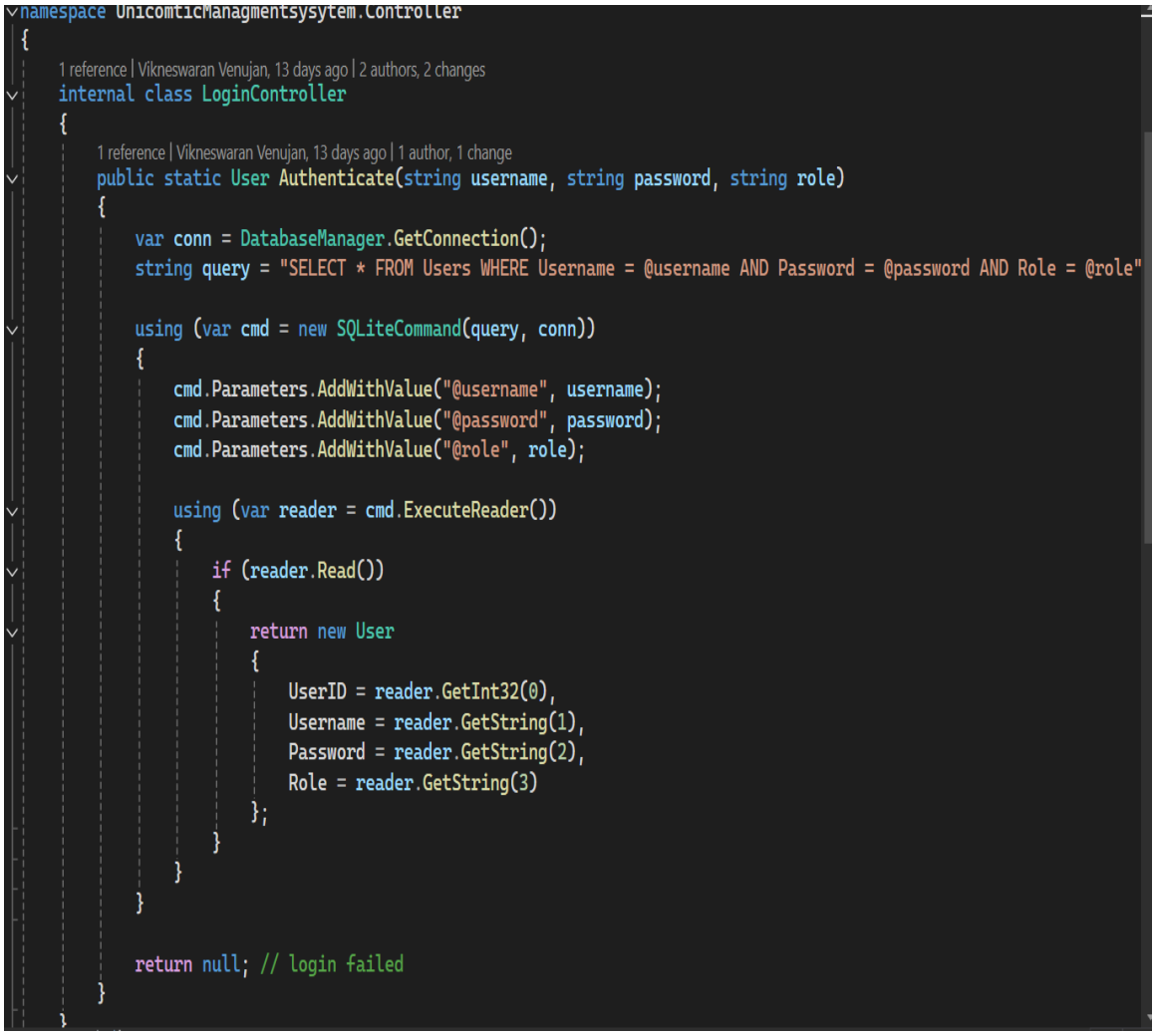
Attach up to 6 screenshots of your code that you believe represent your best work. Each screenshot can include a short caption/description.

1. **On first run of the application, the admin must create their own login credentials manually. This admin information is not pre-inserted into the Admin table. Once created, the admin can log in and access their dashboard**
    - • If users exist, it directly opens the **Login Form**.
    - • If no users are found, it opens the **Admin Registration Form**, allowing the first admin to create their login credentials.
    - • This ensures that the system is secure and cannot be accessed without an admin account.

```
10
11  namespace UnicomticManagmentsysytem
12  {
        0 references | Vikneswaran Venujan, 13 days ago | 1 author, 1 change
13      internal static class Program
14      {
15          /// <summary>
16          /// The main entry point for the application.
17          /// </summary>
18          [STAThread]
            0 references | Vikneswaran Venujan, 13 days ago | 1 author, 1 change
19          static void Main()
20          {
21              var conn = DatabaseManager.GetConnection();
22              Application.EnableVisualStyles();
23              Application.SetCompatibleTextRenderingDefault(false);
24              // Check if any users exist
25              string checkUserQuery = "SELECT COUNT(*) FROM Users";
26              using (var cmd = new SQLiteCommand(checkUserQuery, conn))
27              {
28                  long userCount = (long)cmd.ExecuteScalar();
29                  if (userCount == 0)
30                  {
31                      // No users? Show RegisterForm
32                      Application.Run(new RegisterForm());
33                  }
34                  else
35                  {
36                      // Users exist? Show LoginForm
37                      Application.Run(new LoginForm());
38                  }
39              }
40          }
41      }
42  }
43
```

2. User Login Logic (Role-based): This method checks the username/password and redirects users based on their roles.

*loginController



```
namespace UnicomticManagmentsysytem.Controller
{
    1 reference | Vikneswaran Venujan, 13 days ago | 2 authors, 2 changes
    internal class LoginController
    {
        1 reference | Vikneswaran Venujan, 13 days ago | 1 author, 1 change
        public static User Authenticate(string username, string password, string role)
        {
            var conn = DatabaseManager.GetConnection();
            string query = "SELECT * FROM Users WHERE Username = @username AND Password = @password AND Role = @role"

            using (var cmd = new SQLiteCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@username", username);
                cmd.Parameters.AddWithValue("@password", password);
                cmd.Parameters.AddWithValue("@role", role);

                using (var reader = cmd.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return new User
                        {
                            UserID = reader.GetInt32(0),
                            Username = reader.GetString(1),
                            Password = reader.GetString(2),
                            Role = reader.GetString(3)
                        };
                    }
                }
            }

            return null; // login failed
        }
    }
```

Login form

```
namespace UnicomticManagementsysytem.Views
{
    8 references | Vikraswaran Verujan, 2 days ago | 2 authors, 7 changes
    public partial class LoginForm : Form
    {
        4 references | Vikraswaran Verujan, 13 days ago | 2 authors, 2 changes
        public LoginForm()
        {
            InitializeComponent();
            cmbrole.Items.AddRange(new string[] { "Select Your Role","Admin", "Staff", "Student", "Lecturer" });
            cmbrole.SelectedIndex = 0;
        }

        0 references | Vikraswaran Verujan, 13 days ago | 1 author, 1 change
        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        1 reference | Vikraswaran Verujan, 13 days ago | 1 author, 1 change
        private void LoginForm_Load(object sender, EventArgs e)
        {
            //cmbrole.Items.AddRange(new string[] { "Admin", "Staff", "Student", "Lecturer" });
            //cmbrole.SelectedIndex = 0;

        }

        0 references | Vikraswaran Verujan, 13 days ago | 1 author, 1 change
        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }


        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }


        private void button1_Click(object sender, EventArgs e)
        {
            string username = txtUsername.Text.Trim();
            string password = txtPassword.Text.Trim();
            string role = cmbrole.SelectedItem.ToString();

            if (username == "" || password == "")
            {
                MessageBox.Show("Please enter username and password.", "Message");
                return;
            }
            var user = LoginController.Authenticate(username, password, role);

            if (user != null)
            {
                MessageBox.Show("Login successful!");

                this.Hide();
                MainForm mainForm = new MainForm(user.Role, user.Username,user.UserID);
                mainForm.ShowDialog();
                this.Close();
            }
            else
            {
                MessageBox.Show("Invalid login credentials.");
            }



        }

        1 reference | Vikraswaran Verujan, 2 days ago | 1 author, 2 changes
        private void chkShowPassword_CheckedChanged(object sender, EventArgs e)
        {
            //bool show = chkShowPassword.Checked;

            //txtPassword.UseSystemPasswordChar = !show;
            //txtPassword.UseSystemPasswordChar = !show;
            if (chkShowPassword.Checked) {
                txtPassword.PasswordChar = '\0';
            }
            //hide password
            else
            {
                txtPassword.PasswordChar = '*';
            }
        }

        1 reference | Vikraswaran Verujan, 13 days ago | 1 author, 1 change
        private void txtPassword_TextChanged(object sender, EventArgs e)
        {
```

Note 3: Main Form Role-Based Button Visibility

The Main Form uses **role-based logic** to control which buttons are visible depending on the logged-in user's role.

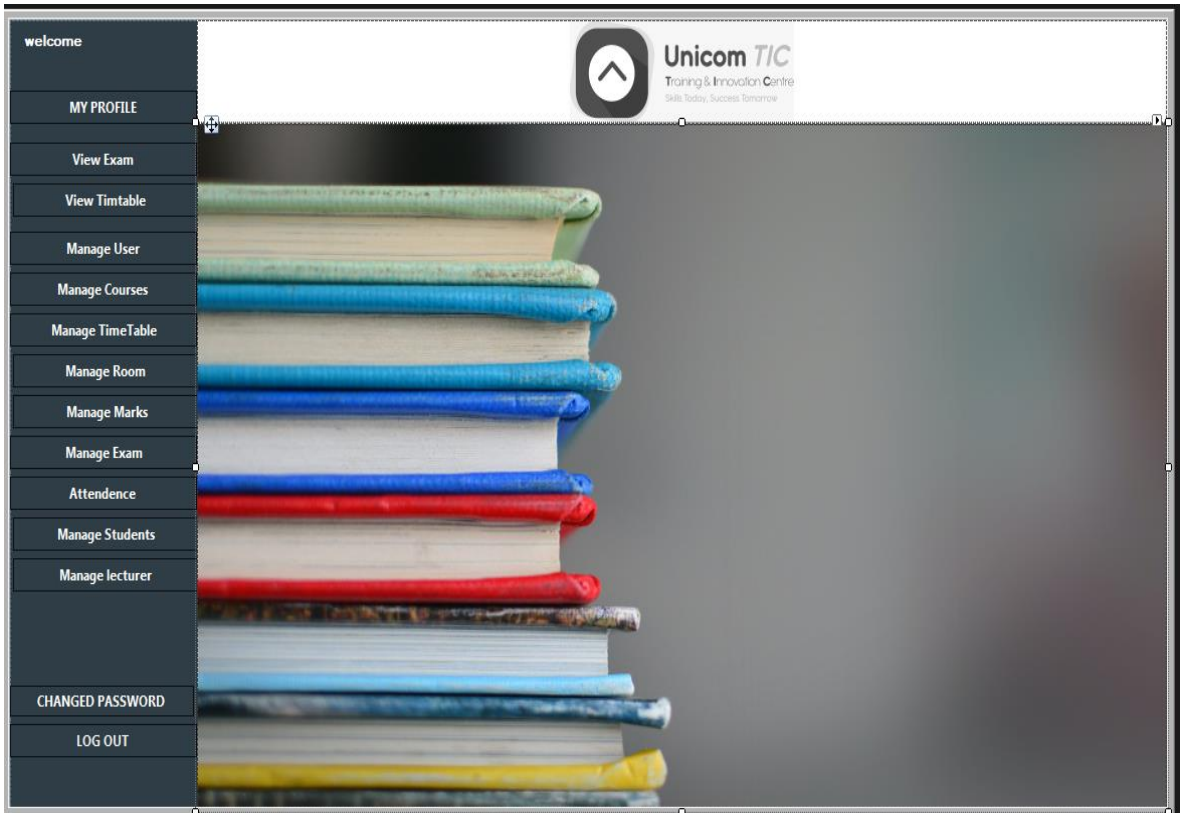
- When a user logs in, the system checks their role (Admin, Staff, Lecturer, or Student).
- Only the buttons/features permitted for that specific role are displayed on the Main Form.
- **For example:**
  - Only Admins can see the **"User Management"** and **"Course Management"** buttons.
  - Lecturers can only see features relevant to subject management and attendance.
- This ensures a clean, secure interface for every role.

```
private void ApplyRolePermissions()
{
    // Example role handling
    if (_role == "Student")
    {
        btnvexam.Visible = true;
        btnatten.Visible = false;
        btnview.Visible = true;
        btnroo.Visible = false;
        btnTimetable.Visible = false;
        btnstu.Visible = false;
        btnlec.Visible = false;
        btnaddusers.Visible = false;
        btnExams.Visible = false;
        btnco.Visible = false;
        btnMarks.Visible = false;
    }
    else if (_role == "Lecturer")
    {
        btnco.Visible = false;
        btnaddusers.Visible = false;
        btnstu.Visible = false;
        btnroo.Visible = false;
        btnlec.Visible = false;
        btnTimetable.Visible =false;
        btnMarks.Visible = true;
    }
    else if (_role == "Admin")
    {
        btnTimetable.Visible = true;
        btnMarks.Visible = true;
        btnaddusers.Visible = true;
        btnExams.Visible = true;
    }
    else if (_role == "Staff")
    {
        btnTimetable.Visible = true;
        btnMarks.Visible = true;
        btnExams.Visible = true;
    }
```

Note 4: Panel-Based Dashboard Design

The application uses a **panel-based model** for the main dashboard interface. This design approach has several benefits:

- **Modular Interface:** Instead of opening new forms for every module, all functional views (like Students, Subjects, Exams, Attendance) are loaded dynamically into a single central panel.
- **Improved Navigation:** Only one form (`MainForm`) stays open, and its central panel gets updated based on button clicks (e.g., clicking "Subjects" loads the subject view into the panel).
- **Cleaner User Experience:** The user stays within the main window, reducing clutter and confusion from multiple open windows.

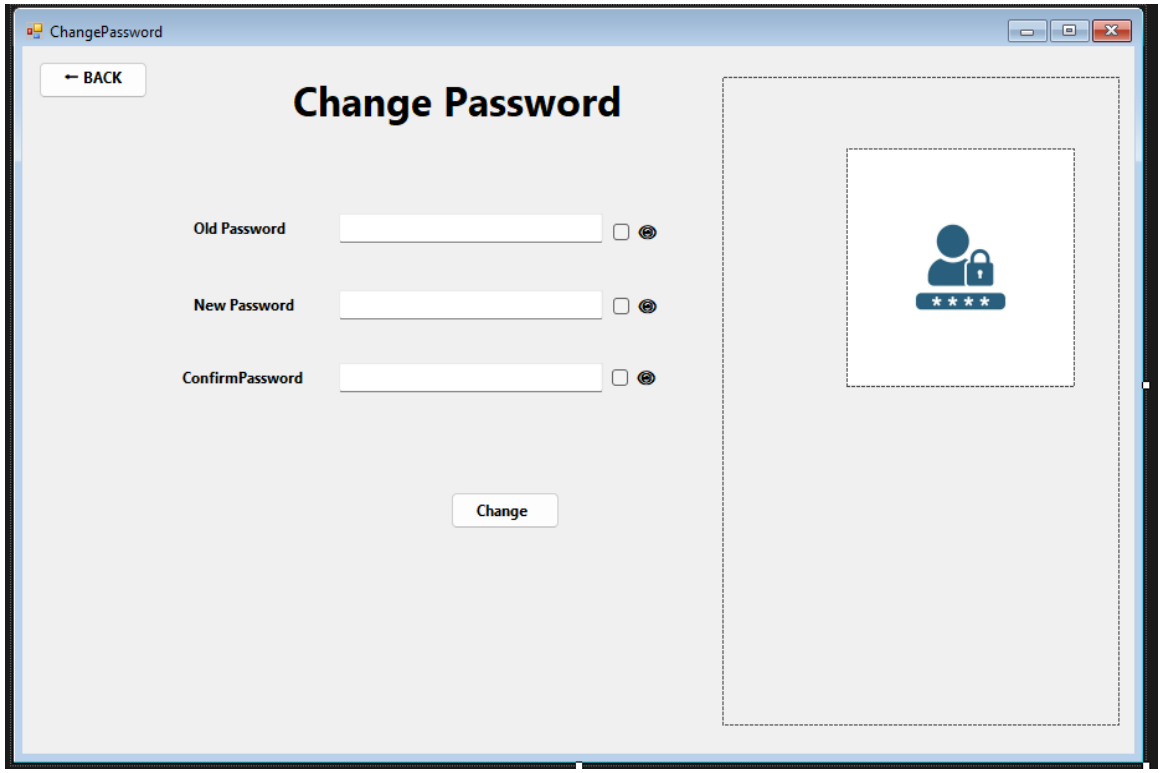


Note 5: Change Password Functionality

The application includes a secure **Change Password** form, accessible to all users after login. Key points of its functionality:

- **Current Password Validation:** The user must enter their current password correctly before setting a new one.
- **New Password Confirmation:** The form requires the user to enter the new password twice to confirm it matches.
- **Role Aware Update:** The password change affects only the currently logged-in user, ensuring no unintended data updates.
- **Seamless Integration:** This form is integrated into the dashboard using the same panel-based model, ensuring consistent UX.

This feature improves account security and allows users to manage their credentials without admin intervention.

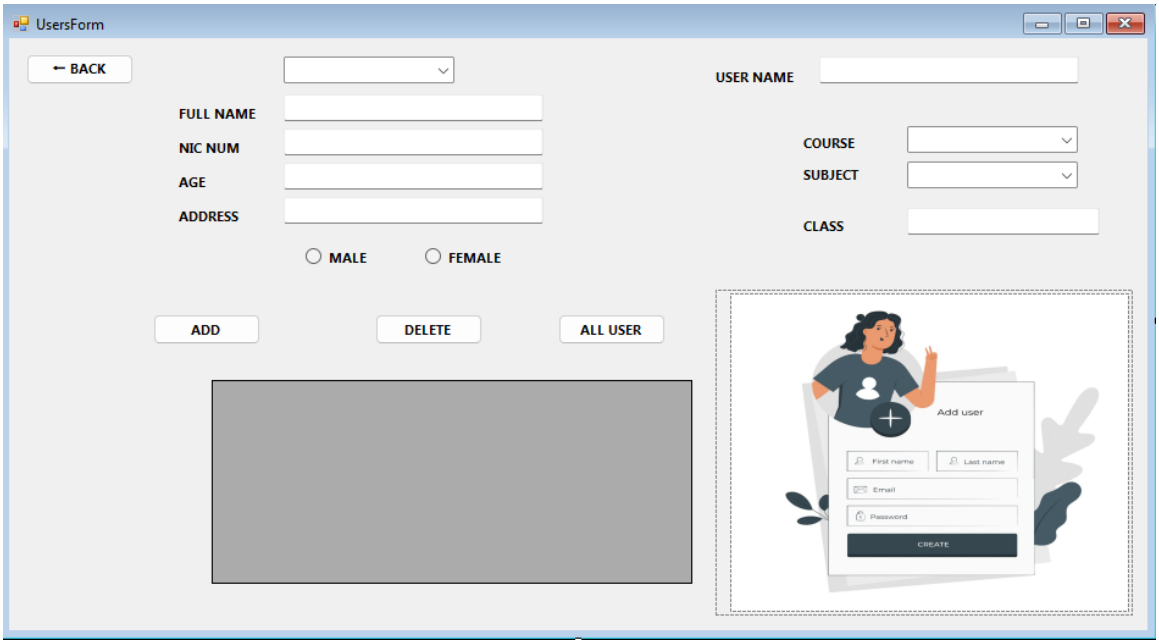


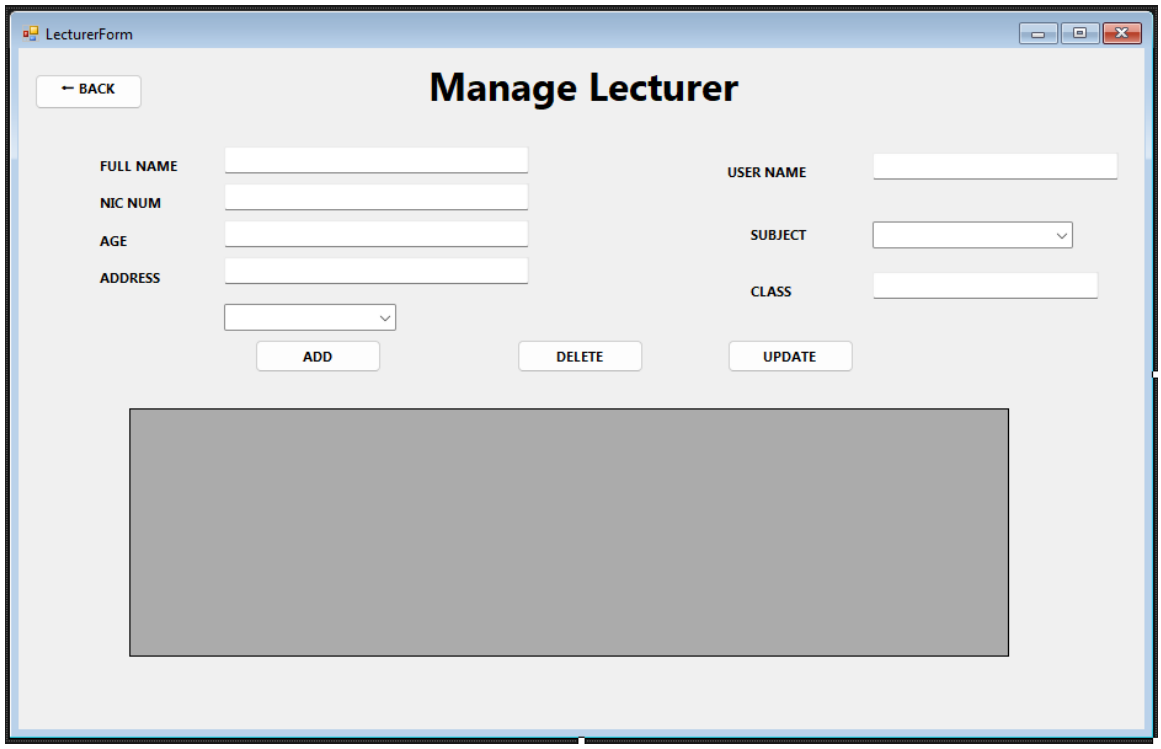
Note 6: User Management Structure

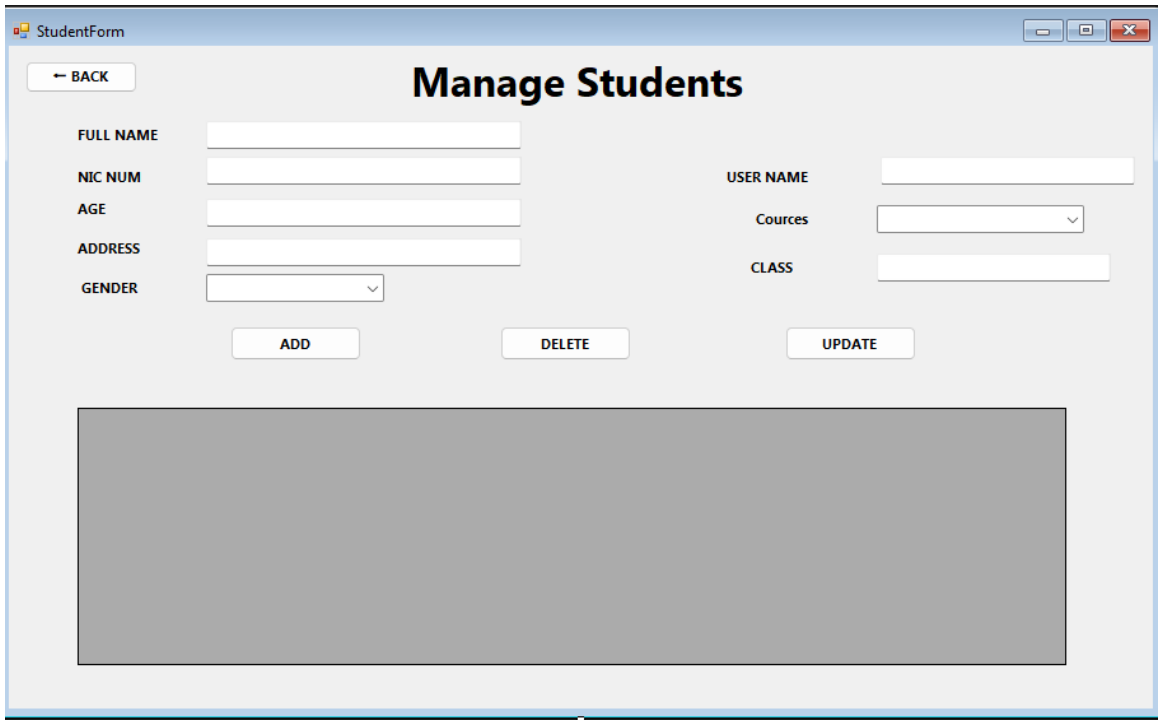
The application handles different user roles using separate forms to maintain clarity and modularity:

- **Manage Users Form:**
  - Used to create and manage **Admin** and **Staff** accounts.
  - Includes functionality to assign roles and credentials.
- **Manage Students Form:**
  - A separate dedicated form specifically for managing **Student** information.
  - Handles student details, course assignment, and subject mapping.
- **Manage Lecturers Form:**
  - A dedicated form to add and manage **Lecturer** data.
  - Also manages the link between lecturers and their assigned subjects.

This separation improves maintainability and ensures that each user role is managed in a focused, role-appropriate interface.




LecturerForm
← BACK
Manage Lecturer
FULL NAME
NIC NUM
AGE
ADDRESS
USER NAME
SUBJECT
CLASS
ADD
DELETE
UPDATE


StudentForm
← BACK
Manage Students
FULL NAME
NIC NUM
AGE
ADDRESS
GENDER
USER NAME
Cources
CLASS
ADD
DELETE
UPDATE