

## User Login Form and connection between Login Form and Users Table



### LoginForm

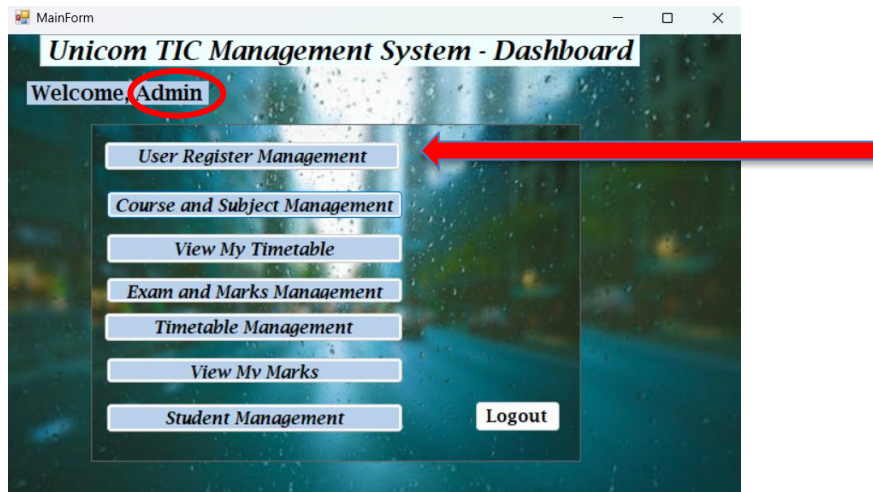
- Initially, the Admin can log in using the **role 'Admin'** with the **username 'admin'** and **password '123'**.
- When you click on the Username, Password, or Role text box, the placeholder text will disappear, allowing you to type or select the appropriate value.
- After entering the **Username, Password**, and selecting the **Role**, the system checks the data against the **Users** table in the database.

If all three fields match a record correctly:

- A "**Login successful**" message will be displayed.
- You will be redirected to the **Main Dashboard** based on your role.

On the Dashboard:

- A **welcome message** will be shown with your **logged-in username**.
- If the user is an **Admin**, all buttons and features will be visible and accessible.
- If the user is **Staff, Lecturer**, or **Student**, access will be limited based on their role, and unauthorized buttons will be **hidden**.



### User Registration Management (Admin Only):

- Only the **Admin** has access to the **User Registration Management** feature.
- The Admin can **add**, **update**, and **delete** user accounts by managing their **username**, **password**, and **role**.
- The **created time** and **last updated time** of each user are displayed in the **DataGridView**.

The screenshot shows the 'UserForm' window titled 'User Register Form'. The form has three input fields: 'User\_Name' with the value 'admin', 'Password' with the value '123', and 'Role' with a dropdown menu showing 'Admin'. Below the form are four buttons: 'ADD', 'UPDATE', 'DELETE', and 'Back To Login'. Below the buttons is a DataGridView with the following data:

	Userid	UserName	Password	Role	created_at	updated
▶	1	admin	123	Admin	2025-06-23 11:3...	2025-06-
	2	Danu	123	Student	2025-06-23 11:3...	
	4	Kathir	Hi123	Lecturer	2025-06-23 3:11 ...	2025-06-
	5	Mayooran	Hello 123	Staff	2025-06-23 3:28 ...	

```
am.cs | UserForm.cs [Design] | LoginController.cs | MainForm.cs | LoginForm.cs [Design] | MainForm.cs [Design] | UserForm.cs
UnicomTICManagementSystem.Controllers.LoginController | DeleteUser(int UserId)

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),
            //created_at = reader.GetDateTime(4), // Use GetDateTime
            updated_at = reader.GetDateTime(5) // Use GetDateTime
        };
    }
}

return null;

1 reference
public void UpdateUser(User user) // To update new user/existing user
{
    using (var conn = Dbconfig.GetConnection())
    {
        var command = new SQLiteCommand("UPDATE Users SET UserName = @Name, Password = @Password, Role = @Role, updated_at = @updatedAt WHERE UserId = @UserID", conn);
        command.Parameters.AddWithValue("@Name", user.UserName);
        command.Parameters.AddWithValue("@Password", user.Password);
        command.Parameters.AddWithValue("@Role", user.Role);
        command.Parameters.AddWithValue("@UserID", user.UserId);
        command.Parameters.AddWithValue("@updatedAt", user.updated_at);
        command.ExecuteNonQuery();
    }
}
```

- Displaying the '**Created At**' and '**Updated At**' timestamps is one of my best features. It allows the admin to easily track when each user was created or last modified, directly within the DataGridView after any creation or update action.

```
===== Below coding for User Login =====

string username = tUsername.Text.Trim();
string password = tPassword.Text.Trim();
string role = comboBox_Role.Text.Trim();

if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) || string.IsNullOrEmpty(role))
{
    MessageBox.Show("Please enter Username, Password, and select a Role.");
    return;
}

using (var conn_login = Dbconfig.GetConnection())
{
    //conn_login.Open();
    string query = @"SELECT COUNT(*) FROM Users
    WHERE UserName = @username AND Password = @password AND Role = @role";

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

        int userCount = Convert.ToInt32(cmd.ExecuteScalar()); // ExecuteScalar() -> Runs the query and returns one value only

        if (userCount > 0)
        {
            MessageBox.Show("Login Successful!");
            // open main form or dashboard here

            string selectedText = comboBox_Role.SelectedItem?.ToString(); // To pass the Role to MainForm welcom [Role] msg...

            if (!string.IsNullOrEmpty(selectedText))
            {
                MainForm form2 = new MainForm();
            }
        }
    }
}
```

- @username, @password, and @role are **parameters** — they are replaced with the values typed in your form (tUsername.Text, tPassword.Text, comboBox\_Role.Text).
- SQL checks how many records in the Users table **match exactly** those 3 values.
- Executes an SQL query that counts how many rows **match** the given username, password, and role.
  - If count is 1 (or more) → it means a match exists.
  - If count is 0 → invalid login.

Code	Meaning
ExecuteScalar()	Runs the query and returns <b>one value only</b>
userCount	Tells how many matching records exist
Convert.ToInt32( )	Converts the result (like 1) to an integer
if (userCount > 0)	Login success if <b>any match is found</b>

**Admin** *Course and Subject Management*

Course Name: DataBase-Technologies

ADD UPDAT DELET

	CourseId	CourseName
	1	Programming
	2	Web Design
▶	4	DataBase-Tech...

Course Name: DataBase-Technologies

Subject Name: MySQL

ADD UPDAT DELET

	SubjectId	SubjectName	CourseId
	1	C#	1
	2	Html	2
▶	3	MySQL	4
	4	Java	1
	5	Python	1
	6	CSS	2
	7	Java Script	2

## Course and Subject Access Control:

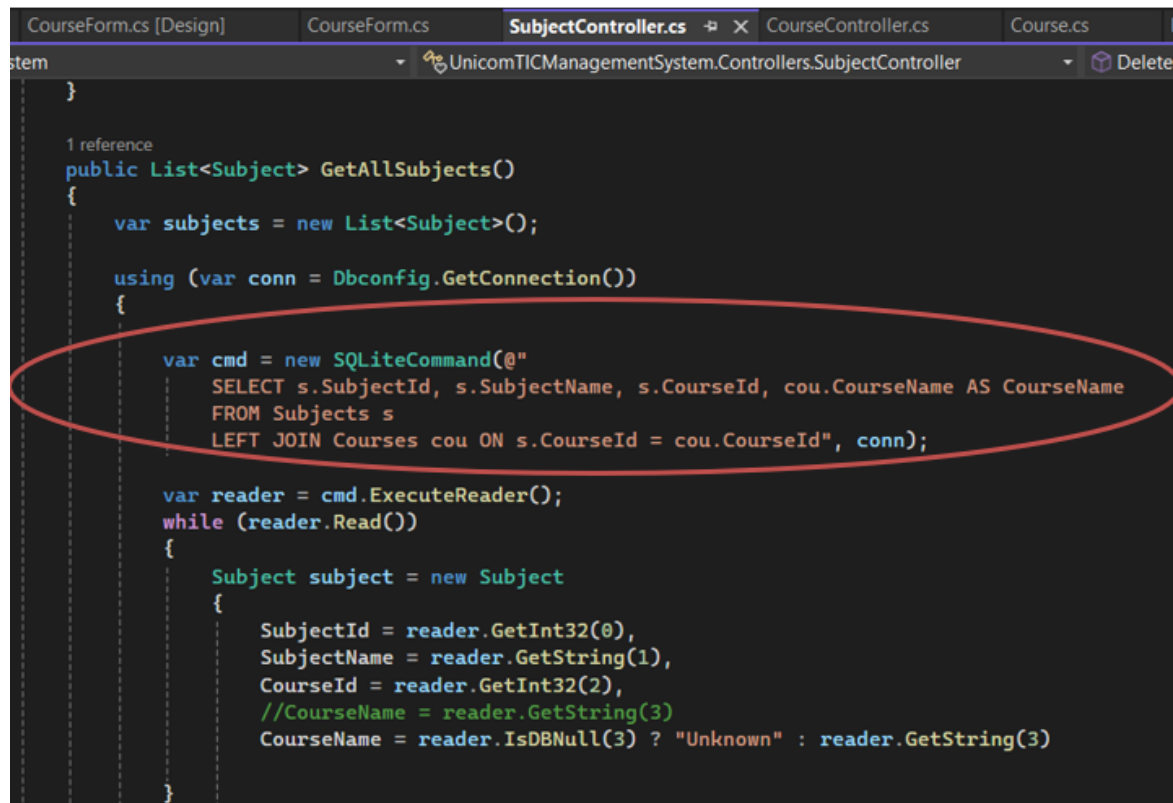
- **Admin** users have full access to the **Course** and **Subject** modules.

They can **add**, **update**, and **delete** courses and related subjects.

- **Staff, Lecturers, and Students** have **read-only access**.

They can **view the list of available courses and subjects** in the grid views.

They **cannot add, update, or delete** any course or subject. (**visible hidden by code**)



```
CourseForm.cs [Design] | CourseForm.cs | SubjectController.cs | CourseController.cs | Course.cs
UnicomTICManagementSystem.Controllers.SubjectController | Delete

}

1 reference
public List<Subject> GetAllSubjects()
{
    var subjects = new List<Subject>();

    using (var conn = Dbconfig.GetConnection())
    {
        var cmd = new SQLiteCommand(@"
        SELECT s.SubjectId, s.SubjectName, s.CourseId, cou.CourseName AS CourseName
        FROM Subjects s
        LEFT JOIN Courses cou ON s.CourseId = cou.CourseId", conn);

        var reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            Subject subject = new Subject
            {
                SubjectId = reader.GetInt32(0),
                SubjectName = reader.GetString(1),
                CourseId = reader.GetInt32(2),
                //CourseName = reader.GetString(3)
                CourseName = reader.IsDBNull(3) ? "Unknown" : reader.GetString(3)
            }
        }
    }
}
```

I used a **LEFT JOIN** to display the **Course Name** in the subject grid view by linking the **CourseId** foreign key from the Subjects table with the Courses table.

Other work is in progress...