Unicom TIC Management System - Project Submission Report

# Project Overview

## Key Features Implemented:

* Login System with role-based access for Admin, Staff, Students, and Lecturers.
* Course and Subject Management module.
* Student Management with course association.
* Exam and Marks Management with entry and viewing privileges by role.
* Timetable Management including Computer Lab and Lecture Hall allocation.
* SQLite database with relationships between tables.
* Role-based dashboards that restrict access to appropriate features.
* Error messages and input validation included.
* Simple and intuitive WinForms user interface.

## Technologies Used:

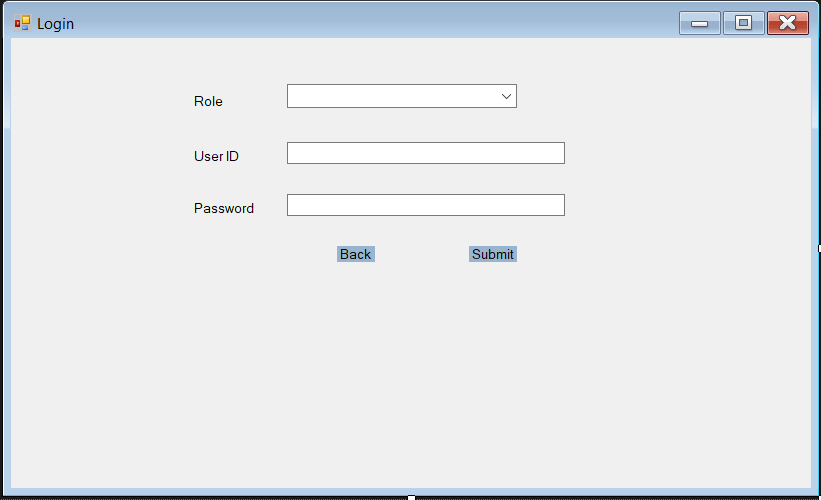
* Programming Language: C#
* Framework: WinForms (.NET Framework)
* Database: SQLite using System.Data.SQLite
* Architecture: MVC (Model–View–Controller)
* IDE: Visual Studio
* Design Patterns: MVC
* UI Elements: Buttons, ComboBoxes, DataGridViews, TextBoxes, Labels

## Challenges Faced and Solutions:

* Role-Based Dashboard Visibility: Solved by checking user role after login and showing/hiding buttons accordingly.
* Room Allocation with Combo Box: Created a separate table for Rooms with room type filtering, and populated Combo Box dynamically.
* Login Validation: Added a simple user validation system querying the Users table and managing sessions based on roles.
* Database Table Creation: Implemented DatabaseManager.cs to auto-create tables if they don’t exist.
* Navigation Between Forms: Used controller logic to open/close forms based on user role without error

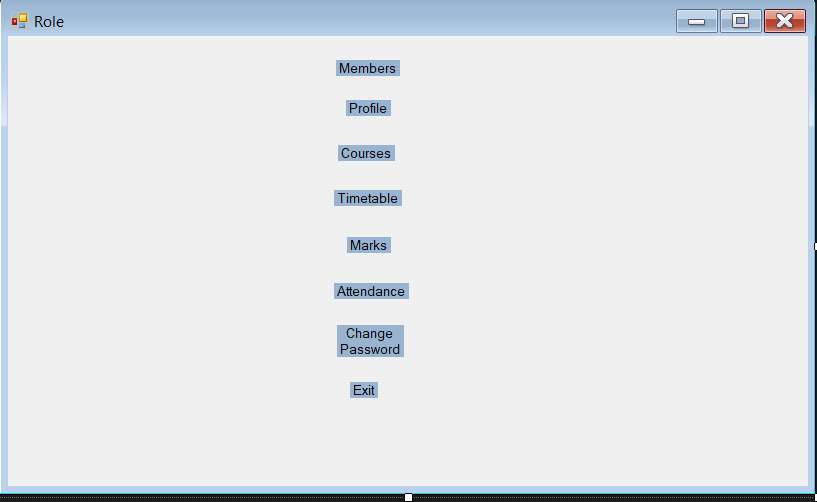
## Login Design

* This is the initial view for all users.
* If the User ID, Password, and Role are correct, you can log in.
* When the user clicks Exit, the application navigates back to the login screen, allowing another user to log in.



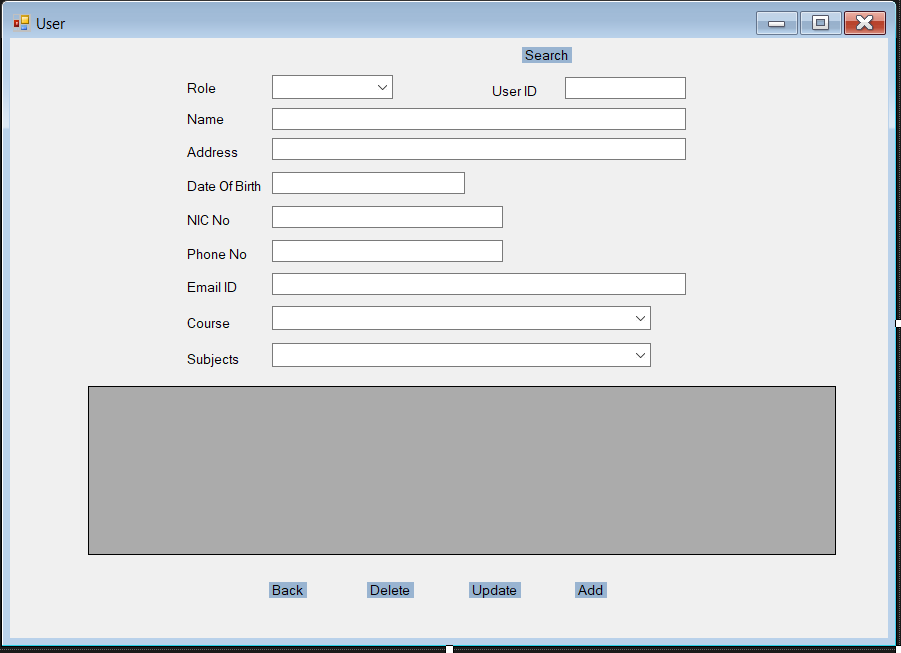
Role.cs – Disign

* Upon successful login, the user is directed to this page.
* The Members option is visible for Admin only.



User.cs –Design

* When the Admin selects the Members option from the Role view, the User view is opened automatically.
* When the application opens, the User ID field and its input box are hidden.
* An admin can enter all required information through the 'Add New User' feature.
* If the Admin doesn't receive complete details, they can still add a new user by entering the Role, Name, Address, and Date of Birth. These four fields are mandatory for user creation. The User can update the remaining details later with the help of the Admin.
* When a new user is added, a UT-style User ID is automatically generated, and the default password is set to 'admin@123'.
* To update or delete a specific user, the admin can navigate through the grid view and select the desired user."
* If the admin has difficulty selecting a user, they can click the Search button. This will make the User ID field and its input box visible. The admin can then enter the User ID to retrieve the user's details.
* After that, the admin can update the user's details or delete the user.
* Once the update or deletion is complete, the User ID field and its input box will be hidden.
* When the admin clicks the Back button, the system navigates back to the Role view.



Profile

# How the App Works

Login System: Authenticates users based on their role (Admin, Staff, Student, Lecturer).

Role Dashboards: Different dashboards are shown depending on the user role.

Data Operations: Add/Edit/Delete supported for Admin; restricted view-only for other roles.

Rooms & Timetable: Admin allocates labs/halls while scheduling; others can only view.

Exam & Marks: Staff and Lecturers can update marks; students can only view their own.

# Features

Filter (Courses, Subjects, Students, Exams, Rooms in dropdown list)

Error Messages (e.g., “Please select a room”, “Login failed”)

Input Validation (e.g., Score must be 0–100)

Descriptive Mappings for Viewing Data:

* Student name | Course name
* Exam name | Subject name
* Timetable – Subject name | Room name | Time slot
* Room name | Room type
* Subject name | Course name

# Conclusion

This project helped in learning:

* How to implement an MVC architecture in a desktop application.
* SQLite CRUD operations in C#.
* Role-based access control with a simple UI.
* Structuring a school management system from scratch.