

# UNICOM TIC

C#

## Unicom Management System Report

**Name: - D.Sujivan**  
**UT No: - UT010504**

## Table of Contents

|   |    |
|---|----|
| 1. Introduction.....                    | 1  |
| 2. Objective .....                      | 2  |
| 3. System Overview .....                | 2  |
| 3.1 Functional Requirement .....        | 2  |
| 3.2 Non-Functional Requirement .....    | 3  |
| 4. Technologies Used.....               | 4  |
| 5. Code Highlights.....                 | 5  |
| 6. Challenges Faced and Solutions ..... | 9  |
| 7. Default Credentials.....             | 11 |
| 8. Conclusion.....                      | 11 |

# 1. Introduction

This project is a Windows-based desktop application developed primarily for **UNICOM TIC**, designed to streamline and automate the management of all academic and non-academic operations within the institution. The system is implemented using **C# Windows Forms** with an **SQLite database**, providing a lightweight, fast, and cost-effective solution.

The **UNICOM TIC Academic Management System** offers a user-friendly interface and comprehensive features to simplify daily administrative tasks. The application supports multiple user roles including **Admin**, **Student**, **Lecturer**, and **Staff**, ensuring that each user has role-specific access to the system's features. It significantly reduces the dependency on manual processes, enhances data integrity, and improves access to real-time academic information.

## Key Features of the Application:

- **User Management:** Admins can register and manage all users including students, lecturers, and staff. Each user has a unique role with predefined permissions, ensuring controlled and secure access to the system.
- **Student and Staff Management:** The system stores detailed profiles for all academic and non-academic personnel including personal information, contact details, and credentials.
- **Course and Subject Management:** Admins can create, update, and delete courses and assign relevant subjects, providing a structured academic flow.
- **Enrollment System:** Students can be enrolled in specific courses, and their enrollment records are maintained systematically in the database.
- **Timetable Management:** The application allows the creation of timetables by assigning subjects to rooms with specified dates and times. These schedules are visible to both students and lecturers, minimizing communication gaps.
- **Exam and Marks Management:** Lecturers can create exams and record students' marks. The system automatically links marks with students and exams, and students can log in to view their own performance. Role-based access ensures that **students can only view their own marks and cannot manipulate** exam or mark data.
- **Role-Based Access Control:** A major feature of the system is its strict access control based on user roles. For example, students do not have access to create or modify academic data but can view their own academic records. Admins and staff, however, have full control over CRUD operations across the system.
- **Clean and Simple UI:** The application features a clean, responsive, and easy-to-navigate user interface designed using Windows Forms, making it accessible for users with minimal technical knowledge.

## 2. Objective

The primary objective of this system is to **automate and streamline academic operations** at UNICOM TIC, replacing manual processes with a digital platform. It aims to:

- **Centralize student and staff information** for easier management and secure storage.
- **Enhance communication and access** to academic resources for students and lecturers.
- **Ensure data security and integrity** through role-based access controls.
- **Minimize administrative workload** and reduce the use of physical paperwork.

Overall, the system provides an efficient, secure, and scalable solution for managing educational activities. Its modular design also allows future enhancements like attendance tracking, notifications, and online platform integration.

## 3. System Overview

This section provides a general understanding of the system's architecture and how it functions. The system is designed as a centralized academic management solution tailored for UNICOM TIC, aiming to simplify and digitize core academic operations.

### 3.1 Functional Requirement

- Users should be able to log in using a valid username and password.
- The system should support role-based access (Admin, Staff, Lecturer, Student) and display features accordingly.
- Admins should be able to manage (add, update, delete) users, courses, subjects, rooms, and timetable entries.
- Admins should be able to view and manage all student and staff records.
- Admins should be able to assign students to courses.
- Lecturers should be able to create exams and enter student marks.
- Lecturers should be able to view assigned subjects and relevant student performance.
- Staff should have limited access to manage non-academic data or view academic records as required.
- Students should be able to view only their own marks and timetable.
- Students should not have access to create or edit any data.
- The system should allow CRUD operations for courses, subjects, rooms, exams, and marks (by authorized roles).
- The system should validate all form inputs before performing database operations.
- The timetable should allow assignment of subject, room, date, and time in a unified time slot.
- The application should securely store all data in a local SQLite database.
- Users should be able to update their personal profiles, except for restricted fields like NIC and DOB.
- Students should receive a view-only experience with limited interface controls enabled.
- The system should display data dynamically in DataGridViews after every operation.
- The system should prevent duplication of entries (e.g., same user, subject, or room).
- Users should be able to log out and return to the login screen securely.

## 3.2 Non-Functional Requirement

- The system should respond to user actions (such as logging in, saving records, or loading data) in less than **2 seconds** under normal usage conditions.
- The application should follow **role-based access control**, ensuring users can only access features relevant to their role.
- The system should be able to handle **concurrent access** from multiple users without performance degradation.
- The application should provide **clear error messages** and feedback for invalid actions or failed operations.
- The system should be developed using **C# Windows Forms** with a **user-friendly and intuitive interface** to ensure ease of use for both technical and non-technical users.
- The database (SQLite) must maintain **data integrity**, preventing partial writes and ensuring rollback on failure.
- The application should be **reliable**, with minimal bugs or crashes during long periods of usage.
- The system should require **minimal hardware** resources and be able to run on standard Windows PCs used within UNICOM TIC.
- The system should support **easy backup and recovery** of the SQLite database.
- The architecture should be **modular**, allowing for future expansion (e.g., attendance tracking, online learning integration).
- The UI should be responsive and compatible with **varied screen resolutions** and display sizes.
- The application should support **offline usage** without needing a constant internet connection.
- System updates and feature changes should be **easy to deploy** without affecting existing data or user preferences.

## 4. Technologies Used

| Technology                      | Purpose  |
|---------------------------------|--|
| <b>C# (.NET Framework)</b>      | Used to develop the core application logic and user interface via Windows Forms (WinForms).                                  |
| <b>Windows Forms (WinForms)</b> | Provides the graphical user interface for the desktop application.   |
| <b>SQLite</b>                   | Lightweight, embedded relational database used to store all system data such as users, students, exams, marks, courses, etc. |
| <b>SQL (SQLite Queries)</b>     | Used for CRUD operations (Create, Read, Update, Delete) and data manipulation within the database.                           |
| <b>MVC Architecture</b>         | Maintains separation of concerns in the application by organizing code into Model, View, and Controller layers.              |
| <b>GDI+ / System.Drawing</b>    | Used to load and display profile pictures within the UI.   |
| <b>.NET System Libraries</b>    | Provides support for essential functionality such as file I/O, memory management, encryption, data validation, etc.          |
| <b>Visual Studio</b>            | Integrated Development Environment (IDE) used to design, code, debug, and manage the application.                            |

## 5. Code Highlights


I have shared the best coding practices that I have implemented.

- In my application, only authorized individuals can register. First, the user must submit their details to UNICOM, where an admin manually enters the information into the pre\_register table. When the user opens the app, they enter their NIC number and click the 'Check' button. If their details are found, the form will automatically display the information entered by the admin. The user can then verify and correct any incorrect details, enter their desired username and password, and successfully complete the registration process.
- Note: - You cannot change your Role or NIC number during registration in the app. Also, note that you cannot use a username that has already been taken

```
private void checkbtn_Click(object sender, EventArgs e)
{
    string nic = nictxt.Text.Trim();

    if (!string.IsNullOrEmpty(nic))
    {
        var adminData = LoginController.GetAdminApprovalByNIC(nic);
        if (adminData != null)
        {
            fullnametxt.Text = adminData.FullName;
            phonetxt.Text = adminData.phone.ToString();
            emailtxt.Text = adminData.Email;
            homeaddresstxt.Text = adminData.Address;
            dobpicker.Text = adminData.DOB;
            roletxt.Text = adminData.Role;
            roletxt.Enabled = false;
        }
        else
        {
            MessageBox.Show("NIC not found or already signed up.");
        }
    }
    else
    {
        MessageBox.Show("Please enter a valid NIC.");
    }
}
```

Figure 1-Coding for check option in Register page



## Unicom TIC Management System

### Create a new account

It's quick and easy

NIC Number

Full Name

Phone Number

Email Address

Home Address

Date of birth

Profile

Roll


User Name

Password


Alredy have an account?

Figure 2-UI for check option in Register page

- In my application, when a user logs in, their profile details are sent to the main dashboard. Their profile picture is displayed along with a greeting message, 'Hello, <name>'. In the Settings page, the user can update their personal details and change their profile picture as well.



## Unicom TIC Management System



Hello, Dayananda

User

Course Subject

Student

Exams Marks

Time Table

Setting

Logout

### SETTING

Full Name

NIC Number

Phone Number

Email Address

Home Address

DOB

Role

User Name

Password

Profile Picture




Figure 3-Profile picture & profile setting UI



```

127     public static UserProfile GetUserProfile(string username)
128     {
129         using (var conn = DatabaseConnection.GetConnection())
130         {
131             string query = "SELECT * FROM Users WHERE Username = @Username";
132             using (var cmd = new SQLiteCommand(query, conn))
133             {
134                 cmd.Parameters.AddWithValue("@Username", username);
135                 using (var reader = cmd.ExecuteReader())
136                 {
137                     if (reader.Read())
138                     {
139                         string role = reader["Role"].ToString();
140                         int userId = Convert.ToInt32(reader["UserID"]);
141                         string password = reader["Password"].ToString();
142
143                         return GetProfileByRole(userId, username, password, role, conn);
144                     }
145                 }
146             }
147         }
148         return null;
149     }
150

```

Figure 5-Profile picture & profile setting coding 01

```

1 reference
private static UserProfile GetProfileByRole(int userId, string username, string password, string role, S
{
    string roleTable = null;
    string loweredRole = role.ToLower();

    if (loweredRole == "student")
        roleTable = "Students";
    else if (loweredRole == "admin")
        roleTable = "Admins";
    else if (loweredRole == "staff")
        roleTable = "Staff";
    else if (loweredRole == "lecturer")
        roleTable = "Lecturers";
    if (roleTable == null) return null;

    string query = $"SELECT * FROM {roleTable} WHERE UserID = @UserID";
    using (var cmd = new SQLiteCommand(query, conn))
    {
        cmd.Parameters.AddWithValue("@UserID", userId);
        using (var reader = cmd.ExecuteReader())
        {
            if (reader.Read())
            {
                return new UserProfile
                {
                    UserID = userId,
                    FullName = reader["FullName"].ToString(),
                    NIC = reader["NIC"].ToString(),
                    Phone = Convert.ToInt32(reader["Phone"]),
                    Email = reader["Email"].ToString(),
                    Address = reader["Address"].ToString(),
                    DOB = reader["DOB"].ToString(),

```

Figure 4-Profile picture & profile setting coding 02

- In my application, I have implemented role-based access.

```
{
    private readonly MainDashboardForm _mainDashboard;
    private readonly string _userRole;
    private int selectedCourseId = -1;

    1 reference
    public CourseForm(MainDashboardForm mainDashboard, UserProfile user)
    {
        InitializeComponent();
        _mainDashboard = mainDashboard;
        _userRole = user.Role?.ToLower();

        LoadCourses();
        dgv.CellClick += dgv_CellClick;
        ApplyRoleRestrictions();
    }

    1 reference
    private void ApplyRoleRestrictions()
    {
        if (_userRole != "admin")
        {
            addbtn.Enabled = false;
            updatebtn.Enabled = false;
            deletebtn.Enabled = false;
            clearbtn.Enabled = false;
            coursetxt.Enabled = false;
        }
    }

    1 reference
}
```

Figure 6-Role base access code

- The coding is implemented using the MVC architecture and OOP concepts, which helps make the application easier to understand and more secure

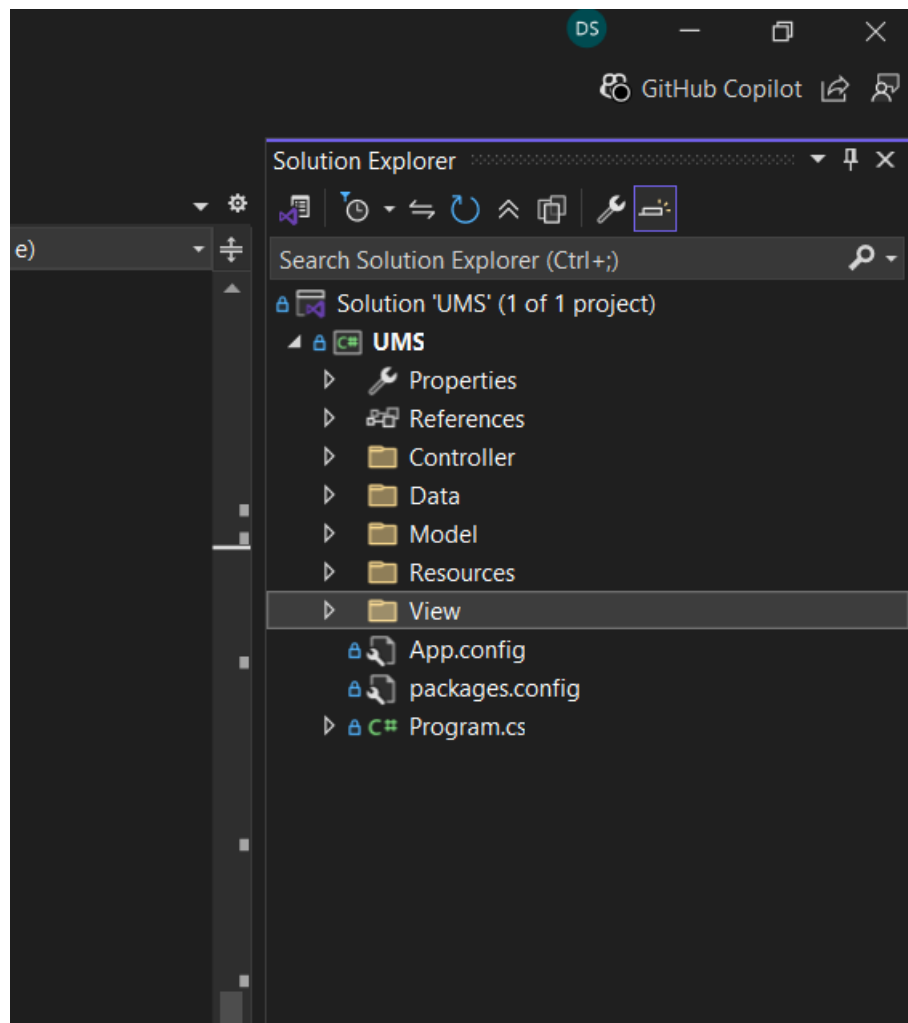


Figure 7-MVC architecture

## 6. Challenges Faced and Solutions

- **Challenge: Handling Role-Based Access Control**  
Initially, implementing separate functionalities for different user roles (Admin, Staff, Lecturer, Student) was complex, especially ensuring that students could only access their own data.  
**Solution:** Implemented strict role-based UI controls and query filters based on the logged-in user's role and ID. This ensured students could only view their own marks, while admins and lecturers had full CRUD access.
- **Challenge: User Registration with Pre-Approval**  
Preventing unauthorized registrations while allowing flexibility in user detail verification posed a unique challenge.  
**Solution:** Designed a `pre_register` table where admins manually pre-approve users. During registration, NIC is checked first, and only if found and approved, users can continue to create accounts with a unique username.
- **Challenge: Displaying Profile Picture and Personalization**  
Loading and displaying user profile pictures dynamically in the dashboard and settings was initially difficult.  
**Solution:** Stored profile pictures as binary data in the database and used `MemoryStream` to convert and display them within `PictureBox` controls.
- **Challenge: Managing Data Integrity Across Multiple Tables**  
Ensuring synchronization between Users and role-specific tables (like Students, Admins, etc.) during updates and deletions required careful design.  
**Solution:** Used centralized methods in the controller layer to perform transactions that update both general and role-specific data consistently.
- **Challenge: Preventing Duplicate Usernames During Registration**  
Users could accidentally try to register with already used usernames, which caused database errors.  
**Solution:** Added real-time validation to check for username uniqueness before allowing registration to proceed.
- **Challenge: Implementing Timetable Integration**  
Linking subjects, rooms, and scheduling into one coherent timetable format was initially tricky.  
**Solution:** Merged Date and Time into a single `TimeSlot` column and used `ComboBoxes` to select subjects and rooms, simplifying user input and ensuring consistency.
- **Challenge: Ensuring a User-Friendly Interface**  
Making the WinForms interface intuitive and clean for all user roles was challenging.  
**Solution:** Followed consistent UI patterns, disabled irrelevant controls for students, and used clear feedback messages to guide user interactions.

## 7. Default Credentials

Initially, user details must be manually inserted into the `pre_register` table using DB Browser for SQLite. To register, the user must enter the NIC that was previously added to the `pre_register` table. By default, the login credentials are: **Username – admin, Password – admin.**

## 8. Conclusion

This application was developed to streamline and digitize academic operations within **UNICOM TIC**, providing a centralized platform for managing students, academic staff, non-academic staff, courses, exams, marks, rooms, and timetables. By leveraging a **C# Windows Forms** interface with an **MVC architecture** and **SQLite database**, the system offers a user-friendly, secure, and efficient experience for users of different roles.

Role-based access control ensures that sensitive data and operations are only accessible to authorized users, while automated workflows such as registration through pre-approved NICs and real-time data interaction greatly reduce manual work and errors.

The system is flexible and scalable, and it lays the foundation for future enhancements like attendance tracking, report generation, push notifications, and online learning integration. Overall, this project has successfully achieved its objectives by delivering a robust solution that simplifies and enhances the educational management process at UNICOM TIC.