Unicom Tic Management System

This is a desktop application built using C# 7.3 and WinForms to manage the academic activities of a Unicom Tic. It includes modules such as Courses, Subjects, Students, Exams, Marks, Timetables, and Attendance.

- **How to Run**

1. Open the solution file (`UnicomTic_ManagementSystem.sln`) in Visual Studio.

2. Make sure the SQLite file (`unicomtic.db`) is in the output directory do not use System.Data.SQLite.Core, Use **"System.Data.SQLite"**

3. Default Login Credentials

User Name: Admin

Password : 6789

1. Project Overview

- Key Features Implemented

- User login system with three roles: **Admin**, **Lecturer, Student**

- Role-based **Main Dashboard** (MainForm) with access control

- Full CRUD functionality for:

- Courses

- Subjects

- Students

- Exams

- Marks

- Timetables

- Attendance Module

- Admin: View, edit, and delete attendance

- Lecturer: Mark attendance by subject/date

- Student: View own attendance records

- SQLite integration with OOP concepts (Encapsulation, Inheritance)

- Consistent UI design with error messages and field validation

- Technologies Used

- Language: C# 7.3

- Framework: Windows Forms (WinForms)

- Database: SQLite (local file-based)

- Architecture: MVC (Model–View–Controller)

- IDE: Visual Studio

- Challenges Faced & Solutions

-SQLite Database Locked Error

- **Problem:** Frequent error: The database file is locked during insert/update.
- **Solution:** Fixed by **properly disposing** connections using conn.Dispose() or using blocks in DatabaseManager.
- **Learning:** Learned about **resource management** and preventing locks in local database apps.

- Incompatible Syntax (C# 8+ in C# 7.3)

- **Problem:** Errors due to using newer C# features like ??=, simplified async lambdas, etc.
- **Solution:** Rewrote all modern syntax (e.g., ??=) into valid C# 7.3 alternatives.
- **Learning:** Got deeper understanding of **language version compatibility**.

- Passing Data Between Forms

- **Problem:** Couldn't pass student ID from login to attendance viewer.
- **Solution:** Passed the logged-in User object using form constructors.
- **Learning:** Learned to **share state between forms** without using static variables.

- Attendance Duplication

- **Problem:** Students were being marked present multiple times for the same subject and date.
- **Solution:** Used a **UNIQUE(StudentID, SubjectID, Date)** constraint in the table to prevent duplicate entries.
- **Learning:** Learned to design **safe database schemas** that prevent logic errors.

-DataGridView Not Refreshing:

- **Problem:** After Add/Update/Delete, the grid wasn't showing the latest data.
- **Solution:** Called LoadData () after operations, and set DataSource = null before rebinding.
- **Learning:** Understood how WinForms DataGridView binds data and how to refresh it properly.

2. Code Samples

Below are screenshots of the most important and well-structured parts of the code.

1. Login Controller and Validation

```
1 reference
private async void btnLogin_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text.Trim();
    string password = txtPassword.Text;
    var user = await controller.Login(username, password);

    if (user != null)
    {
        MessageBox.Show("Welcome, " + user.Username + " (" + user.Role + ")");
        this.Hide();
        MainForm main = new MainForm(user);
        main.Show();
    }
}

1 reference
private async void btnRegister_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text.Trim();
    string password = txtPassword.Text;
    string role = cmbRole.SelectedItem?.ToString();

    if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password) || string.IsNullOrWhiteSpace(role))
    {
        MessageBox.Show("All fields are required for registration.");
        return;
    }

    await controller.Register(username, password, role);
}

1 reference
private void btnExit_Click(object sender, EventArgs e)
```

(Role-based login and redirection to MainForm)

2. Course Management

```
1 reference
public async Task AddCourse(string name)
{
    try
    {
        await service.AddCourse(name);
        MessageBox.Show("Course added successfully.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}

1 reference
public async Task<List<Course>> GetCourses()
{
    return await service.GetAllCourses();
}

1 reference
public async Task UpdateCourse(Course course)
{
    try
    {
        await service.UpdateCourse(course);
        MessageBox.Show("Course updated successfully.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}

1 reference
public async Task DeleteCourse(int courseID)
{
    try
    {
        await service.DeleteCourse(courseID);
        MessageBox.Show("Course deleted successfully.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
```

```
2 references
internal class CourseService
{
    private readonly DatabaseManager db = DatabaseManager.Instance;

    1 reference
    public async Task AddCourse(string courseName)
    {
        if (string.IsNullOrWhiteSpace(courseName))
        {
            throw new ArgumentException("Course name cannot be empty.");
        }

        var course = new Course { CourseName = courseName };
        await db.AddCourseAsync(course);
    }

    1 reference
    public async Task<List<Course>> GetAllCourses()
    {
        return await db.GetCoursesAsync();
    }

    1 reference
    public async Task UpdateCourse(Course course)
    {
        if (string.IsNullOrWhiteSpace(course.CourseName))
        {
            throw new ArgumentException("Course name cannot be empty.");
        }

        await db.UpdateCourseAsync(course);
    }

    1 reference
    public async Task DeleteCourse(int courseID)
    {
        if (courseID <= 0)
        {
            throw new ArgumentException("Invalid Course ID.");
        }

        await db.DeleteCourseAsync(courseID);
    }
}
```

(CourseController handles adding and listing courses using async logic)

3. Student Attendance

```
2 references
public class AttendanceController
{
    private readonly AttendanceService service = new AttendanceService();

    1 reference
    public async Task AddAttendance(Attendance a)
    {
        try { await service.Add(a); MessageBox.Show("Attendance added."); }
        catch (Exception ex) { MessageBox.Show("Error: " + ex.Message); }
    }

    1 reference
    public async Task<List<Attendance>> GetAttendance() => await service.GetAll();

    1 reference
    public async Task UpdateAttendance(Attendance a)
    {
        try { await service.Update(a); MessageBox.Show("Attendance updated."); }
        catch (Exception ex) { MessageBox.Show("Error: " + ex.Message); }
    }

    1 reference
    public async Task DeleteAttendance(int id)
    {
        try { await service.Delete(id); MessageBox.Show("Attendance deleted."); }
        catch (Exception ex) { MessageBox.Show("Error: " + ex.Message); }
    }
}
```

```
5 references
public partial class StudentAttendanceForm : Form
{
    private readonly AttendanceController controller = new AttendanceController();
    private readonly DatabaseManager db = DatabaseManager.Instance;

    2 references
    public StudentAttendanceForm() => InitializeComponent();

    1 reference
    private async void StudentAttendanceForm_Load(object sender, EventArgs e)
    {
        cmbStatus.Items.AddRange(new string[] { "Present", "Absent", "Late", "Excused" });

        cmbStudent.DataSource = await db.GetStudentsAsync();
        cmbStudent.DisplayMember = "FullName";
        cmbStudent.ValueMember = "StudentID";

        cmbSubject.DataSource = await db.GetSubjectsAsync();
        cmbSubject.DisplayMember = "SubjectName";
        cmbSubject.ValueMember = "SubjectID";

        await LoadAttendance();
    }

    4 references
    private async Task LoadAttendance()
    {
        var list = await controller.GetAttendance();
        foreach (var a in list)
        {
            var student = await db.GetStudentByIdAsync(a.StudentID);
            var subject = await db.GetSubjectByIdAsync(a.SubjectID);
            a.StudentName = student.FullName;
            a.SubjectName = subject.SubjectName;
        }
        dgvAttendance.DataSource = null;
        dgvAttendance.DataSource = list;
    }

    1 reference
    private async void btnAdd_Click(object sender, EventArgs e)
    {
        if (cmbStudent.SelectedIndex == -1 || cmbSubject.SelectedIndex == -1 || string.IsNullOrEmpty(cmbStatus.Text))
        {
            MessageBox.Show("Please fill all fields.");
            return;
        }
```

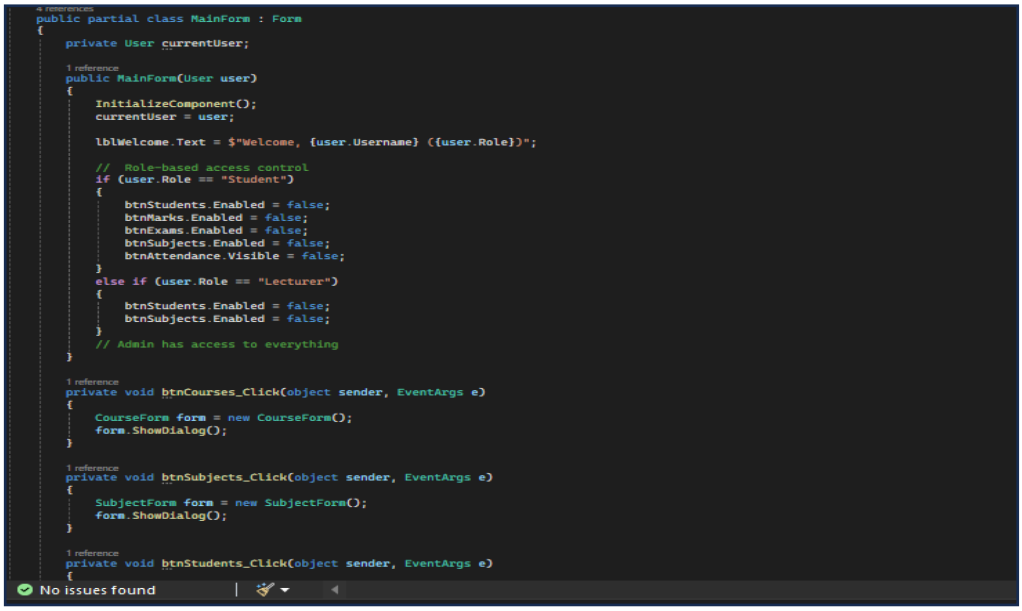
(Lecturer marks attendance per subject and date with status options)

4. Database Connection and Queries

```
    }

    1 reference
    public async Task AddAttendanceAsync(Attendance a)
    {
        var conn = new SQLiteConnection(connectionString);
        await conn.OpenAsync();
        var cmd = new SQLiteCommand("INSERT INTO Attendance (StudentID, SubjectID, Date, Status) VALUES (@StudentID, @SubjectID, @Date, @Status)", conn);
        cmd.Parameters.AddWithValue("@StudentID", a.StudentID);
        cmd.Parameters.AddWithValue("@SubjectID", a.SubjectID);
        cmd.Parameters.AddWithValue("@Date", a.Date);
        cmd.Parameters.AddWithValue("@Status", a.Status);
        await cmd.ExecuteNonQueryAsync();
        conn.Dispose();
    }

    1 reference
    public async Task<Student> GetStudentByIdAsync(int id)
    {
        var conn = new SQLiteConnection(connectionString);
        await conn.OpenAsync();
        var cmd = new SQLiteCommand("SELECT * FROM Students WHERE StudentID = @id", conn);
        cmd.Parameters.AddWithValue("@id", id);
        var reader = await cmd.ExecuteReaderAsync();

        Student s = null;
        if (await reader.ReadAsync())
        {
            s = new Student
            {
                StudentID = Convert.ToInt32(reader["StudentID"]),
                FullName = reader["FullName"].ToString(),
                Gender = reader["Gender"].ToString(),
                DOB = reader["DOB"].ToString(),
                CourseID = Convert.ToInt32(reader["CourseID"]),
                SubjectID = Convert.ToInt32(reader["SubjectID"])
            };
        }
        conn.Dispose();
        return s;
    }

    2 references
    public async Task<List<Attendance>> GetAllAttendanceAsync()
    {
```

(Centralized SQLite database manager with safe `using` statements)

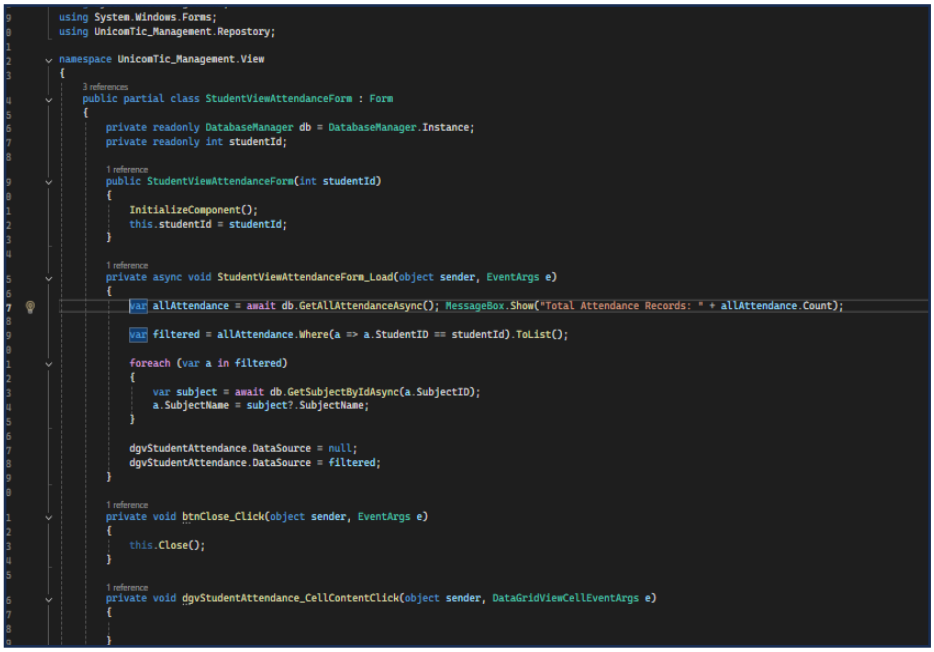
5. Role-Based Main Dashboard



```
4 references
public partial class MainForm : Form
{
    private User currentUser;

    1 reference
    public MainForm(User user)
    {
        InitializeComponent();
        currentUser = user;

        lblWelcome.Text = $"Welcome, {user.Username} ({user.Role})";

        //  Role-based access control
        if (user.Role == "Student")
        {
            btnStudents.Enabled = false;
            btnMarks.Enabled = false;
            btnExams.Enabled = false;
            btnSubjects.Enabled = false;
            btnAttendance.Visible = false;
        }
        else if (user.Role == "Lecturer")
        {
            btnStudents.Enabled = false;
            btnSubjects.Enabled = false;
        }
        // Admin has access to everything
    }

    1 reference
    private void btnCourses_Click(object sender, EventArgs e)
    {
        CourseForm form = new CourseForm();
        form.ShowDialog();
    }

    1 reference
    private void btnSubjects_Click(object sender, EventArgs e)
    {
        SubjectForm form = new SubjectForm();
        form.ShowDialog();
    }

    1 reference
    private void btnStudents_Click(object sender, EventArgs e)
    {
No issues found
```

(MainForm shows/hides buttons depending on user role (Admin, Lecturer, Student))

6. Student Attendance Viewer



```
using System.Windows.Forms;
using UnicomTic_Management.Repostory;

namespace UnicomTic_Management.View
{
    3 references
    public partial class StudentViewAttendanceForm : Form
    {
        private readonly DatabaseManager db = DatabaseManager.Instance;
        private readonly int studentId;

        1 reference
        public StudentViewAttendanceForm(int studentId)
        {
            InitializeComponent();
            this.studentId = studentId;
        }

        1 reference
        private async void StudentViewAttendanceForm_Load(object sender, EventArgs e)
        {
            var allAttendance = await db.GetAllAttendanceAsync(); MessageBox.Show("Total Attendance Records: " + allAttendance.Count);

            var filtered = allAttendance.Where(a => a.StudentID == studentId).ToList();

            foreach (var a in filtered)
            {
                var subject = await db.GetSubjectByIdAsync(a.SubjectID);
                a.SubjectName = subject?.SubjectName;
            }

            dgvStudentAttendance.DataSource = null;
            dgvStudentAttendance.DataSource = filtered;
        }

        1 reference
        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        1 reference
        private void dgvStudentAttendance_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {

        }
```

(Students can view their attendance filtered by subject and date)

3.Author

Mohammed Harees Fathima Nitha

UT010216