Project Report – Unicom TIC Management System

1. Project Overview

- This project is a school management system.
- It helps manage courses, subjects, students, exams, marks, and timetables.
- Users can login with different roles: Admin, Staff, Lecturer, Student.
- Each role can see different parts of the system.
- Admin can add and manage courses, subjects, and students.
- Staff can manage exams and marks.
- Timetables show when and where classes happen.
- We used C#, Windows Forms, and SQLite database.
- The system saves data in a database file.

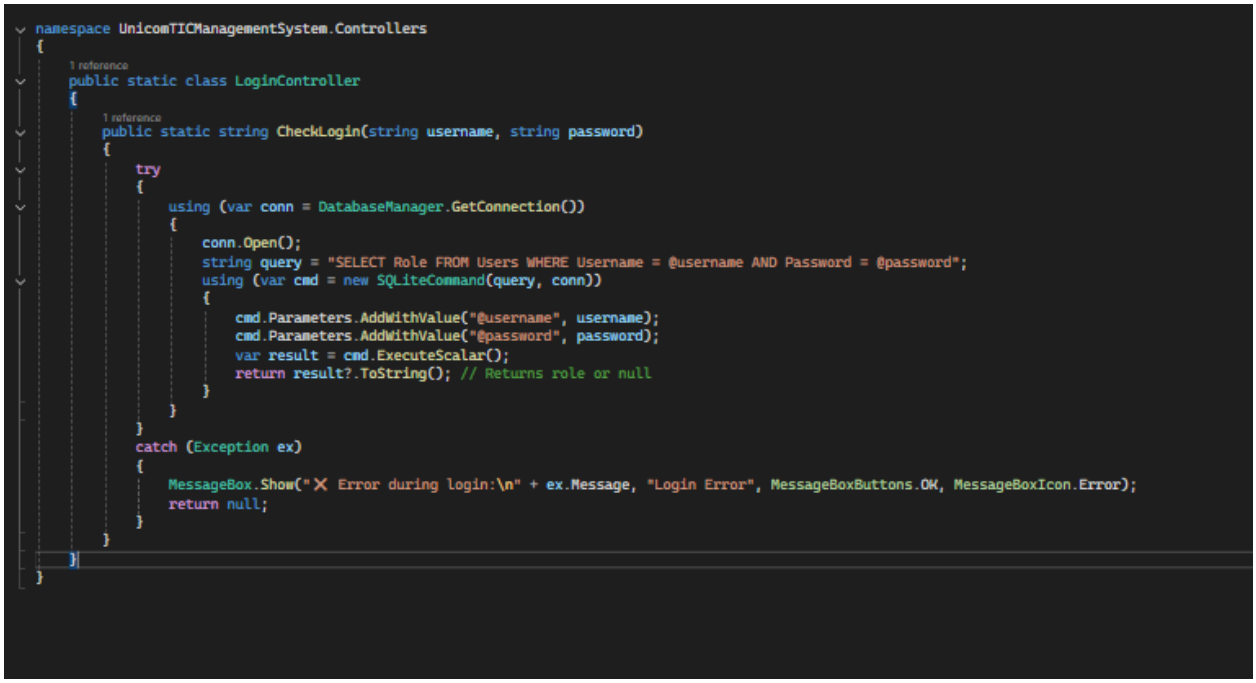
2. Technologies Used

- Programming Language: C#
- Framework: Windows Forms (WinForms)
- Database: SQLite
- IDE: Visual Studio

3. Challenges and Solutions

- **Challenge:** Handling different user roles and permissions.
- **Solution:** Made role-based access control to show or hide buttons.
- **Challenge:** Creating and connecting to the SQLite database.
- **Solution:** Wrote code to create database and tables if not exist.
- **Challenge:** Updating the UI when data changes.
- **Solution:** Reload data in forms after adding, updating, or deleting.
- **Challenge:** Keeping code organized.
- **Solution:** Used controllers to separate logic from UI.

Course



```
namespace UnicomTICManagementSystem.Controllers
{
    1 reference
    public static class LoginController
    {
        1 reference
        public static string CheckLogin(string username, string password)
        {
            try
            {
                using (var conn = DatabaseManager.GetConnection())
                {
                    conn.Open();
                    string query = "SELECT Role FROM Users WHERE Username = @username AND Password = @password";
                    using (var cmd = new SQLiteCommand(query, conn))
                    {
                        cmd.Parameters.AddWithValue("@username", username);
                        cmd.Parameters.AddWithValue("@password", password);
                        var result = cmd.ExecuteScalar();
                        return result?.ToString(); // Returns role or null
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("❌ Error during login:\n" + ex.Message, "Login Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return null;
            }
        }
    }
}
```

```
public partial class MainForm : Form
{
    private string userRole;

    // Constructor receives role after login
    1 reference
    public MainForm(string role)
    {
        InitializeComponent();
        userRole = role;
        lblWelcome.Text = $"Welcome! Role: {role}";
        ApplyRoleAccess();
    }

    // Hide or show buttons based on user role
    1 reference
    private void ApplyRoleAccess()
    {
        // Only Admin sees these
        btnCourses.Visible = userRole == "Admin";
        btnSubjects.Visible = userRole == "Admin";
        btnStudents.Visible = userRole == "Admin";

        // Admin and Staff can see Exams
        btnExams.Visible = userRole == "Admin" || userRole == "Staff";

        // Admin, Staff, Lecturer, Student can see Marks
        btnMarks.Visible = userRole == "Admin" || userRole == "Staff" || userRole == "Lecturer" || userRole == "Student";

        // Everyone can see Timetables
        btnTimetables.Visible = true;
    }

    // Button clicks open respective forms

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

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

    1 reference
    private void btnStudents_Click(object sender, EventArgs e)
    {
        var form = new StudentForm();
        form.ShowDialog();
    }

    1 reference
    private void btnExams_Click(object sender, EventArgs e)
    {
        var form = new ExamForm();
        form.ShowDialog();
    }

    1 reference
    private void btnMarks_Click(object sender, EventArgs e)
    {
        var form = new MarkForm();
        form.ShowDialog();
    }

    1 reference
    private void btnTimetables_Click(object sender, EventArgs e)
    {
        var form = new TimetableForm();
        form.ShowDialog();
    }

    1 reference
    private void btnLogout_Click(object sender, EventArgs e)
    {
        // Close main form and restart application (back to login)
        this.Close();
        Application.Restart();
    }
```

```
{
    private List<Subject> subjects;
    private List<Room> rooms;

    1 reference
    public TimetableForm()
    {
        InitializeComponent();
        LoadSubjects();
        LoadRooms();
        LoadTimetables();
    }

    1 reference
    private void LoadSubjects()
    {
        subjects = SubjectController.GetAllSubjects();
        cmbSubject.DataSource = subjects;
        cmbSubject.DisplayMember = "SubjectName";
        cmbSubject.ValueMember = "SubjectID";
    }

    1 reference
    private void LoadRooms()
    {
        rooms = RoomController.GetAllRooms();
        cmbRoom.DataSource = rooms;
        cmbRoom.DisplayMember = "RoomName";
        cmbRoom.ValueMember = "RoomID";
    }

    4 references
    private void LoadTimetables()
    {
        dgvTimetables.DataSource = TimetableController.GetAllTimetables();
        dgvTimetables.Columns["TimetableID"].Visible = false;
    }

    1 reference
    private void btnAdd_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtTimeSlot.Text))
        {
            MessageBox.Show("Please enter a time slot.");
            return;
        }

        int subjectId = (int)cmbSubject.SelectedValue;
        int roomId = (int)cmbRoom.SelectedValue;

        TimetableController.AddTimetable(subjectId, txtTimeSlot.Text.Trim(), roomId);
        LoadTimetables();
        txtTimeSlot.Clear();
    }

    1 reference
    private void btnUpdate_Click(object sender, EventArgs e)
    {
        if (dgvTimetables.SelectedRows.Count > 0)
        {
            int id = Convert.ToInt32(dgvTimetables.SelectedRows[0].Cells["TimetableID"].Value);
            int subjectId = (int)cmbSubject.SelectedValue;
            int roomId = (int)cmbRoom.SelectedValue;
            string slot = txtTimeSlot.Text.Trim();

            TimetableController.UpdateTimetable(id, subjectId, slot, roomId);
            LoadTimetables();
        }
    }

    1 reference
    private void btnDelete_Click(object sender, EventArgs e)
    {
        if (dgvTimetables.SelectedRows.Count > 0)
        {
            int id = Convert.ToInt32(dgvTimetables.SelectedRows[0].Cells["TimetableID"].Value);
            TimetableController.DeleteTimetable(id);
            LoadTimetables();
        }
    }

    0 references
    private void dgvTimetables_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            txtTimeSlot.Text = dgvTimetables.Rows[e.RowIndex].Cells["TimeSlot"].Value.ToString();
```

```
4 references
public partial class CourseForm : Form
{
    1 reference
    public CourseForm()
    {
        InitializeComponent();
        LoadCourses();  // Load course list when form opens
    }
    // Load all courses into the DataGridView
    4 references
    private void LoadCourses()
    {
        dgvCourses.DataSource = CourseController.GetAllCourses();
        dgvCourses.Columns["CourseID"].Visible = false; // hide ID column
    }

    // Add a new course
    1 reference
    private void btnAdd_Click(object sender, EventArgs e)
    {
        string name = txtCourseName.Text.Trim();
        if (string.IsNullOrWhiteSpace(name))
        {
            MessageBox.Show("Course name cannot be empty.");
            return;
        }

        if (CourseController.CourseExists(name))
        {
            MessageBox.Show("This course already exists.");
            return;
        }

        CourseController.AddCourse(name);
        LoadCourses();
        txtCourseName.Clear();
    }

    // Update selected course
    1 reference
    private void btnUpdate_Click(object sender, EventArgs e)
    {
        if (dgvCourses.SelectedRows.Count == 0)
        {
            MessageBox.Show("Please select a course to update.");
            return;
        }

        string name = txtCourseName.Text.Trim();
        if (string.IsNullOrWhiteSpace(name))
        {
            MessageBox.Show("Course name cannot be empty.");
            return;
        }

        int courseId = Convert.ToInt32(dgvCourses.SelectedRows[0].Cells[0].Value);
        CourseController.UpdateCourse(courseId, name);
        LoadCourses();
        txtCourseName.Clear();
    }

    // Delete selected course
    1 reference
    private void btnDelete_Click(object sender, EventArgs e)
    {
        if (dgvCourses.SelectedRows.Count == 0)
        {
            MessageBox.Show("Please select a course to delete.");
            return;
        }

        int courseId = Convert.ToInt32(dgvCourses.SelectedRows[0].Cells[0].Value);

        DialogResult result = MessageBox.Show("Are you sure you want to delete this course?", "Confirm Delete", MessageBoxButtons.YesNo);
        if (result == DialogResult.Yes)
        {
            CourseController.DeleteCourse(courseId);
            LoadCourses();
            txtCourseName.Clear();
        }
    }
```

```
private static void CreateDatabaseAndTables()
{
    SQLiteConnection.CreateFile(dbPath);
    using (var conn = new SQLiteConnection(connectionString))
    {
        conn.Open();
        using (var cmd = new SQLiteCommand(conn))
        {
            cmd.CommandText = @"
            CREATE TABLE IF NOT EXISTS Users (
                UserID INTEGER PRIMARY KEY AUTOINCREMENT,
                Username TEXT,
                Password TEXT,
                Role TEXT
            );

            CREATE TABLE IF NOT EXISTS Courses (
                CourseID INTEGER PRIMARY KEY AUTOINCREMENT,
                CourseName TEXT
            );

            CREATE TABLE IF NOT EXISTS Subjects (
                SubjectID INTEGER PRIMARY KEY AUTOINCREMENT,
                SubjectName TEXT,
                CourseID INTEGER,
                FOREIGN KEY(CourseID) REFERENCES Courses(CourseID)
            );

            CREATE TABLE IF NOT EXISTS Students (
                StudentID INTEGER PRIMARY KEY AUTOINCREMENT,
                Name TEXT,
                CourseID INTEGER,
                FOREIGN KEY(CourseID) REFERENCES Courses(CourseID)
            );

            CREATE TABLE IF NOT EXISTS Exams (
                ExamID INTEGER PRIMARY KEY AUTOINCREMENT,
                ExamName TEXT,
                SubjectID INTEGER,
                FOREIGN KEY(SubjectID) REFERENCES Subjects(SubjectID)
            );

            CREATE TABLE IF NOT EXISTS Marks (
                MarkID INTEGER PRIMARY KEY AUTOINCREMENT,
                StudentID INTEGER,
                ExamID INTEGER,
                Score INTEGER,
                FOREIGN KEY(StudentID) REFERENCES Students(StudentID),
                FOREIGN KEY(ExamID) REFERENCES Exams(ExamID)
            );

            CREATE TABLE IF NOT EXISTS Rooms (
                RoomID INTEGER PRIMARY KEY AUTOINCREMENT,
                RoomName TEXT,
                RoomType TEXT
            );

            CREATE TABLE IF NOT EXISTS Timetables (
                TimetableID INTEGER PRIMARY KEY AUTOINCREMENT,
                SubjectID INTEGER,
                TimeSlot TEXT,
                RoomID INTEGER,
                FOREIGN KEY(SubjectID) REFERENCES Subjects(SubjectID),
                FOREIGN KEY(RoomID) REFERENCES Rooms(RoomID)
            );
```

```
{
    private List<Subject> subjects;

    1 reference
    public ExamForm()
    {
        InitializeComponent();
        LoadSubjects();
        LoadExams();
    }

    1 reference
    private void LoadSubjects()
    {
        subjects = SubjectController.GetAllSubjects();
        cmbSubject.DataSource = subjects;
        cmbSubject.DisplayMember = "SubjectName";
        cmbSubject.ValueMember = "SubjectID";
    }

    4 references
    private void LoadExams()
    {
        dgvExams.DataSource = ExamController.GetAllExams();
        dgvExams.Columns["ExamID"].Visible = false;
    }

    1 reference
    private void btnAdd_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtExamName.Text)) return;
        int subjectId = (int)cmbSubject.SelectedValue;
        ExamController.AddExam(txtExamName.Text.Trim(), subjectId);
        LoadExams();
        txtExamName.Clear();
    }

    1 reference
    private void btnUpdate_Click(object sender, EventArgs e)
    {
        if (dgvExams.SelectedRows.Count > 0)
        {
            int id = Convert.ToInt32(dgvExams.SelectedRows[0].Cells["ExamID"].Value);
            string name = txtExamName.Text.Trim();
            int subjectId = (int)cmbSubject.SelectedValue;
            ExamController.UpdateExam(id, name, subjectId);
            LoadExams();
        }
    }

    1 reference
    private void btnDelete_Click(object sender, EventArgs e)
    {
        if (dgvExams.SelectedRows.Count > 0)
        {
            int id = Convert.ToInt32(dgvExams.SelectedRows[0].Cells["ExamID"].Value);
            ExamController.DeleteExam(id);
            LoadExams();
        }
    }

    0 references
    private void dgvExams_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            txtExamName.Text = dgvExams.Rows[e.RowIndex].Cells["ExamName"].Value.ToString();
            cmbSubject.SelectedValue = Convert.ToInt32(dgvExams.Rows[e.RowIndex].Cells["SubjectID"].Value);
        }
    }


    1 reference
    private void ExamForm_Load(object sender, EventArgs e)
    {

    }

    1 reference
    private void txtExamName_TextChanged(object sender, EventArgs e)
    {

    }

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