Unicom Tic Management System Report

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 Admin,Attendence,ClassTimetable,Course,Exam,Lecturer,LecturerSubject,Mark,Room,

Student,Subject,User.

- Default Login Credentials
    - Username:admin
    - Password:admin123

Key Features Implemented:

- SQLite intergration with **OOP** concept
- Student Management: Add, Edit, Delete, View student records
- Subject Management: Manage subject data
- Timetable Management: Assign subjects and time slots
- Exam and Marks Management: Enter and display exam results
- Attendance Management: Mark and view attendance per subject, per student, per date
- Role-Based Login System: **Admin, Lecturer, Student**
- Admin-only User Registration

- **Technologies Used:**
- Language: C# (Version 7.3)
- Framework: Windows Forms (WinForms)
- Database: **SQLite**
- IDE: Visual Studio 2022
- Architecture**: MVC (Model–View–Controller)**

1. Challenges Faced and Solutions

Problem: Lock Error on Database File
Solution: This occurred when multiple connections tried to access the .db file simultaneously without proper closure. Fixed by ensuring each SQLiteConnection is explicitly opened and closed inside a using block or after command execution.

Learning: I understood the importance of opening and closing SQLiteConnection objects carefully. If connections remain open unintentionally, it can lock the database file and cause runtime errors. Using proper disposal and ensuring each query handles its own connection improved application stability.

Problem: Build Error During Compilation
Solution: Some missing file references or incorrect namespace declarations caused build failures. These were solved by cleaning and rebuilding the project, ensuring all .cs files were included in the .csproj, and checking for typos in namespaces.

Learning: I gained experience using Visual Studio's build output window and error list to identify missing files, incorrect namespace references, and project misconfigurations. This helped me understand how .csproj files manage included resources and how to properly rebuild a clean project.

Problem: Executable (.exe) File Not Running Properly
Solution: The compiled .exe sometimes failed due to missing SQLite database or DLLs. Fixed by ensuring the .db file is copied to the output directory and setting Copy to Output Directory = Copy if newer for dependent files.

Learning: I learned that the .exe file alone isn't enough—it must be packaged with its dependencies, including the .db file and necessary DLLs. Using the "Copy to Output Directory" property in Visual Studio solved these issues and taught me how to prepare apps for distribution.

Problem: Attendance Duplication
Solution: Multiple entries for the same student/subject/date were being inserted. A check was added before insertion in the controller to prevent duplicate attendance records.

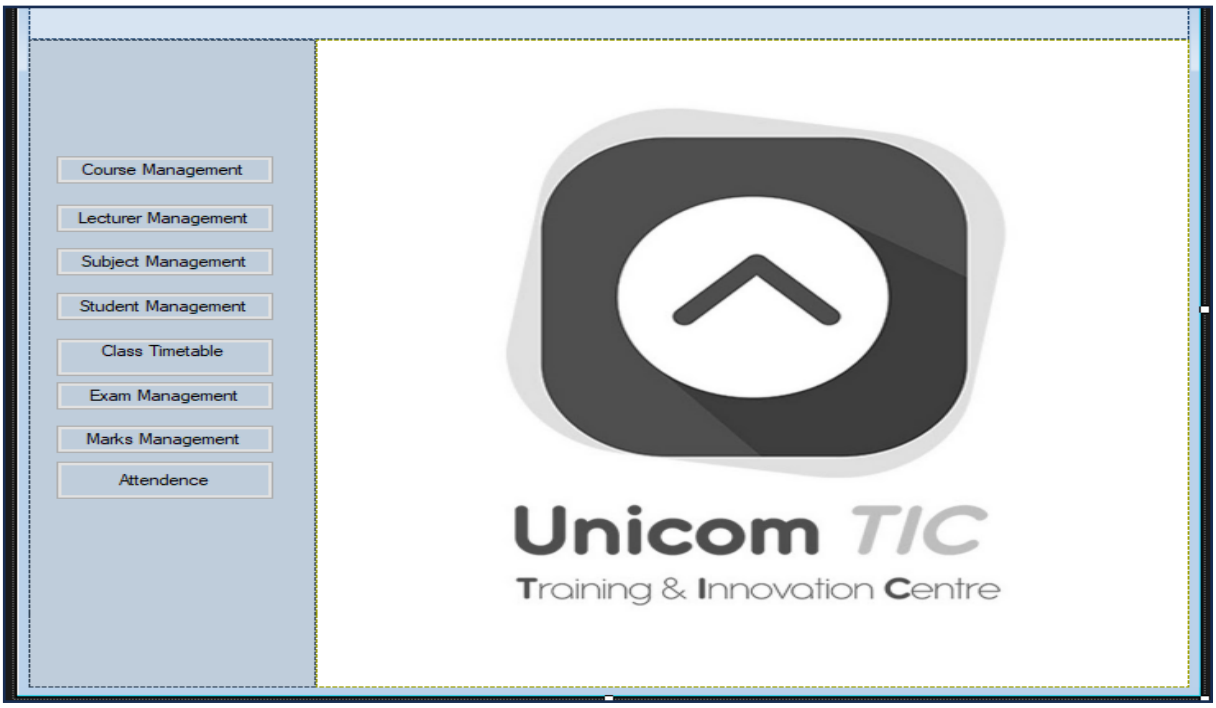
Learning: By checking whether an attendance record already exists before inserting, I learned how to ensure that the system maintains accurate, unique entries—particularly important in academic applications.

Problem: Data Connection Reuse
Solution: Passing database connection across multiple classes led to confusion and reuse issues. Solved by keeping each method responsible for managing its own SQLiteConnection and disposing it after execution.

Learning: Instead of using shared or global connections, I now understand the value of isolating data access logic within each method or service, which makes debugging and error handling easier.

CODE SAMPLES:

1. Admin dash board



2. Attendance Controller

```
                {
                    string query = @"INSERT INTO Attendence (StudentId, Date, Status)
                                     VALUES (@StudentId, @Date, @Status)";
                    using (var cmd = new SQLiteCommand(query, conn))
                    {
                        cmd.Parameters.AddWithValue("@StudentId", attendence.StudentId);
                        cmd.Parameters.AddWithValue("@Date", attendence.Date.ToString("yyyy-MM-dd"));
                        cmd.Parameters.AddWithValue("@Status", attendence.Status);
                        cmd.ExecuteNonQuery();
                    }

                    transaction.Commit();
                    MessageBox.Show("Attendance recorded successfully.");
                }
                catch (SQLiteException ex)
                {
                    transaction.Rollback();
                    if (ex.ResultCode == SQLiteErrorCode.Constraint && ex.Message.Contains("UNIQUE"))
                    {
                        MessageBox.Show("Attendance for this student on this date already exists.");
                    }
                    else
                    {
                        MessageBox.Show("Error adding attendance: " + ex.Message);
                    }
                }
            }
        }

        // Get all attendance records with student name
        1 reference
        public List<Attendence> GetAll()
        {
            var list = new List<Attendence>();
            using (var conn = DatabaseConnection.GetConnection())
            {
                string query = @"SELECT a.StudentId, s.Name AS StudentName, a.Date, a.Status
                                 FROM Attendence a
                                 JOIN Students s ON a.StudentId = s.Id";

                using (var cmd = new SQLiteCommand(query, conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        var att = new Attendence()
No issues found                                                                    Ln: 1
```

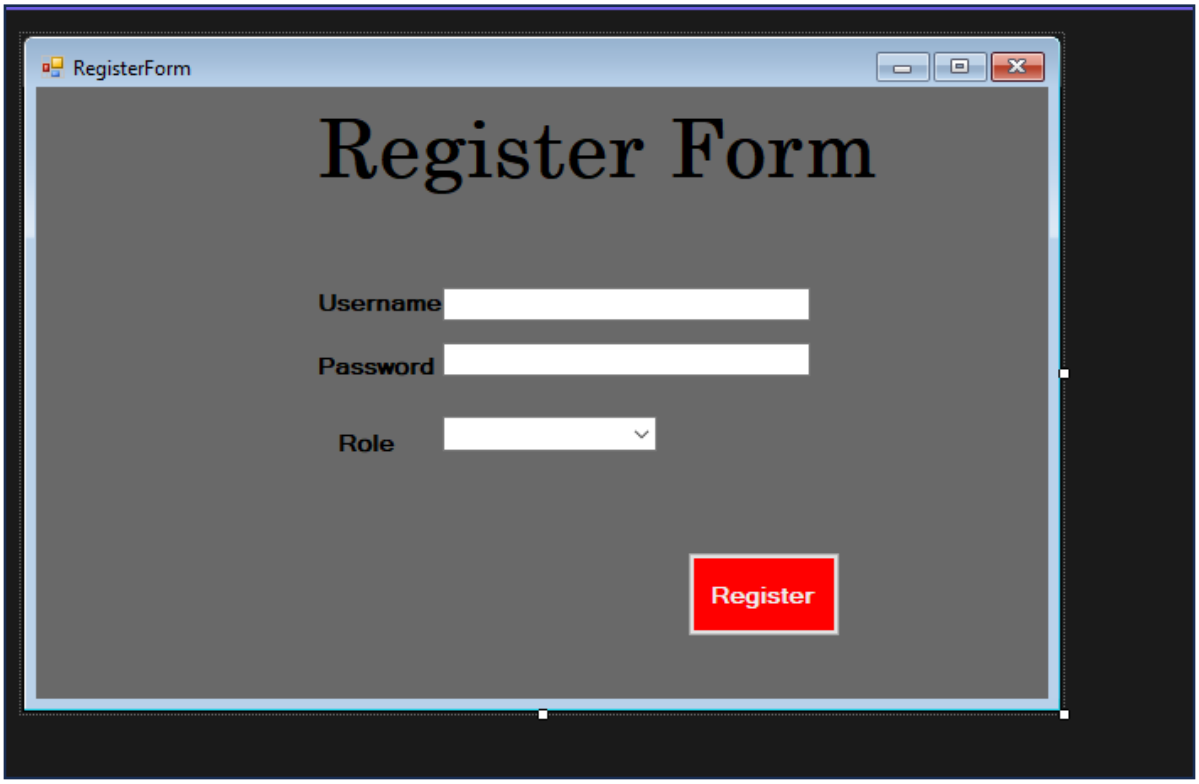
3. Lecture Form code

```
        }

        1 reference
        private void btnLogout_Click(object sender, EventArgs e)
        {
            this.Close();
            new LoginForm().Show();
        }

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

        }

        1 reference
        private void btnViewTimetable_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Timetable view loads.");
            // Connect to timetable
        }

        1 reference
        private void LecturerDashboard_Load(object sender, EventArgs e)
        {
            lblWelcome.Text = "Welcome, Lecturer";
            btnViewTimetable.Visible = true;
            btnManageMarks.Visible = true;

            // Hide all non-lecturer features
            btnManageStudents.Visible = false;
            btnManageCourses.Visible = false;
            btnManageExams.Visible = false;
            btnViewMarks.Visible = false;
        }

        1 reference
        private void btnManageMarks_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Mark management form opens.");
            // Connect to mark form
        }
    }
}

No issues found
```

4. Login Form Code

```
}

1 reference
private void btnLogin_Click(object sender, EventArgs e)
{
    var service = new UserService();
    string username = txtUsername.Text.Trim();
    string password = txtPassword.Text;

    var result = service.ValidateLogin(username, password);
    if (result.success)
    {
        this.Hide();
        switch (result.role)
        {
            case "Admin":
                new AdminDashboardForm().Show();
                break;
            case "Student":
                new StudentDashboard(username).Show();
                break;
            case "Lecturer":
                new LecturerDashboard().Show();
                break;
        }
    }
    else
    {
        MessageBox.Show("Invalid username or password");
    }
}

1 reference
private void btnRegister_Click(object sender, EventArgs e)
{
    var reg = new RegisterForm();
    reg.ShowDialog();
}

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

}
```

5. Register form



6. Table Query

```
using (SQLiteCommand cmd = new SQLiteCommand(insertAdmin, conn))
{
    cmd.ExecuteNonQuery();
}

string examTable = @"
CREATE TABLE IF NOT EXISTS Exams(
    Id INTEGER PRIMARY KEY AUTOINCREMENT,
    Name TEXT NOT NULL,
    SubjectId INTEGER,
    ExamDate DATE NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    UNIQUE (Name, SubjectId),
    UNIQUE (SubjectId, ExamDate, StartTime, EndTime),
    FOREIGN KEY (SubjectId) REFERENCES Subjects(Id)

);";
using (SQLiteCommand cmd = new SQLiteCommand(examTable, conn))
{
    cmd.ExecuteNonQuery();
}

string timetableTable = @"
CREATE TABLE IF NOT EXISTS Timetable(
    Id INTEGER PRIMARY KEY AUTOINCREMENT,
    SubjectId INTEGER,
    Room TEXT,
    ClassDate DATE NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    UNIQUE (Room, ClassDate, StartTime, EndTime),
    FOREIGN KEY (SubjectId) REFERENCES Subjects(Id)

);";
using (SQLiteCommand cmd = new SQLiteCommand(timetableTable, conn))
{
    cmd.ExecuteNonQuery();
}
string markTable = @"
CREATE TABLE IF NOT EXISTS Marks(
    StudentId INTEGER
```

No issues found

7. User Service Code

```
{
    2 references
    internal class UserService
    {
        private static String connectionString = "Data Source=Ut.db;Version=3;";

        1 reference
        public bool RegisterUser(string username, string password, string role)
        {
            using (var con = new SQLiteConnection(connectionString))
            {
                con.Open();

                var cmd = new SQLiteCommand("INSERT INTO Users (Username, Password, Role) VALUES (@username, @password, @role)", con);
                cmd.Parameters.AddWithValue("@username", username);
                cmd.Parameters.AddWithValue("@password", password);
                cmd.Parameters.AddWithValue("@role", role);

                return cmd.ExecuteNonQuery() > 0;
            }
        }

        1 reference
        public (bool success, string role) ValidateLogin(string username, string password)
        {
            using (var con = new SQLiteConnection(connectionString))
            {

                con.Open();

                var cmd = new SQLiteCommand("SELECT Role FROM Users WHERE Username = @username AND Password = @password", con);
                cmd.Parameters.AddWithValue("@username", username);
                cmd.Parameters.AddWithValue("@password", password);

                var reader = cmd.ExecuteReader();
                if (reader.Read())
                    return (true, reader["Role"].ToString());

                return (false, null);
            }
        }
    }
}
```

No issues found

Author

Fathima Shajiya

UT010665