UNICOM TIC

PROJECT REPORT

Introduction

My name is Nixon Yapes , and this report summarizes the design and development of the *Unicom TIC Management System* desktop application. Developed as part of my academic coursework, the project offered a valuable opportunity to apply object-oriented programming principles, implement graphical user interfaces using C# WinForms, and explore modular application architecture through the MVC design pattern. This report outlines the system’s key features, technologies used, development challenges, and the lessons gained during the process.

Table of Contents

Problem Definition

Educational institutions often rely on fragmented systems to manage academic operations like course registration, examinations, and scheduling. The *Unicom TIC Management System* addresses this gap by providing a unified desktop application that simplifies academic administration. It allows role-based access for Admin, Staff, Lecturers, and Students to manage essential modules like students, courses, exams, and timetables.

Project Requirements

Functional Requirements

- Role-based login (Admin, Staff, Lecturer, Student)
- Create, edit, and manage Courses, Subjects, Students, Exams
- View and record student marks
- Allocate classrooms (labs/halls) to timetable entries
- Role-specific dashboards displaying permitted features

Non-Functional Requirements

- Clean and intuitive UI using C# WinForms
- Persistent data storage using SQLite
- Scalable, modular MVC-based architecture
- Input validation and error message handling

Focus on quality attributes and tech constraints.

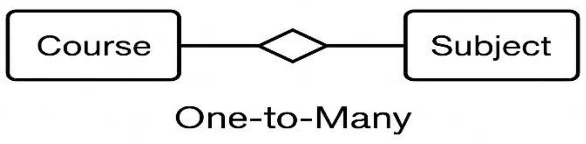
- Simple and intuitive UI using WinForms
- Persistent storage using SQLite
- Scalable and modular structure via MVC
- Error handling and input validation

System Design

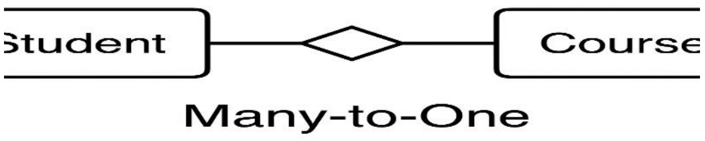
a. Entity-Relationship Diagram

The data model is structured with the following key relationships:

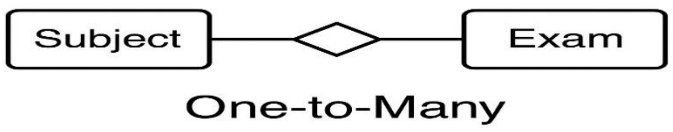
- One Course has many Subjects



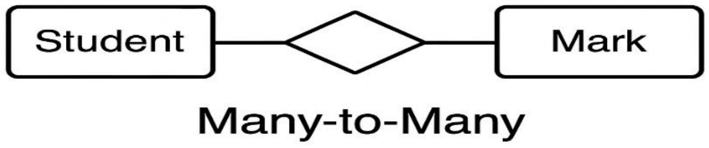
- Many Students belong to one Course



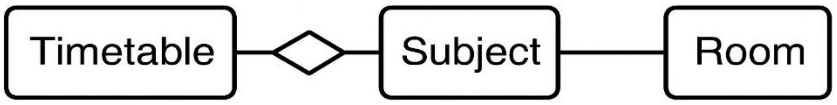
- One Subject has multiple Exams



- One Student has many Marks (linked via Exams)



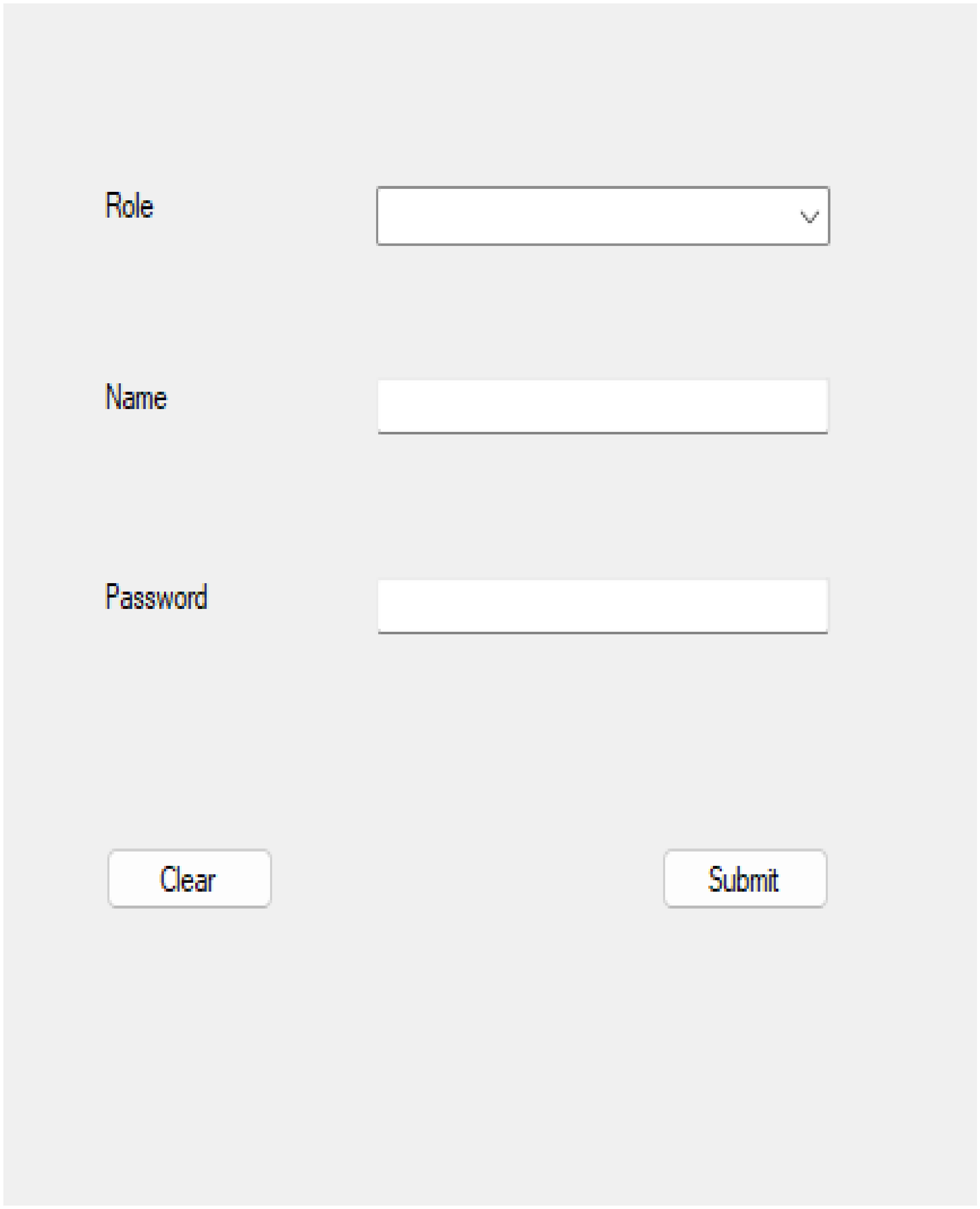
- Each Timetable entry links one Subject to one Room (Lab or Hall)

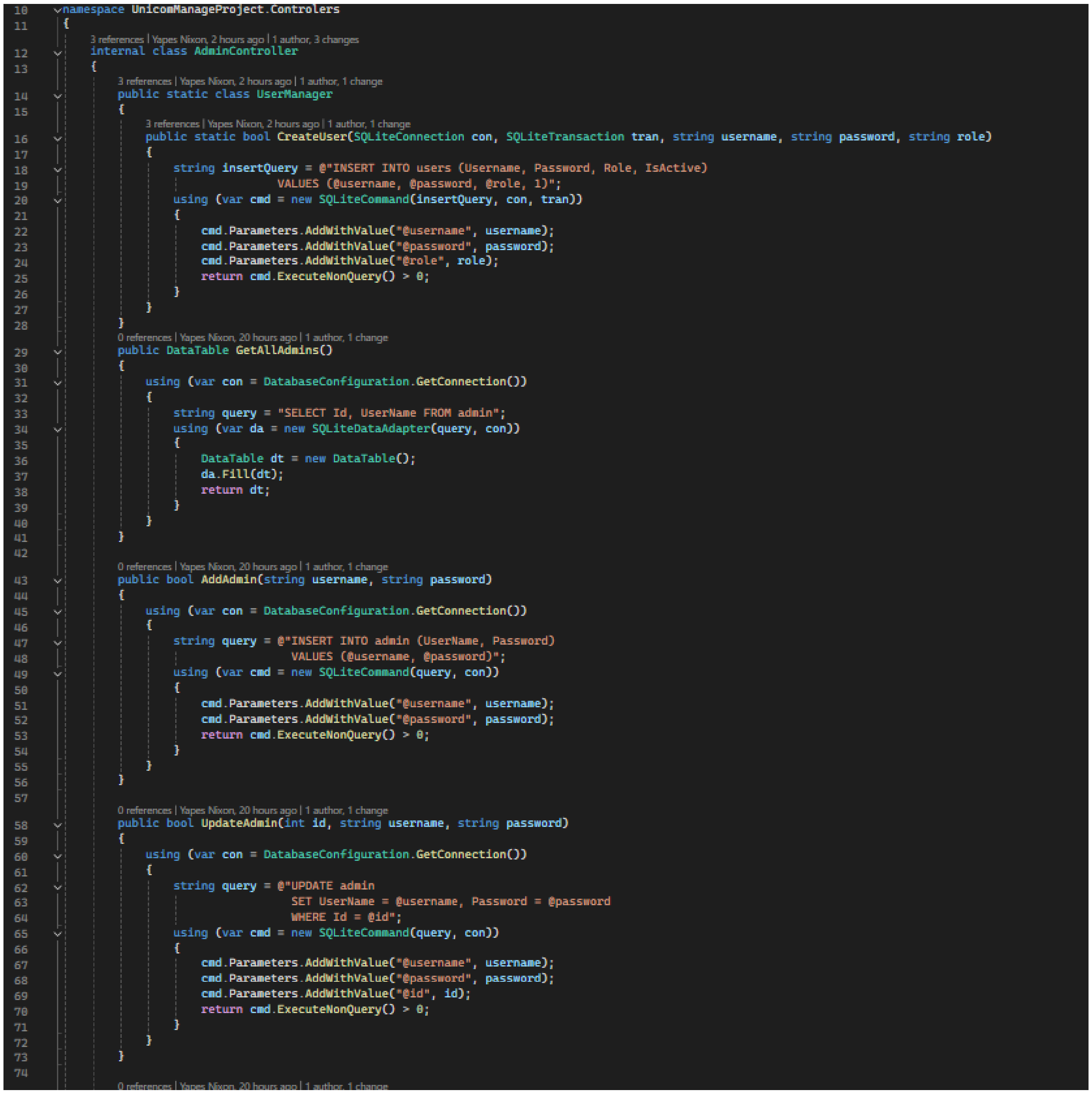


Wireframes

Include UI sketches or screenshots of:

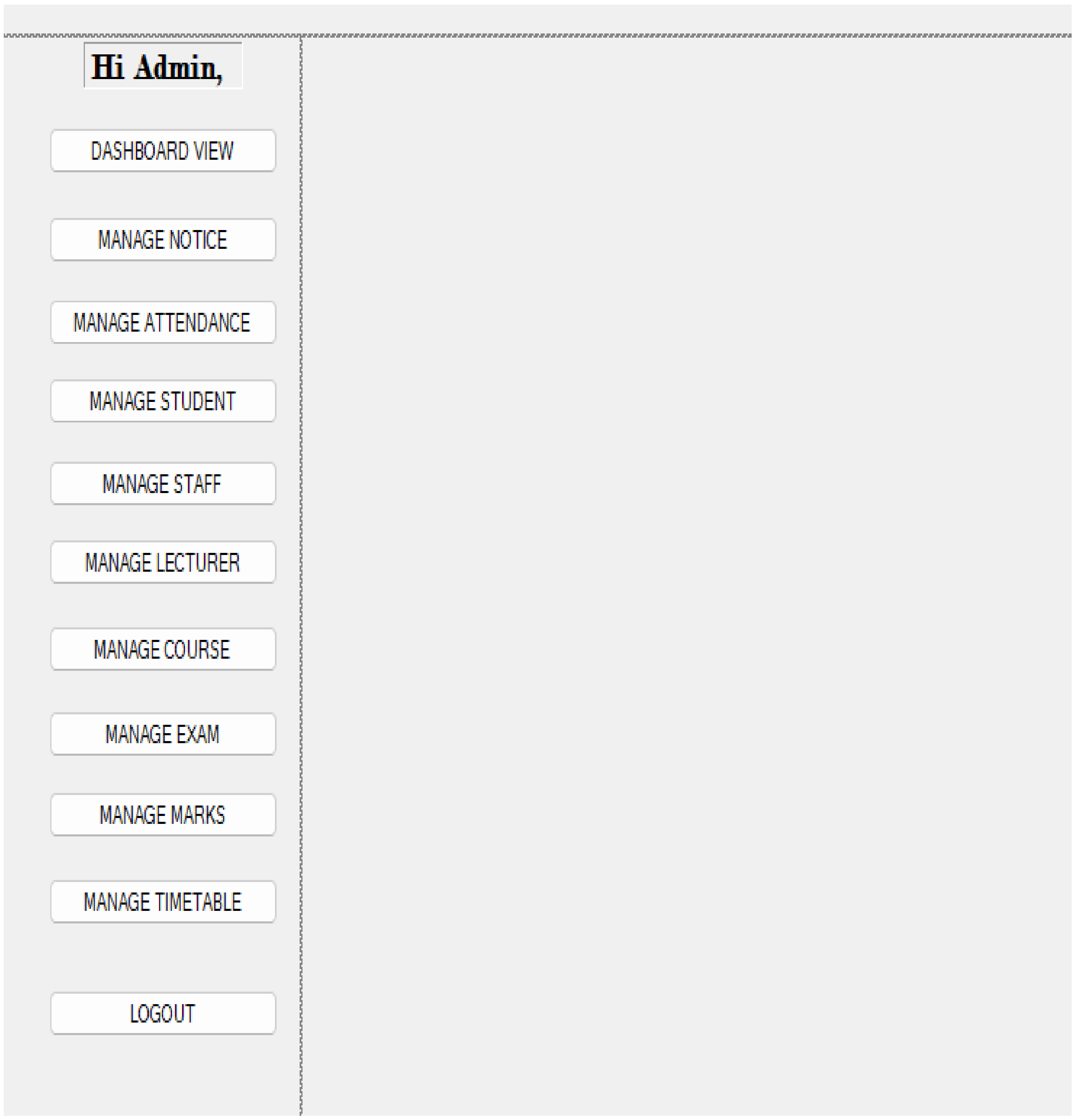
- Login Form



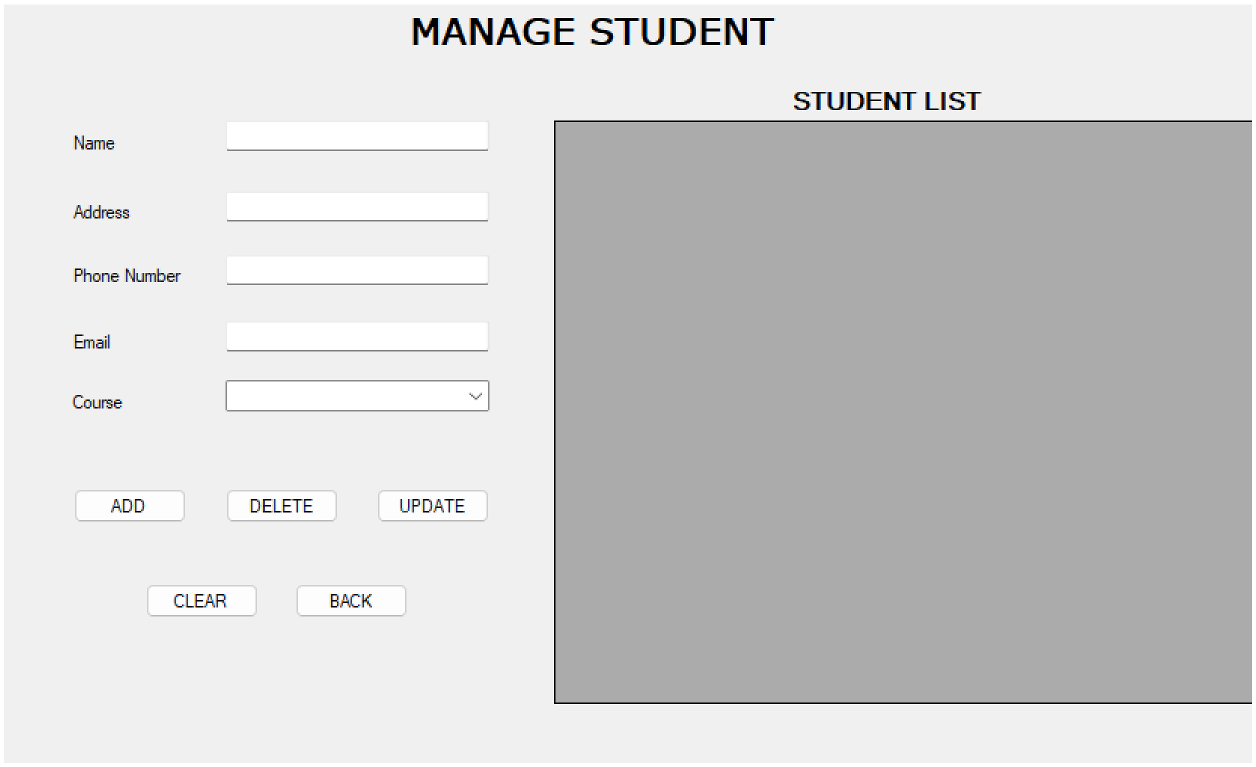


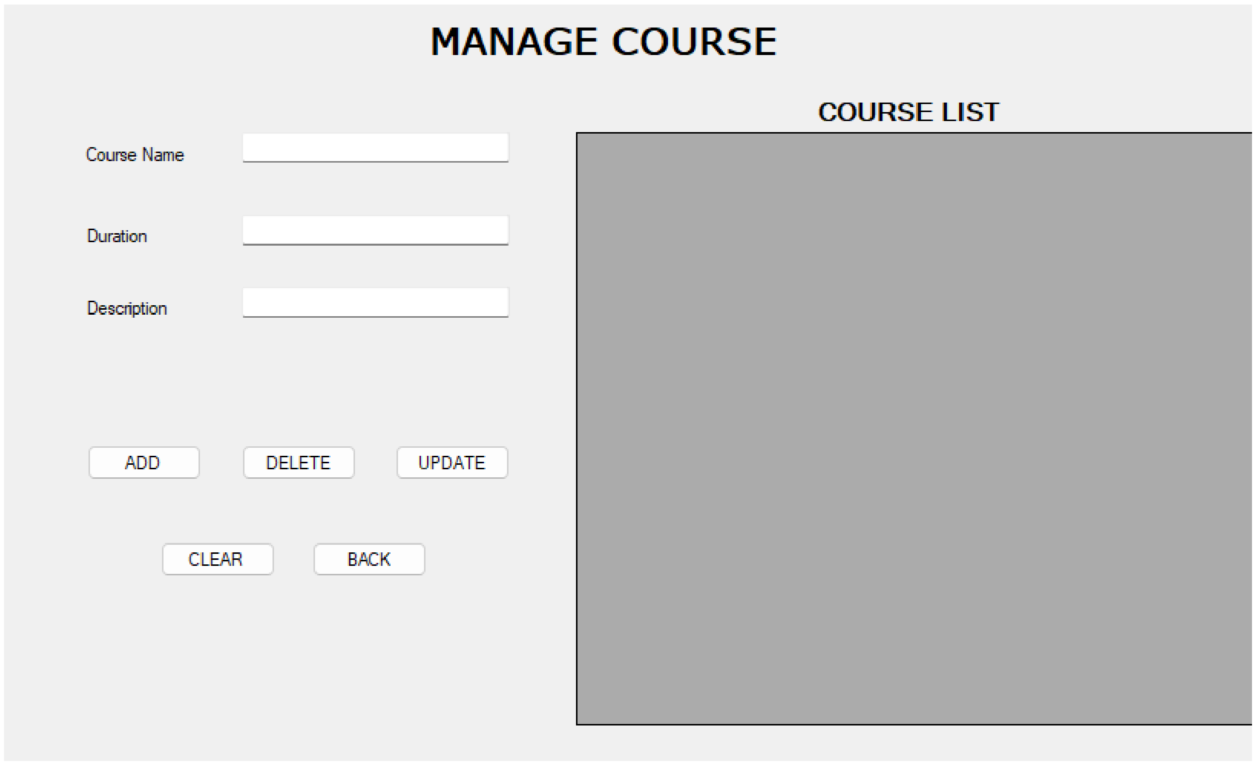
```
namespace UnicomManageProject.Controlers
{
    3 references | Yapes Nixon, 2 hours ago | 1 author, 3 changes
    internal class AdminController
    {
        3 references | Yapes Nixon, 2 hours ago | 1 author, 1 change
        public static class UserManager
        {
            3 references | Yapes Nixon, 2 hours ago | 1 author, 1 change
            public static bool CreateUser(SQLiteConnection con, SQLiteTransaction tran, string username, string password, string role)
            {
                string insertQuery = @"INSERT INTO users (Username, Password, Role, IsActive)
                               VALUES (@username, @password, @role, 1)";
                using (var cmd = new SQLiteCommand(insertQuery, con, tran))
                {
                    cmd.Parameters.AddWithValue("@username", username);
                    cmd.Parameters.AddWithValue("@password", password);
                    cmd.Parameters.AddWithValue("@role", role);
                    return cmd.ExecuteNonQuery() > 0;
                }
            }
        }
        0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
        public DataTable GetAllAdmins()
        {
            using (var con = DatabaseConfiguration.GetConnection())
            {
                string query = "SELECT Id, UserName FROM admin";
                using (var da = new SQLiteDataAdapter(query, con))
                {
                    DataTable dt = new DataTable();
                    da.Fill(dt);
                    return dt;
                }
            }
        }

        0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
        public bool AddAdmin(string username, string password)
        {
            using (var con = DatabaseConfiguration.GetConnection())
            {
                string query = @"INSERT INTO admin (UserName, Password)
                                 VALUES (@username, @password)";
                using (var cmd = new SQLiteCommand(query, con))
                {
                    cmd.Parameters.AddWithValue("@username", username);
                    cmd.Parameters.AddWithValue("@password", password);
                    return cmd.ExecuteNonQuery() > 0;
                }
            }
        }

        0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
        public bool UpdateAdmin(int id, string username, string password)
        {
            using (var con = DatabaseConfiguration.GetConnection())
            {
                string query = @"UPDATE admin
                                 SET UserName = @username, Password = @password
                                 WHERE Id = @id";
                using (var cmd = new SQLiteCommand(query, con))
                {
                    cmd.Parameters.AddWithValue("@username", username);
                    cmd.Parameters.AddWithValue("@password", password);
                    cmd.Parameters.AddWithValue("@id", id);
                    return cmd.ExecuteNonQuery() > 0;
                }
            }
        }

        0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
```

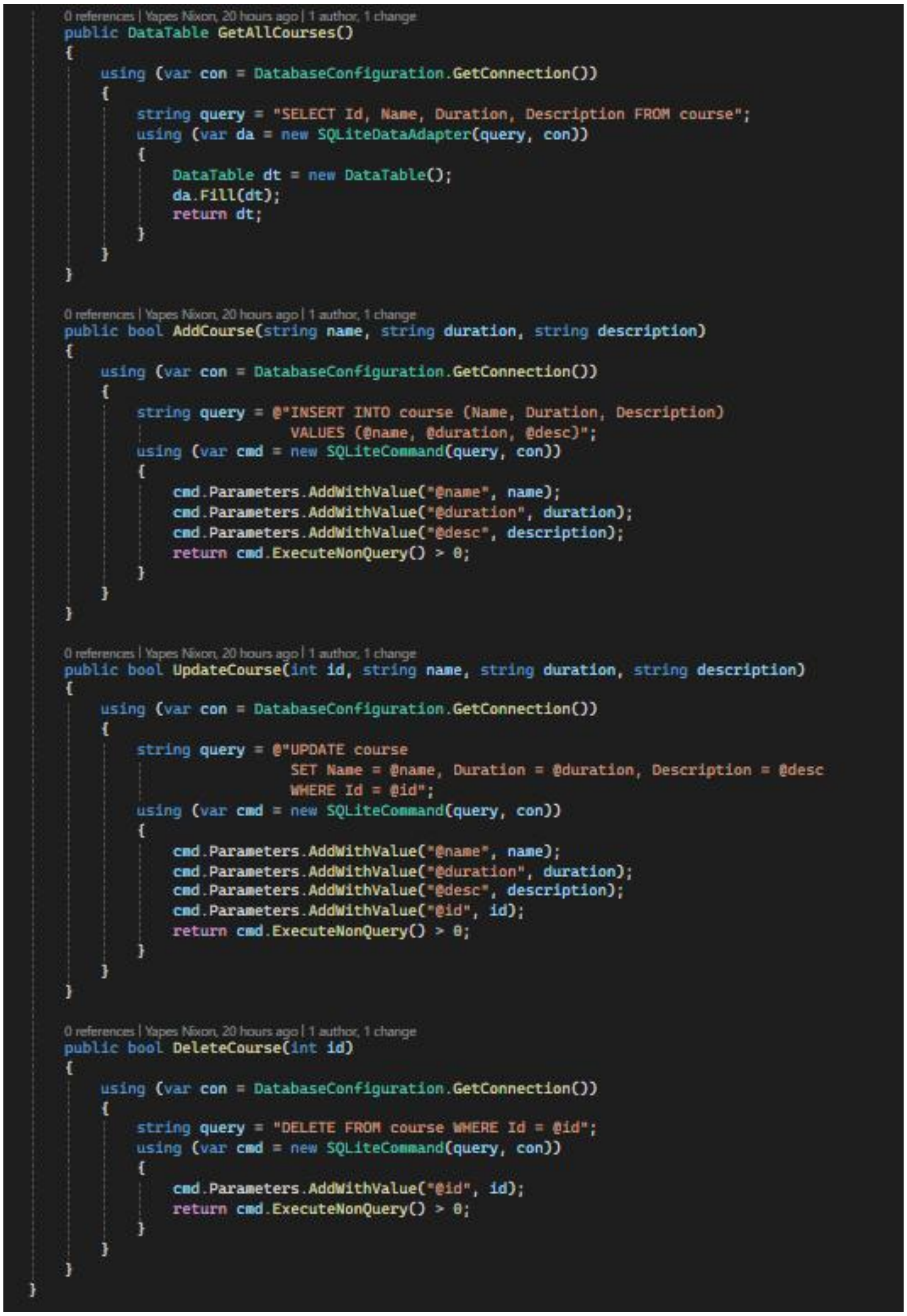
- Main Dashboard Form



- CourseForm / StudentForm

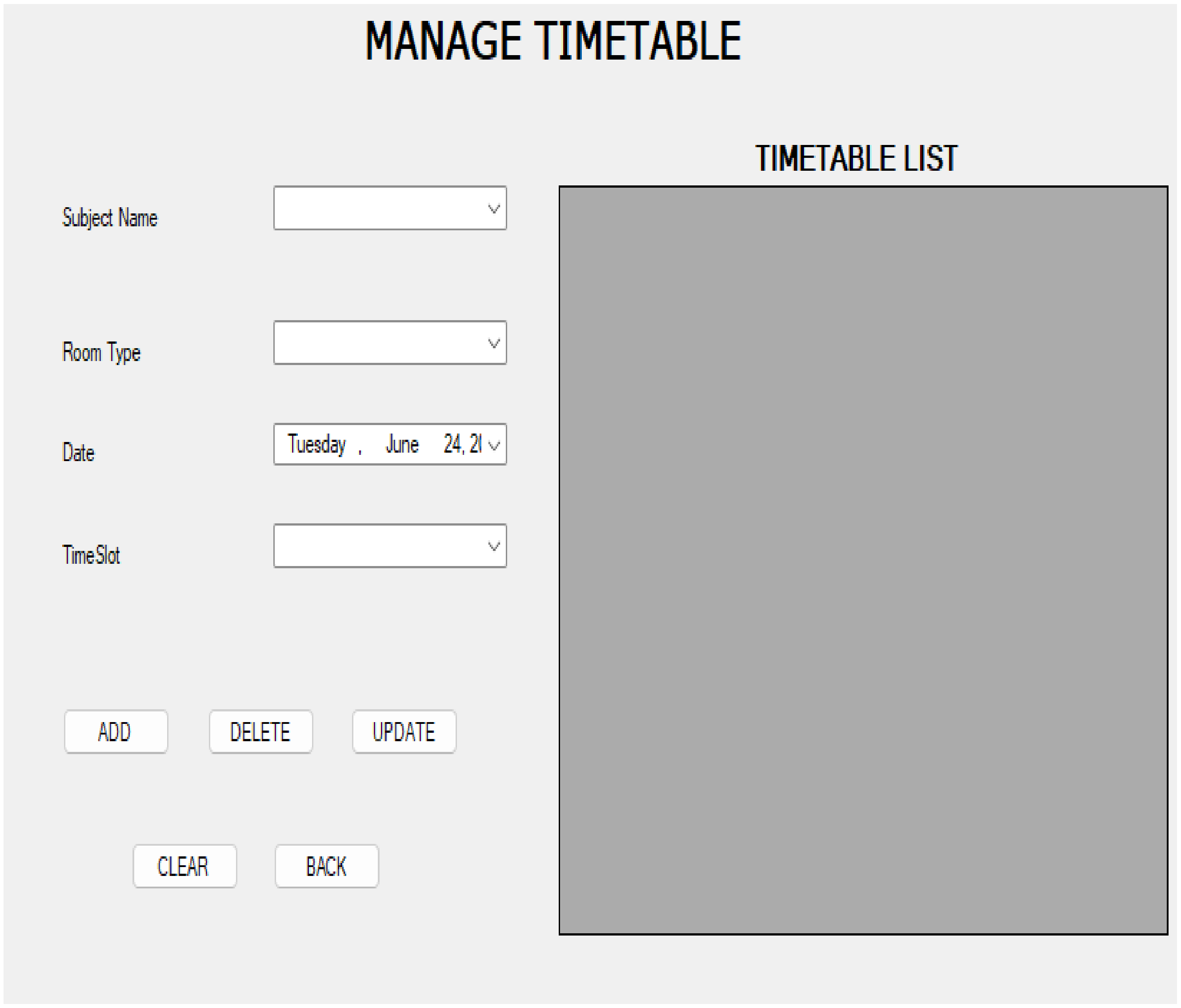


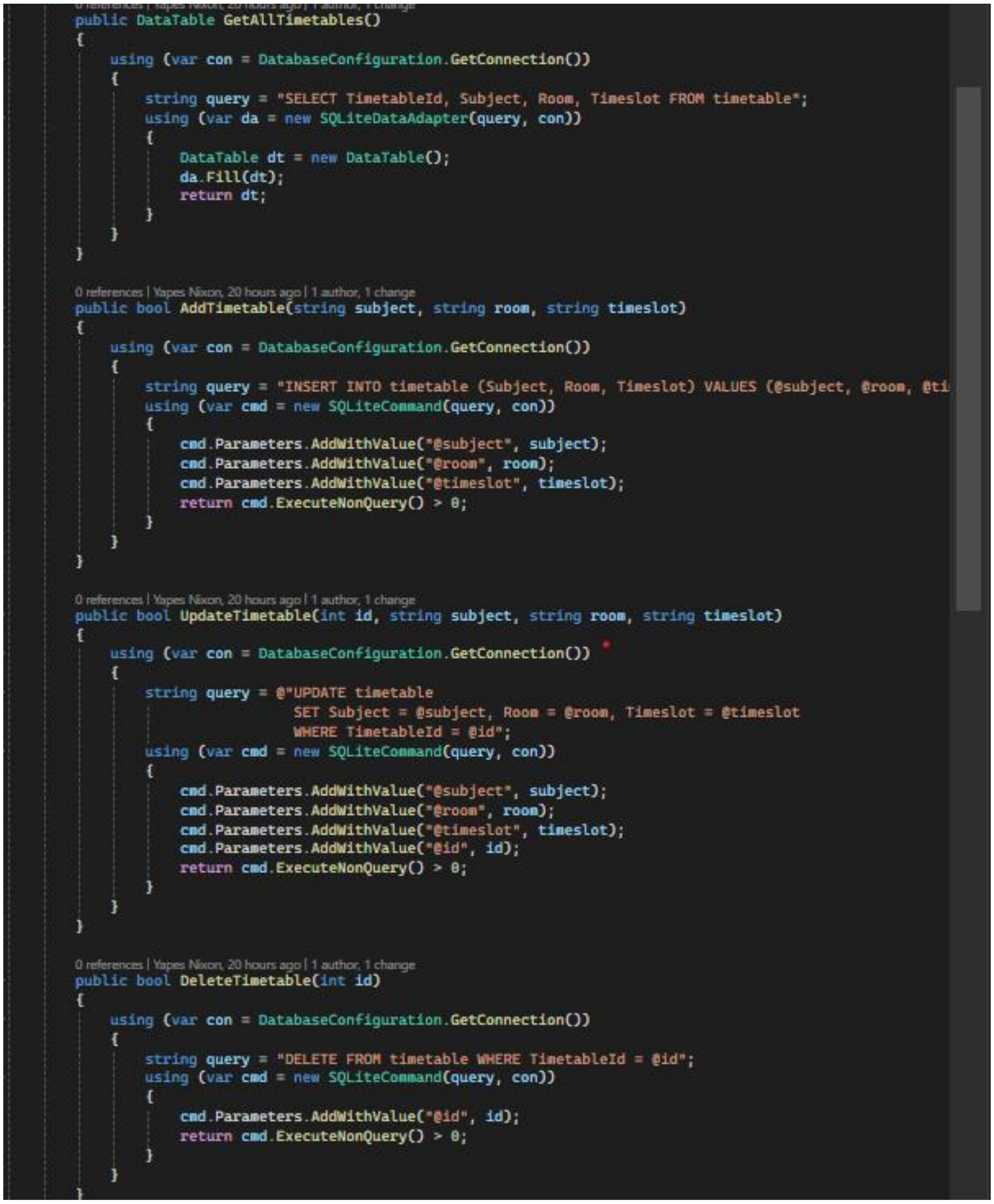




```
    0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
    public DataTable GetAllCourses()
    {
        using (var con = DatabaseConfiguration.GetConnection())
        {
            string query = "SELECT Id, Name, Duration, Description FROM course";
            using (var da = new SQLiteDataAdapter(query, con))
            {
                DataTable dt = new DataTable();
                da.Fill(dt);
                return dt;
            }
        }
    }

    0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
    public bool AddCourse(string name, string duration, string description)
    {
        using (var con = DatabaseConfiguration.GetConnection())
        {
            string query = @"INSERT INTO course (Name, Duration, Description)
                             VALUES (@name, @duration, @desc)";
            using (var cmd = new SQLiteCommand(query, con))
            {
                cmd.Parameters.AddWithValue("@name", name);
                cmd.Parameters.AddWithValue("@duration", duration);
                cmd.Parameters.AddWithValue("@desc", description);
                return cmd.ExecuteNonQuery() > 0;
            }
        }
    }

    0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
    public bool UpdateCourse(int id, string name, string duration, string description)
    {
        using (var con = DatabaseConfiguration.GetConnection())
        {
            string query = @"UPDATE course
                             SET Name = @name, Duration = @duration, Description = @desc
                             WHERE Id = @id";
            using (var cmd = new SQLiteCommand(query, con))
            {
                cmd.Parameters.AddWithValue("@name", name);
                cmd.Parameters.AddWithValue("@duration", duration);
                cmd.Parameters.AddWithValue("@desc", description);
                cmd.Parameters.AddWithValue("@id", id);
                return cmd.ExecuteNonQuery() > 0;
            }
        }
    }

    0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
    public bool DeleteCourse(int id)
    {
        using (var con = DatabaseConfiguration.GetConnection())
        {
            string query = "DELETE FROM course WHERE Id = @id";
            using (var cmd = new SQLiteCommand(query, con))
            {
                cmd.Parameters.AddWithValue("@id", id);
                return cmd.ExecuteNonQuery() > 0;
            }
        }
    }
}
```

- TimetableForm (with room selection dropdown)





```
0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
public DataTable GetAllTimetables()
{
    using (var con = DatabaseConfiguration.GetConnection())
    {
        string query = "SELECT TimetableId, Subject, Room, Timeslot FROM timetable";
        using (var da = new SQLiteDataAdapter(query, con))
        {
            DataTable dt = new DataTable();
            da.Fill(dt);
            return dt;
        }
    }
}

0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
public bool AddTimetable(string subject, string room, string timeslot)
{
    using (var con = DatabaseConfiguration.GetConnection())
    {
        string query = "INSERT INTO timetable (Subject, Room, Timeslot) VALUES (@subject, @room, @ti
        using (var cmd = new SQLiteCommand(query, con))
        {
            cmd.Parameters.AddWithValue("@subject", subject);
            cmd.Parameters.AddWithValue("@room", room);
            cmd.Parameters.AddWithValue("@timeslot", timeslot);
            return cmd.ExecuteNonQuery() > 0;
        }
    }
}

0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
public bool UpdateTimetable(int id, string subject, string room, string timeslot)
{
    using (var con = DatabaseConfiguration.GetConnection())
    {
        string query = @"UPDATE timetable
                         SET Subject = @subject, Room = @room, Timeslot = @timeslot
                         WHERE TimetableId = @id";
        using (var cmd = new SQLiteCommand(query, con))
        {
            cmd.Parameters.AddWithValue("@subject", subject);
            cmd.Parameters.AddWithValue("@room", room);
            cmd.Parameters.AddWithValue("@timeslot", timeslot);
            cmd.Parameters.AddWithValue("@id", id);
            return cmd.ExecuteNonQuery() > 0;
        }
    }
}

0 references | Yapes Nixon, 20 hours ago | 1 author, 1 change
public bool DeleteTimetable(int id)
{
    using (var con = DatabaseConfiguration.GetConnection())
    {
        string query = "DELETE FROM timetable WHERE TimetableId = @id";
        using (var cmd = new SQLiteCommand(query, con))
        {
            cmd.Parameters.AddWithValue("@id", id);
            return cmd.ExecuteNonQuery() > 0;
        }
    }
}
```

Coding

a. Object-Oriented Programming

Inheritance Student, Lecturer, and Admin classes inherit from a base User class to share common properties such as Username, Password, and Role.

Polymorphism Role-based functionality is implemented through polymorphic methods, such as overriding DisplayDashboard() to load features based on the authenticated user's role.

b. MVC Pattern

The application is structured using the Model-View-Controller design pattern:

- **Models** encapsulate business logic and data structures
- **Views** are responsible for user interface components (WinForms)
- **Controllers** coordinate data flow, UI events, and database interaction

This structure promotes separation of concerns, making the system modular, scalable, and easier to test and maintain.

References

- Microsoft Documentation: C# Programming and WinForms
- SQLite Official Documentation
- System.Data.SQLite NuGet package
- Tutorials on MVC implementation in desktop applications

- Stack Overflow community guidance
- Visual Studio Form Designer documentation