Unicom *TIC*

Skills Today, Success Tomorrow

UNICOM TIC MANAGEMENT SYSTEM

FORM REPORT

Unicom TIC Management System - Report

Preview

You want the application to first show a "Welcome" screen with the title "UnicomTIC Management System". When the user clicks this form, it should navigate to the Main Login form.

Login Modules

Main Login

- Admin Login (Full Management)

- Admin
- Admin@123

- User Login (Role-based: Lecturer, Staff, Student)

- User
- User@123
- Lecturer
- Lecturer@123
- NewStaff
- NewStaff@123
- Student
- Student@123

Admin Access Modules

Student Management

- Name
- Address
- Gender
- Date of Birth
- Subject Name
- Mobile Number
- Email ID

Subject Management

- Subject Name
- Date
- Time

Exam Management

- Subject Name
- Date
- Time

Lecturer Management

- Name
- Subject Name
- Address
- Gender
- Date of Birth
- Mobile Number
- Email ID

Staff Management

- Name
- Address
- Gender
- Date of Birth
- Mobile Number
- Email ID

Timetable Management

- Subject Name
- Lecturer Name
- Date

Marks Management

- Student Name
- Subject Name
- Marks

Attendance Management

- Student Name
- Subject Name
- Date
- Status: Present, Absent, Late, Excused

👥 User Login Role-Based Features

Lecturer

- View Timetable
- View Subject
- Access Exam Management
- Access Marks Management

Staff

- View Timetable
- View Attendance
- Access Exam Management
- Access Marks Management

Student

- View Timetable
- View Subject
- View Exams
- View Marks
- View Attendance

MVC Folder Structure Documentation

Folder: Model

File Name	Description
Admin.cs	Admin user data
Attendance.cs	Attendance records
Exam.cs	Exam details
Lecturer.cs	Lecturer information
Mark.cs	Student marks

NewStaff.cs	New staff information
Students.cs	Student data (name, dob, etc.)
Subject.cs	Subject information
Timetable.cs	Timetable entries
User.cs	Generic user login information

Folder: Controllers

File Name	Description
AttendanceController.cs	Manages attendance logic
ExamController.cs	Manages exam logic
LecturerController.cs	Handles lecturer data
MarkController.cs	Controls marks-related operations
StaffController.cs	Handles staff logic
StudentController.cs	CRUD operations for student data
SubjectController.cs	Logic for managing subjects
TimetableController.cs	Manages timetable data

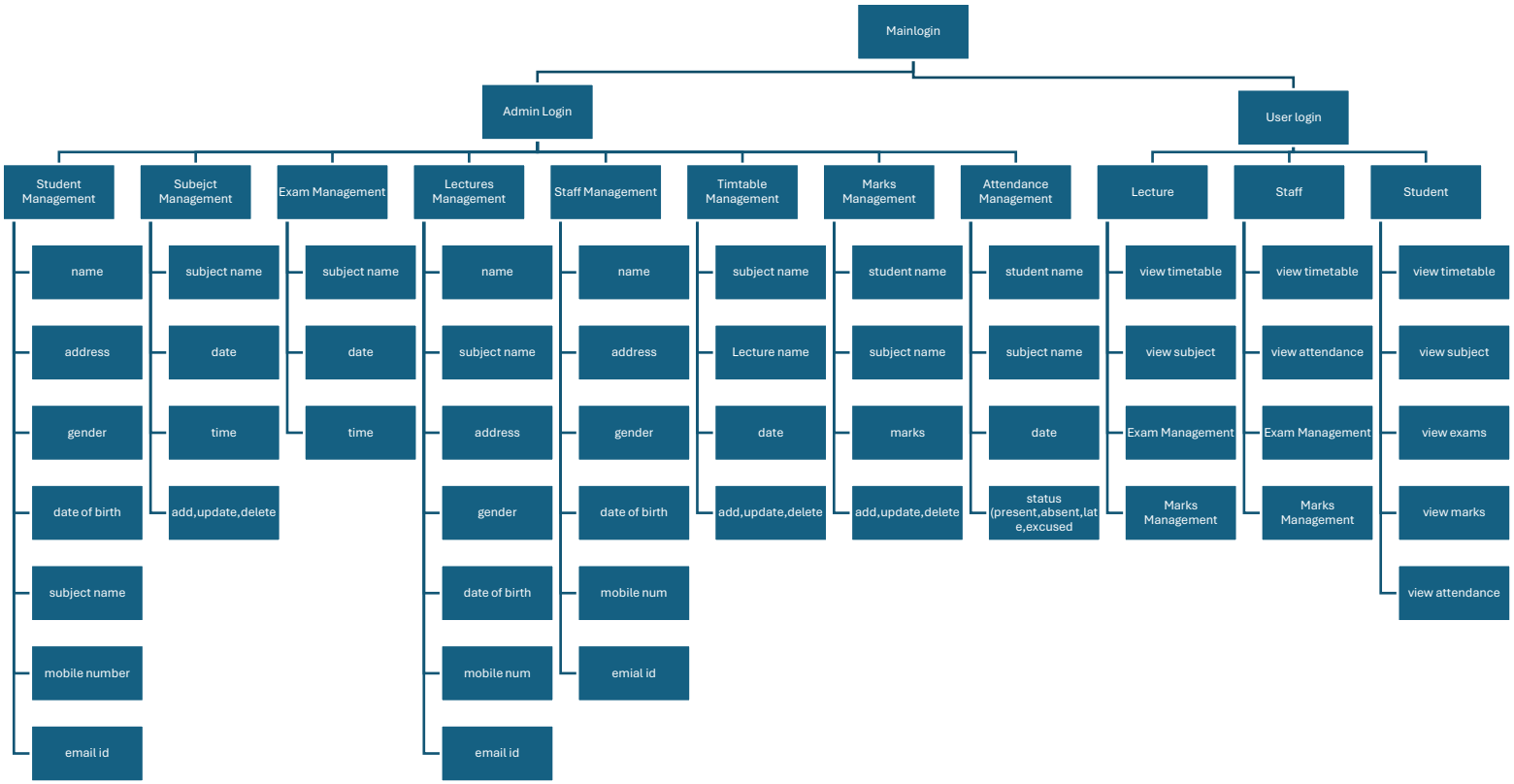
Folder: Views

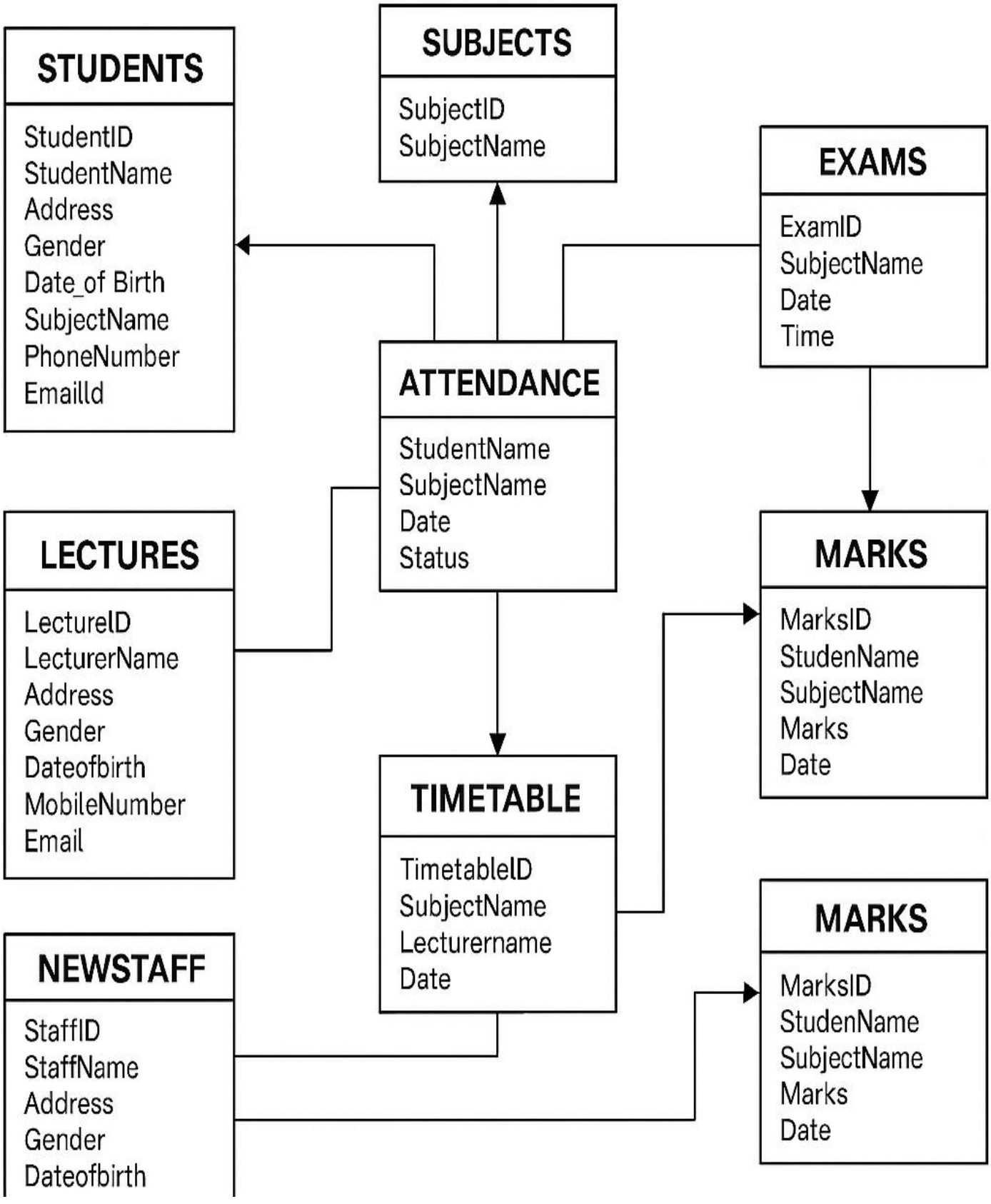
File Name	Description
Admin Interface.cs	Admin dashboard
Attendance Management.cs	Attendance input UI
Exam Management.cs	UI for managing exams
Lecturer Management.cs	UI for managing lecturers
LecturerForm.cs	Lecturer-side form
Mainlogin.cs	Main login form
Management.cs	Possibly a shared or main form
Marks Management.cs	Manage student marks
Staff Management.cs	Staff management UI
StaffForm.cs	Form used by staff
Student Management.cs	UI for managing students
Student.cs	Possibly a student dashboard
Subject Management.cs	Form to manage subjects
Timetable Management.cs	Create/update/delete timetable
User Login.cs	Login form for normal users
ViewAttendance.cs	Lecturer/Student view attendance
ViewExams.cs	Lecturer/Student view exams

ViewMarks.cs	Lecturer/Student view marks
ViewSubjects.cs	Subject viewer form
ViewTimetable.cs	View timetable form

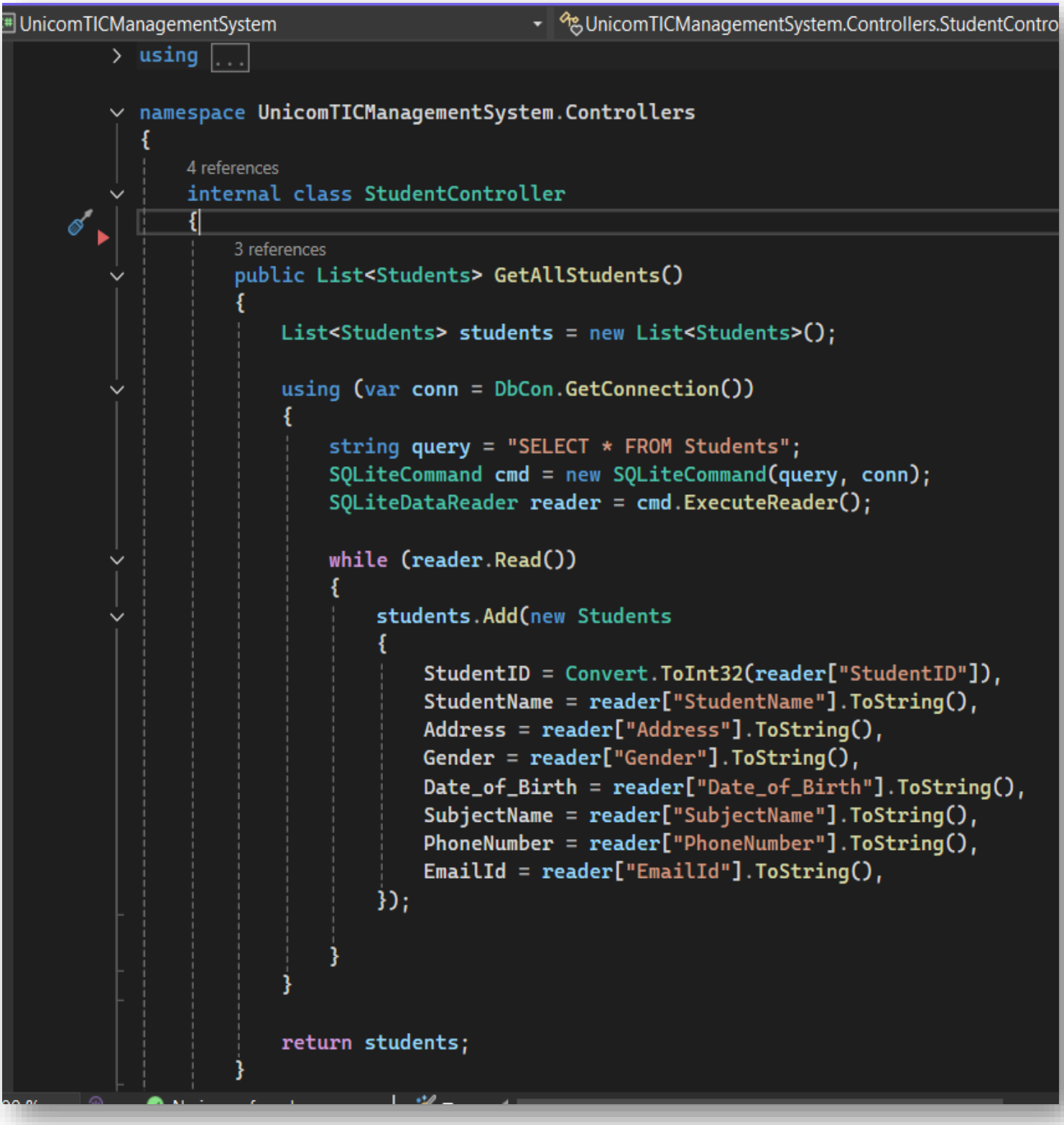
Folder: Repositories

File Name	Description
DbCon.cs	SQLite connection management
DatabaseManager.cs	Table creation, initialization scripts etc.



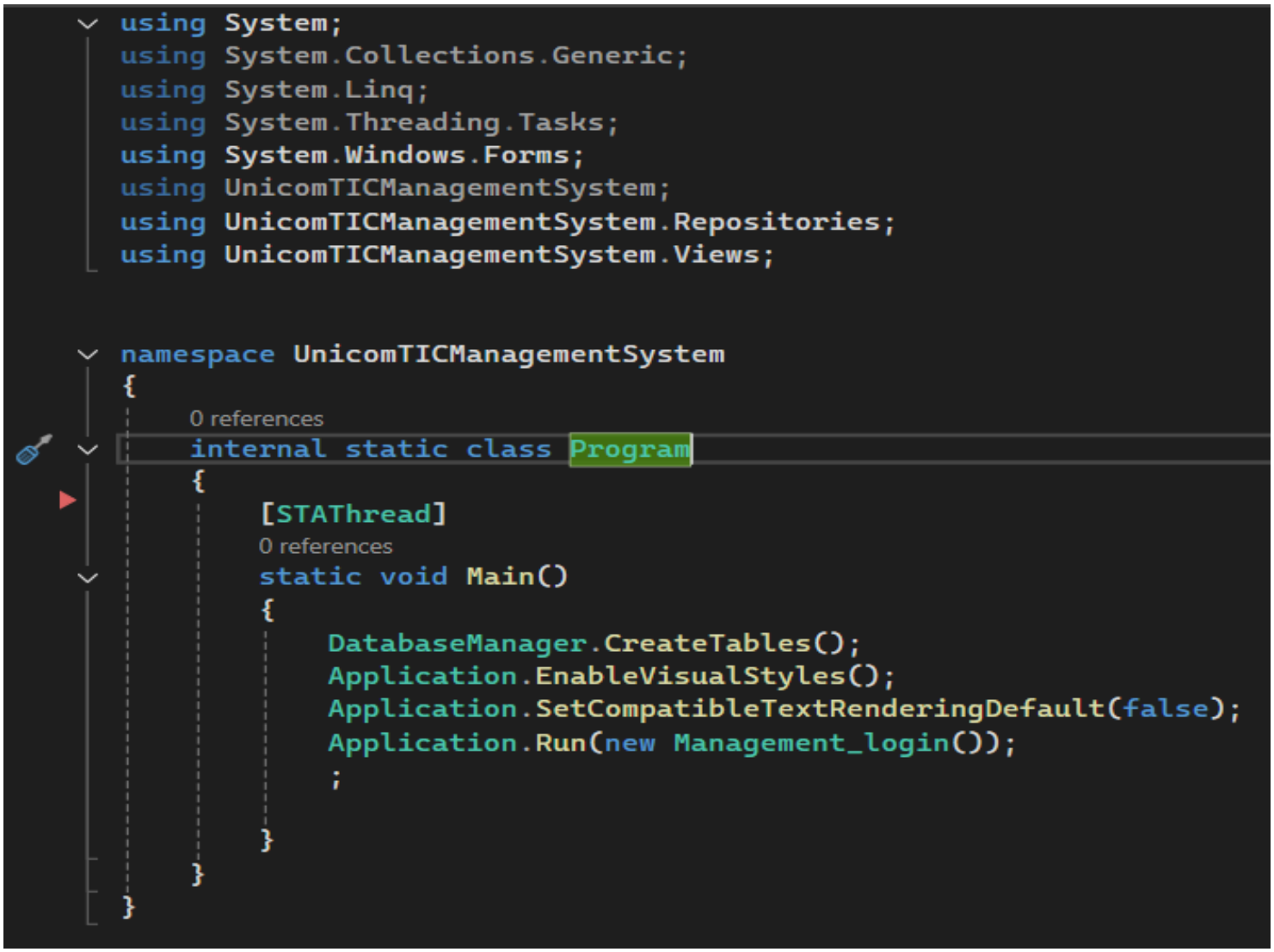

STUDENTS
StudentID
StudentName
Address
Gender
Date_of Birth
SubjectName
PhoneNumber
Emailld
SUBJECTS
SubjectID
SubjectName
EXAMS
ExamID
SubjectName
Date
Time
ATTENDANCE
StudentName
SubjectName
Date
Status
LECTURES
LectureID
LecturerName
Address
Gender
Dateofbirth
MobileNumber
Email
MARKS
MarksID
StudenName
SubjectName
Marks
Date
TIMETABLE
TimetableID
SubjectName
Lecturername
Date
NEWSTAFF
StaffID
StaffName
Address
Gender
Dateofbirth
MARKS
MarksID
StudenName
SubjectName
Marks
Date

Code Samples (Screenshots)



```
1 reference
public void AddStudent(Students student)
{
    using (var conn = DbCon.GetConnection())
    {
        string query = "INSERT INTO Students (StudentName, Address, Gender, Date_of_Birth, SubjectName, PhoneNumber, EmailId) VALUES (@StudentName, @Address, @Gender, @Date_of_Birth, @SubjectName, @PhoneNumber, @EmailId)";

        SQLiteCommand cmd = new SQLiteCommand(query, conn);
        cmd.Parameters.AddWithValue("@StudentName", student.StudentName);
        cmd.Parameters.AddWithValue("@Address", student.Address);
        cmd.Parameters.AddWithValue("@Gender", student.Gender);
        cmd.Parameters.AddWithValue("@Date_of_Birth", student.Date_of_Birth);
        cmd.Parameters.AddWithValue("@SubjectName", student.SubjectName);
        cmd.Parameters.AddWithValue("@PhoneNumber", student.PhoneNumber);
        cmd.Parameters.AddWithValue("@EmailId", student.EmailId);
        cmd.ExecuteNonQuery();
    }
}


1 reference
public void UpdateStudent(Students student)
{
    using (var conn = DbCon.GetConnection())
    {
        string query = "UPDATE Students SET StudentName=@StudentName, Address=@Address, Gender=@Gender, Date_of_Birth=@Date_of_Birth, SubjectName=@SubjectName, PhoneNumber=@PhoneNumber, EmailId=@EmailId  WHERE StudentID=@StudentID";
        SQLiteCommand cmd = new SQLiteCommand(query, conn);
        cmd.Parameters.AddWithValue("@StudentName", student.StudentName);
        cmd.Parameters.AddWithValue("@Address", student.Address);
        cmd.Parameters.AddWithValue("@Gender", student.Gender);
        cmd.Parameters.AddWithValue("@Date_of_Birth", student.Date_of_Birth);
        cmd.Parameters.AddWithValue("@SubjectName", student.SubjectName);
        cmd.Parameters.AddWithValue("@PhoneNumber", student.PhoneNumber);
        cmd.Parameters.AddWithValue("@EmailId", student.EmailId);
        cmd.Parameters.AddWithValue("@StudentID", student.StudentID);
        cmd.ExecuteNonQuery();
    }
}


1 reference
public void DeleteStudent(int id)
{
    using (var conn = DbCon.GetConnection())
    {

        string query = "DELETE FROM Students WHERE StudentID=@StudentID";
        SQLiteCommand cmd = new SQLiteCommand(query, conn);
        cmd.Parameters.AddWithValue("@StudentID",id);
        cmd.ExecuteNonQuery();
    }
}
```

```
1 reference
private void loginbtn1_Click(object sender, EventArgs e)
{
    string selectedRole = rolecomboBox1.SelectedItem?.ToString();
    string enteredUsername = username.Text.Trim();
    string enteredPassword = password.Text.Trim();

    if (string.IsNullOrEmpty(selectedRole))
    {
        MessageBox.Show("Please select a role.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.Warnir
        return;
    }

    if (string.IsNullOrEmpty(enteredUsername) || string.IsNullOrEmpty(enteredPassword))
    {
        MessageBox.Show("Please enter both username and password.", "Input Error", MessageBoxButtons.OK, Me
        return;
    }

    if (selectedRole == "Lecturer")
    {
        if (enteredUsername == LecturerUsername && enteredPassword == LecturerPassword)
        {
            MessageBox.Show("Lecturer login successful!", "Success", MessageBoxButtons.OK, MessageBoxIcon.I
            Lectrer lecturerForm = new Lectrer();
            this.Hide();
            lecturerForm.Show();
        }
        else
        {
            MessageBox.Show("Invalid Admin credentials!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Err
        }
    }
    else if (selectedRole == "NewStaff")
    {
        if (enteredUsername == StaffUsername && enteredPassword == StaffPassword)
        {
            MessageBox.Show("NewStaff login successful!", "Success", MessageBoxButtons.OK, MessageBoxIcon.I
            StaffForm staff = new StaffForm();
            this.Hide();
            staff.Show();
        }
        else
        {
            MessageBox.Show("Invalid User credentials!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Erro
        }
    }
    else if (selectedRole == "Student")
    {
        if (enteredUsername == StudentUsername && enteredPassword == StudentPassword)
        {
            MessageBox.Show("Student login successful!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Ir
            Student student = new Student();
            this.Hide();
            student.Show();
        }
        else
        {
            MessageBox.Show("Invalid User credentials!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Erro
        }
    }
    else
    {
        MessageBox.Show("Please select a valid role.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using UnicomTICManagementSystem;
using UnicomTICManagementSystem.Repositories;
using UnicomTICManagementSystem.Views;


namespace UnicomTICManagementSystem
{
    0 references
    internal static class Program
    {
        [STAThread]
        0 references
        static void Main()
        {
            DatabaseManager.CreateTables();
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Management_login());
            ;

        }
    }
}
```

```
using ...

namespace UnicomTICManagementSystem.Views
{
    10 references
    public partial class Marks_Management : Form
    {
        private MarkController markcontroller = new MarkController();
        private Form previousForm;
        private string selectedRole;
        0 references
        public Marks_Management()
        {
            InitializeComponent();
            LoadMarks();
        }
        3 references
        public Marks_Management(Form prevForm, string role)
        {
            InitializeComponent();
            previousForm = prevForm;
            selectedRole = role;
            LoadMarks();
        }
        5 references
        private void LoadMarks()
        {
            marksdataGridView1.DataSource = null;
            marksdataGridView1.DataSource = markcontroller.GetAllMarks();
            marksdataGridView1.ClearSelection();
        }

        1 reference
        private void backbtn_Click(object sender, EventArgs e)
        {
            this.Hide();
            previousForm.Show();
        }

        1 reference
        private void Marks_Management_Load(object sender, EventArgs e)
        {
            var subjects = SubjectController.GetAllSubject();
            subjectname1.DataSource = subjects;
            subjectname1.DisplayMember = "SubjectName";
            subjectname1.ValueMember = "Id";

            var studentController = new StudentController();
            var students =studentController.GetAllStudents();
            studentname1.DataSource = students;
            studentname1.DisplayMember = "StudentName";
            studentname1.ValueMember = "StudentID";
        }
```

```
private void addbtn_Click(object sender, EventArgs e)
{
    var Marks = new Mark
    {
        StudentName = studentname1.Text,
        SubjectName = subjectname1.Text,
        Marks = marks1.Text,
        Date = dateTextBox1.Text,
    };
    markcontroller.AddMark(Marks);
    LoadMarks();

}

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

    if (marksdataGridView1.SelectedRows.Count > 0)
    {
        int MarksID = Convert.ToInt32(marksdataGridView1.SelectedRows[0].Cells["MarksID"].Value);
        var Marks = new Mark
        {
            MarksID = MarksID,
            StudentName = studentname1.Text,
            SubjectName = subjectname1.Text,
            Marks = marks1.Text.Trim(),
            Date = dateTextBox1.Text,
        };
        markcontroller.UpdateMark(Marks);
        LoadMarks();

    }
}

1 reference
private void deletebtn_Click(object sender, EventArgs e)
{
    if (marksdataGridView1.SelectedRows.Count > 0)
    {
        int MarksID = Convert.ToInt32(marksdataGridView1.SelectedRows[0].Cells["MarksId"].Value);
        markcontroller.DeleteMark(MarksID);
        LoadMarks();

    }
}

1 reference
private void marksdataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (marksdataGridView1.SelectedRows.Count > 0)
    {
        marksdataGridView1.Text = marksdataGridView1.SelectedRows[0].Cells["MarksID"].Value.ToString();
    }
}
```