Project Guide

Table of Contents

1. Introduction to Unicom Management System

The Unicom Management System is a considerable software solution designed to efficiently manage and streamline the day-to-day administrative and academic operations of a college. This system supports the organization in handling essential functions related to students, lecturers, and staff, and integrates several key modules such as:

- Course and Subject Management
- Class, Examination and Marks Management
- Timetable and Hall Allocation
- Attendance Tracking
- Student Information Management

Each module is interconnected to ensure seamless data flow and operational efficiency, reducing manual workload and improving accuracy across departments. With user-friendly interfaces and secure role-based access.

2. System Overview

2.1. System Functionality

The system includes a centralized dashboard accessible by the Admin, which provides access to all system features. Students have restricted access, allowing them to view only their personal details within the Student Management module. Lecturers can access modules related to Exams and Marks, Attendance Management, and Student Information for all students. Staff members have access to Timetable and Hall Management as well as Attendance Management.

The system operation begins with the Admin, who is responsible for creating and managing user accounts by assigning specific roles. This ensures that only registered users can access the system. When a new user is created with an assigned role by the Admin, they can then register their account and receive a username and password. Upon logging in, the system automatically detects the user's role and grants access permissions accordingly, ensuring that each user can only access features relevant to their role.

2.2. Steps of Operation in the Unicom Management System

1. **Register users by role**
   The Admin registers users by assigning roles (Student, Lecturer, Staff), ensuring role-based access to system features.
2. **Create courses and assign subjects of student**
   Courses are created to student and linked with relevant subjects to organize academic structure.
3. **Create exams/classes**
   Exams and classes are added and assigned to subjects for academic planning and assessment.
4. **Class/Lap Allocation**
   Halls are scheduled and allocated based on sessions and course requirements.
5. **Enter marks and generate grades**
   Lecturers input student marks for each exam under the appropriate subject.
6. **Generate Timetables**
   Timetables are generated based on lecturers, halls, dates, and time slots to manage class schedules efficiently.
7. **Record attendance by subject/date**
   Attendance is recorded for students, lecturers, and staff for each session.
8. **Student Access to Information**
   Students can securely view their **timetable**, **exam marks**, and **attendance records** from their dashboard.

3. Key Features Implemented

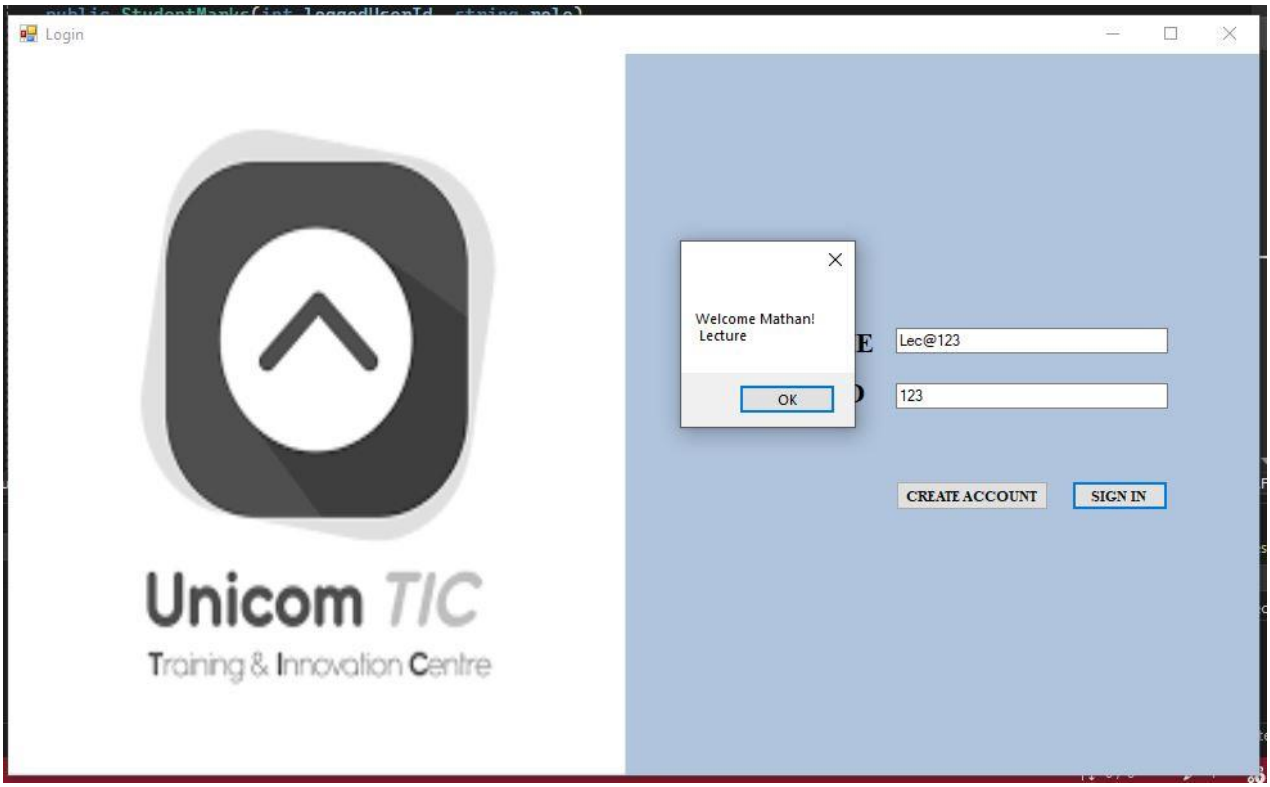
3.1. Role-based notifications on login

I have implemented role-based notifications that appear when a user logs into the system, displaying a personalized welcome message and informing them of their available features and access level based on their assigned role (Admin, Lecturer, Student, or Staff).

Default User Name Password is:

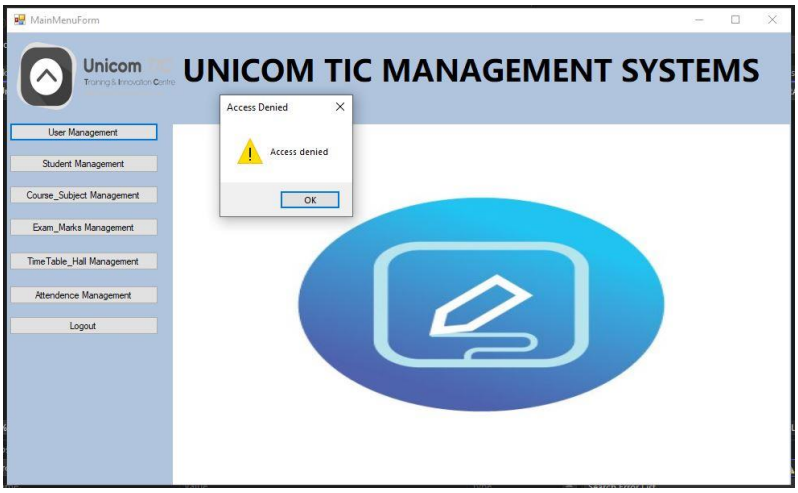
UserName : Admin

PassWord : 123



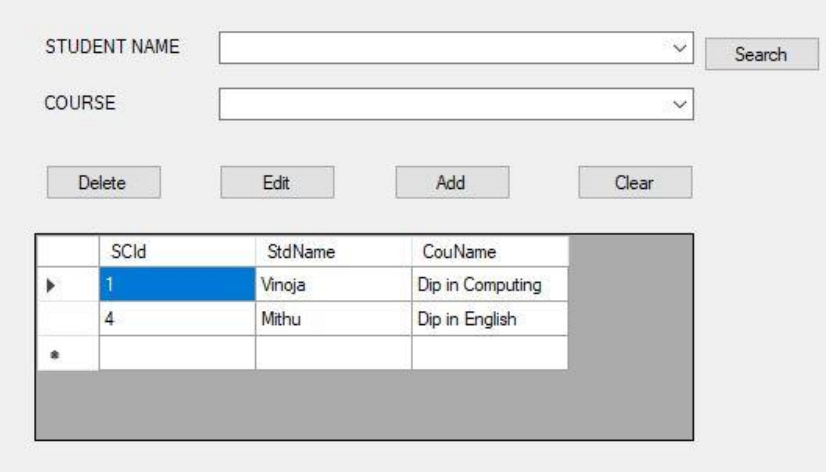
3.2. Secure role-based access control

The system provides user-based access control, allowing each user to access only the modules and features permitted by their assigned role



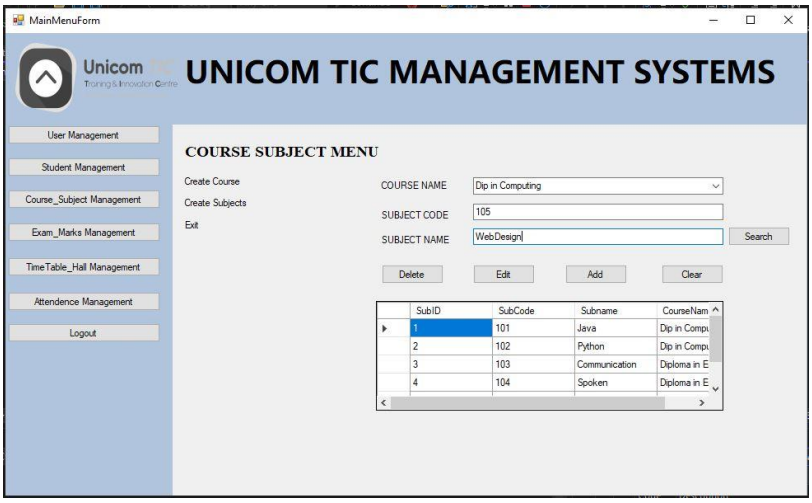
3.3. CRUD operations features in all forms

All forms in the system support full CRUD operations, including Add, Edit, Delete, Search, and Clear functionalities, ensuring efficient data management across all modules.



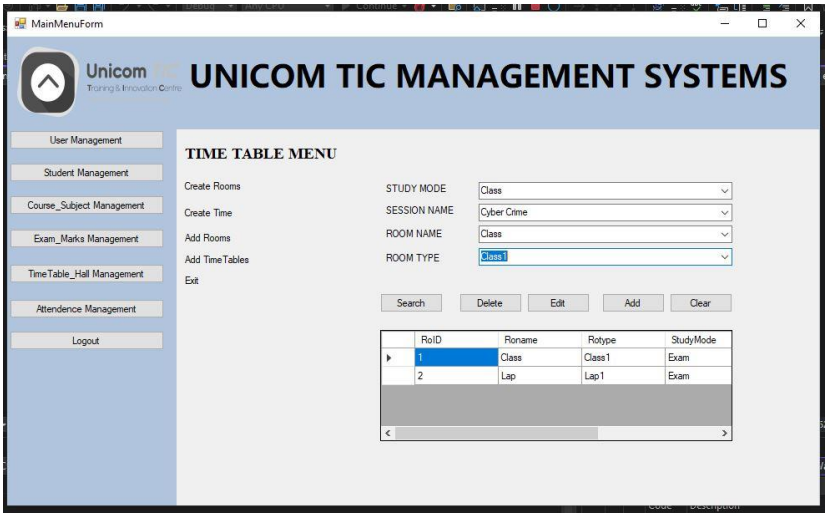
3.4. Course_Subject management under specific courses

The system allows subjects to be added based on the selected course of enrolled student, ensuring proper organization and alignment of subjects under their respective courses.



3.5. Dynamic hall/classroom allocation

The system allocates classrooms and Laps based on the requirements of scheduled exams or class sessions, ensuring efficient room utilization and session planning.

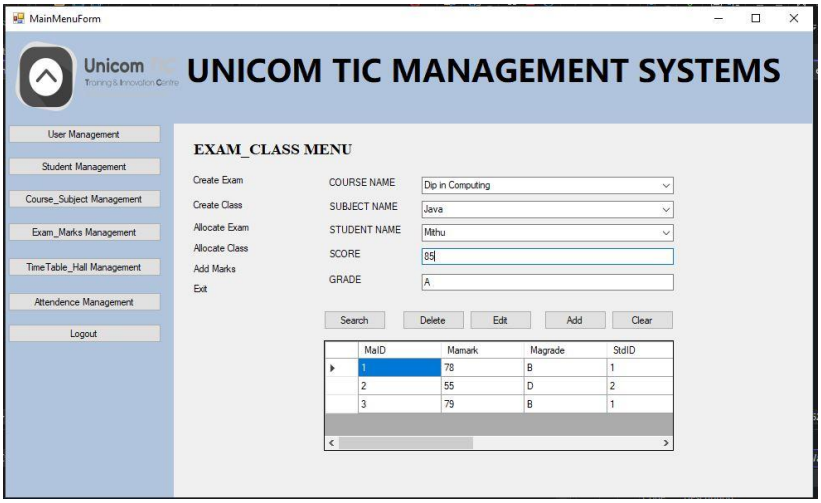


3.6. Automatic Student Attendance System

When a student logs in, the system automatically records attendance by marking them as "Present" for subjects in their enrolled course, using the current date.

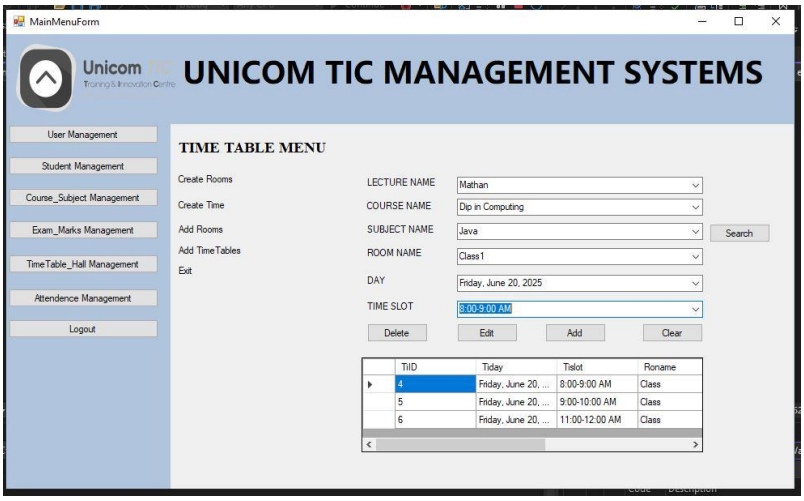
3.7. Auto-grade generation on mark entry

Marks are added based on the student's name and subject. Upon entering the mark, the system automatically calculates and displays the corresponding grade, which is then stored along with the record



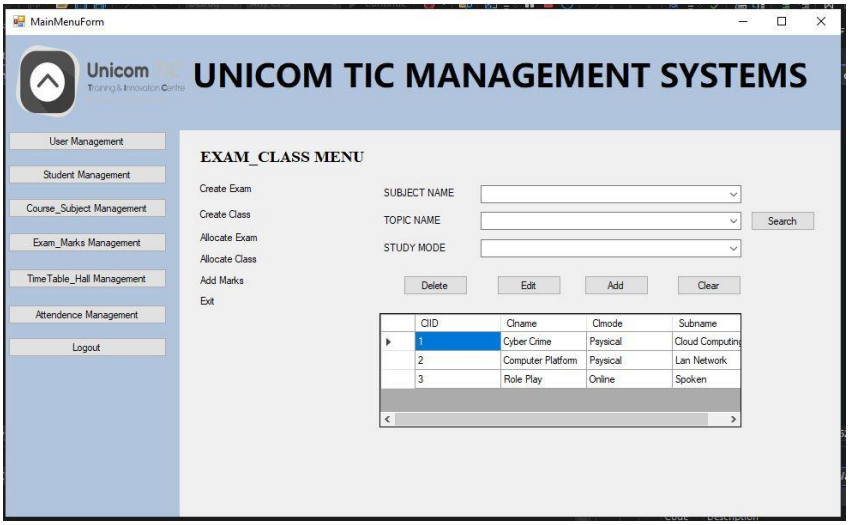
3.8. Timetable generation with clash detection(Without Duplicates)

The timetable is created using the lecturer's name, course name, subject name, room, day, and time. The system prevents duplicate time allocations, ensuring that a classroom cannot be assigned to more than one lecture at the same time.



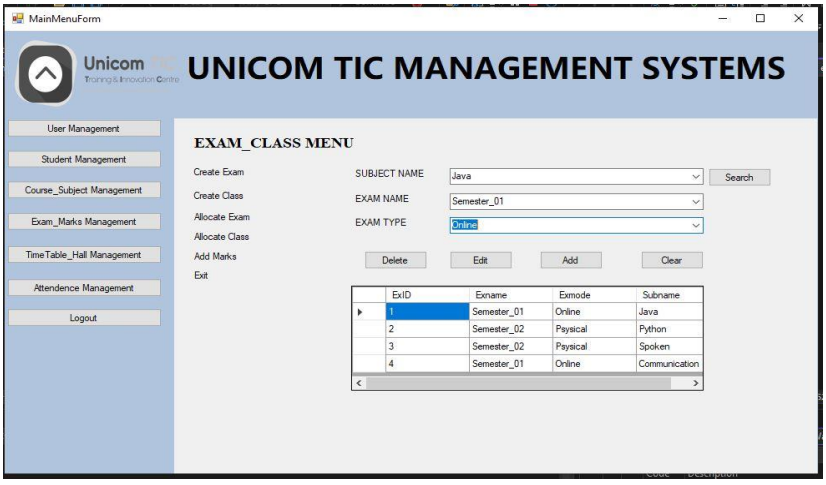
3.9. Class Allocation by Topic name

Classes are allocated with specific topic names to organize and manage the academic sessions effectively



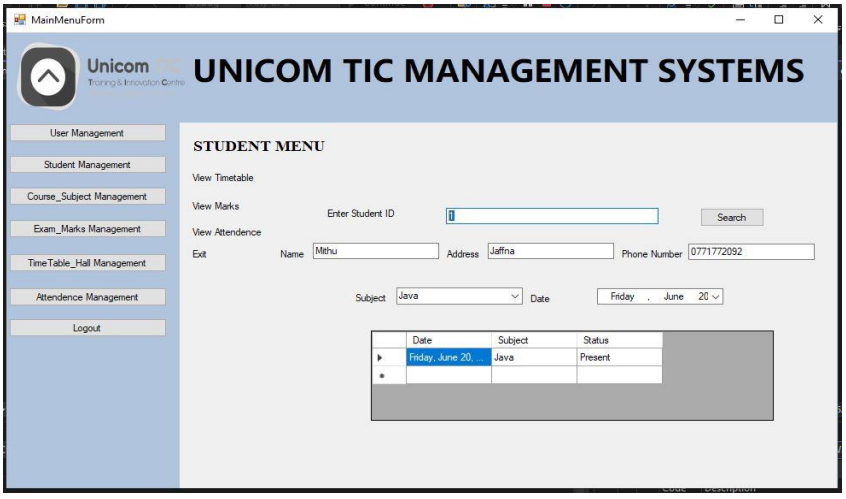
3.10. Exam Allocation by Specific Subject

Exams are allocated to specific subjects to organize assessments accurately and streamline exam management.



3.11. Student portal for marks/timetable/attendance view

Admin, Staff, and Lecturers can view a student's timetable, marks, and attendance by entering the student ID, enabling comprehensive monitoring of student performance and schedules. Students can view their timetable, marks, and attendance by logging in with their username and password, providing secure access to their academic information.



4. Technologies used

Environment: Visual Studio with C# WinForms.

Database: SQLite, using System.Data.SQLite.Core (install via NuGet).

User Interface: WinForms with Form Designer, using Button, TextBox, ComboBox, DataGridView,Date timePicker.

OOP:

- Class and Object : Student, Course, Subject, Lecture, Timetable, etc. are classes.When used, each class creates specific objects like a particular student or course.
- Encapsulation : Wrapping related properties and methods into classes. TimetableController encapsulates all logic for adding, updating, and deleting timetable entries. Data access is hidden behind these methods.
- Abstraction : Hides complex logic behind simple interfaces. Dbconfig.GetConnection() hides SQLite connection details.Forms interact with Controllers instead of directly running SQL.
- Inheritance : We have a BaseForm for shared UI code, other forms inherit from it.
- Association: Student linked with Course through StudentCourse table.
- Composition: Room composed of Exam or Class details depending on context.

Design Pattern: MVC: Keep data, forms, and logic separate.

Error Handling: Check inputs (e.g., Phone number is 10 digits), Show messages (e.g., “Wrong User name and password” or “Select Subject”), Handle database errors (e.g., “Database is locked , No such column name m.score ”).

5. Code Highlights

```
namespace UnicomTICManagementSystem.Controllers
{
    internal class AddClassController
    {
        public void InsertClass(string name, string code)
        {
            string insertQuery = "INSERT INTO AddClasses (ClName, ClMode) VALUES (@ClName, @ClMode)";
            using (var conn = Dbconfig.GetConnection())
            using (var cmd = new SQLiteCommand(insertQuery, conn))
            {
                cmd.Parameters.AddWithValue("@ClName", name);   // Clname.Text (Topic Name)
                cmd.Parameters.AddWithValue("@ClMode", code);   // Clcode.Text (Study Mode)
                cmd.ExecuteNonQuery();
                MessageBox.Show("Class inserted successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        public void UpdateClass(int id, string name, string code)
        {
            string updateQuery = "UPDATE AddClasses SET ClName = @ClName, ClMode = @ClMode WHERE ClId = @ClId";
            using (var conn = Dbconfig.GetConnection())
            using (var cmd = new SQLiteCommand(updateQuery, conn))
            {
                cmd.Parameters.AddWithValue("@ClId", id);
                cmd.Parameters.AddWithValue("@ClName", name);   // Clname.Text
                cmd.Parameters.AddWithValue("@ClMode", code);   // Clcode.Text
                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                    MessageBox.Show("Class updated successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                else
                    MessageBox.Show("Class not found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        public void DeleteClass(int id)
```

CRUD Operations

```
1   using System;
2   using System.Collections.Generic;
3   using System.Data.SQLite;
4   using System.Linq;
5   using System.Text;
6   using System.Threading.Tasks;
7
8   namespace UnicomTICManagementSystem.Data
9   {
10      internal class Dbconfig
11      {
12          private static string connectionString = "Data Source=Unicomtic.db;Version=3;";
13
14          public static SQLiteConnection GetConnection()
15          {
16              SQLiteConnection conn = new SQLiteConnection(connectionString);
17              conn.Open();
18              return conn;
19          }
20      }
21  }
22
```

Database Connection

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnicomTICManagementSystem.Models
{
    internal class Attendence
    {
        public int AttendID { get; set; }
        public string Statusday { get; set; }
        public int StatusID { get; set; }
        public string Status { get; set; }
        public int SubID { get; set; }
        public string Subname { get; set; }
        public int StudentID { get; set; }
        public string StudentName { get; set; }
    }
}
```

Encapsulation class

```
        cmd.CommandText = @"
                RoomId INTEGER NOT NULL,
                CourseID INTEGER NOT NULL,
                SubID INTEGER NOT NULL,
                LecID INTEGER NOT NULL,
                FOREIGN KEY (RoomId) REFERENCES Rooms(RoomId),
                FOREIGN KEY (CourseID) REFERENCES Courses(CouId),
                FOREIGN KEY (SubID) REFERENCES Subjects(SubjectId),
                FOREIGN KEY (LecID ) REFERENCES lectures(LecId)
            );

                CREATE TABLE IF NOT EXISTS Attendances (
                AttenId INTEGER PRIMARY KEY AUTOINCREMENT,
                Date Text NOT NULL,
                StdId INTEGER NOT NULL,
                SubjectId INTEGER NOT NULL,
                StatusId INTEGER NOT NULL,
                FOREIGN KEY (StdId) REFERENCES Students(StdId),
                FOREIGN KEY (StatusId) REFERENCES AddStatus(StatusId),
                FOREIGN KEY (SubjectId) REFERENCES Subjects(SubjectId)

            );

                CREATE TABLE IF NOT EXISTS Marks (
                MarksId INTEGER PRIMARY KEY AUTOINCREMENT,
                MarkScore INTEGER NOT NULL,
                MarksGrade TEXT NOT NULL,
                StdId INTEGER NOT NULL,
                CourseId INTEGER NOT NULL,
                SubjectId INTEGER NOT NULL,
                FOREIGN KEY (StdId) REFERENCES Students(StdId),
                FOREIGN KEY (CourseId) REFERENCES Courses(CouId),
                FOREIGN KEY (SubjectId) REFERENCES Subjects(SubjectId)
```

Foreign Key for Create Tables

```
        );

            INSERT INTO Roles (RoleCode, RoleName)
            SELECT 'R001', 'Admin'
            WHERE NOT EXISTS (SELECT 1 FROM Roles WHERE RoleName = 'Admin');

            INSERT INTO Users (UserName, UserPass, UserRole)
            SELECT 'Admin', 'Admin@123', 'Admin'
            WHERE NOT EXISTS (SELECT 1 FROM Users WHERE UserName = 'Admin');

            INSERT INTO Staffs (StaffName, StaffPhone, StaffAddress, UserId)
            SELECT 'Default Admin', '0000000000', 'Admin Office', UserId
            FROM Users
            WHERE UserName = 'Admin'
            AND NOT EXISTS (
            SELECT 1 FROM Staffs
            WHERE UserId = (SELECT UserId FROM Users WHERE UserName = 'Admin')
        );
    ";

    cmd.ExecuteNonQuery();

}
```

Default username Password for Admin

```
{
    string userQuery = "SELECT * FROM Users WHERE UserName = @username AND UserPass = @password";
    using (var userCmd = new SQLiteCommand(userQuery, conn))
    {
        userCmd.Parameters.AddWithValue("@username", username);
        userCmd.Parameters.AddWithValue("@password", password);

        using (var reader = userCmd.ExecuteReader())
        {
            if (reader.Read())
            {
                int userId = Convert.ToInt32(reader["UserId"]);
                string role = reader["UserRole"].ToString();

                string fullName = "";

                if (role == "Student")
                {
                    fullName = GetNameFromTable(conn, "Students", "StdName", userId);
                }
                else if (role == "Staff")
                {
                    fullName = GetNameFromTable(conn, "Staffs", "StaffName", userId);
                }
                else if (role == "Lecture")
                {
                    fullName = GetNameFromTable(conn, "Lectures", "LecName", userId);
                }
                else
                { fullName = "Admin";
                }

                MessageBox.Show($"Welcome {fullName}!\n {role}");
```

Log in Role and name Display

```
public List<Timetable> GetAllTimetables()
{
    var list = new List<Timetable>();
    using (var conn = Dbconfig.GetConnection())
    {
        var cmd = new SQLiteCommand(@"
    SELECT t.TimeId, t.TimeDay, t.TimeSlot,
           t.RoomId, r.RooomName,
           t.CourseID, c.CouName,
           t.SubID, s.SubjectName,
           t.LecID, l.LecName
    FROM TimeTables t
    LEFT JOIN Rooms r ON t.RoomId = r.RoomId
    LEFT JOIN Courses c ON t.CourseID = c.CouId
    LEFT JOIN Subjects s ON t.SubID = s.SubjectId
    LEFT JOIN lectures l ON t.LecID = l.LecId", conn);

        using (var reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                list.Add(new Timetable
                {
                    TiID = reader.GetInt32(0),
                    Tiday = reader.GetString(1),
                    Tislot = reader.GetString(2),
                    RoID = reader.GetInt32(3),
                    Roname = reader.IsDBNull(4) ? "" : reader.GetString(4),
                    CourseID = reader.GetInt32(5),
                    CourseName = reader.IsDBNull(6) ? "" : reader.GetString(6),
                    SubID = reader.GetInt32(7),
                    Subname = reader.IsDBNull(8) ? "" : reader.GetString(8),
                    LecID = reader.GetInt32(9),
                    LecName = reader.IsDBNull(10) ? "" : reader.GetString(10)
                });
```

Return List Method

```
public List<Course> GetCoursesByStudent(int studentId)
{
    var courses = new List<Course>();

    using (var conn = Dbconfig.GetConnection())
    {
        var cmd = conn.CreateCommand();
        cmd.CommandText = @"
            SELECT c.CouId, c.CouCode, c.CouName
            FROM Courses c
            INNER JOIN StudentCourses sc ON c.CouId = sc.CourseId
            WHERE sc.StudentId = @studentId";
        cmd.Parameters.AddWithValue("@studentId", studentId);

        using (var reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                courses.Add(new Course
                {
                    CourseID = reader.GetInt32(0),
                    CourseCode = reader.GetString(1),
                    CourseName = reader.GetString(2)
                });
            }
```

Get Course for student

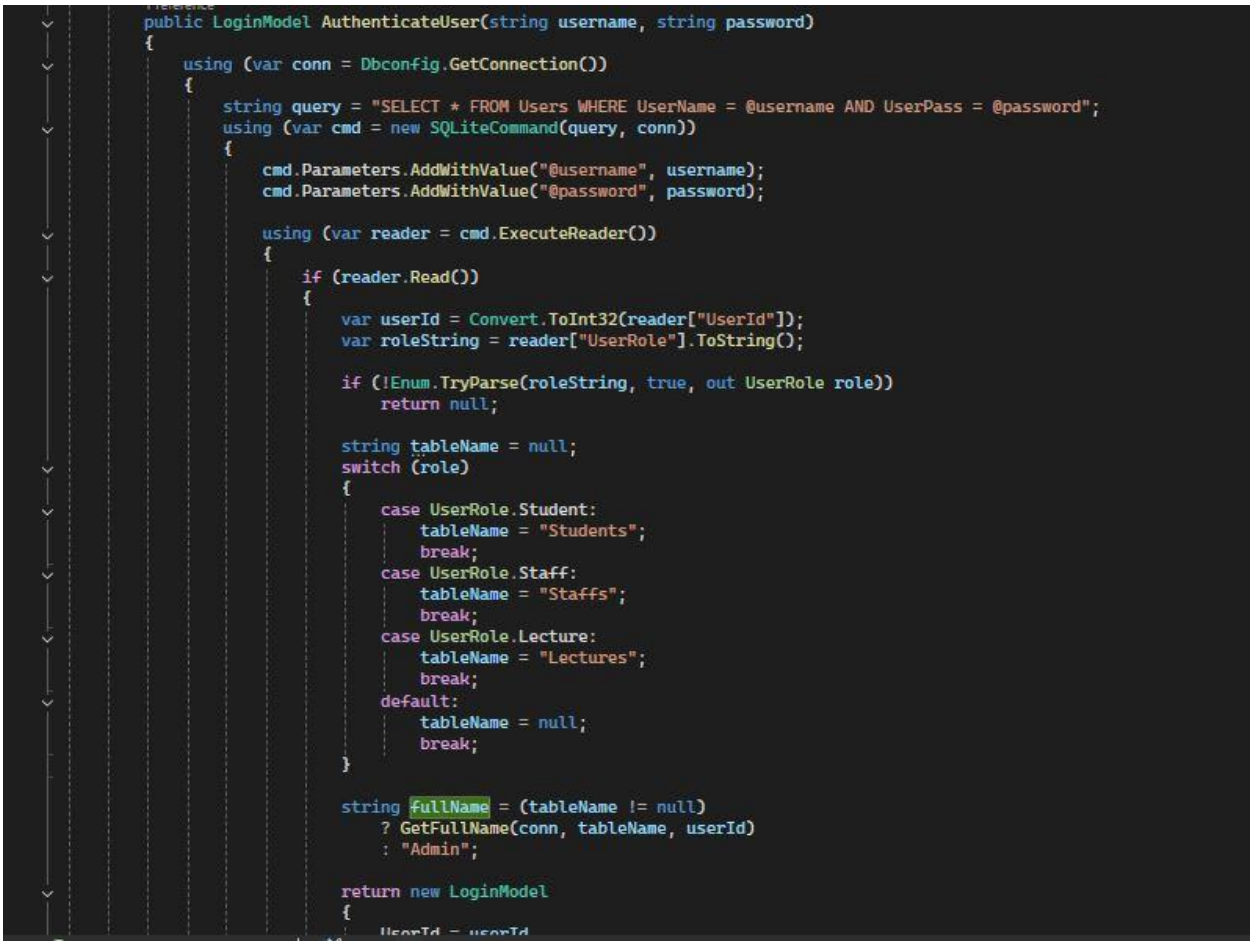
```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using UnicomTICManagementSystem.Models;

namespace UnicomTICManagementSystem
{
    public partial class MainMenuForm : Form
    {
        private readonly string userRole;
        private readonly int loggedInUserId;

        public MainMenuForm(int userId, string role)
        {
            InitializeComponent();
            loggedInUserId = userId;
            userRole = role;
        }

        public void loadform(object Form)
        {
            if (this.mainpanel.Controls.Count > 0)
                this.mainpanel.Controls.RemoveAt(0);
            Form f = Form as Form;
            f.TopLevel = false;
            f.Dock = DockStyle.Fill;
            this.mainpanel.Controls.Add(f);
            this.mainpanel.Tag = f;
            f.Show();
        }
        private void MainMenuForm_Load(object sender, EventArgs e)
```

User Information Transfer Through Form Constructors



```
public LoginModel AuthenticateUser(string username, string password)
{
    using (var conn = Dbconfig.GetConnection())
    {
        string query = "SELECT * FROM Users WHERE UserName = @username AND UserPass = @password";
        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@username", username);
            cmd.Parameters.AddWithValue("@password", password);

            using (var reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    var userId = Convert.ToInt32(reader["UserId"]);
                    var roleString = reader["UserRole"].ToString();

                    if (!Enum.TryParse(roleString, true, out UserRole role))
                        return null;

                    string tableName = null;
                    switch (role)
                    {
                        case UserRole.Student:
                            tableName = "Students";
                            break;
                        case UserRole.Staff:
                            tableName = "Staffs";
                            break;
                        case UserRole.Lecture:
                            tableName = "Lectures";
                            break;
                        default:
                            tableName = null;
                            break;
                    }

                    string fullName = (tableName != null)
                        ? GetFullName(conn, tableName, userId)
                        : "Admin";

                    return new LoginModel
                    {
                        UserId = userId
```

User Authentication Ennum Method To view Details

```
private void MarkScore_TextChanged(object sender, EventArgs e)
{
    if (int.TryParse(MarkScore.Text.Trim(), out int score))
    {
        Markgrade.Text = GetGradeFromScore(score);
    }
    else
    {
        Markgrade.Text = string.Empty;
    }
}

private void LoadCourses()
{
    var Courses = CourseControll.GetAllCourse();
    CoursecomboBox.DataSource = Courses;
    CoursecomboBox.DisplayMember = "CourseName";
    CoursecomboBox.ValueMember = "CourseID";
}
private void LoadSubjects()
{
    var subjects = subjectController.GetAllSubject();
    SelectcomboBox.DataSource = subjects;
    SelectcomboBox.DisplayMember = "Subname";
    SelectcomboBox.ValueMember = "SubID";
}

private void LoadStudents()
{
```

Load Data From Controller methods

```
    }
    1 reference
    public bool IsConflictExists(int roomId, int lecId, string timeDay, string timeSlot)
    {
        using (var conn = Dbconfig.GetConnection())
        {
            string checkQuery = @"
        SELECT COUNT(*) FROM TimeTables
        WHERE
        (LecID = @LecID AND TimeDay = @TimeDay AND TimeSlot = @TimeSlot)
        OR
        (RoomId = @RoomId AND TimeDay = @TimeDay AND TimeSlot = @TimeSlot)";

            using (var cmd = new SQLiteCommand(checkQuery, conn))
            {
                cmd.Parameters.AddWithValue("@LecID", lecId);
                cmd.Parameters.AddWithValue("@RoomId", roomId);
                cmd.Parameters.AddWithValue("@TimeDay", timeDay);
                cmd.Parameters.AddWithValue("@TimeSlot", timeSlot);

                int count = Convert.ToInt32(cmd.ExecuteScalar());
                return count > 0;
            }
        }
    }


}
```

Duplicate validation Fro Time table Entry

```
public void RecordLoginAttendance(int studentId)
{
    using (var conn = Dbconfig.GetConnection())
    {
        string formattedDate = DateTime.Now.ToString("dddd, MMMM dd, yyyy");


        string checkQuery = @"SELECT COUNT(*) FROM Attendances
                    WHERE StdId = @StdId AND Date = @Date";
        var checkCmd = new SQLiteCommand(checkQuery, conn);
        checkCmd.Parameters.AddWithValue("@StdId", studentId);
        checkCmd.Parameters.AddWithValue("@Date", formattedDate);

        long exists = (long)checkCmd.ExecuteScalar();
        if (exists > 0)
            return;

        int subjectId = GetAssignedSubjectIdForStudent(studentId, conn);
        int statusId = GetStatusIdByName("Present", conn);

        if (subjectId > 0 && statusId > 0)
        {
            var insertCmd = new SQLiteCommand(@"INSERT INTO Attendances (Date, StatusId, SubjectId, StdId)
                                    VALUES (@Date, @StatusId, @SubjectId, @StdId)", conn);
            insertCmd.Parameters.AddWithValue("@Date", formattedDate);
            insertCmd.Parameters.AddWithValue("@StatusId", statusId);
            insertCmd.Parameters.AddWithValue("@SubjectId", subjectId);
            insertCmd.Parameters.AddWithValue("@StdId", studentId);
            insertCmd.ExecuteNonQuery();
        }
    }
}
```

Auto Attendance detection

```
public Mark SearchMarkBySubjectName(string subjectName)
{
    using (var conn = Dbconfig.GetConnection())
    {
        string query = @"
    SELECT m.MarksId, m.MarkScore, m.MarksGrade,
           s.StdId, s.StdName,
           c.CouId, c.CouName,
           sub.SubjectId, sub.SubjectName
    FROM Marks m
    JOIN Students s ON m.StdId = s.StdId
    JOIN Courses c ON m.CourseId = c.CouId
    JOIN Subjects sub ON m.SubjectId = sub.SubjectId
    WHERE sub.SubjectName LIKE @subjectName
    LIMIT 1";

        using (var cmd = new SQLiteCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@subjectName", $"%{subjectName}%");

            using (var reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    return new Mark
                    {
                        MaID = Convert.ToInt32(reader["MarksId"]),
                        Mamark = reader["MarkScore"].ToString(),
                        Magrade = reader["MarksGrade"].ToString(),
                        StdID = Convert.ToInt32(reader["StdId"]),
                        Stdname = reader["StdName"].ToString(),
                        CourseId = Convert.ToInt32(reader["CouId"])
```

Search mark By Subject name

6. Challenges Faced and Solutions

1.Initial difficulty with database table relationships.

2.Understanding WinForms structure and flow.

3.Debugging “database locked” errors.

Solution: Gradual learning of table joins, UI layout best practices, and structured debugging

7. Default Credentials

Username: Admin

Password: Admin@123

8. Conclusion

The Unicom Management System successfully integrates core academic functionalities into a cohesive, secure, and user-friendly desktop application, improving the college’s operational efficiency and information access.

9. System Flow Diagram