Project Report: Unicom TIC Management System (UMS)

1. Introduction

The Unicom TIC Management System (UMS) is a desktop application developed using C# WinForms, aimed at managing the basic operations of a school. The system provides functionalities for managing courses, subjects, students, exams, marks, and timetables, along with a room allocation system for computer labs and lecture halls. It also incorporates a login system with role-based access for Admin, Staff, Students, and Lecturers. The data is stored in an SQLite database, and the application follows a Model-View-Controller (MVC) architectural pattern to separate logic, user interface, and data management.

2. Objectives

The primary objectives of the project include:

- Building a simple desktop application using C# WinForms.
- Developing core modules such as Course & Subject Management, Student Management, Exam & Marks Management, and Timetable Management.
- Creating a basic role-based login system to allow access for Admin, Staff, Students, and Lecturers.
- Using the MVC design pattern to organize data, forms, and logic.
- Integrating SQLite for data storage and management.
- Implementing simple validation and error handling.

3. System Scope and Features

The UMS offers several essential features:

- **Login System:** Role-based access for Admin, Staff, Students, and Lecturers.
- **Course & Subject Management:** Admin can add, view, edit, and delete courses and subjects. All users can view courses and subjects.
- **Student Management:** Admin can manage student details, while students can view their personal details.
- **Exam & Marks Management:** Admin and Lecturers can manage exams and marks, while students can view their results.
- **Timetable Management:** Admin can allocate rooms (labs or halls) to subjects, and all users can view the timetables.

4. Architecture and Design

- **Model-View-Controller (MVC):**
    - **Model:** Classes that represent data entities such as Course, Student, Room, etc.
    - **View:** WinForms-based UI for interaction, including forms for login, course management, and timetable management.
    - **Controller:** Handles user interactions and updates the view and model accordingly.
- **Database:** SQLite is used to store the application data, including users, courses, subjects, students, exams, marks, rooms, and timetables. The database schema includes tables like `Users`, `Courses`, `Subjects`, `Exams`, `Marks`, `Rooms`, and `Timetables`.

5. Key Features

- **Login System:** Provides different access levels based on roles (Admin, Staff, Students, Lecturer).
- **Course & Subject Management:** Admin can create courses and assign subjects. Students and other users can view them.
- **Timetable and Room Allocation:** Admin can assign rooms (either labs or halls) to specific subjects, and users can view their timetables, including room assignments.
- **Role-Based Access:** Depending on the user's role, they will have different access to features like adding data or viewing reports.

6. Technical Details

- **Programming Language:** C# using WinForms for the UI.
- **Database:** SQLite for data persistence.
- **Design Patterns:** MVC to separate logic and UI components for maintainability.
- **Error Handling:** The application uses basic validation to check for empty fields and invalid input. Appropriate error messages are displayed for common issues, such as invalid usernames or password errors.

7. Challenges and Solutions

- **Login System:** Ensuring correct role-based access was a challenge. It was resolved by using a clear role identification system based on the login credentials.
- **Database Management:** Setting up SQLite and integrating it with the application required handling asynchronous operations for smooth performance.

- **User Interface:** Designing a clean, user-friendly interface that was easy to navigate and interacted seamlessly with the backend.

8. Conclusion

The Unicom TIC Management System is a simple, yet effective application for managing school operations. It integrates role-based access with a clean UI, while utilizing an MVC architecture for maintainable and scalable code. By leveraging SQLite, the system provides efficient data storage and management. The project successfully meets its goals and provides an easy-to-use solution for basic school management tasks.

9. Deliverables

- **Source Code:** Complete project files including all forms, controllers, and models.
- **Database:** The SQLite database (`unicomtic.db`) containing all the required tables.
- **Demo:** A demonstration of the app, showing how to login, manage courses, and generate timetables.

```
private string CheckTime()
{
    if (DateTime.TryParseExact(time_from.Text, "hh:mm tt", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime timeFrom) &&
        DateTime.TryParseExact(time_to.Text, "hh:mm tt", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime timeTo))
    {
        if (timeFrom < timeTo)
        {
            List<Date_Time_mo> examSlots = examController.GetTime(date.Text);

            foreach (var slot in examSlots)
            {
                string[] times = slot.Time.Split('-');
                if (times.Length != 2)
                    continue;

                DateTime existingStart = DateTime.Parse(times[0].Trim());
                DateTime existingEnd = DateTime.Parse(times[1].Trim());

                if (timeFrom < existingEnd && existingStart < timeTo)
                {
                    if (check != 3)
                    {
                        MessageBox.Show("Time conflict detected with another exam.");

                        return null;
                    }
                }

            }

            return timeFrom.ToString("hh:mm tt") + " - " + timeTo.ToString("hh:mm tt");
        }
        else
        {
            MessageBox.Show("Start time must be earlier than end time.");
            return null;
        }
    }
    else
    {
        MessageBox.Show("Invalid time format. Please use format like 07:00 AM.");
        return null;
    }
}
```

- **We can only have one exam at a time on the same day.**

```
private string CheckTime()
{
    string[] formats = { "h:mm tt", "hh:mm tt" };

    if (DateTime.TryParseExact(time_from.Text.Trim(), formats, CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime timeFrom) &&
        DateTime.TryParseExact(time_to.Text.Trim(), formats, CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime timeTo))
    {
        if (timeFrom < timeTo)
        {

            List<Date_Time_mo> examSlots = timeTableController.GetTime_Date(roomname_combobox.Text);

            foreach (var slot in examSlots)
            {

                string[] times = slot.Time.Split('-');
                if (times.Length != 2)
                    continue;

                DateTime existingStart = DateTime.Parse(times[0].Trim());
                DateTime existingEnd = DateTime.Parse(times[1].Trim());


                if (date.Text == slot.Date &&
                    timeFrom < existingEnd &&
                    existingStart < timeTo)
                {
                    if (check_all_time != 3)
                    {
                        MessageBox.Show("Time conflict detected with another exam.");
                        return null;
                    }
                }
            }

            return timeFrom.ToString("hh:mm tt") + " - " + timeTo.ToString("hh:mm tt");
        }
        else
        {
            MessageBox.Show("Start time must be earlier than end time.");
            return null;
        }
    }
    else
    {
        MessageBox.Show("Invalid time format. Please use format like 07:00 AM.");
        return null;
    }
```

- **We can only have one subject at a time in a room in a day.**

```
12 references | UT010529, 18 hours ago | 2 authors, 2 changes
internal class UserController
{
    3 references | UT010529, 18 hours ago | 1 author, 1 change
    public string CreateUserID(string Age,string NIC_Number,string role)
    {
        string Name = null;
        List<char> NIC = new List<char>();
        foreach (char NIC_Num in NIC_Number)
        {
            NIC.Add(NIC_Num);
        }
       if (role == "Student")
        {
            Name = "UT";
        }
       else if (role == "Lecturers")
        {
            Name = "LC";
        }
        else
        {
            Name = "ST";
        }
        string ID = Name + NIC[3] + NIC[1] + Age + NIC[2] + NIC[4];
        return ID;
    }
    3 references | UT010529, 18 hours ago | 1 author, 1 change
    public void AddUsers(string userid,string role)
    {

        using (var Dbconn = Dbconfing.GetConnection())
        {
            var cmd = Dbconn.CreateCommand();

                    string AddscourseQuery = "INSERT INTO Users(UserID,Password,Role) VALUES (@userid,@password,@role)";
                    SQLiteCommand SQLite = new SQLiteCommand(AddscourseQuery, Dbconn);

                    SQLite.Parameters.AddWithValue("@userid", userid);
                    SQLite.Parameters.AddWithValue("@password","4321");
                    SQLite.Parameters.AddWithValue("@role", role);
                    SQLite.ExecuteNonQuery();
        }
```

- **Create user Id by role automatically.**

```
public void searchveiw()
{

    List<Courses_mo> courses = coursecontroller.GetCourse();
    string Name = course_name.Text;
    foreach (var course in courses)
    {
        if (course.Name == Name)
        {
            List<Courses_mo> search = new List<Courses_mo>();
            search.Add(course);
            veiw_courses.DataSource = search;

        }
```

- **Able to search course by name.**

```
private void Add_btn_Click(object sender, EventArgs e)
{
    if (add > 0)
    {
        rooms.AddRoomtype(roomtype.Text);

        addview();
        addcombobox();
        Hide_01(true);
        Hide_02(false);
        add = 0;
    }
    else
    {
        if (room_name.Text.Trim() != "")
        {
            Rooms_mo rooms_ = new Rooms_mo();
            rooms_.RoomName = room_name.Text.Trim();
            rooms_.RoomType = roomtype.Text;
            rooms.AddRooms(rooms_);
            addview();
            room_name.Clear();


        }
        else
        {
            MessageBox.Show("Invalid Room Name", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

        }
    }

}
```

- **Able to add both room name and room type on a single add button**

```
public int AddStudents(Students_mo students_mo)
{

    using (var Dbconn = Dbconfing.GetConnection())
    {
        var cmd = Dbconn.CreateCommand();
        cmd.CommandText = @"SELECT NICNumber FROM Students WHERE NICNumber = @nicnumber";
        cmd.Parameters.AddWithValue("@nicnumber", students_mo.NIC_Number);
        using (var reader = cmd.ExecuteReader())
        {


            if (!reader.HasRows)
            {
                reader.Close();

                string AddsstudentsQuery = "INSERT INTO Students (Name,Age,Gender,Address,CoursesName,UserID,NICNumber,CoursesID,Date) "
                    "VALUES (@name,@age,@gender,@address,@coursesname,@userid,@nicnumber,@coursesid,@date)";
                SQLiteCommand SQLite = new SQLiteCommand(AddsstudentsQuery, Dbconn);

                SQLite.Parameters.AddWithValue("@name", students_mo.Name);
                SQLite.Parameters.AddWithValue("@age", students_mo.Age);
                SQLite.Parameters.AddWithValue("@gender", students_mo.Gender);
                SQLite.Parameters.AddWithValue("@address", students_mo.Address);
                SQLite.Parameters.AddWithValue("@coursesname", students_mo.coursesName);
                SQLite.Parameters.AddWithValue("@userid", students_mo.UserId);
                SQLite.Parameters.AddWithValue("@nicnumber", students_mo.NIC_Number);
                SQLite.Parameters.AddWithValue("@coursesid", students_mo.CoursesID);
                SQLite.Parameters.AddWithValue("@date", students_mo.Date);
                SQLite.ExecuteNonQuery();
                return 1;

            }
            else
            {
                MessageBox.Show("NIC Number Already EXISTS ");
                return 0;
```

- **Avoid adding users with same NIC number**

S.Neathiran.

UT010529