Project Report: UnicomTic Management System

1. Project Overview

• Key Features Implemented:

- Multi-role user authentication system (Admin, Staff, Lecturer, Student).
- Complete academic management (Courses, Groups, Subjects, Exams).
- Timetable scheduling with conflict detection (date, time slot, room, lecturer).
- Attendance tracking and mark management per exam.
- CRUD (Create, Read, Update, Delete) operations for students, lecturers, rooms, subjects, and more.
- Role-based dashboards with custom functionalities.

• Technologies Used:

- **Frontend:** C# WinForms
- **Backend:** .NET Framework
- **Database:** SQLite
- **Development Environment:** Visual Studio

• Challenges Faced & Solutions:

- **Challenge:** Preventing timetable conflicts (e.g., same room or lecturer double-booked).
  Solution: Added logic to validate room and lecturer availability before saving.
- **Challenge:** Managing role-based data access (e.g., lecturers seeing only their students).
  Solution: Filtered data using queries linked to logged-in user IDs.
- **Challenge:** Handling 'Absent' students in mark entry.
  Solution: Used `NULL` in SQLite to store absence cleanly.

---

2. Role-Based Functionalities

• Admin:

- Access full system dashboard.
- Manage all entities: Students, Lecturers, Staff, Courses, Groups, Subjects.
- Create timetable entries.
- Assign lecturers to courses.
- Manage exams and marks.

• Staff:

- Limited administrative view.
- Can assist in scheduling and viewing data.

- May help with room or slot assignments.

• Lecturer:

- View their assigned courses and timetable.
- Mark and update attendance.
- View and enter marks for their students.
- View students by course/group.

• Student:

- View personal timetable.
- View own marks and attendance status.

---

3. Workflow Summary

1. **Login:**
   User logs in using role credentials (Admin, Staff, Lecturer, Student). Each role sees a dedicated dashboard.
2. **Admin Activities:**
   Admin registers users, assigns courses/groups, manages exams and attendance.
3. **Timetable Creation:**
   Admin or Staff creates a timetable with validations to prevent conflicts using date, time slot, and room availability.
4. **Lecturer Interaction:**
   Lecturers access their dashboard to manage marks and attendance only for assigned students.
5. **Student Access:**
   Students log in to view personal data like exam results, subject schedules, and attendance records.

Bonus: Attendance Management

- **Admin:**
  - Can view, edit, or delete any attendance records across all students, subjects, and dates.
  - Has full visibility into attendance trends and history for academic auditing or reporting.
- **Lecturer:**
  - Can **mark attendance** for students in their assigned subjects based on the timetable.
  - Can **update or correct attendance** on the same day or retrospectively if needed.

    - Ensures accurate tracking of student participation for their classes only.
- **Student:**
    - Can **view their own attendance records** per subject and date.
    - Helps them monitor their class participation and stay informed on absences, lateness, or excused entries.

Code Samples (Screenshots)

INSERTING LECTURER DETAILS INTO USER TABLE AND LECTURER TABLE :

```
    // Insert into User table
    string insertUser = "INSERT INTO User (UserName, Password, UserType) VALUES (@uname, @pwd, 'lecturer');";
    SQLiteCommand cmd = new SQLiteCommand(insertUser, con);
    cmd.Parameters.AddWithValue("@uname", lecturer.userName);
    cmd.Parameters.AddWithValue("@pwd", lecturer.password);
    cmd.ExecuteNonQuery();

    long userId = con.LastInsertRowId; // Retrieve the newly created user ID

    // Insert into Lecturer table
    string insertLecturer = "INSERT INTO Lecturer (LecturerName, UserId, GroupId) VALUES (@lname, @uid, @gid);";
    SQLiteCommand lecCmd = new SQLiteCommand(insertLecturer, con);
    lecCmd.Parameters.AddWithValue("@lname", lecturer.lecturerName);
    lecCmd.Parameters.AddWithValue("@uid", userId);
    lecCmd.Parameters.AddWithValue("@gid", lecturer.Gr
    lecCmd.ExecuteNonQuery();
}
```

string Lecturer.lecturerName { get; set; }

CHECKING WHETHER THE PASSWORD ALREADY EXISTS :

```
// Check if the password is unique (not already in use)
1 reference
public bool IsPasswordUnique(string password)
{
    try
    {
        using (SQLiteConnection con = DataBaseCon.Connection())
        {
            string query = "SELECT COUNT(*) FROM User WHERE Password = @Password";
            SQLiteCommand cmd = new SQLiteCommand(query, con);
            cmd.Parameters.AddWithValue("@Password", password);
            int count = Convert.ToInt32(cmd.ExecuteScalar());
            return count == 0;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error while checking password uniqueness: " + ex.Message);
        return false;
    }
}
```

GETTING TIMETABLE FOR A CERTAIN LECTURER TO MARK ATTENDANCE :

```
// Get timetables for a certain Lecturer
1 reference
public List<Timetable> GetTimetablesByLecturerUserId(int userId)
{
    var list = new List<Timetable>();
    try
    {
        using (var conn = DataBaseCon.Connection())
        {
            // First, get LecturerId from UserId
            string getLecturerIdQuery = "SELECT LecturerId FROM Lecturer WHERE UserId = @UserId";
            int lecturerId = -1;

            using (var cmd = new SQLiteCommand(getLecturerIdQuery, conn))
            {
                cmd.Parameters.AddWithValue("@UserId", userId);
                var result = cmd.ExecuteScalar();
                if (result != null)
                    lecturerId = Convert.ToInt32(result);
                else
                    return list; // No matching lecturer found
            }

            // Now fetch timetables for this LecturerId
            string query = @"
                SELECT t.TimetableId, t.SubjectId, s.SubjectName,
                       t.RoomId, r.RoomName,
                       t.TimeSlotId, ts.StartTime, ts.EndTime,
                       t.LecturerId, l.LecturerName,
```

GETTING SUBJECTS FOR A GIVEN COURSE :

```
// Retrieves all subjects for a given course.
1 reference
public List<(int subjectId, string subjectName)> GetSubjectsByCourse(int courseId)
{
    var subjects = new List<(int, string)>();

    try
    {
        using (var conn = DataBaseCon.Connection())
        {
            string query = "SELECT SubjectId, SubjectName FROM Subjects WHERE CourseId = @courseId;";
            using (var cmd = new SQLiteCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@courseId", courseId);
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        subjects.Add((Convert.ToInt32(reader["SubjectId"]), Convert.ToString(reader["SubjectName"])));
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Failed to load subjects for the course. " + ex.Message);
    }

    return subjects;
```

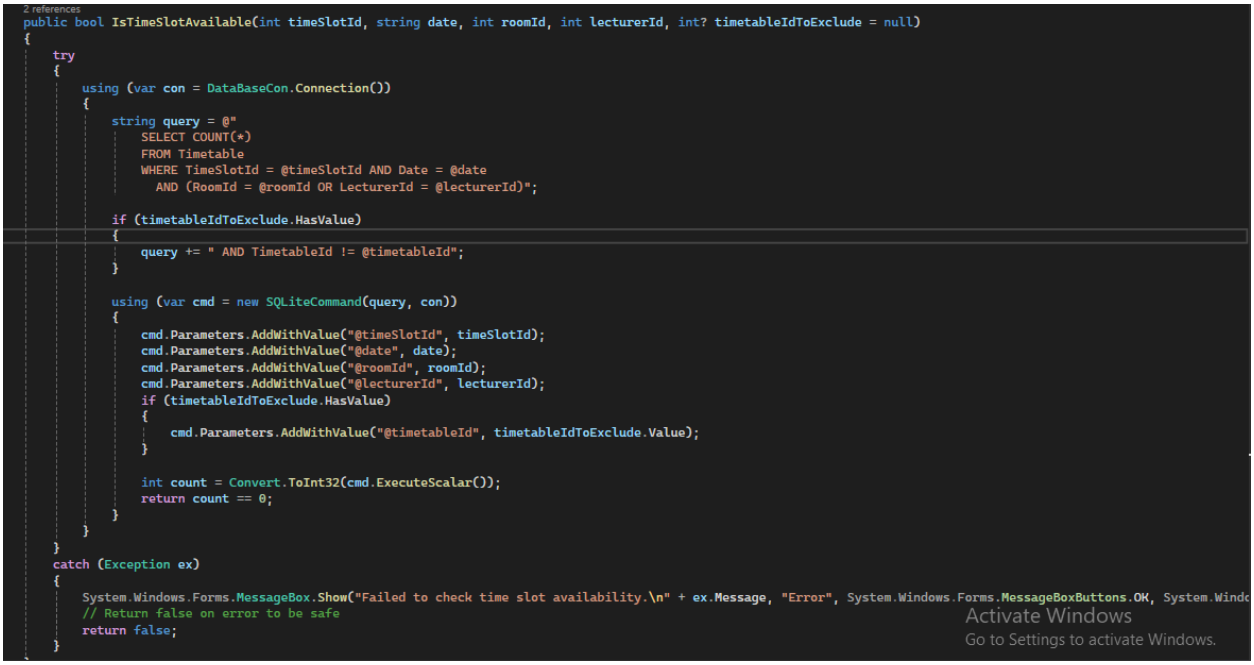
ADD OR UPDATE ATTENDANCE :

```
// Add or update attendance for each student
1 reference
public void AddOrUpdateAttendance(List<Attendance> records)
{
    try
    {
        using (var conn = DataBaseCon.Connection())
        {
            foreach (var att in records)
            {
                string checkQuery = "SELECT COUNT(*) FROM Attendance WHERE TimetableId=@Tid AND StudentId=@Sid";
                using (var checkCmd = new SQLiteCommand(checkQuery, conn))
                {
                    checkCmd.Parameters.AddWithValue("@Tid", att.TimetableId);
                    checkCmd.Parameters.AddWithValue("@Sid", att.StudentId);
                    long exists = (long)checkCmd.ExecuteScalar();

                    string query = exists > 0 ?
                        "UPDATE Attendance SET Status=@Status WHERE TimetableId=@Tid AND StudentId=@Sid" :
                        "INSERT INTO Attendance (TimetableId, StudentId, Status) VALUES (@Tid, @Sid, @Status)";

                    using (var cmd = new SQLiteCommand(query, conn))
                    {
                        cmd.Parameters.AddWithValue("@Tid", att.TimetableId);
                        cmd.Parameters.AddWithValue("@Sid", att.StudentId);
                        cmd.Parameters.AddWithValue("@Status", att.Status);
                        cmd.ExecuteNonQuery();
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Something went wrong: " + ex.Message, "Error");
```

CHECKING IF ALREADY A SAME LECTURER OR A SAME ROOM BOOKED FOR A SLECTED TIMESLOT IN A SELECTED DATE :



```
2 references
public bool IsTimeSlotAvailable(int timeSlotId, string date, int roomId, int lecturerId, int? timetableIdToExclude = null)
{
    try
    {
        using (var con = DataBaseCon.Connection())
        {
            string query = @"
                SELECT COUNT(*)
                FROM Timetable
                WHERE TimeSlotId = @timeSlotId AND Date = @date
                  AND (RoomId = @roomId OR LecturerId = @lecturerId)";

            if (timetableIdToExclude.HasValue)
            {
                query += " AND TimetableId != @timetableId";
            }

            using (var cmd = new SQLiteCommand(query, con))
            {
                cmd.Parameters.AddWithValue("@timeSlotId", timeSlotId);
                cmd.Parameters.AddWithValue("@date", date);
                cmd.Parameters.AddWithValue("@roomId", roomId);
                cmd.Parameters.AddWithValue("@lecturerId", lecturerId);
                if (timetableIdToExclude.HasValue)
                {
                    cmd.Parameters.AddWithValue("@timetableId", timetableIdToExclude.Value);
                }

                int count = Convert.ToInt32(cmd.ExecuteScalar());
                return count == 0;
            }
        }
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show("Failed to check time slot availability.\n" + ex.Message, "Error", System.Windows.Forms.MessageBoxButtons.OK, System.Windo
        // Return false on error to be safe
        return false;
    }
```