Unicom TIC Management System - Report

1. Introduction

The **Unicom TIC Management System (UMS)** is a desktop application designed to help manage basic institute operations such as course management, student records, exams, and timetables. It includes a login system with role-based access for Admin, Staff, Students, and Lecturers. The system is built using **C# WinForms** and **SQLite** to store and manage data.

2. Key Features Implemented

Login System

- A login form where users enter their username and password.
- A default admin is created when the system first runs. The **username** is **admin123** and the **password** is **12345**.
- Based on roles, users are given access to different features:
  - **Admin**: Full access to all modules.
  - **Staff**: Can view timetables and manage exams/marks.
  - **Student**: Can only view their own timetable, exams and marks.
  - **Lecturer**: Can view timetables and manage exams/marks.

2. **Profile Form**
   - Users can view their profile and change their username and password from the profile form.
3. **Course & Subject Management**
   - Admin can add, edit, and delete courses and subjects.
4. **Student Management**
   - Admin can add, edit, and delete student records.
   - Students can view their own details.
5. **Exam & Marks Management**
   - Admin , lecturers and staff can add and edit exam marks.
   - Students can only view their marks.
6. **Timetable Management**

- Admin can add and edit timetable entries, assigning subjects to specific time slots and rooms (labs or halls).
- When a **student** logs in, they will see only the timetable for their own course.
- The **exams** and **marks** sections will also show data related only to the student's course.

3. Technologies Used

1. **C# and WinForms**
   - Used for building the application logic and user interface, including forms, buttons, data grids, and combo boxes.
2. **SQLite Database**
   - Data such as users, courses, students, exams, marks, rooms, and timetables are stored in SQLite.
   - Relationships between data are handled using foreign keys.
3. **MVC Design Pattern**
   - **Model**: Represents the data (e.g., Course, Student).
   - **View**: The UI elements (e.g., forms, buttons, and data grids).
   - **Controller**: Manages the logic, responding to user actions and updating the UI.

4. Challenges Faced and How I Overcame Them

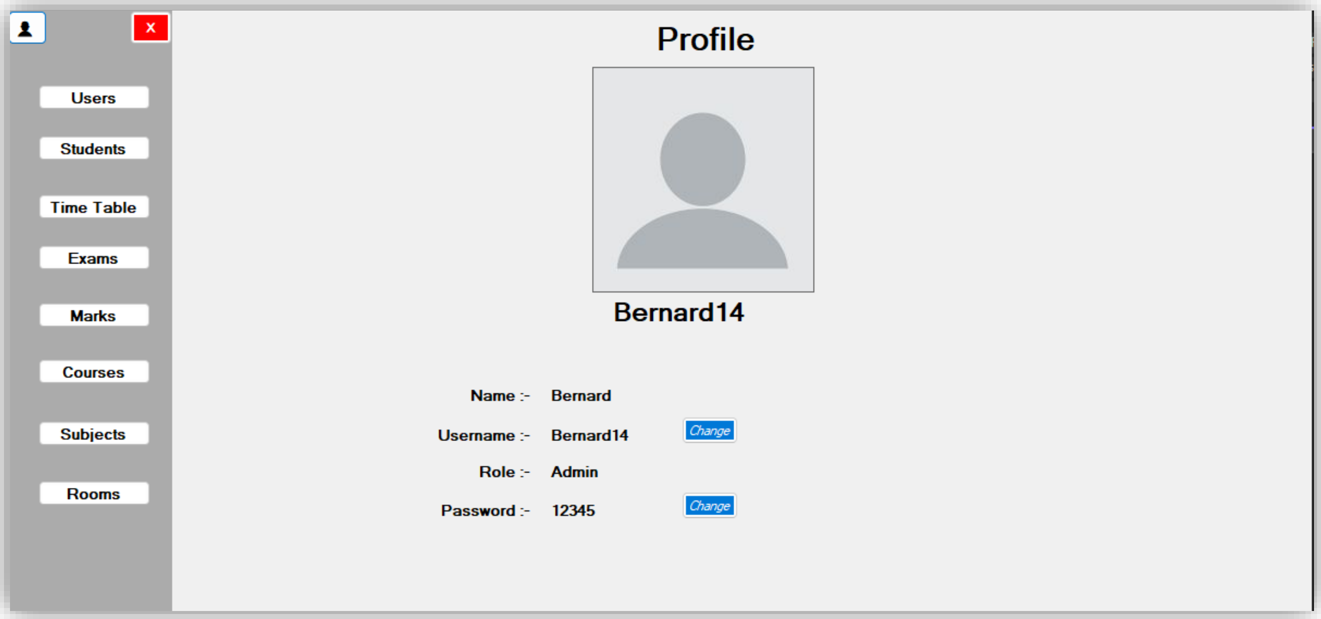
1. **Learning C# and WinForms**
   - Challenge: I had been learning C# and WinForms for less than a month, so it took some time to understand key concepts, especially **MVC** and database interactions.
   - Solution: I sought help from my mentor and used resources like **ChatGPT** to get through the tough parts. With practice, I became more comfortable with the language and framework.
2. **Understanding MVC**
   - Challenge: Implementing the **Model-View-Controller** design pattern was initially difficult, as I wasn't familiar with how to separate data, UI, and logic properly.

    - Solution: After experimenting with small examples and reading up on MVC, I was able to break down the application into manageable parts. ChatGPT helped me understand the structure better.
3. **Role-Based Access Control**
    - Challenge: Managing role-based access, ensuring that users see only the features they are allowed to access based on their role, was tricky.
    - Solution: I added logic to check the user's role after login and customized the dashboard and forms accordingly. Admins can access all features, while students only see their own timetable, exams, and marks.
4. **Timetable and Course-Specific Views for Students**
    - Challenge: Making sure that students could only see timetables, exams, and marks related to their course was a bit challenging.
    - Solution: I used filters to ensure that when a student logs in, they see only the data that corresponds to their course (e.g., timetables and exams).
5. **Profile Form for User Details**
    - Challenge: Allowing users to view and change their username and password securely.
    - Solution: I created a simple **profile form** where users can update their credentials. I ensured the changes were reflected in the database immediately after saving.

5. Conclusion

The **Unicom TIC Management System** is a successful implementation of an institute management tool. Despite the initial learning curve with C#, WinForms, and the MVC design pattern, the project was completed with all major features implemented. The key features like the role-based login, course management, timetable management, and the profile form work smoothly. Although it took some time to get familiar with C# and Windows Forms, I was able to overcome these challenges with help from my mentor and **ChatGPT**, gaining valuable experience in programming and problem-solving.

By clicking the profile icon user can view their details and can change their credentials.



Profile

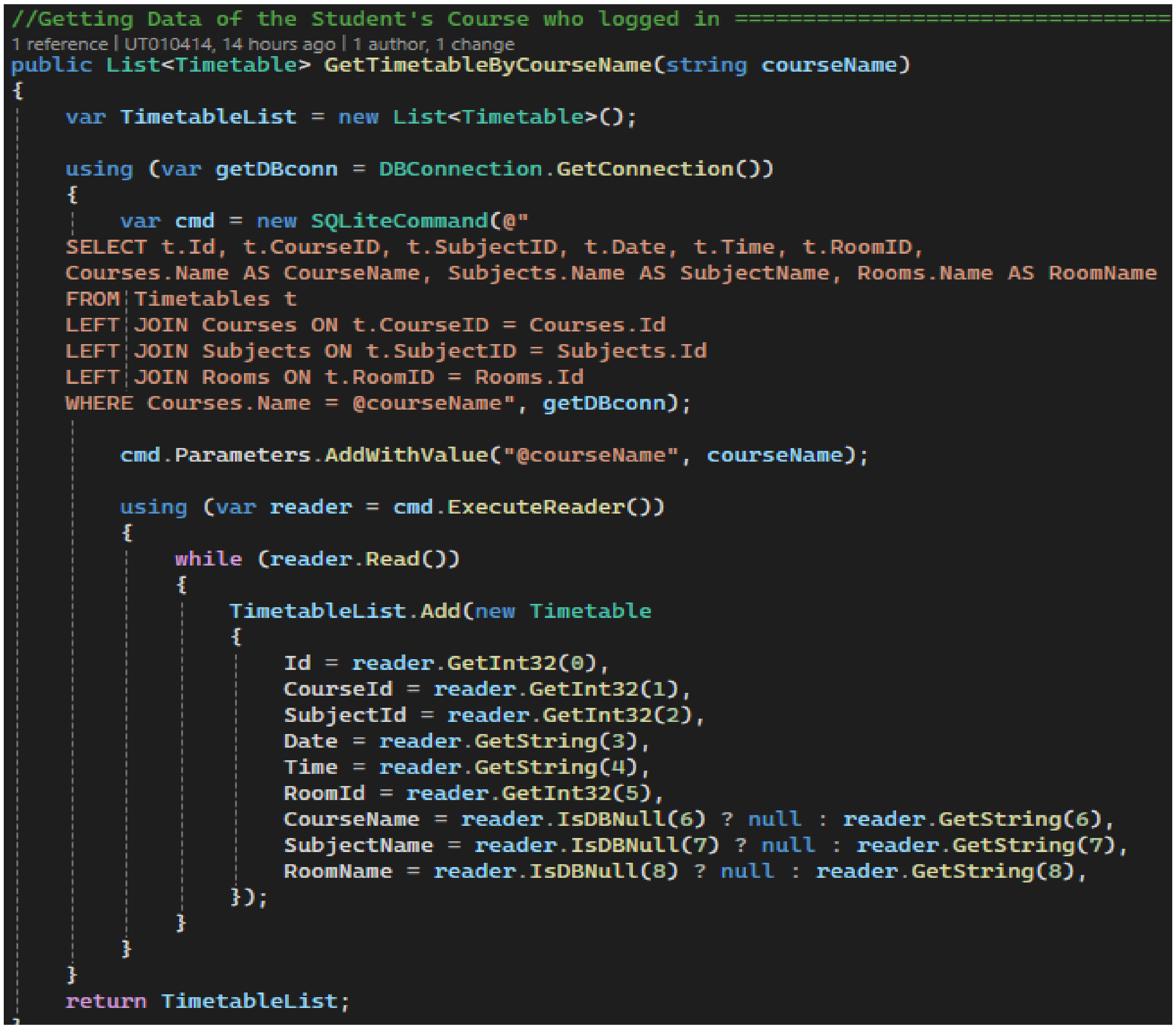
Bernard14

Name :- Bernard

Username :- Bernard14

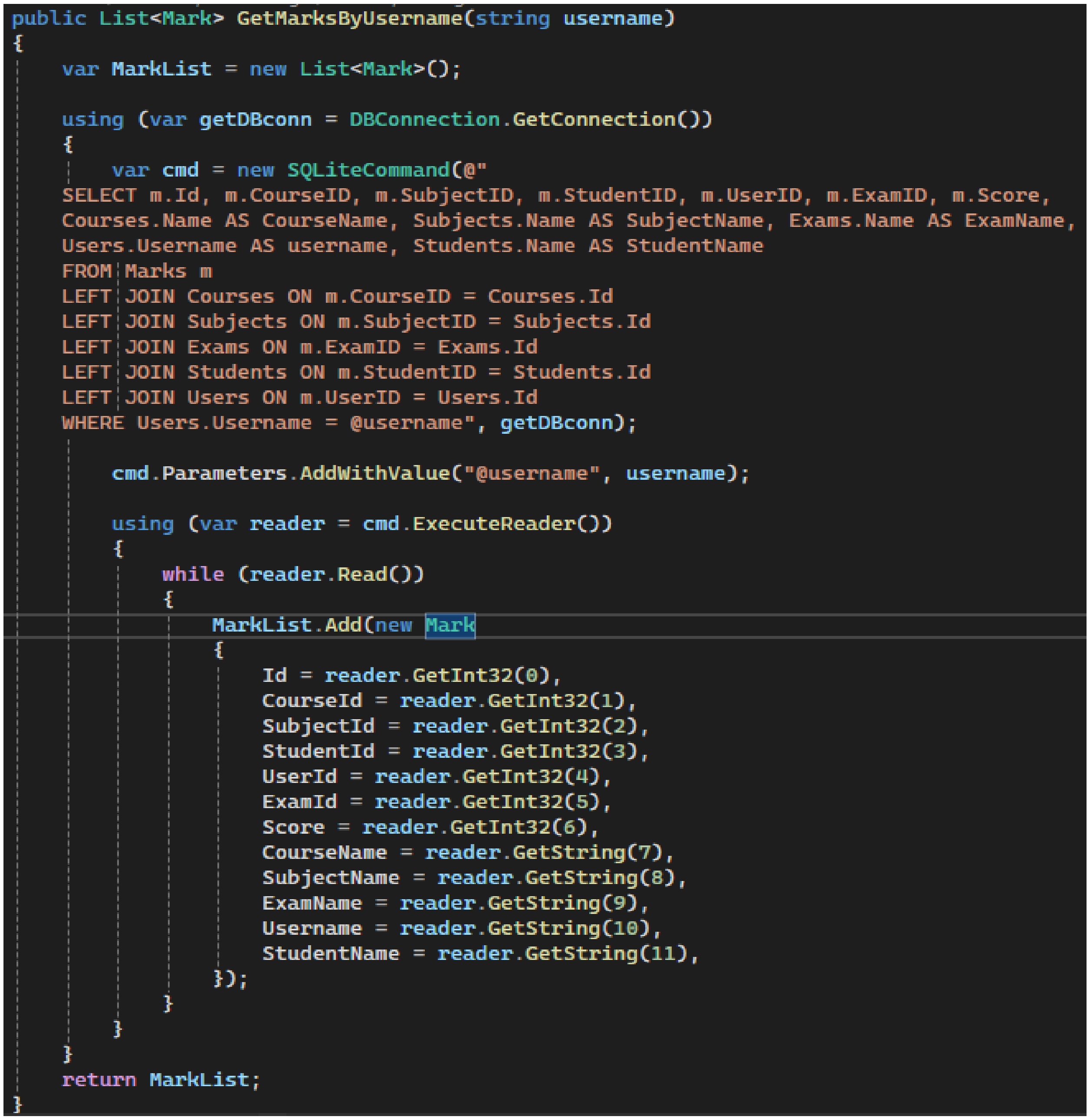
Role :- Admin

Password :- 12345



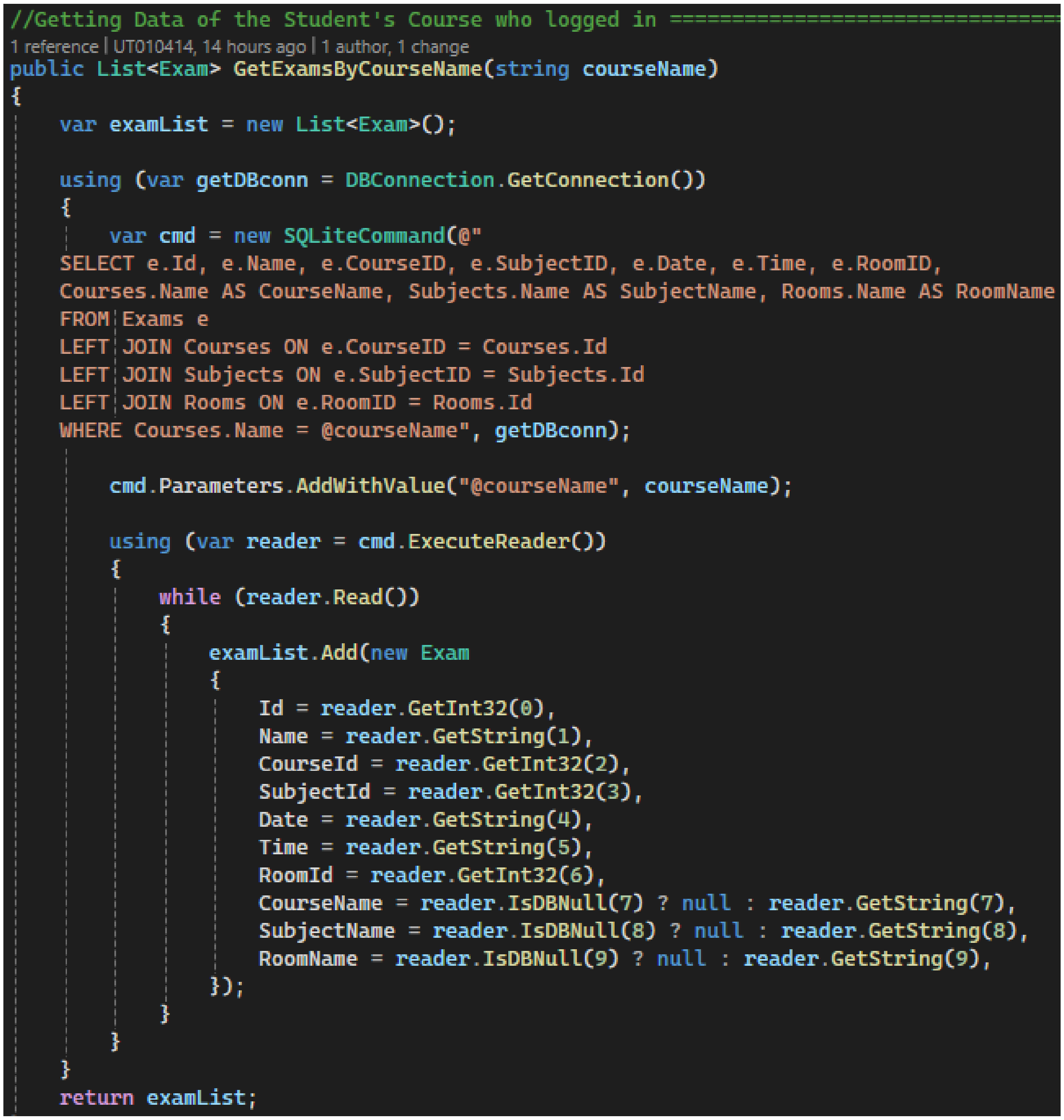
```
//Getting Data of the Student's Course who logged in ================================
1 reference | UT010414, 14 hours ago | 1 author, 1 change
public List<Timetable> GetTimetableByCourseName(string courseName)
{
    var TimetableList = new List<Timetable>();

    using (var getDBconn = DBConnection.GetConnection())
    {
        var cmd = new SQLiteCommand(@"
    SELECT t.Id, t.CourseID, t.SubjectID, t.Date, t.Time, t.RoomID,
    Courses.Name AS CourseName, Subjects.Name AS SubjectName, Rooms.Name AS RoomName
    FROM Timetables t
    LEFT JOIN Courses ON t.CourseID = Courses.Id
    LEFT JOIN Subjects ON t.SubjectID = Subjects.Id
    LEFT JOIN Rooms ON t.RoomID = Rooms.Id
    WHERE Courses.Name = @courseName", getDBconn);

        cmd.Parameters.AddWithValue("@courseName", courseName);

        using (var reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                TimetableList.Add(new Timetable
                {
                    Id = reader.GetInt32(0),
                    CourseId = reader.GetInt32(1),
                    SubjectId = reader.GetInt32(2),
                    Date = reader.GetString(3),
                    Time = reader.GetString(4),
                    RoomId = reader.GetInt32(5),
                    CourseName = reader.IsDBNull(6) ? null : reader.GetString(6),
                    SubjectName = reader.IsDBNull(7) ? null : reader.GetString(7),
                    RoomName = reader.IsDBNull(8) ? null : reader.GetString(8),
                });
            }
        }
    }
    return TimetableList;
```

- **DataGridView show only the timetable of the course of the student who logged in.**



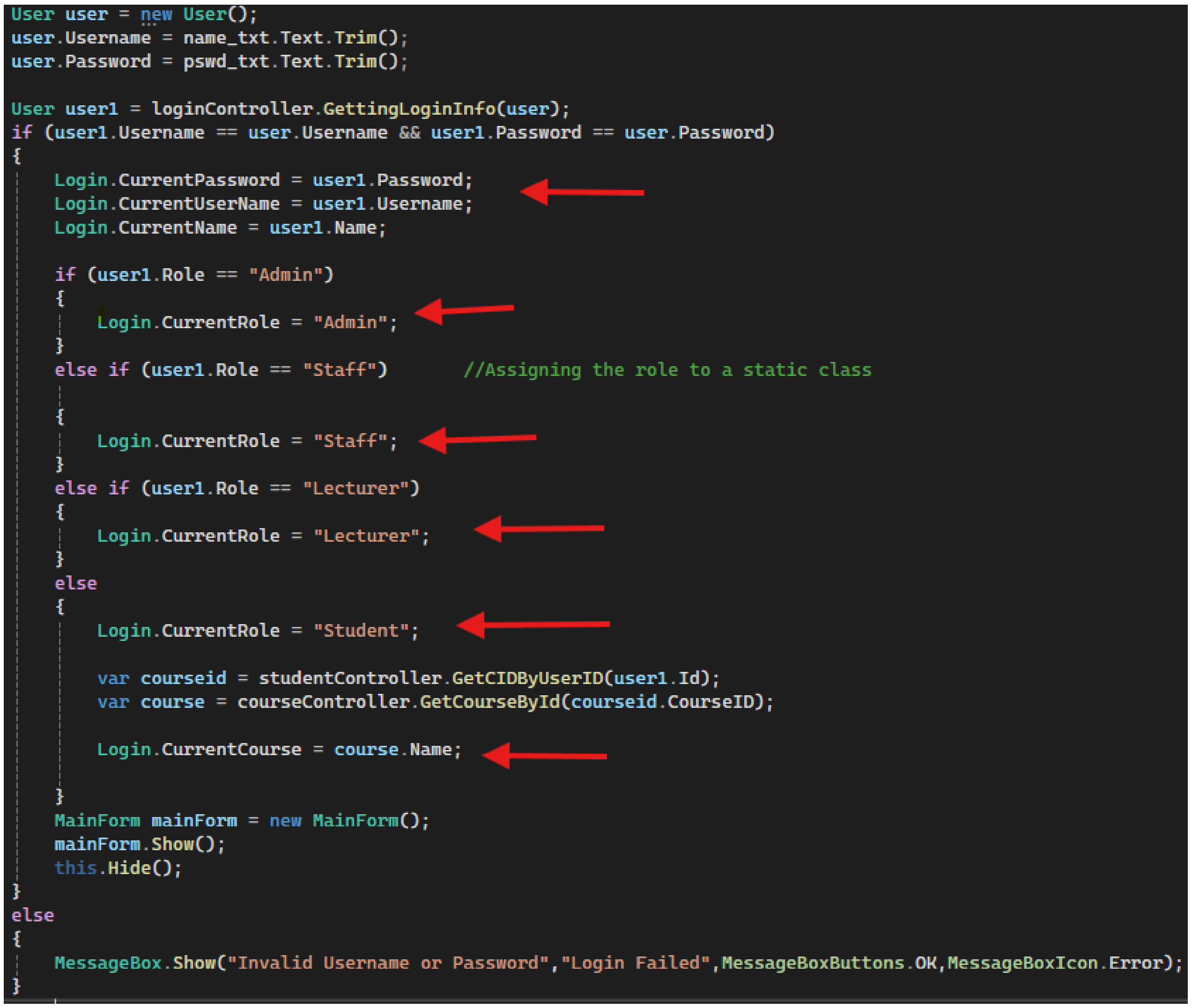
```
public List<Mark> GetMarksByUsername(string username)
{
    var MarkList = new List<Mark>();

    using (var getDBconn = DBConnection.GetConnection())
    {
        var cmd = new SQLiteCommand(@"
    SELECT m.Id, m.CourseID, m.SubjectID, m.StudentID, m.UserID, m.ExamID, m.Score,
    Courses.Name AS CourseName, Subjects.Name AS SubjectName, Exams.Name AS ExamName,
    Users.Username AS username, Students.Name AS StudentName
    FROM Marks m
    LEFT JOIN Courses ON m.CourseID = Courses.Id
    LEFT JOIN Subjects ON m.SubjectID = Subjects.Id
    LEFT JOIN Exams ON m.ExamID = Exams.Id
    LEFT JOIN Students ON m.StudentID = Students.Id
    LEFT JOIN Users ON m.UserID = Users.Id
    WHERE Users.Username = @username", getDBconn);

        cmd.Parameters.AddWithValue("@username", username);

        using (var reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                MarkList.Add(new Mark
                {
                    Id = reader.GetInt32(0),
                    CourseId = reader.GetInt32(1),
                    SubjectId = reader.GetInt32(2),
                    StudentId = reader.GetInt32(3),
                    UserId = reader.GetInt32(4),
                    ExamId = reader.GetInt32(5),
                    Score = reader.GetInt32(6),
                    CourseName = reader.GetString(7),
                    SubjectName = reader.GetString(8),
                    ExamName = reader.GetString(9),
                    Username = reader.GetString(10),
                    StudentName = reader.GetString(11),
                });
            }
        }
    }
    return MarkList;
}
```

- **Student can view their marks only so other students cannot view Your marks (Privacy).**



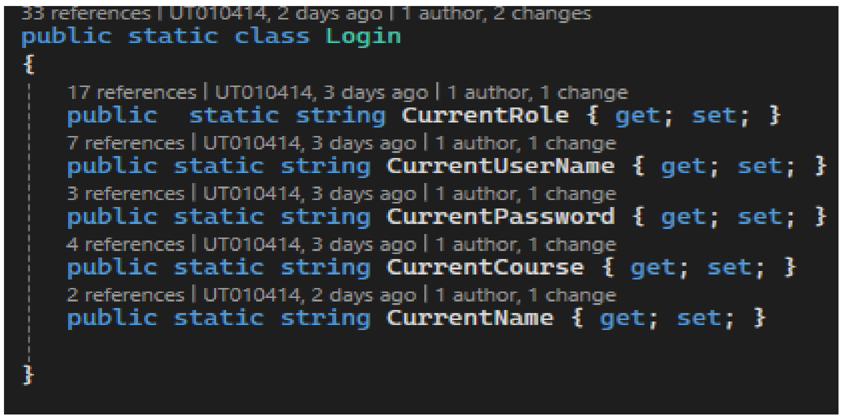
```
//Getting Data of the Student's Course who logged in ==============================
1 reference | UT010414, 14 hours ago | 1 author, 1 change
public List<Exam> GetExamsByCourseName(string courseName)
{
    var examList = new List<Exam>();

    using (var getDBconn = DBConnection.GetConnection())
    {
        var cmd = new SQLiteCommand(@"
    SELECT e.Id, e.Name, e.CourseID, e.SubjectID, e.Date, e.Time, e.RoomID,
    Courses.Name AS CourseName, Subjects.Name AS SubjectName, Rooms.Name AS RoomName
    FROM Exams e
    LEFT JOIN Courses ON e.CourseID = Courses.Id
    LEFT JOIN Subjects ON e.SubjectID = Subjects.Id
    LEFT JOIN Rooms ON e.RoomID = Rooms.Id
    WHERE Courses.Name = @courseName", getDBconn);

        cmd.Parameters.AddWithValue("@courseName", courseName);

        using (var reader = cmd.ExecuteReader())
        {
            while (reader.Read())
            {
                examList.Add(new Exam
                {
                    Id = reader.GetInt32(0),
                    Name = reader.GetString(1),
                    CourseId = reader.GetInt32(2),
                    SubjectId = reader.GetInt32(3),
                    Date = reader.GetString(4),
                    Time = reader.GetString(5),
                    RoomId = reader.GetInt32(6),
                    CourseName = reader.IsDBNull(7) ? null : reader.GetString(7),
                    SubjectName = reader.IsDBNull(8) ? null : reader.GetString(8),
                    RoomName = reader.IsDBNull(9) ? null : reader.GetString(9),
                });
            }
        }
    }
    return examList;
```

- **DataGridView show the exam schedule of the course of the student who logged in.**



```
User user = new User();
user.Username = name_txt.Text.Trim();
user.Password = pswd_txt.Text.Trim();

User user1 = loginController.GettingLoginInfo(user);
if (user1.Username == user.Username && user1.Password == user.Password)
{
    Login.CurrentPassword = user1.Password;
    Login.CurrentUserName = user1.Username;
    Login.CurrentName = user1.Name;

    if (user1.Role == "Admin")
    {
        Login.CurrentRole = "Admin";
    }
    else if (user1.Role == "Staff")       //Assigning the role to a static class

    {
        Login.CurrentRole = "Staff";
    }
    else if (user1.Role == "Lecturer")
    {
        Login.CurrentRole = "Lecturer";
    }
    else
    {
        Login.CurrentRole = "Student";

        var courseid = studentController.GetCIDByUserID(user1.Id);
        var course = courseController.GetCourseById(courseid.CourseID);

        Login.CurrentCourse = course.Name;

    }
    MainForm mainForm = new MainForm();
    mainForm.Show();
    this.Hide();
}
else
{
    MessageBox.Show("Invalid Username or Password","Login Failed",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
```

- **Assigning values to a static class at the time of user login for future use.**



```
33 references | UT010414, 2 days ago | 1 author, 2 changes
public static class Login
{
    17 references | UT010414, 3 days ago | 1 author, 1 change
    public  static string CurrentRole { get; set; }
    7 references | UT010414, 3 days ago | 1 author, 1 change
    public static string CurrentUserName { get; set; }
    3 references | UT010414, 3 days ago | 1 author, 1 change
    public static string CurrentPassword { get; set; }
    4 references | UT010414, 3 days ago | 1 author, 1 change
    public static string CurrentCourse { get; set; }
    2 references | UT010414, 2 days ago | 1 author, 1 change
    public static string CurrentName { get; set; }

}
```

(Note:- If any error related to SQLITE during build kindly uninstall System.Data.SQLite.Core package and reinstall it)

T. Bernard Niruban

UT010414