Table of Contents

Unicom TIC Management System - Project Submission Report

1. Project Overview (Point-Wise)

Key Features Implemented:

- Login System with role-based access for Admin, Staff, Students, and Lecturers.
- Course and Subject Management module.
- Student Management with course association.
- Exam and Marks Management with entry and viewing privileges by role.
- Timetable Management including Computer Lab and Lecture Hall allocation.
- SQLite database with relationships between tables.
- Role-based dashboards that restrict access to appropriate features.
- Error messages and input validation included.
- Simple and intuitive WinForms user interface.

Technologies Used:

- Programming Language: C#
- Framework: WinForms (.NET Framework)
- Database: SQLite using System.Data.SQLite
- Architecture: MVC (Model–View–Controller)
- IDE: Visual Studio
- Design Patterns: MVC

- UI Elements: Buttons, ComboBoxes, DataGridViews, TextBoxes, Labels

Challenges Faced and Solutions:

- Role-Based Dashboard Visibility: Solved by checking user role after login and showing/hiding buttons accordingly.

- Room Allocation with Combo Box: Created a separate table for Rooms with room type filtering, and populated Combo Box dynamically.
- Login Validation: Added a simple user validation system querying the Users table and managing sessions based on roles.
- Database Table Creation: Implemented DatabaseManager.cs to auto-create tables if they don’t exist.
- Navigation Between Forms: Used controller logic to open/close forms based on user role without error.

2. Code Samples (Screenshots)

Create table and default admin username: admin and password:admin123

```
namespace UnicomTICManagementSystem.Repositories
{
    1 reference
    internal class Migration
    {
        1 reference
        public static void CreateTables()
        {
            string tablesQuery = @"
            CREATE TABLE IF NOT EXISTS Users(UserID INTEGER PRIMARY KEY AUTOINCREMENT,
                Username TEXT NOT NULL, Password TEXT NOT NULL, Role TEXT NOT NULL);
            CREATE TABLE IF NOT EXISTS Courses(CourseID INTEGER PRIMARY KEY AUTOINCREMENT,
                CourseName TEXT NOT NULL);
            CREATE TABLE IF NOT EXISTS Subjects(SubjectID INTEGER PRIMARY KEY AUTOINCREMENT,
                SubjectName TEXT NOT NULL, CourseID INTEGER, FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
            CREATE TABLE IF NOT EXISTS Students(StudentID INTEGER PRIMARY KEY AUTOINCREMENT, Name TEXT NOT NULL,
                CourseID INTEGER NOT NULL, FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
            CREATE TABLE IF NOT EXISTS Exams(ExamID INTEGER PRIMARY KEY AUTOINCREMENT,ExamName TEXT NOT NULL,
                SubjectID INTEGER NOT NULL, FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID));
            CREATE TABLE IF NOT EXISTS Marks(MarkID INTEGER PRIMARY KEY AUTOINCREMENT,StudentID INTEGER NOT NULL,ExamID INTEGER NOT NULL,Score INTEGER NOT NULL,
                FOREIGN KEY (StudentID) REFERENCES Students(StudentID), FOREIGN KEY (ExamID) REFERENCES Exams(ExamID));
            CREATE TABLE IF NOT EXISTS Rooms(RoomID INTEGER PRIMARY KEY AUTOINCREMENT,RoomName TEXT NOT NULL,
                RoomType TEXT NOT NULL);
            CREATE TABLE IF NOT EXISTS Timetables(TimetableID INTEGER PRIMARY KEY AUTOINCREMENT,SubjectID INTEGER NOT NULL,
                TimeSlot TEXT NOT NULL,RoomID INTEGER NOT NULL,FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID),FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID));
            ";


            using (var conn = DatabaseManager.GetConnection())
            {
                SQLiteCommand cmd = new SQLiteCommand(tablesQuery, conn);
                cmd.ExecuteNonQuery();

                // Insert a default admin user if none exists
                var checkAdminCmd = new SQLiteCommand("SELECT COUNT(*) FROM Users WHERE Role = 'Admin';", conn);
                long adminCount = (long)checkAdminCmd.ExecuteScalar();
                if (adminCount == 0)
                {
                    var insertAdminCmd = new SQLiteCommand("INSERT INTO Users (Username, Password, Role) VALUES ('admin', 'admin123', 'Admin');", conn);
                    insertAdminCmd.ExecuteNonQuery();
                }
            }

        }
    }
}
```

MainForm.cs – Dashboard with Role-Based Feature Access

```
namespace UnicomTICManagementSystem.View
{
    4 references
    public partial class MainForm : Form
    {
        1 reference
        public MainForm()
        {
            InitializeComponent();
        }
        1 reference
        public void SetWelcomeText(string name, string role)
        {
            labelmainwelcome.Text = $"Welcome {name} .Your role is {role}";
            if (role == "Admin") { btnmainmarks.Visible = false; }
            else if (role == "Student")
            {
                btnmaincourses.Visible = false; btnmainstudents.Visible = false; btnmainexams.Visible = false;
                btnmaintimetable.Visible = false;btnmainroom.Visible = false;btnmainsubjects.Visible = false;
            }
            else if (role == "Lecturer") { btnmaincourses.Visible = false; btnmainstudents.Visible = false; }
            else if (role == "Staff") { }

        }

        1 reference
        private void btnmaincourses_Click(object sender, EventArgs e)
        {
            CourseForm courseForm = new CourseForm();
            courseForm.ShowDialog();
        }

        1 reference
        private void btnmainstudents_Click(object sender, EventArgs e)
        {
            StudentForm studentForm = new StudentForm();
            studentForm.ShowDialog();
        }
```

RoomForm.cs – Manage Lecture Halls and Computer Labs

```
13      namespace UnicomTICManagementSystem.View
15          public partial class RoomForm : Form
23              private void btnroomadd_Click(object sender, EventArgs e)
27                  roomController.AddRoom(room);
28                  LoadRoomdata();
29
30              }
                1 reference
31              public void EditData()
32              {
33                  RoomController editController = new RoomController();
34                  Room room = new Room { RoomName = textboxroomroomname.Text, RoomId = int.Parse(textroomroomID.Text), RoomType = textboxroomroomtype.Text };
35                  editController.EditRoomData(room);
36              }
                1 reference
37              public void DeleteData() {
38                  RoomController deleteController = new RoomController();
39                  Room room = new Room { RoomId = int.Parse(textroomroomID.Text)};
40                  deleteController.DeleteRoomData(room);
41              }
                4 references
42              public void LoadRoomdata()
43              {
44                  RoomController getroomdata = new RoomController();
45                  var roomData = getroomdata.GetRoomData();
46                  dataroom.DataSource = roomData;
47                  dataroom.Columns["RoomID"].Visible = false;
48              }
49
                1 reference
50              private void btnroomedit_Click(object sender, EventArgs e)
51              {
52                  EditData();
53                  LoadRoomdata();
54              }
55
                1 reference
56              private void dataroom_CellClick(object sender, DataGridViewCellEventArgs e)
57              {
58                  if (e.RowIndex >= 0)
59                  {
60                      var selectedrow = dataroom.Rows[e.RowIndex].DataBoundItem as Room;
61                      if (selectedrow != null)
62                      {
63                          textroomroomID.Text = selectedrow.RoomId.ToString();
64                          textboxroomroomname.Text = selectedrow.RoomName;
65                          textboxroomroomtype.Text = selectedrow.RoomType;
66                      }
67                  }
68              }
69
70
```

TimetableForm.cs

```
14      namespace UnicomTICManagementSystem.View
16          public partial class TimetableForm : Form
46              public void LoadTimetabledata()
47              {
48                  TimetableController gettimetabledata = new TimetableController();
49                  var timetableData = gettimetabledata.GettimetableData();
50                  datatimetable.DataSource = timetableData;
51                  datatimetable.Columns["RoomID"].Visible = false;
52                  datatimetable.Columns["SubjectID"].Visible = false;
53              }
54              //=====================================================
                1 reference
55              public void loadSubject()
56              {
57
58                  var subjectList = subjectController.GetSubjectList();
59                  if (subjectList.Count > 0)
60                  {
61                      comboboxtimetablesubject.DataSource = subjectList;
62                      comboboxtimetablesubject.DisplayMember = "SubjectName";
63                      comboboxtimetablesubject.ValueMember = "SubjectID";
64                  }
65                  else
66                  {
67                      labeltimetableerror.Text = "No subject found. Please add a subject first.";
68                  }
69              }
                1 reference
70              public void loadRoom()
71              {
72                  var roomList = timetableController.GetRoomList();
73                  if (roomList.Count > 0)
74                  {
75                      comboboxtimetableroom.DataSource = roomList;
76                      comboboxtimetableroom.DisplayMember = "RoomName";
77                      comboboxtimetableroom.ValueMember = "RoomId";
78                  }
79                  else { labeltimetableerror.Text = "No room found. Please add a room first."; }
80              }
81
82
83
                0 references
84              private void comboboxtimetablesubject_SelectedIndexChanged(object sender, EventArgs e)
85              {
86                  selectedSubjectID = comboboxtimetablesubject.SelectedIndex+1;
87              }
88
                0 references
89              private void comboboxtimetableroom_SelectedIndexChanged(object sender, EventArgs e)
90              {
91                  selectedRoomID = comboboxtimetableroom.SelectedIndex+1;
92              }
```

ExamController.cs

```
namespace UnicomTICManagementSystem.Controller
{
    10 references
    internal class ExamController
    {
        2 references
        public List<Subject> GetSubjectList()
        {
            List<Subject> subjectList = new List<Subject>();
            using (var connection = DatabaseManager.GetConnection())
            {
                var cmd = new SQLiteCommand("select * from Subjects", connection);
                var reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    subjectList.Add(new Subject { SubjectID = reader.GetInt32(0), SubjectName = reader.GetString(1) });
                }
            }
            return subjectList;
        }
        1 reference
        public void Addexam(Exam exam)
        {
            try
            {
                using (var connection = DatabaseManager.GetConnection())
                {
                    SQLiteCommand command = new SQLiteCommand("INSERT INTO Exams (ExamName,SubjectID) VALUES (@examname,@subjectID)", connection);
                    command.Parameters.AddWithValue("@examname", exam.ExamName);
                    command.Parameters.AddWithValue("@subjectID", exam.SubjectID);
                    command.ExecuteNonQuery();
                    MessageBox.Show("successfully.");
                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message); }
        }

        1 reference
        public List<Student> GetStudentList()
        {
            List<Student> studentList = new List<Student>();
            using (var connection = DatabaseManager.GetConnection())
            {
                SQLiteCommand cmd = new SQLiteCommand("select * from Students", connection);
                var reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    studentList.Add(new Student { Name = reader.GetString(1), StudentId = reader.GetInt32(0) });
                }
            }
            return studentList;
        }
```

```
namespace UnicomTICManagementSystem.Controller
    internal class ExamController
        public List<Exam> GetExamData()
            using (var connection = DatabaseManager.GetConnection())
            {
                SQLiteCommand cmd = new SQLiteCommand("select ex.ExamID, ex.ExamName, sub.SubjectID as SubjectID,sub.SubjectName as SubjectName from Exams ex Left join Subjects sub on sub.SubjectID = ex.SubjectID", connection);
                var reader = cmd.ExecuteReader();
                while (reader.Read()) { examList.Add(new Exam { ExamID = reader.GetInt32(0), ExamName = reader.GetString(1),
                    SubjectID = reader.IsDBNull(2) ? (int?)null : reader.GetInt32(2),
                    SubjectName = reader.IsDBNull(3) ? null : reader.GetString(3)
                }); }
            }
            return examList;
        }

        1 reference
        public void EditExamData(Exam exam)
        {
            try
            {
                using (var connection = DatabaseManager.GetConnection())
                {
                    SQLiteCommand cmd = new SQLiteCommand(@"UPDATE Subjects SET SubjectName = @subjectname WHERE SubjectID = @subjectID;update Exams SET ExamName = @examname where ExamID = @examID", connection);
                    cmd.Parameters.AddWithValue("@examID", exam.ExamID);
                    cmd.Parameters.AddWithValue("@examname", exam.ExamName);
                    cmd.Parameters.AddWithValue("@subjectID", exam.SubjectID);
                    cmd.Parameters.AddWithValue("@subjectname", exam.SubjectName);

                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Edit successfully");

                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message); }
        }
        0 references
        public void DeleteExamData(Room room1)
        {
            try
            {
                using (var connection = DatabaseManager.GetConnection())
                {
                    SQLiteCommand cmd = new SQLiteCommand("Delete from Rooms WHERE RoomID = @roomID", connection);
                    cmd.Parameters.AddWithValue("@roomID", room1.RoomId);
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Delete successfully");

                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message); }
        }
```

3. How the App Works

Login System: Authenticates users based on their role (Admin, Staff, Student, Lecturer).
Role Dashboards: Different dashboards are shown depending on the user role.
Data Operations: Add/Edit/Delete supported for Admin; restricted view-only for other roles.
Rooms & Timetable: Admin allocates labs/halls while scheduling; others can only view.
Exam & Marks: Staff and Lecturers can update marks; students can only view their own.

4. Features

Filter (Courses, Subjects, Students, Exams, Rooms in dropdown list)
Error Messages (e.g., “Please select a room”, “Login failed”)
Input Validation (e.g., Score must be 0–100)
Descriptive Mappings for Viewing Data:

- Student name | Course name
- Exam name | Subject name
- Timetable – Subject name | Room name | Time slot
- Room name | Room type
- Subject name | Course name

5. Folder Structure Overview

UnicomTICManagementSystem/

- Models
  - Course.cs
  - Subject.cs
  - Student.cs
  - Exam.cs
  - Mark.cs
  - Room.cs
  - Timetable.cs
  - User.cs
- Views
  - LoginForm.cs
  - MainForm.cs
  - CourseForm.cs
  - StudentForm.cs
  - ExamForm.cs
  - MarkForm.cs
  - TimetableForm.cs
  - RoomForm.cs
  - SubjectForm.cs
- Controllers
  - LoginController.cs
  - CourseController.cs
  - StudentController.cs
  - ExamController.cs
  - MarkController.cs
  - TimetableController.cs
  - RoomController.cs

    - SubjectController.cs
- Repositories
    - DatabaseManager.cs
    - Migration.cs
- Program.cs

6. Conclusion

This project helped in learning:

- How to implement an MVC architecture in a desktop application.
- SQLite CRUD operations in C#.
- Role-based access control with a simple UI.
- Structuring a school management system from scratch.