Unicom Management System — Project Report

****

**👤 Student Information**

**Name:** S.Mathushanth  
**Course:** Web Development  
**Reg. Number:** UT010643  
**IDE:** Visual Studio 2022  
**Language:** C# (.NET Windows Forms)  
**Database:** SQLite

### 📘 Project Overview

This is a Windows Forms–based University Management System developed in C# using SQLite. It helps administrators manage student data, subject lists, exam marks, and timetable scheduling through an interactive desktop interface.

----------------------------------------------------------

### ✔️ Key Features Implemented

1. **Student Module**
   * Add, update, and delete students
   * Secure login credentials with role-based user access
2. **Subject Management**
   * Create and manage subject lists
   * Display subject names and IDs via ComboBoxes
3. **Exam Marks Management**
   * Add, edit, and view exam marks for students
   * Linked with subjects and student IDs
   * Data validation and error handling included
4. **Timetable Scheduling**
   * Assign subjects to time slots and rooms
   * Combines subject and room management
   * Room auto-fill via ComboBox selection
5. **UI Features**
   * Responsive Windows Forms with consistent layout
   * Data displayed in DataGridView controls
   * Error messages and input validations
6. **Database**
   * Local database powered by SQLite
   * Separate service classes for each table
   * Uses parameterized queries to avoid SQL injection

--------------------------------------------------------------

### 🛠️ Technologies Used

| **Component** | **Technology** |
| --- | --- |
| Programming | C# |
| Framework | .NET Framework (WinForms) |
| Database | SQLite |
| IDE | Visual Studio 2022 |
| UI Components | Windows Forms, DataGridView, ComboBox |

### ⚠️ Challenges Faced & Solutions

* **Constraint failed (NOT NULL)**  
  🛠 Fixed by validating input fields before executing database insert queries.
* **ComboBox data binding errors**  
  🛠 Resolved by binding strongly typed object lists (e.g., List<Subject>) and setting DisplayMember / ValueMember.
* **InvalidCastException and FormatException**  
  🛠 Fixed by using as keyword for safe casting and checking null values. Also avoided parsing non-numeric strings with int.Parse.
* **Room auto-fill in timetable**  
  🛠 Solved by binding ComboBox to Room objects and handling SelectedIndexChanged to populate the room name field.

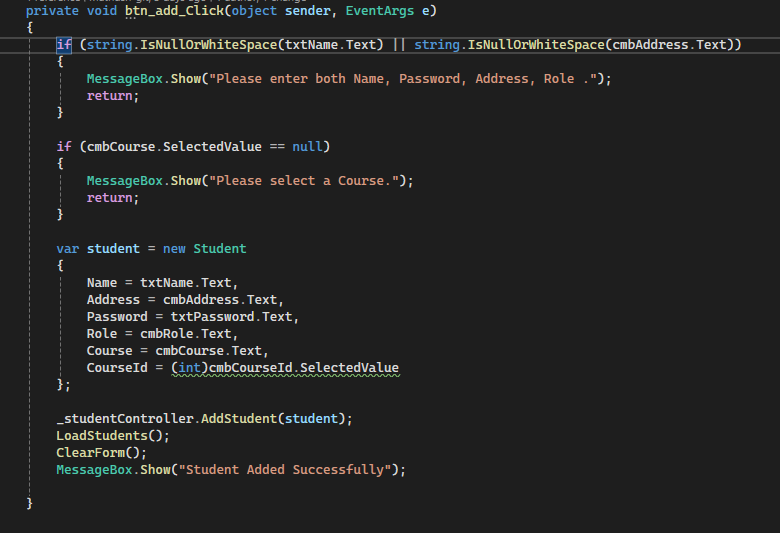
--------------------------------------------------------------

### 📷 Code Sample Screenshots (to be included on GitHub)

You can include the following screenshots in your GitHub repository under a folder like /screenshots/:

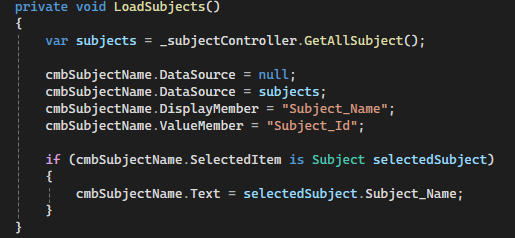
1. **Student Form Code** – with validation and ComboBox binding

**StudentForm**.CS ---(validation + ComboBox)

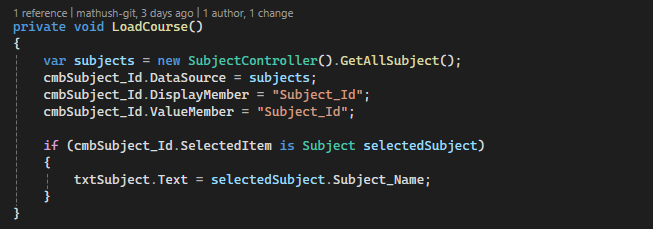


1. **Subject ComboBox Binding** – showing DisplayMember and ValueMember

Exam\_MarkForm.CS

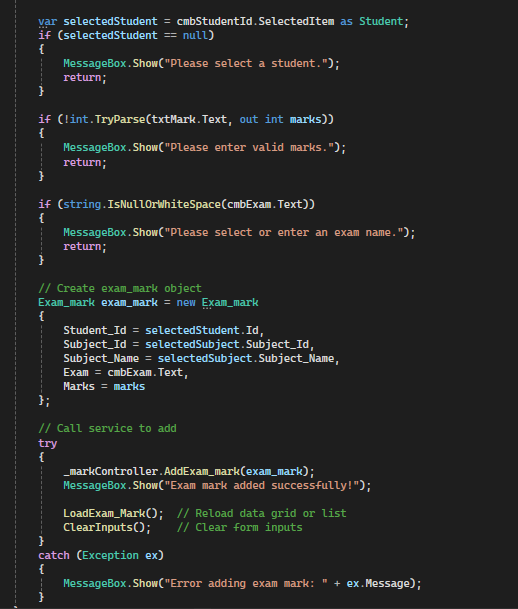


Time\_Table.CS



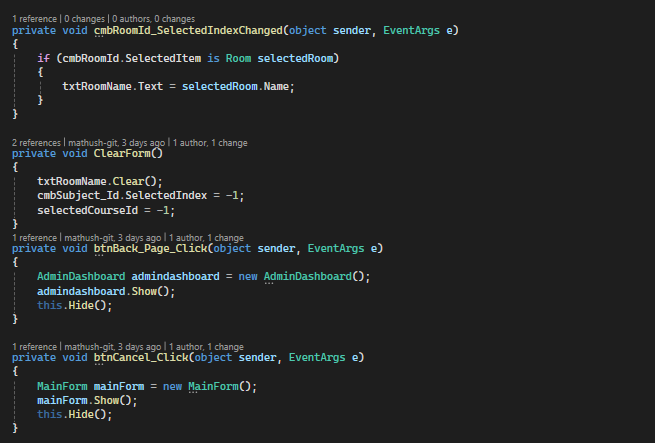
1. **Exam Mark Entry Code** – with null-checks and safe casting

Exam\_MarkForm.cs → btnAdd\_Click



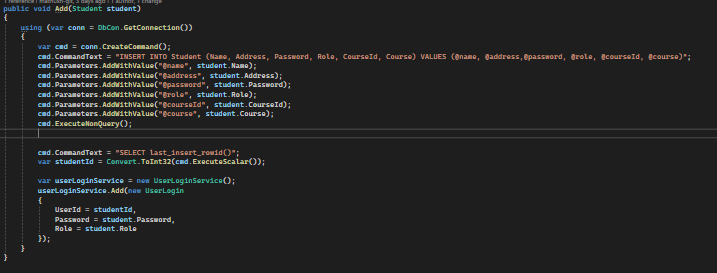
1. **Timetable Form Code** – auto-filling Room Name from selected Room ID

Time\_Table.cs → cmbRoomId\_SelectedIndexChanged (after fix)



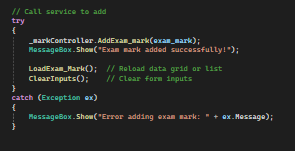
1. **SQLite Command Execution** – using cmd.Parameters.AddWithValue(...)

cmd.Parameters.AddWithValue(...)



1. **Error Handling Code** – e.g., handling System.FormatException

btnAdd\_Click()



## 📋 **Project Summary by Module**

### 1️⃣ ****Student Module****

* Add, update, delete student records.
* Assign students to courses.
* Automatically creates login credentials upon registration.

### 2️⃣ ****Lecturer Module****

* Manage lecturer details (name, contact, etc.).
* Link lecturers to specific subjects or timetables.
* Supports role-based login access.

### 3️⃣ ****Course Module****

* Add and maintain courses offered by the university.
* Courses are linked with students and subjects.

### 4️⃣ ****Subject Module****

* Add and assign subjects to courses.
* Each subject has a unique ID and name.
* Used in Exam Mark and Timetable modules.

### 5️⃣ ****Exam Marks Module****

* Record exam marks for students per subject and exam type.
* Features validation and update functionality.
* Displays data in a DataGridView.

### 6️⃣ ****Timetable Module****

* Assign subjects to rooms and timeslots.
* ComboBox selection for room and subject.
* Automatically displays subject name when selected by ID.

### 7️⃣ ****Room Module****

* Manage rooms (Room ID and Room Name).
* Used in timetable scheduling.
* Room ID selected from ComboBox, auto-fills Room Name.

### 8️⃣ ****Authentication Module****

* Role-based login system (Student, Lecturer, Admin).
* Prevents unauthorized access to admin features.

### 9️⃣ ****Error Handling and Validation****

* Input validation for every form field.
* Handles:
  + FormatException when parsing input.
  + NullReferenceException when ComboBox selections are missing.
* Prevents invalid database inserts.

## 🖥️ **Dashboard Module Explanation**

### 📌 What is the Dashboard?

The **Dashboard** is the **main control panel** of your University Management System. After logging in, users are redirected to the Dashboard based on their role (Admin, Student, or Lecturer). It provides quick access to all the major functions of the application.

### 👥 Role-Based Access

The Dashboard is **dynamic**, meaning the features shown depend on the user role:

| **Role** | **Features Accessed from Dashboard** |
| --- | --- |
| **Admin** | Full access to Student, Lecturer, Course, Subject, Exam Marks, Timetable, Rooms |
| **Lecturer** | Limited access to Exam Marks and Timetable |
| **Student** | View personal marks and timetable |

### 🧩 Modules Linked via Dashboard

Each button or tile in the Dashboard opens a corresponding form:

* **👨‍🎓 Student Management:** Opens the Student form to manage students.
* **📚 Subject Management:** Opens the Subject form to add/edit subjects.
* **📖 Course Management:** Manages course details and links with students.
* **🧑‍🏫 Lecturer Management:** Adds or edits lecturer details.
* **📝 Exam Mark Entry:** Opens the mark entry form with ComboBoxes for subject & student.
* **📅 Timetable:** Assigns subjects and rooms to timeslots.
* **🏫 Room Management:** Manages room IDs and names.
* **🚪 Logout Button:** Returns to the login form.

### 🖼️ Visual Elements

* **Background Image:** A custom background (e.g., A.jpg) gives it a clean look.
* **Buttons:** Styled and clickable — open respective forms.
* **Title Label:** Usually shows the application name or user role.
* **User Info:** Optionally displays username or role for context.

### ✅ Key Features

* **User-Friendly UI:** Easy navigation for all user levels.
* **Centralized Access:** One screen to access all features.
* **Role Filtering:** Prevents unauthorized users from accessing admin-only features.
* **Event Handling:** Button click events use form.Show(); this.Hide(); for smooth transitions.

**The Project Works** — Step by Step

### 🔐 ****Step 1: Login Page****

* When the application starts, users see a **login form**.
* Users enter **username & password**.
* The system checks the credentials from the UserLogin table.
* Based on the user's **role** (Admin, Student, or Lecturer), the correct **dashboard** is loaded.

### 🧭 ****Step 2: Dashboard Navigation****

After login:

* The user is directed to a **role-specific Dashboard**.
* The Dashboard shows buttons (or tiles) for modules like:
  + Student
  + Course
  + Subject
  + Exam Marks
  + Time Table
  + Room Management

✅ **Admin** sees all modules  
✅ **Lecturer** sees exam and timetable modules  
✅ **Student** sees only their marks and timetable

### 👨‍🎓 ****Step 3: Student Module****

* Add, update, delete student records.
* Enter details: name, address, password, course, etc.
* When a student is added:
  + The system also creates a login for them (stored in UserLogin table).
* Student list is displayed using a **DataGridView**.

### 📚 ****Step 4: Subject & Course Management****

* Admin can add or edit subjects and courses.
* Subjects are assigned unique Subject\_Id and name.
* Courses are stored with CourseId and course name.
* Subjects can be linked to students or timetable later.

### 📝 ****Step 5: Exam Marks Entry****

* Choose a **student** and **subject** using ComboBoxes.
* Enter exam type (e.g., Midterm, Final) and marks.
* Data is saved into the Exam\_marks table.
* Marks can be updated or deleted.
* Data shown in a DataGridView for review.

### 🕒 ****Step 6: Time Table Scheduling****

* Select a subject and assign a **room ID** and **time slot**.
* When a room is selected, the **room name auto-fills**.
* Schedule is saved into the TimeTable table.
* Full timetable shown in a DataGridView.

### 🏫 ****Step 7: Room Management****

* Admin can create room records with Room\_Id and Room\_Name.
* Rooms are used in the timetable scheduling.
* ComboBox selection allows auto-fill of room info.

### 🛡️ ****Step 8: Validation & Error Handling****

* Before any data is saved, forms check for:
  + Empty fields
  + Wrong formats (e.g., marks must be numbers)
  + Missing ComboBox selections
* All SQL statements are parameterized to prevent SQL injection.
* Errors like FormatException, InvalidCastException, and NullReferenceException are caught and shown using MessageBox.

### 💾 ****Step 9: Data Storage in SQLite****

* All data (students, subjects, marks, timetable) is saved in a local **SQLite** database.
* Each module uses a corresponding service class to:
  + Connect to the DB
  + Run INSERT, SELECT, UPDATE, DELETE queries