

Unicom TIC Management System – Final Project Report

Developed Using:

- Language: C#
- Framework: Windows Forms
- Database: SQLite

Submitted by:

Sathusha Santhiravarnan

UT Number: UT010651

Introduction

The Unicom TIC Management System (UMS) is a role-based Windows desktop application for educational institutions. It supports user management, academic scheduling, attendance, exams, and analytics within a secure, modular structure built using a fully layered architecture.

Project Objectives

- Design a user-friendly interface using C# Windows Forms
- Implement a complete layered architecture using Models, DTOs, Enums, Mappers, Repositories, Services, and Controllers
- Use SQLite for persistent storage with full CRUD operations

- Restrict registration to Admins only (no public sign-up)
- Enable role-based login for Admin, Student, Staff, Lecturer, and Mentor
- Add core modules like Timetable, Attendance, Marks, Top Performers, and Activity Logs
- Auto-generate IDs: Admission Numbers, Employee IDs, Lecturer IDs
- Ensure secure password handling using a custom PasswordHasher utility
- Validate all forms using Try-Catch blocks and user-friendly error feedback

System Architecture

The system follows a clean, scalable, and maintainable layered architecture:

DTO → Mapper → Model → Enums → I<Repository> → Repository → I<Service> → Service → Controller → Form (UI)

This ensures separation of concerns, testability, and clean code organization.

Modules Implemented

- User Management: Admin-only account creation and filtering by roles
- Login and Role Access: Secure login and redirection
- NIC Verification: Validates NIC before registration
- Course, Subject, Department Management: Full CRUD functionality
- Mentor Module: Bonus feature to register and manage mentors
- Student Module: Admin approval with auto-generated Admission Numbers
- Staff Module: Admin-managed staff records with auto-generated Employee IDs
- Lecturer Module: Department selection with auto-generated IDs
- Timetable Module: Course scheduling with Room, TimeSlot, Lecturer, and Group
- Attendance Module: Daily attendance tracking per student and group
- Marks Module: Record and view exam scores
- Top Performers Module: Identify top-scoring students
- Activity Logs Module: Audit and track admin actions
- Group Structure: Manage MainGroups and SubGroups for class splits

- Password Utility: Secure password hashing with PasswordHasher.cs

Validations and Utilities

- Registration restricted to Admins
- Auto ID generation for Students, Staff, and Lecturers
- Passwords hashed via a reusable utility
- Try-Catch error handling and validation in all forms
- Enum logic used to manage roles, statuses, gender, days, and leave types

Database Tables

Auto-generated using Migration.cs:

- Users, NICDetails, Departments, Courses, Subjects, CourseSubjects, Mentors
- Students, Staff, Lecturers, Exams, Marks, Timetables, TimeSlots, Attendance
- TopPerformers, ActivityLogs, MainGroups, SubGroups

Key Highlights

- Fully implemented: Interfaces, DTOs, Mappers, Repositories, Services, Enums, Controllers
- Enums used: UserRole, AttendanceStatus, GenderType, LeaveStatus, DayOfWeekType
- Mentor Module developed beyond requirement
- Attendance integrated with Timetable and Marks
- Timetable supports full scheduling workflow
- Role-based permissions secured via logic
- Password encryption handled via utility
- Clean UI and layered business logic

Bonus Features

- Mentor Management (extra module)
- Full-featured Timetable with Room, Group, Slot, Lecturer assignment
- Activity Log for admin monitoring
- Top Performers Module (rank students using marks)
- Attendance and Marks tightly linked for performance tracking
- Auto-generation of all IDs
- Centralized use of Enums across modules
- Strict access: Only Admin can register users
- Fully layered structure with reusable components:

DTO → Mapper → Model → Enums → I<Repository> → Repository → I<Service>
→ Service → Controller → Form (UI)

Final Status

- All modules completed, tested, and integrated
- Bonus modules like Mentor, Attendance, and Top Performers included
- Secure, validated, and submission-ready system
- Ready for academic submission, internship demo, or portfolio use

References

1. **ChatGPT** – Technical consultation and design suggestions

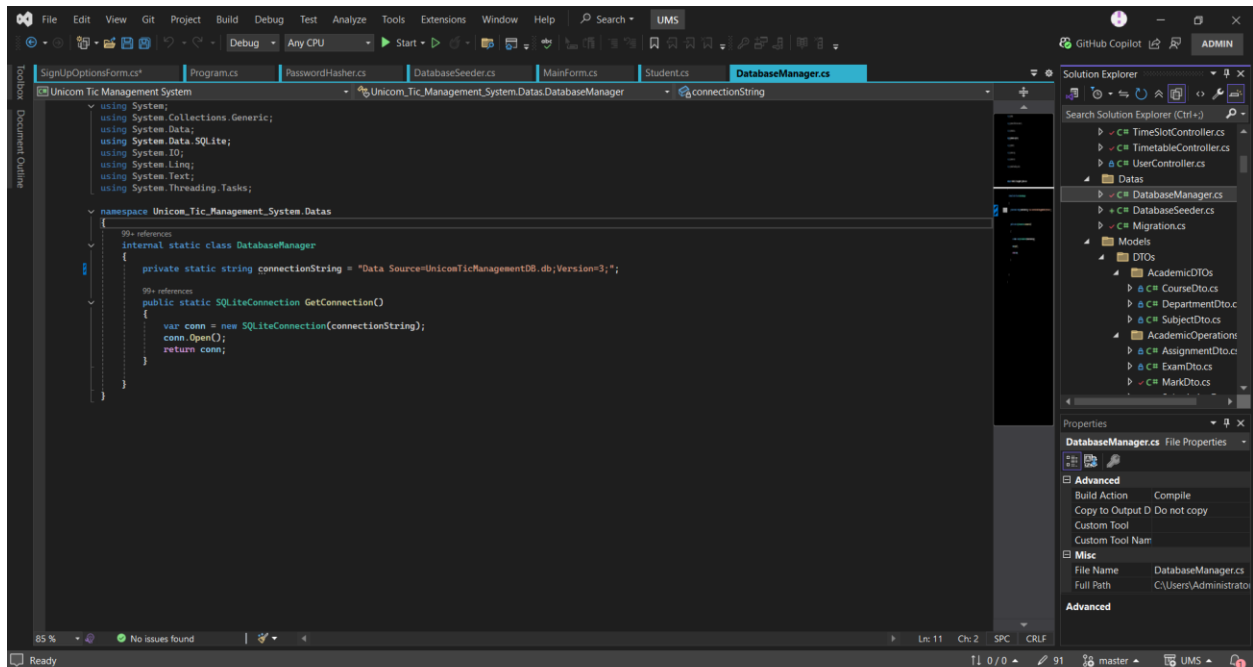
OpenAI, <https://chat.openai.com>

2. **Gemini (Google AI)** – Code samples, logic design guidance

Google, <https://gemini.google.com>

3. Official documentation for:
 - a. C# and Windows Forms (docs.microsoft.com)

b. SQLite (sqlite.org)



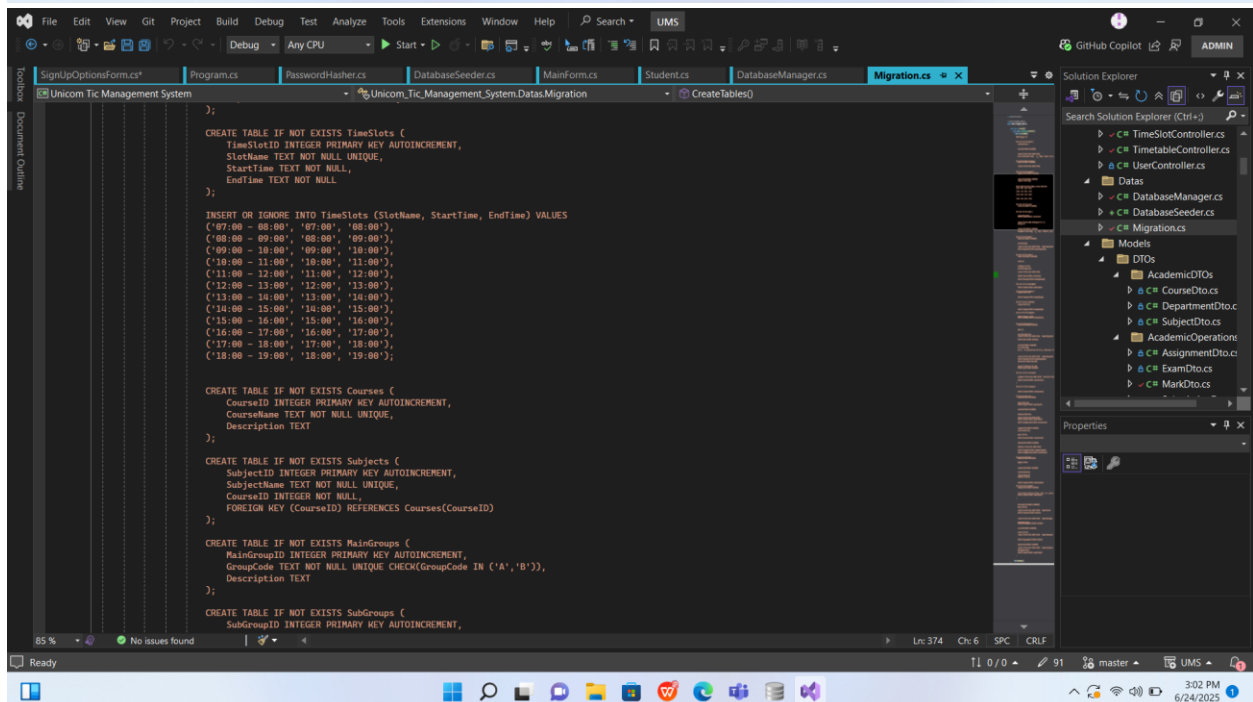
This screenshot shows the Visual Studio IDE with the `DatabaseManager.cs` file open. The code defines a static class `DatabaseManager` within the `Unicom_Tic_Management_System_Datas` namespace. It includes a private static `connectionString` and a public static `GetConnection()` method that returns a `SQLiteConnection` object.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SQLite;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Unicom_Tic_Management_System_Datas
{
    //+ references
    internal static class DatabaseManager
    {
        private static string connectionString = "Data Source=UnicomTicManagementDB.db;Version=3;";

        //+ references
        public static SQLiteConnection GetConnection()
        {
            var conn = new SQLiteConnection(connectionString);
            conn.Open();
            return conn;
        }
    }
}
```

The Solution Explorer on the right shows the project structure, including folders for `Controllers`, `Models`, `Dtos`, and `Operations`. The Properties window shows the file name `DatabaseManager.cs` and its full path.



This screenshot shows the Visual Studio IDE with the `Migration.cs` file open. The code contains SQL statements to create tables for `TimeSlots`, `Courses`, `Subjects`, `MainGroups`, and `SubGroups`. It also includes an `INSERT OR IGNORE` statement for the `TimeSlots` table.

```
);

CREATE TABLE IF NOT EXISTS TimeSlots (
    TimeSlotID INTEGER PRIMARY KEY AUTOINCREMENT,
    SlotName TEXT NOT NULL UNIQUE,
    StartTime TEXT NOT NULL,
    EndTime TEXT NOT NULL
);

INSERT OR IGNORE INTO TimeSlots (SlotName, StartTime, EndTime) VALUES
('07:00 - 08:00', '07:00', '08:00'),
('08:00 - 09:00', '08:00', '09:00'),
('09:00 - 10:00', '09:00', '10:00'),
('10:00 - 11:00', '10:00', '11:00'),
('11:00 - 12:00', '11:00', '12:00'),
('12:00 - 13:00', '12:00', '13:00'),
('13:00 - 14:00', '13:00', '14:00'),
('14:00 - 15:00', '14:00', '15:00'),
('15:00 - 16:00', '15:00', '16:00'),
('16:00 - 17:00', '16:00', '17:00'),
('17:00 - 18:00', '17:00', '18:00'),
('18:00 - 19:00', '18:00', '19:00');

CREATE TABLE IF NOT EXISTS Courses (
    CourseID INTEGER PRIMARY KEY AUTOINCREMENT,
    CourseName TEXT NOT NULL UNIQUE,
    Description TEXT
);

CREATE TABLE IF NOT EXISTS Subjects (
    SubjectID INTEGER PRIMARY KEY AUTOINCREMENT,
    SubjectName TEXT NOT NULL UNIQUE,
    CourseID INTEGER NOT NULL,
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);

CREATE TABLE IF NOT EXISTS MainGroups (
    MainGroupID INTEGER PRIMARY KEY AUTOINCREMENT,
    GroupCode TEXT NOT NULL UNIQUE CHECK(GroupCode IN ('A','B')),
    Description TEXT
);

CREATE TABLE IF NOT EXISTS SubGroups (
    SubGroupID INTEGER PRIMARY KEY AUTOINCREMENT,
```

The Solution Explorer on the right shows the project structure, including folders for `Controllers`, `Models`, `Dtos`, and `Operations`. The Properties window shows the file name `Migration.cs` and its full path.

