UnicomTIC Management System – Project Report

Submitted by: Sathusha Santhiravarnan

Student ID: UT010651

1. Introduction

- The **UnicomTIC Management System (UMS)** is a desktop-based academic and administrative software designed for educational institutions.
- Developed using **C# WinForms** with **SQLite** database.
- Follows the **Model-View-Controller (MVC)** architecture for clean separation of concerns.

2. Objective

- Automate academic and user management processes.
- Allow only pre-verified NIC holders (UnicomTIC users) to register.
- Implement **automatic ID generation** for Students and Employees.
- Use structured coding with **Models, DTOs, Enums, Mappers, Repositories, Services, Controllers**.
- Ensure secure password handling and full data validation.
- Limit critical actions like Add/Update/Delete to **Admin only**.
- Record all major activities and identify top-performing students.

3. Key Features

3.1 User Registration and Validation

- Only users whose NIC exists in the **NICDetails** table can register.
- Unauthorized NICs are blocked from registering.
- Automatic ID generation:
    - Students → `UTI0001`, `UTI0002`, ...
    - Lecturers → `LecUTI0001`, `LecUTI0002`, ...

    - Staff → `EmpUTI0001`, `EmpUTI0002`, ...
- Passwords are hashed using a separate utility class for secure login.

3.2 Role-Based Access Control

- User roles: **Admin, Student, Lecturer, Staff, Mentor**.
- Only **Admin** can perform Add, Update, and Delete operations.
- Role-based form access and data visibility enforced.

3.3 Academic and Group Management

- Create and manage **Courses** and **Subjects**.
- Support for **Group hierarchy**:
    - **Main Groups** (e.g., Year 1, Year 2)
    - **Sub Groups** (e.g., A, B)
- Assign **Mentors** to groups.
- Add and view timetable entries based on group and subject.

3.4 Timetable and Attendance

- Create timetables with support for **parallel sessions** for different subgroups.
- Assign **Lecture Halls** or **Labs** to specific activities.
- Record **student attendance** by group and session.

3.5 Performance and Logging

- **Top Performers** module identifies and displays highest-scoring students.
- **Activity Log** tracks login, registration, update actions, and other events.

3.6 Validations and Error Handling

- **Try-Catch** blocks used in all layers (UI, Controllers, Services, Repositories).
- Validates all user inputs (e.g., empty fields, email format, NIC uniqueness).
- All registration steps are securely handled with exception feedback.

4. Project Architecture

- Follows **Modular and Layered Architecture**:

4.1 Models

- Represent database tables (User, Course, Subject, etc.)

4.2 DTOs (Data Transfer Objects)

- Used to safely transfer filtered data between UI and logic layers.

4.3 Enums

- Define constants for roles, activity types, etc.

4.4 Mappers

- Convert between Models ↔ DTOs automatically.

4.5 Interfaces and Repositories

- Repositories perform all **CRUD operations**, abstracted via **Interfaces**.

4.6 Services and Interfaces

- Contain **business logic**, validations, and processing rules.

4.7 Controllers

- Bridge between Services and Forms (UI), handle form events and exceptions.

4.8 Utilities

- Includes a separate folder with a class called **PasswordHasher**:
    - Used to hash and validate passwords during login and registration.
    - Secure password handling avoids plain-text storage.

5. Technologies Used

Layer	**Technology**
Frontend	C# WinForms
Backend	.NET (MVC Pattern)
Database	SQLite
Architecture Style	Layered MVC
Security	Password Hasher Class
Validation	Try-Catch + Field Checks

6. Admin Privileges

- Only **Admin** users can:
    - Add, update, or delete Courses, Subjects, Groups, Users
    - Assign Mentors

    - Generate and manage IDs
    - View full logs and performance reports
- Other roles have **read-only or restricted permissions**.

7. Extra Modules

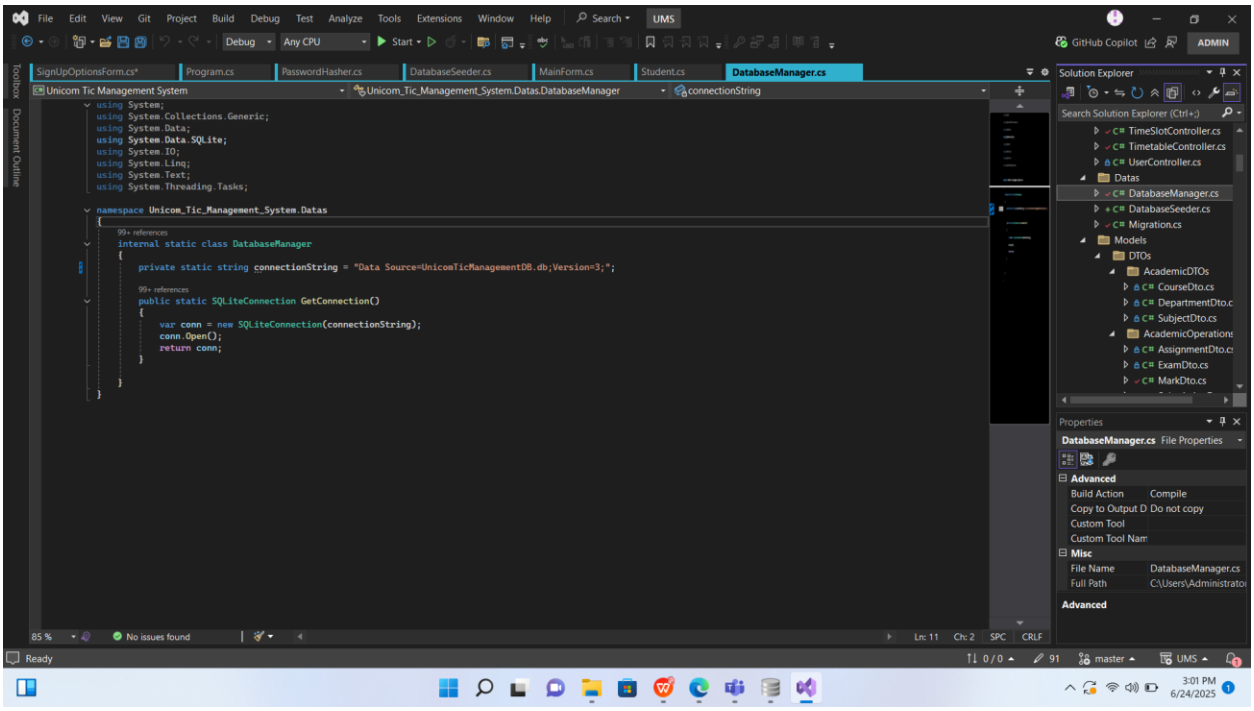
- **Mentor Assignment** – Assign mentors to groups.
- **Top Performers** – Monitor and display top-ranking students by marks.
- **Group Management** – Separate Main and Sub Groups.
- **Timetable Management** – Handle concurrent group sessions.
- **Attendance Module** – Record attendance based on group/subject/session.
- **Activity Logging** – Log all major events for audit and transparency.

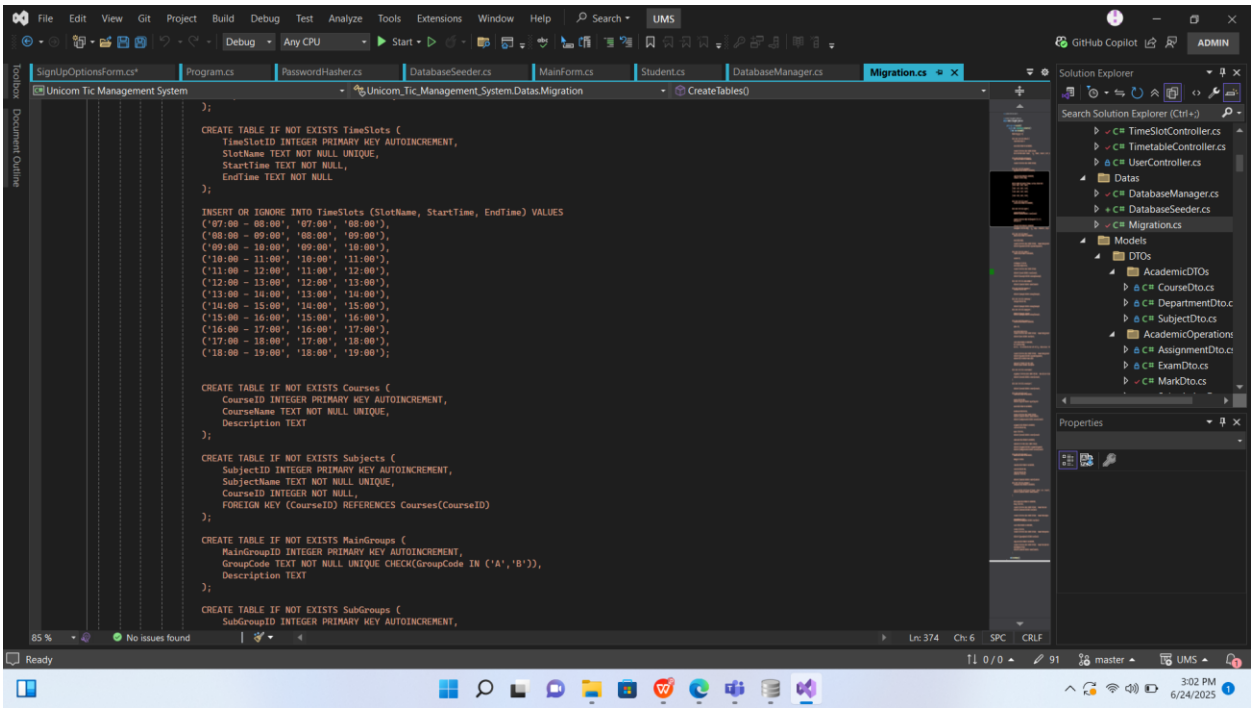
8. Validation and Exception Handling

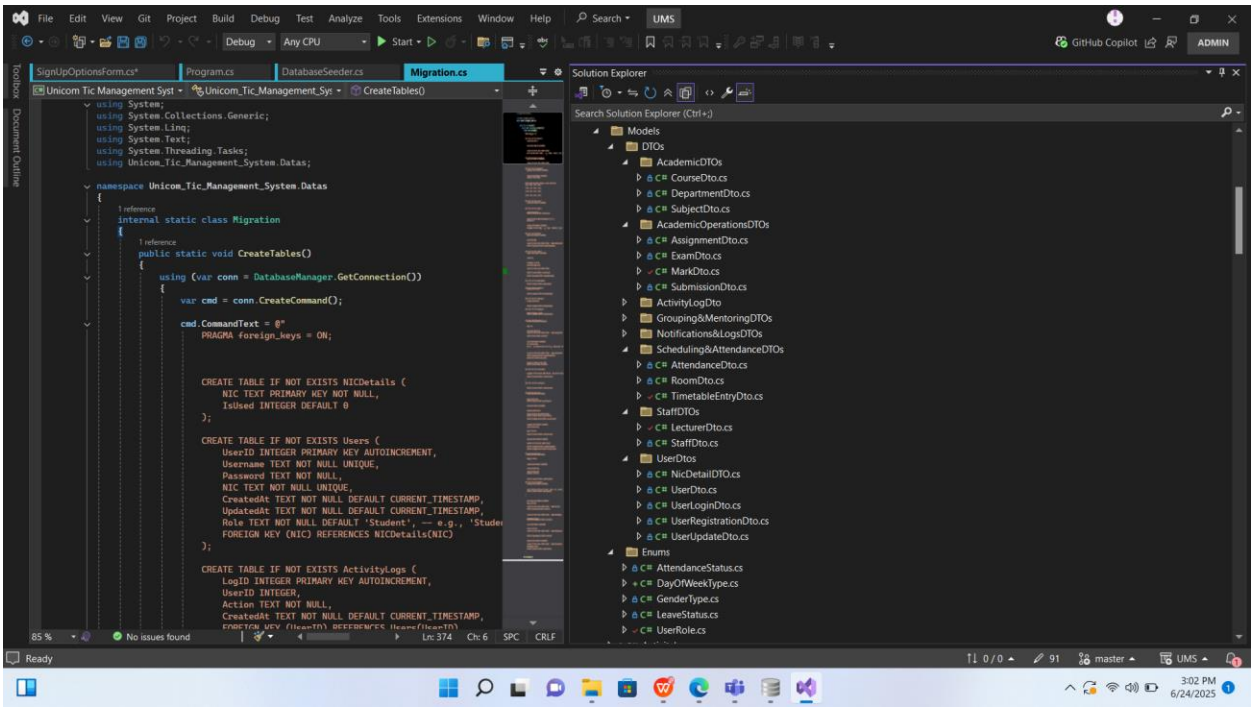
- Input validation on every form field.
- NIC validation ensures only approved users register.
- Passwords are securely hashed and validated.
- **Try-Catch blocks** in every layer to catch and handle errors gracefully.

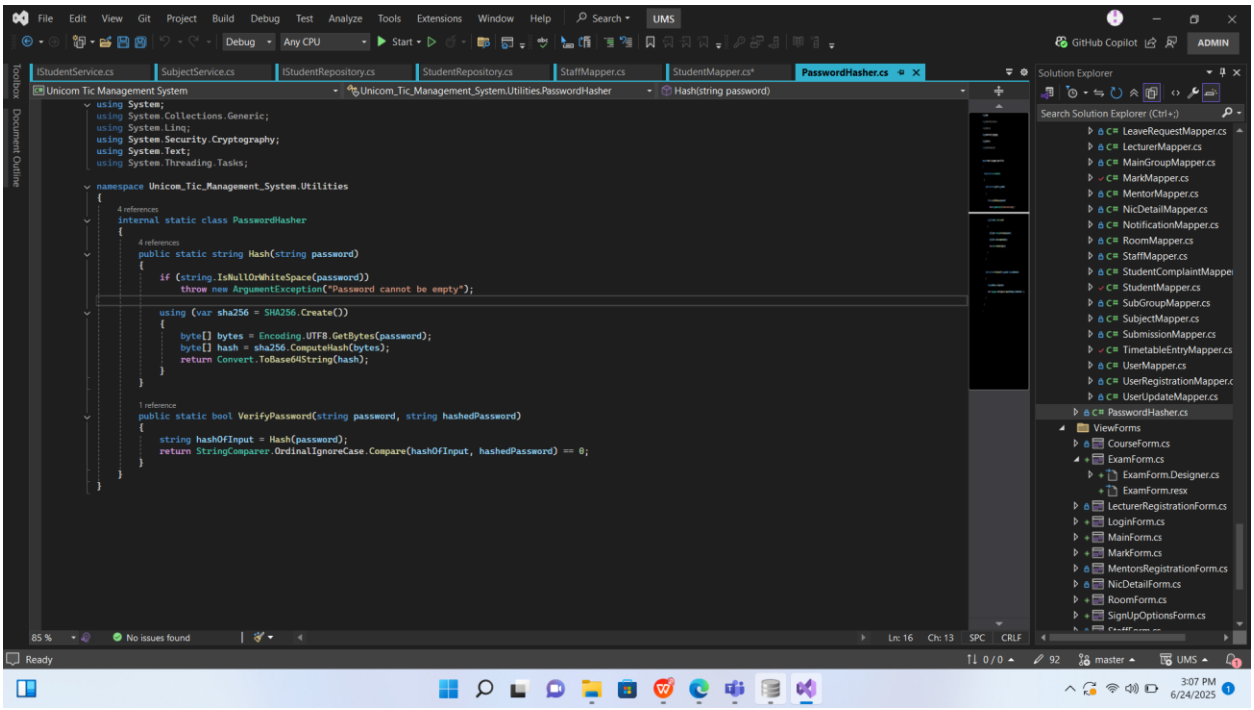
9. Conclusion

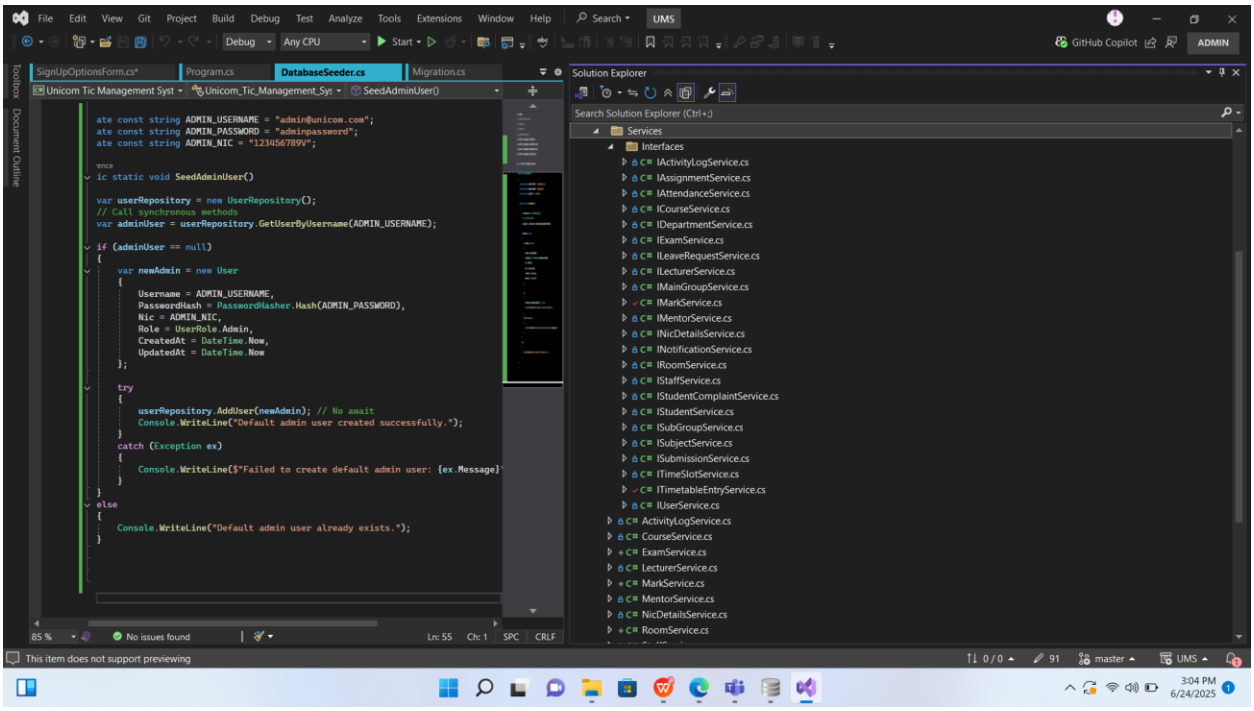
- The **UnicomTIC Management System** is a secure, well-structured, and scalable academic platform.
- It follows **clean coding standards**, supports layered separation, and offers real-world functionality like registration validation, automatic ID generation, timetable scheduling, and mentor management.
- With strong exception handling and password security, the system is suitable for practical use in schools or institutes.

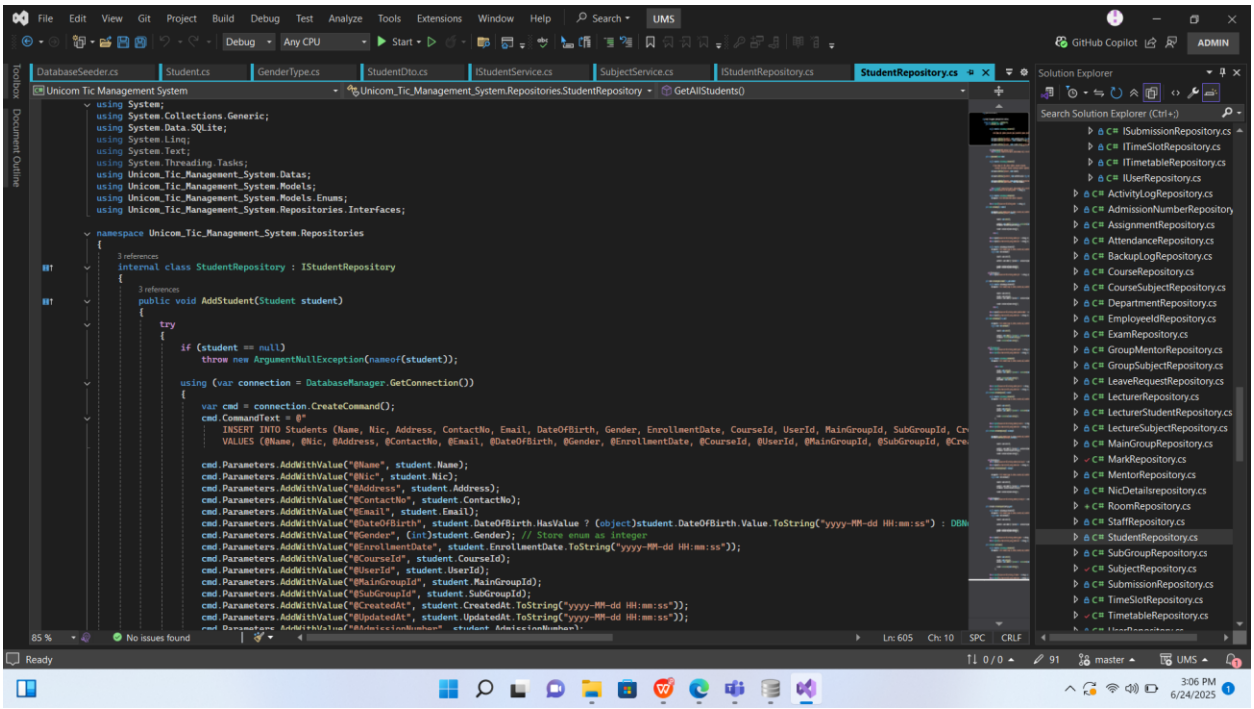

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
UMS
Debug
Any CPU
Start
GitHub Copilot
ADMIN
SignUpOptionsForm.cs*
Program.cs
PasswordHasher.cs
DatabaseSeeder.cs
MainForm.cs
Student.cs
DatabaseManager.cs
Unicom Tic Management System
Unicom_Tic_Management_System.Datas.DatabaseManager
connectionString
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
{
99+ references
internal static class DatabaseManager
{
private static string connectionString = "Data Source=UnicomTicManagementDB.db;Version=3;";
99+ references
public static SQLiteConnection GetConnection()
{
var conn = new SQLiteConnection(connectionString);
conn.Open();
return conn;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
TimeSlotController.cs
TimetableController.cs
UserController.cs
Datas
DatabaseManager.cs
DatabaseSeeder.cs
Migration.cs
Models
DTOs
AcademicDTOs
CourseDto.cs
DepartmentDto.c
SubjectDto.cs
AcademicOperations
AssignmentDto.cs
ExamDto.cs
MarkDto.cs
Properties
DatabaseManager.cs File Properties
Advanced
Build Action Compile
Copy to Output D Do not copy
Custom Tool
Custom Tool Nam
Misc
File Name DatabaseManager.cs
Full Path C:\Users\Administrator
Advanced
85 %
No issues found
Ln: 11
Ch: 2
SPC
CRLF
Ready
master
UMS
3:01 PM
6/24/2025


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
UMS
Debug
Any CPU
Start
GitHub Copilot
ADMIN
SignUpOptionsForm.cs*
Program.cs
PasswordHasher.cs
DatabaseSeeder.cs
MainForm.cs
Student.cs
DatabaseManager.cs
Migration.cs
Unicom Tic Management System
Unicom_Tic_Management_System.Datas.Migration
CreateTables()
);
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,
Solution Explorer
Search Solution Explorer (Ctrl+;)
TimeSlotController.cs
TimetableController.cs
UserController.cs
Datas
DatabaseManager.cs
DatabaseSeeder.cs
Migration.cs
Models
DTOs
AcademicDTOs
CourseDto.cs
DepartmentDto.c
SubjectDto.cs
AcademicOperations
AssignmentDto.cs
ExamDto.cs
MarkDto.cs
Properties
85 %
No issues found
Ln: 374
Ch: 6
SPC
CRLF
Ready
master
UMS
3:02 PM
6/24/2025


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
UMS
Debug
Any CPU
Start
GitHub Copilot
ADMIN
SignUpOptionsForm.cs*
Program.cs
DatabaseSeeder.cs
Migration.cs
Unicom Tic Management Syst
Unicom_Tic_Management_Sys
CreateTables()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Unicom_Tic_Management_System.Datas;
namespace Unicom_Tic_Management_System.Datas
{
1 reference
internal static class Migration
{
1 reference
public static void CreateTables()
{
using (var conn = DatabaseManager.GetConnection())
{
var cmd = conn.CreateCommand();
cmd.CommandText = @"
PRAGMA foreign_keys = ON;
CREATE TABLE IF NOT EXISTS NICDetails (
NIC TEXT PRIMARY KEY NOT NULL,
IsUsed INTEGER DEFAULT 0
);
CREATE TABLE IF NOT EXISTS Users (
UserID INTEGER PRIMARY KEY AUTOINCREMENT,
Username TEXT NOT NULL UNIQUE,
Password TEXT NOT NULL,
NIC TEXT NOT NULL UNIQUE,
CreatedAt TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
UpdatedAt TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
Role TEXT NOT NULL DEFAULT 'Student', -- e.g., 'Stude
FOREIGN KEY (NIC) REFERENCES NICDetails(NIC)
);
CREATE TABLE IF NOT EXISTS ActivityLogs (
LogID INTEGER PRIMARY KEY AUTOINCREMENT,
UserID INTEGER,
Action TEXT NOT NULL,
CreatedAt TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
85 %
No issues found
Ln: 374
Ch: 6
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Models
DTOs
AcademicDTOs
CourseDto.cs
DepartmentDto.cs
SubjectDto.cs
AcademicOperationsDTOs
AssignmentDto.cs
ExamDto.cs
MarkDto.cs
SubmissionDto.cs
ActivityLogDto
Grouping&MentoringDTOs
Notifications&LogsDTOs
Scheduling&AttendanceDTOs
AttendanceDto.cs
RoomDto.cs
TimetableEntryDto.cs
StaffDTOs
LecturerDto.cs
StaffDto.cs
UserDtos
NicDetailDTO.cs
UserDto.cs
UserLoginDto.cs
UserRegistrationDto.cs
UserUpdateDto.cs
Enums
AttendanceStatus.cs
DayOfWeekType.cs
GenderType.cs
LeaveStatus.cs
UserRole.cs
Ready
0 / 0
91
master
UMS
3:02 PM
6/24/2025


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
UMS
Debug
Any CPU
Start
GitHub Copilot
ADMIN
IStudentService.cs
SubjectService.cs
IStudentRepository.cs
StudentRepository.cs
StaffMapper.cs
StudentMapper.cs*
PasswordHasher.cs
Unicom Tic Management System
Unicom_Tic_Management_System.Utilities.PasswordHasher
Hash(string password)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
namespace Unicom_Tic_Management_System.Utilities
{
4 references
internal static class PasswordHasher
{
4 references
public static string Hash(string password)
{
if (string.IsNullOrWhiteSpace(password))
throw new ArgumentException("Password cannot be empty");
using (var sha256 = SHA256.Create())
{
byte[] bytes = Encoding.UTF8.GetBytes(password);
byte[] hash = sha256.ComputeHash(bytes);
return Convert.ToBase64String(hash);
}
}
1 reference
public static bool VerifyPassword(string password, string hashedPassword)
{
string hashOfInput = Hash(password);
return StringComparer.OrdinalIgnoreCase.Compare(hashOfInput, hashedPassword) == 0;
}
}
}
85 %
No issues found
Ln: 16
Ch: 13
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
LeaveRequestMapper.cs
LecturerMapper.cs
MainGroupMapper.cs
MarkMapper.cs
MentorMapper.cs
NicDetailMapper.cs
NotificationMapper.cs
RoomMapper.cs
StaffMapper.cs
StudentComplaintMapper
StudentMapper.cs
SubGroupMapper.cs
SubjectMapper.cs
SubmissionMapper.cs
TimetableEntryMapper.cs
UserMapper.cs
UserRegistrationMapper.c
UserUpdateMapper.cs
PasswordHasher.cs
ViewForms
CourseForm.cs
ExamForm.cs
ExamForm.Designer.cs
ExamForm.resx
LecturerRegistrationForm.cs
LoginForm.cs
MainForm.cs
MarkForm.cs
MentorsRegistrationForm.cs
NicDetailForm.cs
RoomForm.cs
SignUpOptionsForm.cs
Ready
0 / 0
92
master
UMS
3:07 PM
6/24/2025


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
UMS
Debug
Any CPU
Start
GitHub Copilot
ADMIN
SignUpOptionsForm.cs*
Program.cs
DatabaseSeeder.cs
Migration.cs
Unicom Tic Management Syst
Unicom_Tic_Management_Sys
SeedAdminUser()
ate const string ADMIN_USERNAME = "admin@unicom.com";
ate const string ADMIN_PASSWORD = "adminpassword";
ate const string ADMIN_NIC = "123456789V";
ic static void SeedAdminUser()
var userRepository = new UserRepository();
// Call synchronous methods
var adminUser = userRepository.GetUserByUsername(ADMIN_USERNAME);
if (adminUser == null)
{
var newAdmin = new User
{
Username = ADMIN_USERNAME,
PasswordHash = PasswordHasher.Hash(ADMIN_PASSWORD),
Nic = ADMIN_NIC,
Role = UserRole.Admin,
CreatedAt = DateTime.Now,
UpdatedAt = DateTime.Now
};
try
{
userRepository.AddUser(newAdmin); // No await
Console.WriteLine("Default admin user created successfully.");
}
catch (Exception ex)
{
Console.WriteLine($"Failed to create default admin user: {ex.Message}
}
}
else
{
Console.WriteLine("Default admin user already exists.");
}
85 %
No issues found
Ln: 55
Ch: 1
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Services
Interfaces
IActivityLogService.cs
IAssignmentService.cs
IAttendanceService.cs
ICourseService.cs
IDepartmentService.cs
IExamService.cs
ILeaveRequestService.cs
ILecturerService.cs
IMainGroupService.cs
IMarkService.cs
IMentorService.cs
INicDetailsService.cs
INotificationService.cs
IRoomService.cs
IStaffService.cs
IStudentComplaintService.cs
IStudentService.cs
ISubGroupService.cs
ISubjectService.cs
ISubmissionService.cs
ITimeSlotService.cs
ITimetableEntryService.cs
IUserService.cs
ActivityLogService.cs
CourseService.cs
ExamService.cs
LecturerService.cs
MarkService.cs
MentorService.cs
NicDetailsService.cs
RoomService.cs
This item does not support previewing
0 / 0
91
master
UMS
3:04 PM
6/24/2025


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
UMS
Debug
Any CPU
Start
GitHub Copilot
ADMIN
DatabaseSeeder.cs
Student.cs
GenderType.cs
StudentDto.cs
IStudentService.cs
SubjectService.cs
IStudentRepository.cs
StudentRepository.cs
Unicom Tic Management System
Unicom_Tic_Management_System.Repositories.StudentRepository
GetAllStudents()
using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Unicom_Tic_Management_System.Datas;
using Unicom_Tic_Management_System.Models;
using Unicom_Tic_Management_System.Models.Enums;
using Unicom_Tic_Management_System.Repositories.Interfaces;
namespace Unicom_Tic_Management_System.Repositories
{
3 references
internal class StudentRepository : IStudentRepository
{
3 references
public void AddStudent(Student student)
{
try
{
if (student == null)
throw new ArgumentNullException(nameof(student));
using (var connection = DatabaseManager.GetConnection())
{
var cmd = connection.CreateCommand();
cmd.CommandText = @"
INSERT INTO Students (Name, Nic, Address, ContactNo, Email, DateOfBirth, Gender, EnrollmentDate, CourseId, UserId, MainGroupId, SubGroupId, Cr
VALUES (@Name, @Nic, @Address, @ContactNo, @Email, @DateOfBirth, @Gender, @EnrollmentDate, @CourseId, @UserId, @MainGroupId, @SubGroupId, @Cre
cmd.Parameters.AddWithValue("@Name", student.Name);
cmd.Parameters.AddWithValue("@Nic", student.Nic);
cmd.Parameters.AddWithValue("@Address", student.Address);
cmd.Parameters.AddWithValue("@ContactNo", student.ContactNo);
cmd.Parameters.AddWithValue("@Email", student.Email);
cmd.Parameters.AddWithValue("@DateOfBirth", student.DateOfBirth.HasValue ? (object)student.DateOfBirth.Value.ToString("yyyy-MM-dd HH:mm:ss") : DBN
cmd.Parameters.AddWithValue("@Gender", (int)student.Gender); // Store enum as integer
cmd.Parameters.AddWithValue("@EnrollmentDate", student.EnrollmentDate.ToString("yyyy-MM-dd HH:mm:ss"));
cmd.Parameters.AddWithValue("@CourseId", student.CourseId);
cmd.Parameters.AddWithValue("@UserId", student.UserId);
cmd.Parameters.AddWithValue("@MainGroupId", student.MainGroupId);
cmd.Parameters.AddWithValue("@SubGroupId", student.SubGroupId);
cmd.Parameters.AddWithValue("@CreatedAt", student.CreatedAt.ToString("yyyy-MM-dd HH:mm:ss"));
cmd.Parameters.AddWithValue("@UpdatedAt", student.UpdatedAt.ToString("yyyy-MM-dd HH:mm:ss"));
85 %
No issues found
Ln: 605
Ch: 10
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
ISubmissionRepository.cs
ITimeSlotRepository.cs
ITimetableRepository.cs
IUserRepository.cs
ActivityLogRepository.cs
AdmissionNumberRepository
AssignmentRepository.cs
AttendanceRepository.cs
BackupLogRepository.cs
CourseRepository.cs
CourseSubjectRepository.cs
DepartmentRepository.cs
EmployeeIdRepository.cs
ExamRepository.cs
GroupMentorRepository.cs
GroupSubjectRepository.cs
LeaveRequestRepository.cs
LecturerRepository.cs
LecturerStudentRepository.cs
LectureSubjectRepository.cs
MainGroupRepository.cs
MarkRepository.cs
MentorRepository.cs
NicDetailsrepository.cs
RoomRepository.cs
StaffRepository.cs
StudentRepository.cs
SubGroupRepository.cs
SubjectRepository.cs
SubmissionRepository.cs
TimeSlotRepository.cs
TimetableRepository.cs
Ready
0 / 0
91
master
UMS
3:06 PM
6/24/2025