Unicom TIC Management System

Project Report

C.Thilakshan UT010709

Contents

CHAPTER 1 Introduction

The Unicom TIC Management System UMS) is a beginner-friendly desktop application designed to streamline and automate basic school management operations. Developed using C# WinForms with an MVC Model-View-Controller) architecture and SQLite for persistent data storage, UMS enables efficient handling of courses, subjects, students, exams, marks, and timetable management, including the allocation of computer labs and lecture halls. The system features a role-based login for Admin, Staff, Students, and Lecturers, ensuring appropriate access control and data security.

CHAPTER 2 Background & Problem Statement

Background: Traditional school management relies heavily on manual processes, leading to inefficiencies, data inconsistencies, and limited accessibility. The increasing complexity of educational institutions necessitates digital solutions for managing courses, students, exams, and resources.

Problem Statement: Schools face challenges in managing multiple operations, such as student enrollment, exam scheduling, mark recording, and classroom allocation. Manual methods are error-prone and time-consuming, often resulting in scheduling conflicts, data loss, and unauthorized access. There is a need for a simple, robust, and user-friendly system to automate and centralize these processes, especially for institutions with limited IT resources.

CHAPTER 3 Requirement Analysis

3.1 Functional Requirements

- User Authentication: Role-based login for Admin, Staff, Students, and Lecturers.
- Course & Subject Management: Add, edit, delete, and view courses and subjects.
- Student Management: Add, edit, delete, and view student records.
- Exam & Marks Management: Schedule exams, record, and view marks.
- Timetable Management: Assign subjects to time slots and allocate computer labs or lecture halls.
- Role-Based Access Control: Restrict features based on user roles.
- Data Persistence: Store all data in a SQLite database.
- Input Validation & Error Handling: Ensure data integrity and user feedback.

3.2 Non-Functional / Operational Requirements

- Usability: Simple, intuitive WinForms interface.
- Performance: Responsive UI using async/await for database operations.
- Reliability: Robust error handling and input validation.
- Security: Basic authentication with role-based access (note: passwords stored in plain text for simplicity, but hashing is recommended for production).
- Portability: Desktop application compatible with Windows OS.
- Maintainability: Modular MVC structure for easy updates and bug fixes

CHAPTER 4 Project Management

- Development Environment: Visual Studio C# WinForms), System.Data.SQLite via NuGet.
- Team Structure: Single developer or small team; roles include developer, tester, and documenter.
- Timeline:
    1. Requirement Gathering & Analysis: 5 Days
    2. Database & Model Design:5 Days
    3. UI Development: 2 Days
    4. Controller & Logic Implementation: 8 Days
    5. Testing & Debugging: 2 Days
    6. Documentation & Demo Preparation: 2 Day
- Version Control: Git (optional for small projects).
- Milestones: Database schema finalized Core modules functional Role-based access implemented System tested and documented

CHAPTER 5 Feasibility Study

Technical Feasibility:

- Tools C#, WinForms, SQLite) are readily available and suitable for the project scope.
- The MVC pattern ensures maintainability and scalability.

Operational Feasibility:

- The system addresses core school management needs.
- Simple UI ensures ease of use for non-technical staff.

Economic Feasibility:

- No licensing costs for development tools.
- Minimal hardware requirements (runs on standard Windows PCs).

CHAPTER 6 System Design

6.1 Architecture

MVC Pattern:

- **Model:** Classes for Course, Subject, Student, Exam, Mark, Timetable, Room, User…Etc
- **View:** AttendanceForm, CourseForm, ExamForm, LecturerForm, LoginForm…… Etc
- **Controller:** AttendanceController, CourseController, LecturerController……Etc

6.2 Database Design

Tables: Users, Courses, Subjects, Students, Exams, Marks, Rooms, Timetables.

Relationships:

One-to-many, Course-Subject, Subject-Exam, Course-Student

Many-to-one: Student-Course

Many-to-many (via Marks): Lecturer-Student, Student-Subject

6.3 UI Design

Login Form: Username/password fields, role-based dashboard navigation.

Module Forms: DataGridView for lists, ComboBox for selections, buttons for CRUD operations.

CHAPTER 7 Implementation of the System

7.1 Setup

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using Unicom_TIC_Management_System.Repositories;
using Unicom_TIC_Management_System.View;

namespace Unicom_TIC_Management_System
{
    0 references
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        0 references
        static void Main()
        {
            DatabaseManager.CreateTables(); // Ensure tables are created before running the application
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LoginForm());
        }
    }
}
```

Figure 1.Program.cs

7.2 Models

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Unicom_TIC_Management_System.Model
{

    public class User
    {

        public int User_ID {  get; set; }

        public string Name { get; set; }

        public string Role {  get; set; }

        public string Username {  get; set; }

        public string Password { get; set; }
        /// <summary>
        ///
        /// </summary>

        public int  Student_ID {  get; set; }

        public string Student_Name { get;  set; }
        /// <summary>
        ///
        /// </summary>

        public int Lecturer_ID { get; set; }

        public string Lecturer_Name { get; set; }
        /// <summary>
        ///
        /// </summary>
        3 references
        public int Course_ID { get; set; }
        0 references
        public string Course_Name { get;set; }
    }
}
```

Figure 2.User Model

7.3 Repositories

```
1 reference
internal class DatabaseManager
{
    1 reference
    public static void CreateTables()
    {
        using (var conn = DBConnection.GetConnection())
        {
            var cmd = conn.CreateCommand();

            cmd.CommandText = @"
                CREATE TABLE IF NOT EXISTS Attendances (
                    Attendance_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                    Attendance_Name TEXT NOT NULL,
                    Attendance_Role TEXT NOT NULL,
                    Attendance_Time TEXT NOT NULL,
                    Student_ID INTEGER,
                    Course_ID INTEGER ,
                    Lecturer_ID INTEGER,
                    FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID),
                    FOREIGN KEY (Course_ID) REFERENCES Courses(Course_ID),
                    FOREIGN KEY (Lecturer_ID) REFERENCES Lecturers(Lecturer_ID)
                );
                CREATE TABLE IF NOT EXISTS Courses (
                    Course_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                    Course_Name TEXT NOT NULL
                );
                CREATE TABLE IF NOT EXISTS Exams (
                    Exam_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                    Exam_Name TEXT NOT NULL,
                    Exam_Date TEXT NOT NULL,
                    Subject_ID INTEGER NOT NULL,
                    Course_ID INTEGER NOT NULL,
                    FOREIGN KEY (Subject_ID) REFERENCES Subjects(Subject_ID),
                    FOREIGN KEY (Course_ID) REFERENCES Courses(Course_ID)
                );
                CREATE TABLE IF NOT EXISTS Lecturers (
                    Lecturer_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                    Lecturer_Name TEXT NOT NULL,
                    Lecturer_Phone_No TEXT NOT NULL,
                    Lecturer_Email TEXT NOT NULL,
                    Subject_ID INTEGER NOT NULL,
                    Course_ID INTEGER NOT NULL,
                    FOREIGN KEY (Subject_ID) REFERENCES Subjects(Subject_ID),
                    FOREIGN KEY (Course_ID) REFERENCES Courses(Course_ID)
                );
                CREATE TABLE IF NOT EXISTS Marks (
                    Mark_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                    Exam_Marks INTEGER NOT NULL,
                    Grade TEXT NOT NULL
```

Figure 3.Repositories

7.4 Views

```
private User currentUser;

1 reference
public MainForm(User user)
{
    InitializeComponent();
    currentUser = user;
    LoadUserDetails();

}

1 reference
private void LoadUserDetails()
{


    if (currentUser.Role == "Student")
    {
        // Hide specific labels for students
        Course_label.Visible = false;
        Attendance_label.Visible = false;
        UserDetails_label.Visible = false;
        lecturer_label.Visible = false;
        Staff_Label.Visible = false;
        lbexam.Visible = false;
        lbmark.Visible = false;
    }
    else if (currentUser.Role == "Lecturer")
    {
        // Show labels for lecturers
        Course_label.Visible = false;
        Attendance_label.Visible = false;
        UserDetails_label.Visible = false;
        lecturer_label.Visible = true;
        lbexam.Visible = false;
        lbmark.Visible = false;
        lbstudent.Visible = false; // Hide student label for lecturers

    }
    else if (currentUser.Role == "Staff")
    {
        // Show labels for staff
        Course_label.Visible = false;
        Attendance_label.Visible = true;
        UserDetails_label.Visible = false;
        lecturer_label.Visible = false; // Hide lecturer label for staff
        Staff_Label.Visible = true;
        lbexam.Visible = false;
        lbmark.Visible = false;
    }

}
9 references
public void LoadForm(object formObj)
{
    if (this.Centerpanel.Controls.Count > 0)
    {
        this.Centerpanel.Controls.RemoveAt(0);
    }

    if (formObj is Form form)
    {
```

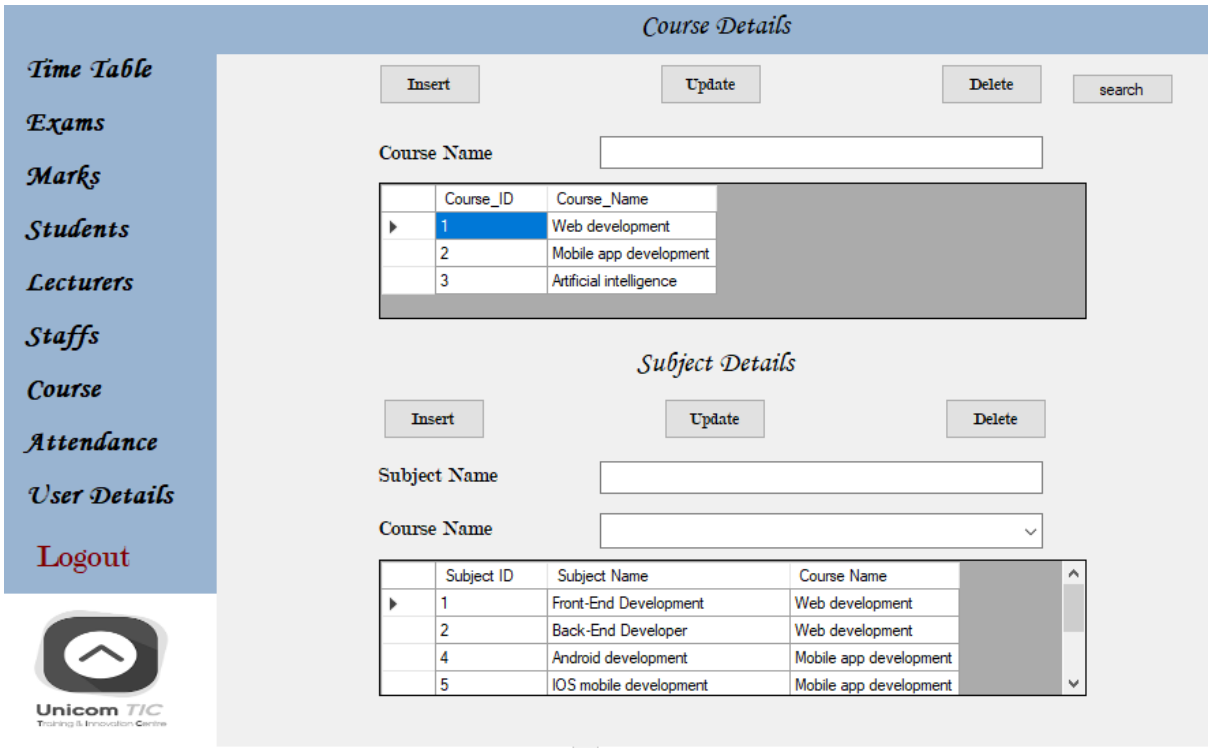
Figure 4.Views

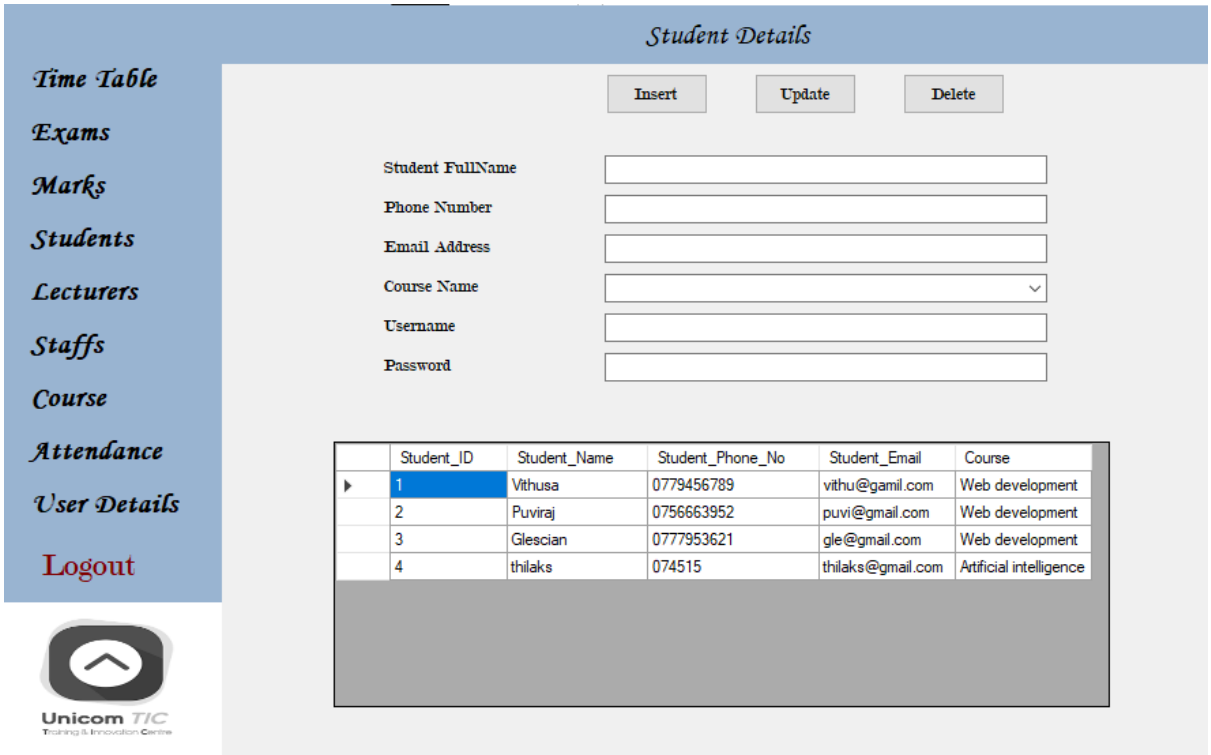
7.5 Controllers

```
using Unicom_TIC_Management_System.Repositories;

namespace Unicom_TIC_Management_System.Controllers
{
    2 references
    internal class LoginController
    {
        1 reference
        public User Login(string username, string password)
        {
            // Hardcoded Admin Check
            if (username == "admin" && password == "admin123")
            {
                return new User
                {
                    Name = "Administrator",
                    Role = "Admin"
                };
            }

            string query = "SELECT * FROM Users WHERE Username = @Username AND Password = @Password";

            using (var conn = DBConnection.GetConnection())
            {

                using (var cmd = new SQLiteCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@Username", username);
                    cmd.Parameters.AddWithValue("@Password", password);

                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            string role = reader["Role"].ToString();

                            var user = new User
                            {
                                Name = reader["Name"].ToString(),
                                Role = role
                            };

                            if (role == "Student")
                            {
                                user.Student_ID = Convert.ToInt32(reader["Student_ID"]);
                                user.Course_ID = Convert.ToInt32(reader["Course_ID"]);
                            }
                            else if (role == "Lecturer")
                            {
                                user.Lecturer_ID = Convert.ToInt32(reader["Lecturer_ID"]);
                                user.Course_ID = Convert.ToInt32(reader["Course_ID"]);
                            }

                            return user;
                        }
                    }
                }
            }
```

Figure 5.Controllers

7.6 Implementation





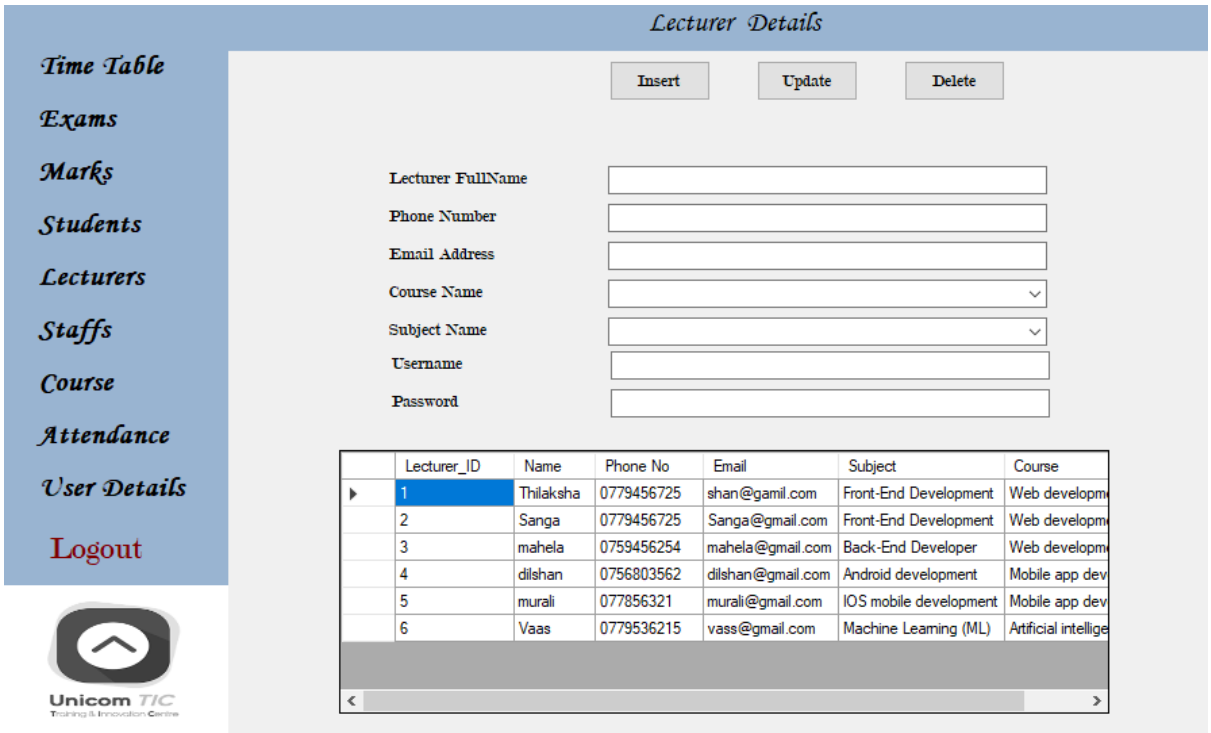


Figure 6.6 Implementation

CHAPTER 8 Testing & Verification

Unit Testing: Tested CRUD operations for each module. Verified input validation and error messages.

Integration Testing: Ensured smooth navigation between forms. Checked data consistency across modules.

User Acceptance Testing: Simulated real-world scenarios for each user role. Confirmed role-based access and feature restrictions.

Error Handling: Tested for invalid inputs, database connection failures, and unauthorized access

CHAPTER 9 Conclusions & Future Work

Conclusions: The Unicom TIC Management System successfully automates core school management functions, providing a simple, reliable, and user-friendly solution. The MVC architecture and SQLite database ensure maintainability and data persistence. Role-based access enhances security and usability for different stakeholders.

9.2 Future Work:

Password Security: Implement password hashing and secure authentication. Conflict Checking: Add logic to prevent double-booking of rooms/timeslots. Reporting: Generate printable reports for marks, timetables, and attendance. Notifications: Add email/SMS alerts for exam schedules and results. Web/Mobile Version: Extend system accessibility beyond desktop. User Activity Logging: Track actions for audit and troubleshooting.