Unicom TIC Management System

Project Report

Contents

[CHAPTER 1 Introduction 3](#_Toc201731313)

[CHAPTER 2 Background & Problem Statement 3](#_Toc201731314)

[CHAPTER 3 Requirement Analysis 4](#_Toc201731315)

[3.1 Functional Requirements 4](#_Toc201731316)

[3.2 Non-Functional / Operational Requirements 4](#_Toc201731317)

[CHAPTER 4 Project Management 5](#_Toc201731318)

[CHAPTER 5 Feasibility Study 5](#_Toc201731319)

[CHAPTER 6 System Design 6](#_Toc201731320)

[6.1 Architecture 6](#_Toc201731321)

[6.2 Database Design 6](#_Toc201731322)

[6.3 UI Design 6](#_Toc201731323)

[CHAPTER 7 Implementation of the System 7](#_Toc201731324)

[7.1 Setup 7](#_Toc201731325)

[7.2 Models 8](#_Toc201731326)

[7.3 Repositories 9](#_Toc201731327)

[7.4 Views 10](#_Toc201731328)

[7.5 Controllers 11](#_Toc201731329)

[CHAPTER 8 Testing & Verification 14](#_Toc201731330)

[CHAPTER 9 Conclusions & Future Work 14](#_Toc201731331)

[9.2 Future Work: 14](#_Toc201731332)

# 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

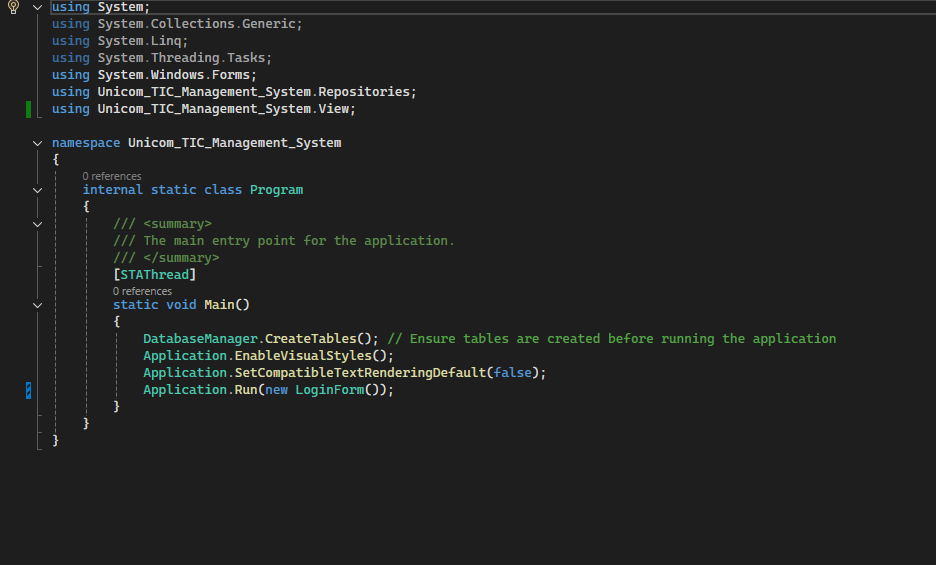


Figure 1.Program.cs

## 7.2 Models

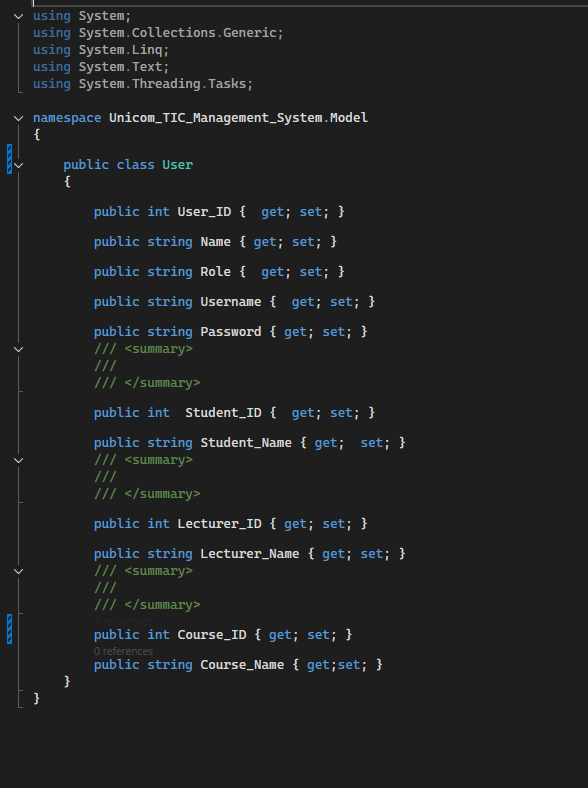


Figure 2.User Model

## 7.3 Repositories



Figure 3.Repositories

## 7.4 Views

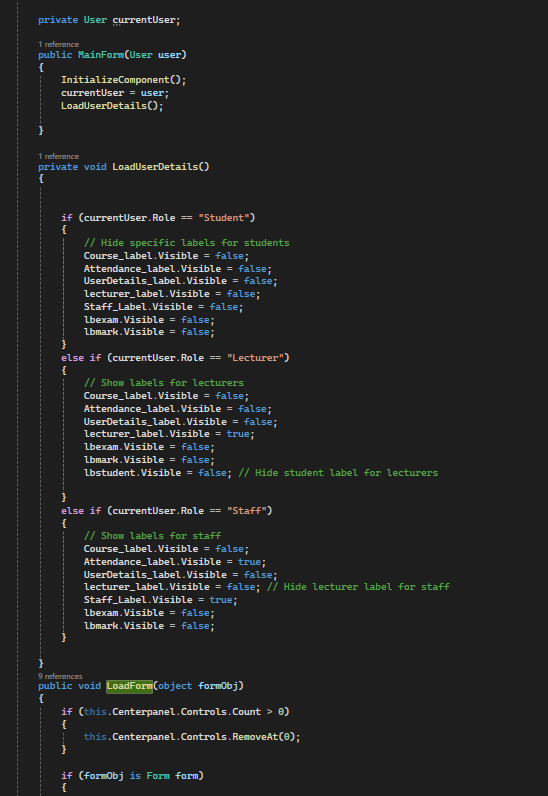
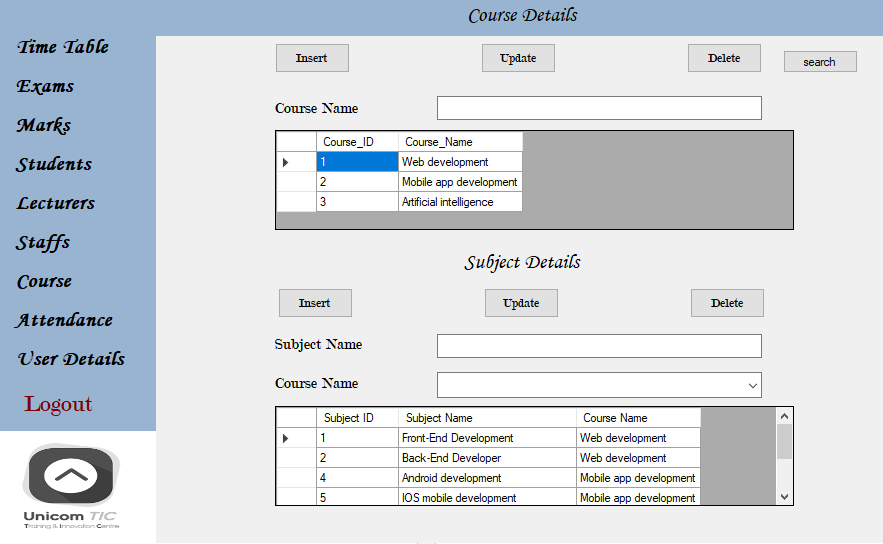


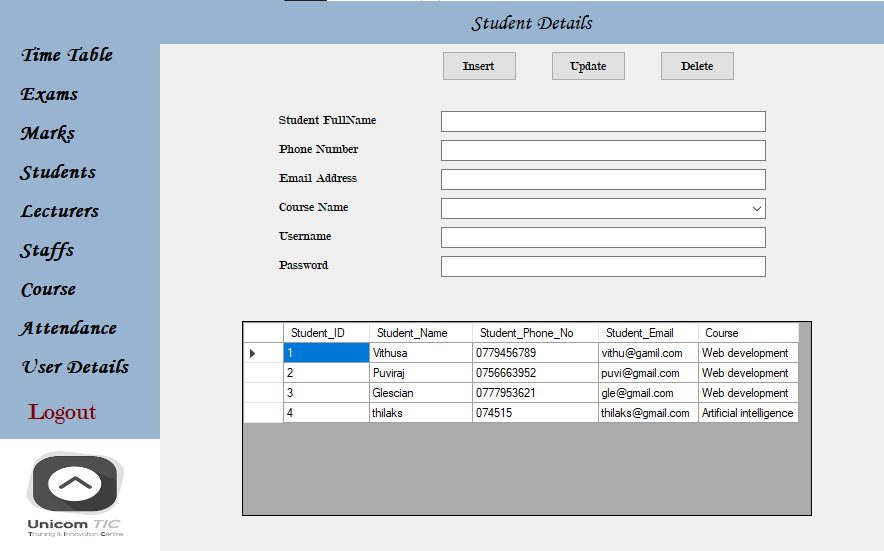
Figure 4.Views

## 7.5 Controllers

Figure 5.Controllers

7.6 Implementation





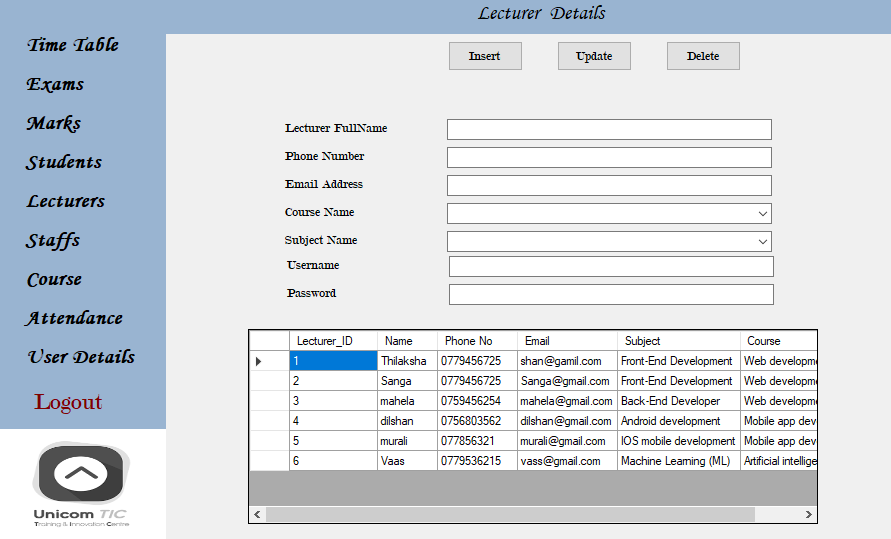


Figure 6.6 Implementation

# 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.