**武汉大学计算机学院软件构造基础**

**结课报告**

**基于.NET开发的教学作业考试一体化平台**

**专业名称： 2023级软件工程**

**小组编号： 8**

**学生学号： 2023302111048**

**学生姓名： 周锦耀**

**指导教师： 叶刚**

**2025年5月30日**

**目录**

[一、系统介绍 3](#_Toc24738)

[1.集中式管理：所有教学相关数据集中存储，避免信息孤岛。 3](#_Toc3812)

[二、设计思路 4](#_Toc31373)

[2.1 角色导向式功能划分 4](#_Toc5172)

[2.2 模块化与分层架构设计 4](#_Toc12682)

[2.3 数据驱动与界面逻辑联动 5](#_Toc23069)

[2.4 交互体验与可扩展性 5](#_Toc24662)

[三、关键技术 5](#_Toc2245)

[3.1 C# WinForms 桌面开发 5](#_Toc27994)

[3.2 SQLite 本地数据库与 ADO.NET 数据访问 6](#_Toc14437)

[3.3 三层架构与面向对象设计 6](#_Toc30099)

[3.4 现代 UI 组件库集成 6](#_Toc2310)

[3.5 数据驱动与界面联动 6](#_Toc15331)

[3.6 权限控制与角色分离 6](#_Toc12569)

[3.7 交互体验优化 7](#_Toc26563)

[四、个人贡献 7](#_Toc466)

[4.1所作任务概览 7](#_Toc13997)

[4.2开发日志 7](#_Toc6008)

[4.3开发系统功能介绍 7](#_Toc7609)

[五、个人成果展示 8](#_Toc23806)

[六、开发体会 23](#_Toc1347)

[6.1开发时遇到的问题和解决办法 23](#_Toc1447)

[6.2本项目[教师端]未来可以拓展的方向 25](#_Toc22162)

[6.3开发结束心得 25](#_Toc29825)

# 一、系统介绍

随着信息化教学的普及，传统的学生管理、作业发布、考试组织等环节面临着信息割裂、数据冗余、用户体验差等诸多问题。这不仅加重了教师与学生的负担，也降低了整体教学效率。本项目《教学作业考试一体化管理平台》的设计初衷，正是基于这些现实痛点，构建一个集中式、模块化、角色分明的教学管理平台。

本系统集教师教学、学生学习、管理员管理于一体，三类用户分别拥有专属的操作权限和界面，能够实现如下核心功能：

1.教师模块：支持课程创建、题库管理、作业与考试布置、在线批阅与评价等功能，帮助教师高效组织教学活动。

2.学生模块：提供作业与考试参与、课程查看、答疑互动、成绩查询等功能，提升学生的学习主动性和参与度。

3.管理员模块：实现对师生用户的集中管理，包括用户信息的增删查改、数据导出、内容审核等，保障系统数据安全和规范运行。

本平台以 C# 为开发语言，采用 Windows Forms (.NET Framework) 构建用户界面，结合 SQLite 实现本地数据存储，通过 ADO.NET 和 LINQ 进行数据交互与处理。界面集成 SunnyUI 和 MaterialSkin 等现代 UI 组件库，整体结构清晰、界面美观、操作直观，能够满足实际教学场景的需求。

系统的主要亮点包括：

1.集中式管理：所有教学相关数据集中存储，避免信息孤岛。

2.角色分明：不同用户拥有独立的操作权限和界面，保障数据安全。

3.模块化设计：各功能模块独立开发，便于维护和扩展。

4.现代化界面：采用美观的 UI 组件，提升用户体验。

5.数据驱动：所有操作均以数据库为核心，保证数据一致性和实时性。

# 二、设计思路

为了实现一体化教学管理平台的目标，本系统在设计之初便坚持“角色分离、功能聚合、数据统一、交互简洁”的原则，力求以最清晰的业务流程、最直观的界面体验，为用户（教师、学生、管理员）提供高效实用的操作支持。整体设计采用模块化架构和面向对象编程思想，兼顾功能性与可扩展性。

## 2.1 角色导向式功能划分

系统按照实际教学参与者划分为三个主要角色：教师、学生、管理员，每个角色拥有独立的登录认证与界面功能，仅可访问其权限范围内的模块，实现权限控制与职责分明。

·教师端：聚焦于教学活动的主导方，提供课程创建、题目管理、作业与考试发布、在线批阅与反馈等功能，提升教学组织效率。

·学生端：面向学习者，提供作业考试参与、课程查看、成绩查询、在线答疑等功能，引导学生自主完成学习任务。

·管理员端：作为系统维护核心，承担用户信息管理职责，实现对教师和学生数据的增删查改，以及数据一致性维护。

通过角色隔离实现系统逻辑上的“横向解耦”，避免功能混杂，提高用户操作的专注性和效率。

## 2.2 模块化与分层架构设计

系统采用典型的三层架构模型（三层即：表示层 UI、业务逻辑层 BLL、数据访问层 DAL），每层职责如下：

·表示层（UI）：采用 WinForms 构建，负责用户界面展示和交互响应，调用业务逻辑层的方法，实现信息传输与展示。所有界面均采用卡片式、分区式布局，支持自适应调整，提升可用性。

·业务逻辑层（BLL）：处理系统核心逻辑，如用户权限验证、成绩统计、题目组装、作业与考试状态判定等，确保平台业务的正确性和稳定性。

·数据访问层（DAL）：负责所有与 SQLite 数据库的交互操作，通过 ADO.NET 封装连接、查询、更新等操作接口，支持数据持久化和一致性校验。

此外，系统采用面向对象的方式设计了诸如 Course、Student、Teacher、Exam、Homework、CommitMent 等实体类，清晰表示各模块的数据结构与行为，有助于后期维护与扩展。

## 2.3 数据驱动与界面逻辑联动

系统的各类数据（如课程、题目、学生信息、作业记录等）统一存储于本地 SQLite 数据库中，通过 ADO.NET 实现增删查改功能，并使用 LINQ 技术在内存中对数据进行筛选、匹配、汇总等操作。所有数据操作均以数据库为核心，保证数据的实时性和一致性。

在界面设计中，表格控件（如 DataGridView）与后端数据紧密绑定，结合分页、排序与搜索功能，提高信息的查阅与管理效率。作业、考试、答疑等模块采用卡片式信息展示，支持状态高亮、剩余时间提示、动态刷新等交互设计，提升了系统的易用性和响应速度。

## 2.4 交互体验与可扩展性

系统界面集成 SunnyUI、MaterialSkin 等现代 UI 组件库，采用扁平化、色彩分明的设计风格，支持按钮悬停高亮、模态输入框、动态刷新等细节优化。各功能模块均以独立 UserControl 实现，便于后续功能扩展和维护。

# 三、关键技术

本系统在开发过程中，综合运用了多项现代软件开发技术，确保平台的高效性、可维护性与良好用户体验。主要关键技术如下。

## 3.1 C# WinForms 桌面开发

系统采用 C# 语言和 Windows Forms 技术作为主要开发框架。WinForms 提供了丰富的控件库和事件驱动机制，便于实现复杂的用户交互界面。通过自定义控件、事件处理和界面布局优化，实现了高度模块化和美观的操作界面。

## 3.2 SQLite 本地数据库与 ADO.NET 数据访问

平台采用 SQLite 作为本地数据存储方案，具有轻量级、易部署、跨平台等优点。所有数据（包括用户、课程、作业、考试、答疑等）均以结构化方式存储于 SQLite 数据库中。数据访问层（DAL）通过 ADO.NET 实现数据库的连接、查询、插入、更新和删除等操作，保证数据的安全性和一致性。

## 3.3 三层架构与面向对象设计

系统严格遵循三层架构（UI、BLL、DAL），各层职责分明，便于后期维护和扩展。实体类（如 Course、Student、Teacher、Homework、Exam 等）采用面向对象方式设计，属性与方法清晰，支持数据的封装与复用。

## 3.4 现代 UI 组件库集成

为提升用户体验，系统集成了 SunnyUI 和 MaterialSkin 等现代化 UI 组件库。SunnyUI 提供了丰富的美化控件和主题支持，MaterialSkin 则带来更现代的扁平化风格。通过自定义控件样式、按钮悬停效果、卡片式布局等方式，显著提升了界面美观度和交互友好性。

## 3.5 数据驱动与界面联动

系统大量采用数据绑定技术（如 DataGridView 绑定数据源），实现前后端数据的实时同步。通过 LINQ 技术对数据进行筛选、排序、分组等操作，提升了数据处理效率和代码可读性。

## 3.6 权限控制与角色分离

系统实现了基于角色的权限控制机制。不同用户（教师、学生、管理员）登录后，仅能访问和操作其权限范围内的功能模块，保证数据安全和操作规范。

## 3.7 交互体验优化

在作业、考试、答疑等模块，采用了卡片式信息展示、动态刷新、模态对话框、输入验证等交互设计，提升了系统的易用性和响应速度。例如，作业和考试列表支持状态高亮、剩余时间提示、双击进入详情等功能。

# 四、个人贡献

## 4.1所作任务概览

担任组长，负责统筹开发时间、开发进度检查等。

同时独立负责教师管理系统的前后端开发，采用winform框架实现响应式布局与用户交互功能，并整合后端API完成数据渲染。主导数据库架构设计，规范化数据模型，确保信息完整性与查询效率。通过给每个按钮添加update功能实现异步请求，优化多界面间的数据同步流程。基于最小化原则设计核心信息表结构，运用范式化理论减少冗余，提升系统整体性能。

## 4.2开发日志

我们组首先确定开发周期大致分为三周：第一周完成初步前端构建和数据库字段设计，方便下一个周期的任务展开，同时经过充分的思考，可以确保后续数据交互的高效、简洁；第二周进行较为复杂的后端逻辑处理，实现关键的同步异步交互、不同用户之间的信息传递；第三周进行界面美化，采用SunnyUI等组件库进一步优化人机关系，美化界面。

## 4.3开发系统功能介绍

教师端实现如下功能：

1. 创建课程：在首页点按创建课程后，输入相关信息并选择学生名单即可创建目标课程。
2. 题库管理：
   1. 实现编辑题目、删除题目：在题目概览-操作一栏，点击对应按钮，回跳转到编辑页面/直接把该题移动到回收站并刷新题目列表。
   2. 录入题目：在录入题目页面输入关键信息，提交后可在题目概览中选择。
   3. 回收站：保存已删除的题目，可以进一步实现完全删除或者恢复。
3. 作业编辑、考试编辑
   1. 实现作业概览/考试概览：查看**本班级**已有的作业/考试。支持已有作业的编辑，例如重新选题、重新输入作业基本信息等。
   2. 作业布置：在**本班级**选题并输入基本信息后即可布置作业。
   3. 回收站：支持彻底删除/恢复目标作业/考试。
4. 数据活动管理
   1. 支持对学生已经写好的作业、考试进行批阅，如果该学生本次作业没有提交任何一题的任何一份答案，那么显示无法批阅。

# 五、个人成果展示



图1教师页面首页



图2-1创建课程



图2-2创建课程

图2-3创建课程



图1-2完成后的首页







图3-1题库界面[支持分类查看]





图3-2编辑题目



图3-3删除题目

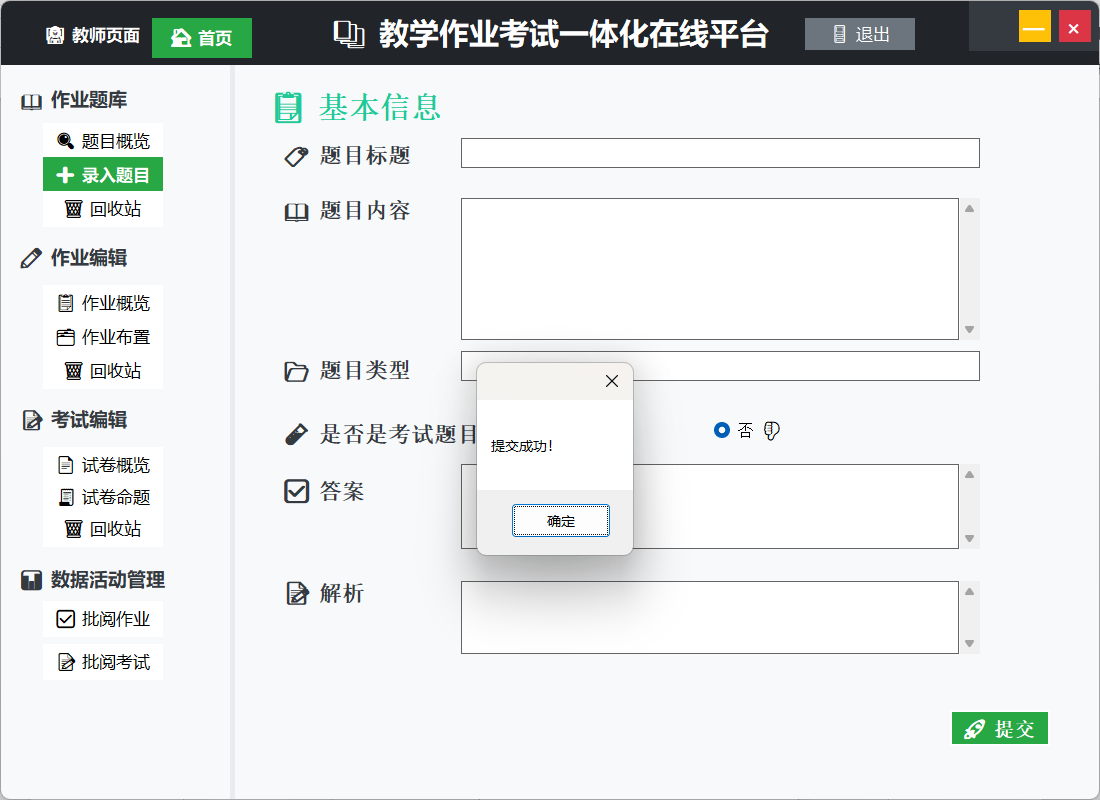


图3-4添加题目



图3-5回收站







图4-1添加作业

图4-2编辑作业

图4-3作业回收站



图5-1批阅作业

# 六、开发体会

## 6.1开发时遇到的问题和解决办法

**问题**：点按父组件某一按钮后，只能实现子组件的切换，无法刷新子组件的内容。比如添加新问题之后，点按切换页面，发现并没有出现新问题。

**初级解决方法**：经过对代码分析知：切换页面只是通过hide、show实现，并没有对创建的页面进行数据刷新，所以需要实现function：update()；也就是每次点按让某个页面从hide到show就刷新一次数据，自然可以实现动态展示。

**高级解决方法：**考虑数据库中的数据非常庞大，那么以上方案直接刷新获取所有数据会变得非常慢，所以采用**增量更新机制、观察者模式等。**具体而言：就是为每个操作按钮绑定自定义DataSyncController，同时仅传输变更字段，并且建立中央事件总线，如回收站删除操作自动触发题库界面的局部重绘而不是所有的都改变。

**问题：**登录页面会传入uid作为用户身份标识，有没有简单的办法实现传参？因为多个组件和他们的子组件都会用到，写入各自的构造函数又特别繁琐。

**解决方法：**让我想到了直接通过新建一个包含static数据的params.cs文件，每次需要参数时直接从里面拿就行了。

**问题：**需要维护全局动态变化的参数，如创建题目时需要给题目一个ProID进行标识并维护ProID没有冲突；创建班级时需要给班级ClassID作为标识符。

**解决方法：**数据库中维护一个全局的ProID，ClassID。每次创建对应的即实现get、increase即可，能保证不冲突。

**问题**[这个准确来说应该是改进的想法]：由于每次写连接数据库对应数据表的语句有点麻烦，怎么进行改进？

**解决方法**：和上个问题差不多，我采用的方法是写静态类的静态函数进行表的动态绑定。

**参考代码**：

public static void BindToTable(DataGridView dgv, string tableName,string judge,int typer)

{

try

{

using var conn = new SQLiteConnection(ConnectionString);

SQLiteDataAdapter adapter;

if(typer == 2)

adapter = new SQLiteDataAdapter($"SELECT \* FROM {tableName} WHERE isValid="+judge, conn);

else if(typer == 1) { adapter = new SQLiteDataAdapter($"SELECT \* FROM {tableName} WHERE isValid=" + judge +" AND IsTest ='考试题'", conn); }

else { adapter = new SQLiteDataAdapter($"SELECT \* FROM {tableName} WHERE isValid=" + judge + " AND IsTest ='作业题'", conn); }

var dataTable = new System.Data.DataTable();

adapter.Fill(dataTable);

dgv.DataSource = dataTable;

}

catch

{

MessageBox.Show("绑定失败，请检查表是否存在");

}

}

## 6.2本项目[教师端]未来可以拓展的方向

以下是有想法去做，但是限于时间因素等尚未完成的、有创新性、挑战性的部分：

1. 添加**一课程多教师**的功能，实现邀请制添加教师，可以应对为某课程添加多位教师、助教的个性化需求。
2. 在1的基础上添加**权限控制**概念，把权限分立如查看作业、编辑作业、添加作业、删除作业、批阅作业等，对每个权限加入位来控制访问。创建本课程的教师拥有所有权限，并可对邀请制添加的教师进行权限控制。
3. 在作业页面可以添加共享作业的功能，将某份作业对应到多个班级。
4. 在3的基础上添加组的概念，统筹多门课程为课程组，增设组长，负责为课程组添加固定的要求，结合2进行课程内的权限控制。
5. 作业数据统计可以添加图表实现作业答题情况可视化，并生成对应报告。
6. 引入ai对学生答案进行自动评分，同时也可以编写脚本对编程题进行评测。

## 6.3开发结束心得

在独立承担教师管理系统的全栈开发过程中，我深刻体会到基础架构的严谨性决定系统生命力。采用WinForm实现响应式布局时，曾面临数据渲染卡顿的挑战。通过为每个按钮注入Update异步回调功能，不仅解决了多界面数据同步问题，更让我理解到：异步就是精准控制数据流的艺术。在数据库设计阶段，坚持最小化原则构建核心表结构，用三范式理论消除冗余字段。

作为组长统筹三周开发周期，最大的收获是学会在进度与质量间寻找平衡点。第一周的前端原型与数据库设计，我们反复推翻多次方案，只为确保后期交互逻辑的简洁性。第二周实现异步通信时，组员对信号处理逻辑都有不解，最终通过拆解教师批阅作业的完整流程，用事件委托机制统一了技术路线。这个过程让我明白：技术分歧的化解之道，在于回归用户场景。第三周的界面美化看似简单，但各种组件的运用教会我**用户体验的提升，往往藏在细节的克制中**。

这个用C#编织的教师管理系统，最终交付的不只是WinForm窗口里的功能模块。它让我亲身验证：优雅的代码源于对业务逻辑的敬畏，高效的协作始于对开发节奏的掌控，而真正的技术创新，永远生长在解决实际问题的土壤里。当看到测试同学流畅地创建课程、增删查改题目和收发批改作业时，那些调试异步信号的日子都有了意义——最好的结课证书，是用户无言的顺畅体验。