**武汉大学计算机学院**

**本科生课程报告**

## 

**基于.NET开发的教学作业考试一体化平台**

专 业 名 称 ：软件工程

课 程 名 称 ：软件构造基础

指 导 教 师 ：叶刚

学 生 学 号 ：2023302111182

学 生 姓 名 ：莫韬

二○二五年五月

**郑 重 声 明**

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名： 莫韬 日期： 2025.5.31

摘 要

随着教育信息化的发展，传统的教学管理方式已难以满足高效教学的需求。本文介绍了一款基于 C# 和 Windows Forms 技术开发的《教学作业考试一体化管理平台》。该系统面向教师、学生与管理员三类用户，分别提供课程管理、作业考试、信息维护等功能。项目采用模块化与分层架构设计，结合 SQLite 本地数据库和 ADO.NET 实现数据存储与访问，同时通过 LINQ 简化数据处理逻辑。界面采用 SunnyUI 与 MaterialSkin 控件进行美化，具备良好的交互体验与系统扩展性。文中详细阐述了系统的设计思路、关键技术实现、开发过程中的挑战与解决方案，以及未来优化方向。本平台具有结构清晰、功能完整、部署便捷等优点，可广泛应用于高校教学管理场景。

**关键词：**教学管理平台；C# WinForms；SQLite；三层架构；LINQ；SunnyUI；MaterialSkin；分层架构；模块化设计

目录

**[一、系统介绍 5](#_Toc1928961914)**

**[二、设计思路 5](#_Toc1649753486)**

[2.1 角色导向式功能划分 5](#_Toc1245472785)

[2.2 模块化与分层架构设计 6](#_Toc1137990186)

[2.3 数据驱动与界面逻辑联动 6](#_Toc711695920)

**[三、关键技术与个人贡献 6](#_Toc2136897297)**

[3.1 Windows Forms（WinForms）图形用户界面开发 7](#_Toc316358251)

[3.2 SQLite 数据库与 ADO.NET 数据访问 7](#_Toc2011098232)

[3.3 LINQ 技术进行内存数据操作 7](#_Toc1282865091)

[3.4 UI 美化：SunnyUI 与 MaterialSkin 控件库 8](#_Toc377768557)

[3.5 异常处理与用户输入校验（正则表达式） 8](#_Toc1194476967)

[3.6 多线程编程 9](#_Toc897252213)

[3.7 文件操作与数据导出 9](#_Toc487774657)

[3.8 面向对象设计与模块封装 9](#_Toc1083579600)

[3.9 哈希加密：保护用户密码安全 10](#_Toc1061010640)

**[四、个人成果展示 11](#_Toc1849105439)**

[4.1 管理员主界面设计 11](#_Toc1679257536)

[4.2 教师与学生信息管理模块 12](#_Toc1051318678)

[4.3 信息审核与内容管理模块 13](#_Toc17573630)

[4.4 学生界面：课程信息与个人信息展示模块 14](#_Toc1154739771)

**[五、遇到的困难与解决方法 15](#_Toc901613258)**

[5.1 UI 美化库的选择与冲突问题 16](#_Toc769413974)

[5.2 全局皮肤设置对其他界面的干扰 16](#_Toc1541622431)

[5.3 窗体最大化后控件与字体无法自适应 16](#_Toc657996762)

**[六、后续发展与优化展望 17](#_Toc1558280531)**

[6.1 功能模块拓展 17](#_Toc1457809352)

[6.2 用户体验优化 17](#_Toc760850441)

[6.3 数据安全与权限控制强化 18](#_Toc1495727649)

[6.4 系统智能化升级 18](#_Toc251024961)

[6.5小结 18](#_Toc1318636819)

**[七、开发体会 19](#_Toc297779893)**

## 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。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：

• 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；

• 学生模块：提供作业与考试参与、课程查看、答疑互动等功能，提升学习参与度；

• 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、数据导出等。

本平台以C#为开发语言，使用 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，支持使用 SunnyUI 和 MaterialSkin 两种现代 UI 组件库实现良好的用户体验。系统结构清晰、界面美观、操作直观，适合教学场景实际落地。

## 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，旨在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。

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。

此外，使用面向对象的方式设计了诸如 Course、Student、Teacher、Exam、Homework 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。

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。

## 三、关键技术与个人贡献

本系统作为一个涵盖“教学、作业、考试”全过程的管理平台，要求在用户交互、数据存储、权限控制、界面美观与操作流畅性等方面实现全面支持。为了满足上述需求，系统综合应用了多项核心编程技术与辅助工具。

在此过程中，我主要负责系统中的管理员界面开发，参与了部分学生界面（课程信息展示与个人信息展示）的设计实现，主导了学生、教师、课程三大核心数据结构的设计与数据库建模工作，并承担了系统界面美化的任务。

以下是我在开发中使用的关键技术。

#### 3.1 Windows Forms（WinForms）图形用户界面开发

系统采用 Windows Forms 构建图形界面，是 .NET Framework 中广泛使用的 UI 开发技术。我在开发过程中主要负责管理员界面的整体设计与实现，同时协助开发了部分学生端页面。通过 WinForms 提供的窗体控件（如 Label、TextBox、ComboBox、DataGridView、Button 等），构建了完整的交互操作界面，实现了信息展示、数据管理、交互反馈等功能。界面支持多角色切换、数据表单验证、弹窗提示等，构建出功能丰富、操作直观的桌面级应用。

#### 3.2 SQLite 数据库与 ADO.NET 数据访问

系统选用轻量级嵌入式数据库 SQLite 进行数据存储，适合本地部署的教学系统。我参与了数据库的结构设计，特别是学生（Student）、教师（Teacher）、课程（Course）三大数据表的建模与关系绑定。

通过 ADO.NET 实现与 SQLite 的连接管理、SQL 命令执行、数据读取与更新等功能，确保数据的持久化与一致性。为便于界面数据绑定和更新，我利用 SQLiteDataAdapter 和 DataSet 实现 DataGridView 与数据库的动态绑定。

#### 3.3 LINQ 技术进行内存数据操作

我在数据处理过程中引入了 LINQ（Language Integrated Query） 技术，主要用于对数据库读取后的集合数据进行高效、简洁的筛选与排序操作。相比传统的循环判断方式，LINQ 具有更强的表达力和更高的代码可读性。

以教师信息管理模块为例，从数据库获取完整的教师列表后，我使用 LINQ 实现了如下操作：

1. var filteredTeachers = teachers
2. .Where(t => !**string**.IsNullOrEmpty(t.Phone)) // 过滤手机号为空的数据
3. .OrderBy(t => t.Id)                         // 按工号升序排列
4. .ToList();

这一方式避免了手动遍历和条件判断，提高了数据处理效率，也使界面展示更加符合用户预期。经过 LINQ 处理后的数据被绑定至 DataGridView，实时刷新界面内容，提升了用户交互体验。

通过 LINQ 技术的引入，系统在数据展示层实现了更灵活的处理逻辑，同时提升了代码模块化程度和运行效率。

#### 3.4 UI 美化：SunnyUI 与 MaterialSkin 控件库

在我负责的界面中，我主导引入了 SunnyUI 和 MaterialSkin 两种现代化 UI 控件库，对管理员和学生界面进行整体美化设计。

SunnyUI：开源现代风格 UI 库，支持卡片式布局、分页控件、图标按钮等，极大提升了界面的一致性和美观度，适合管理系统开发。

MaterialSkin：基于 Google Material Design 设计语言的 UI 控件，提供标准化的按钮、对话框、导航栏等，使界面更具现代感与交互反馈。

结合原生控件与第三方控件，自定义多种控件样式（如圆角按钮、阴影边框、信息面板），提高用户操作的舒适性和界面层次感。

#### 3.5 异常处理与用户输入校验（正则表达式）

充分考虑到实际运行中可能遇到的异常，如数据库连接中断、非法输入、空值操作等，我使用 try-catch 块进行异常捕获与提示反馈，防止程序崩溃并引导用户正确操作。

同时，为增强系统健壮性与安全性，我在管理员界面中通过正则表达式对用户输入数据（如邮箱、手机号、姓名等）进行校验。例如：

1. Regex nameRegex = **new** Regex(@"^[\u4e00-\u9fa5A-Za-z]{2,20}$");
2. Regex phoneRegex = **new** Regex(@"^\d{11}$");
3. Regex emailRegex = **new** Regex(@"^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$");

#### 3.6 多线程编程

我使用了 .NET 中的 多线程编程机制，通过 Task.Run() 启动后台线程，执行数据访问与处理逻辑，确保主线程始终保持响应，避免界面在加载数据过程中出现卡顿。

在教师管理模块中，点击按钮后会异步调用数据库读取方法 TeacherDAL.GetAll()，并使用 this.Invoke() 将最终结果回传主线程，更新 DataGridView：

1. Task.Run(() =>
2. {
3. var teachersResult = TeacherDAL.GetAll();
4. // ... 数据处理 ...
5. **this**.Invoke((MethodInvoker)**delegate**
6. {
7. dataGridViewMain.DataSource = filteredTeachers;
8. });
9. });

这种设计大大提升了用户体验，尤其在数据量较大或处理逻辑复杂时，界面始终保持流畅。

#### 3.7 文件操作与数据导出

我负责的管理员界面支持将老师、学生的信息导出为本地文件，便于线下备份与分析。通过文件读写操作，将结构化信息写入.csv 文件格式，未来也可扩展为导出为PDF 等其他格式。

#### 3.8 面向对象设计与模块封装

整个系统中，我使用面向对象编程（OOP）思想进行开发，参与设计了多种核心实体类（如 Student, Teacher, Course等），将数据与方法封装在统一对象中，使得各模块职责清晰、耦合度低，便于维护与扩展。

此外，UI事件逻辑、数据封装、验证机制都采用了模块化封装，增强代码复用性与工程规范性。

#### 3.9 哈希加密：保护用户密码安全

在系统中，我通过自定义工具类 PasswordHelper 实现了对用户密码的哈希加密，有效提升了数据库中用户信息的安全性。

具体做法是，在用户注册或更新密码时，调用 HashPassword 方法对原始密码进行哈希处理。该方法基于 SHA256 算法，将明文密码编码为 UTF-8 字节数组后计算其摘要值，并将结果转为十六进制字符串存储到数据库中。代码如下：

1. **public** **static** **class** PasswordHelper
2. {
3. **public** **static** **string** HashPassword(**string** password)
4. {
5. **using** (SHA256 sha = SHA256.Create())
6. {
7. **byte**[] bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(password));
8. StringBuilder sb = **new** StringBuilder();
9. **foreach** (**byte** b **in** bytes)
10. sb.Append(b.ToString("x2"));
11. **return** sb.ToString();
12. }
13. }
14. }

在用户登录时，输入的密码同样会经过相同方式加密并与数据库中存储的哈希值进行比对，确保验证逻辑的一致性与安全性。

该机制避免了明文密码的存储风险，即使数据库被非法访问，攻击者也无法直接获取有效密码。后续系统还可进一步引入“加盐（Salt）机制”来防止彩虹表攻击，进一步增强系统抗破解能力。

## 四、个人成果展示

#### 4.1 管理员主界面设计

我设计并实现了一个功能完整、布局清晰的管理员主界面，涵盖了对教师、学生、课程的增删查改、数据导出、快速检索等核心功能模块，并对界面进行了整体美化处理，提升了系统的可用性与专业性。

• 界面采用 卡片式布局 + 分页容器，将各类管理功能模块清晰划分。

• 结合使用 SunnyUI 控件实现图标按钮、数据卡片、分页组件等。

• 使用 MaterialSkin 控件统一界面风格，使整体风格更具现代感。

• 引入 DataGridView + ADO.NET + 多线程 + LINQ 技术，确保操作流畅、响应迅速。

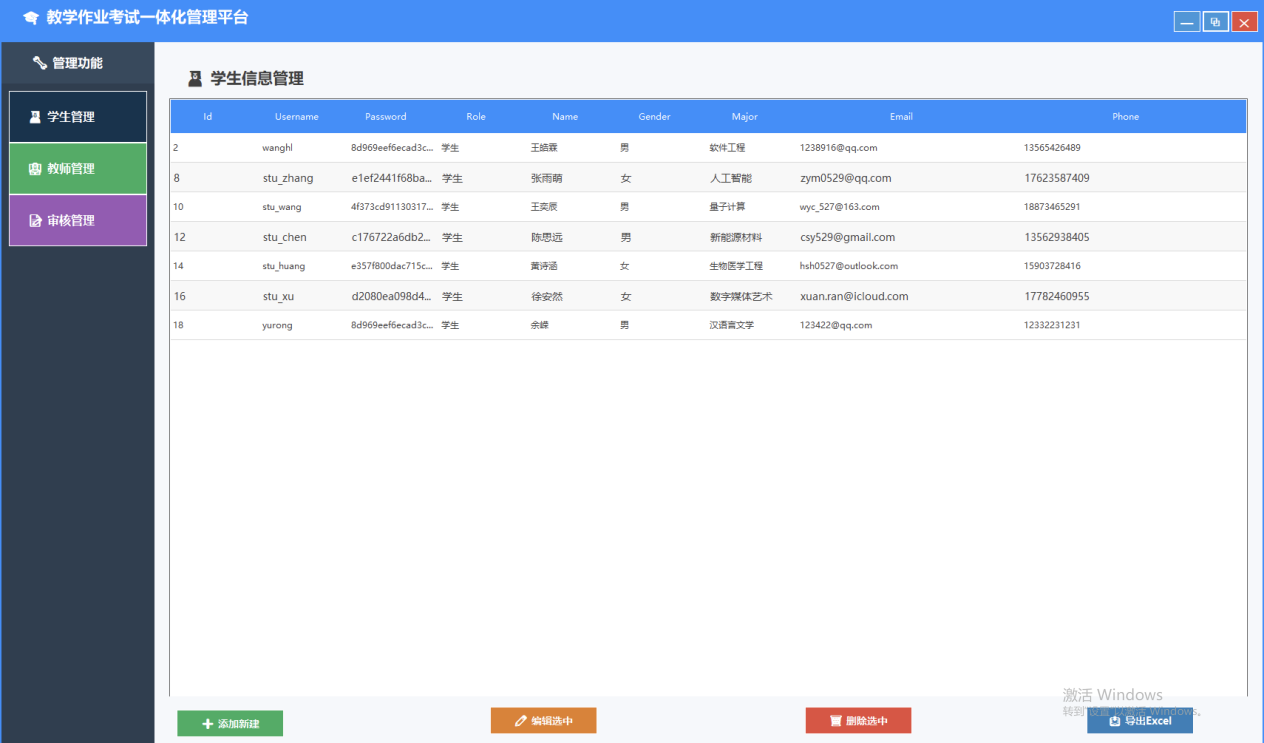


图4.1

该图展示了我设计并实现的管理员主界面中的“学生信息管理”模块。界面整体采用卡片式布局，信息排列清晰、层次分明，使用了 SunnyUI 控件增强视觉体验。图中展示的学生信息按照学号（ID）升序排序，排序逻辑通过 LINQ（Language Integrated Query）技术 实现，实现了对内存中学生数据集合的快速排序与筛选。

通过 LINQ 的 .OrderBy(s => s.Id) 表达式，系统能够自动按照学号从小到大排序，用户无需手动操作即可获得井然有序的数据展示效果，提升了数据管理的效率与用户体验。

#### 4.2 教师与学生信息管理模块

我分别设计了教师管理和学生管理模块，支持完整的信息维护功能：

• 实现教师/学生信息的添加、删除、编辑、快速搜索与排序；

• 使用正则表达式进行输入校验，确保手机号、姓名、邮箱等格式合规；

• 引入 LINQ 实现内存中数据的高效筛选与动态更新；

• 支持将表格数据一键导出为 .csv 文件，便于线下保存与分析。

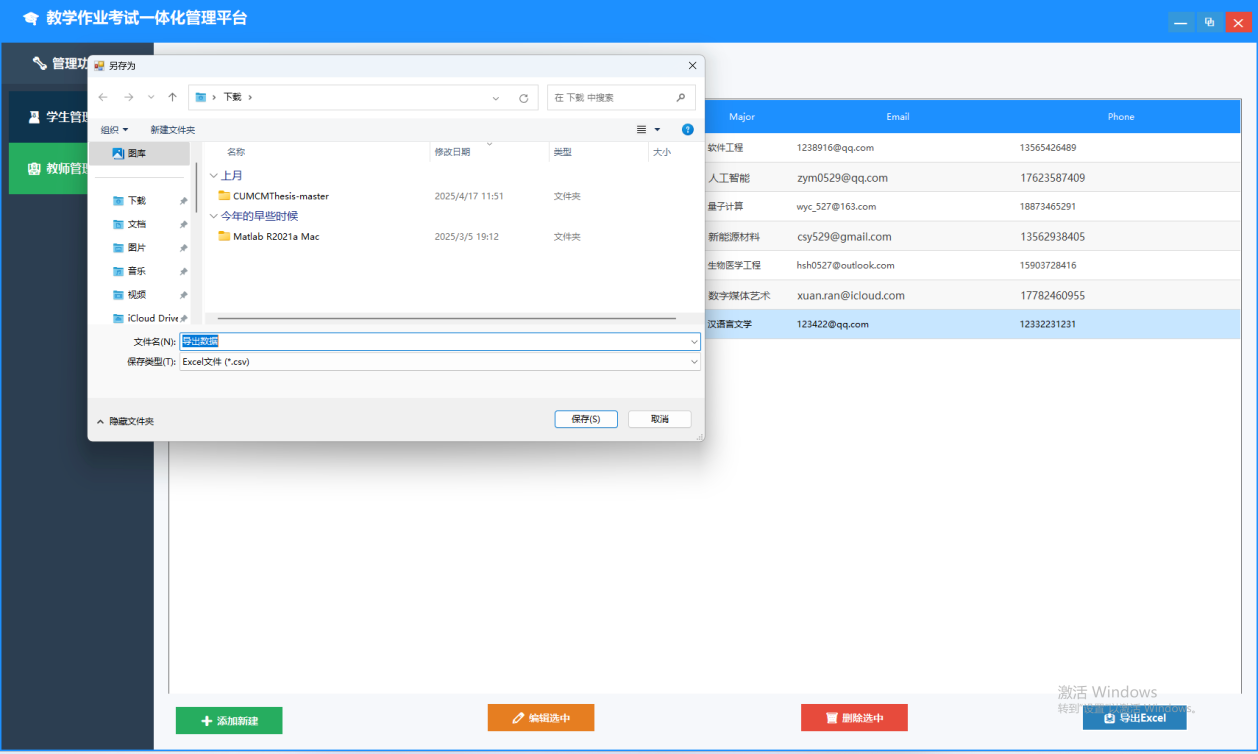


图4.2（导出为Excel文件）

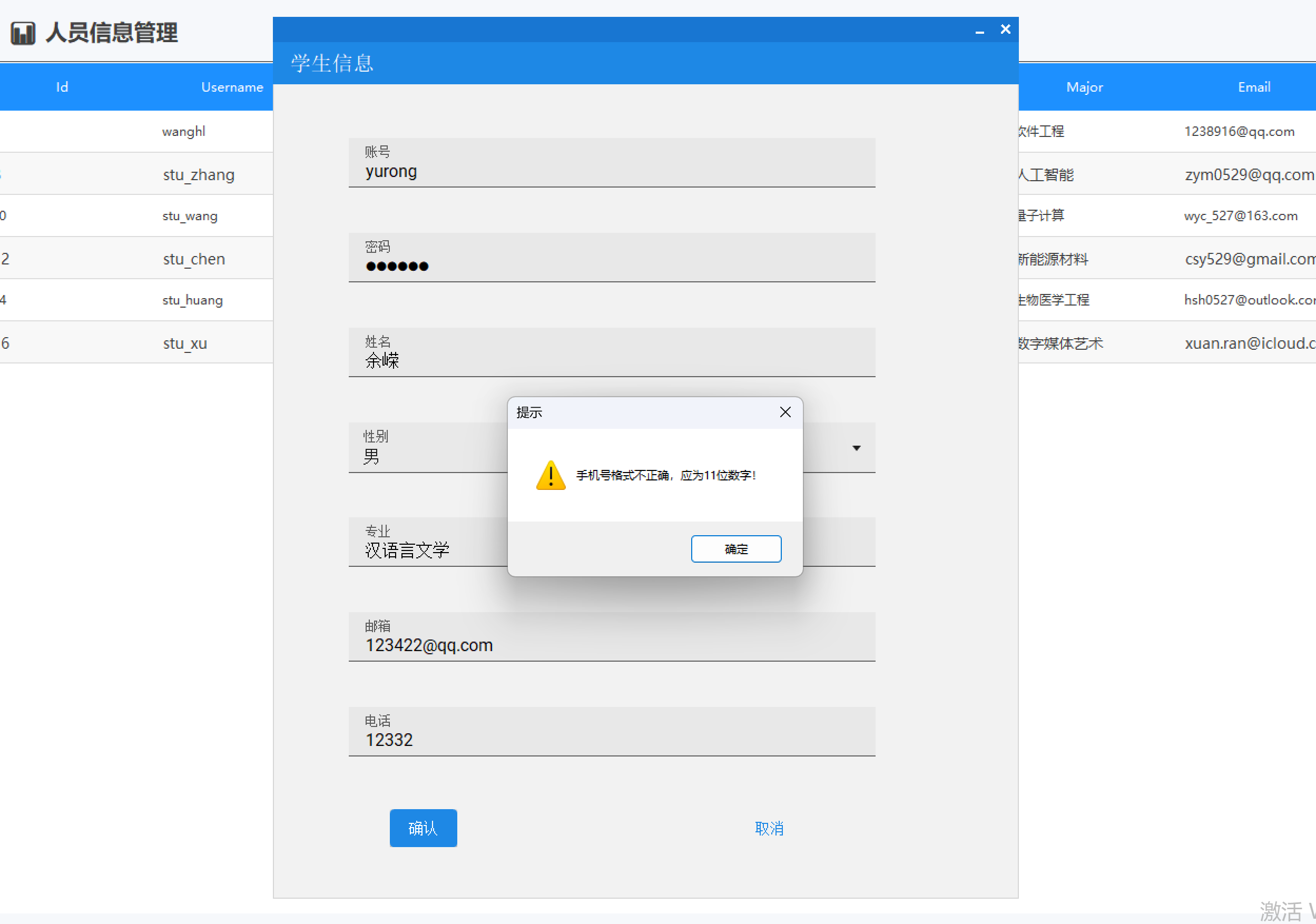


图4.3

在该图所示的操作场景中，我通过管理员界面对学生信息进行添加。当用户输入的电话号码格式不符合规范（如不是标准的11位手机号）时，系统会立即弹出提示窗口，提醒用户检查并重新填写输入内容。

该功能基于 正则表达式（Regex）校验机制 实现，能够在用户提交数据前，对输入的姓名、手机号、邮箱等字段进行格式验证，从而有效防止无效数据写入数据库，保障系统数据的准确性与一致性。该设计不仅提升了系统的健壮性，也优化了用户输入体验，避免了因错误数据导致的后续操作异常。

#### 4.3 信息审核与内容管理模块

我设计并实现了一个简洁直观的信息审核界面，帮助管理员对用户发布的帖子内容进行有效监管，防止不当信息的传播，保障平台内容的健康与合规。

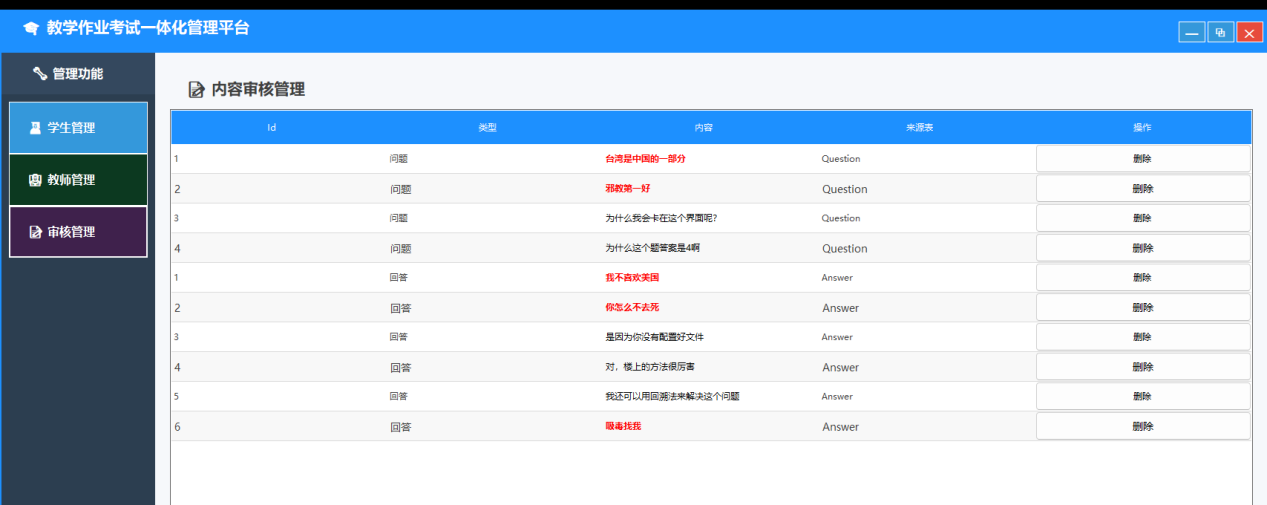


图4.4

该模块主要功能包括：

帖子列表展示：以表格形式展示所有用户发布的帖子内容，支持按用户、时间、关键词等维度筛选。

敏感词自动识别：系统内置敏感词库，通过字符串匹配技术对帖子内容进行实时检测，若发现敏感词（如政治敏感、低俗用语等），会自动将敏感内容高亮标红。

违规内容处理：管理员可一键删除存在敏感词的帖子，操作便捷、反馈明确。

模块可扩展性：支持后期引入词库自定义管理机制和审核日志记录功能，便于持续优化审核策略。

该功能实现不仅提高了系统的内容治理能力，也符合校园平台对信息审查的实际管理需求。

#### **4.4 学生界面：课程信息与个人信息展示模块**

我还设计并实现了学生端的两个关键界面：课程信息展示界面与学生个人信息界面，注重布局层次与视觉清晰度，提升了整体用户体验。

• 使用 SunnyUI 控件库 构建界面，采用卡片化风格与分块展示方式，增强视觉层次感；

• 课程信息部分清晰展示了课程编号、课程名称、授课教师、课程时间等关键字段；

• 学生个人信息界面支持基础信息的查看与部分字段的编辑，布局结构清晰，交互自然；

• 界面设计遵循“信息分区明确 + 操作按钮直观”的原则，适合学生日常高频使用场景。



图4.5（课程信息展示界面）



图4.6（学生个人信息展示界面）

## 五、遇到的困难与解决方法

在本系统的开发过程中，尽管我已在课上学习了很多 C# 和 WinForms 编程知识，但实际开发中仍然遇到了多个具有挑战性的技术问题，尤其集中在界面美化、界面适配、项目整合等方面。通过查阅文档资料、调试测试和总结经验，我逐一找到了解决思路和方法，以下列举几个具有代表性的技术难点与应对措施：

#### 5.1 UI 美化库的选择与冲突问题

为了提升用户体验，我在项目中引入了多个第三方 UI 控件库，如 SunnyUI 与 MaterialSkin，但在集成过程中遇到了如下困难：

兼容性问题：两种 UI 控件库在部分控件注册机制上存在冲突，容易导致控件显示异常或界面响应卡顿。

与原生控件风格不一致：WinForms 自带控件的样式较为传统，与现代化控件风格差异较大，导致界面整体风格不统一。

控件覆盖与层级问题：部分美化控件如分页容器、图标按钮在不同界面中出现重叠或层级混乱，影响视觉效果与操作体验。

解决方法：

严格统一界面风格规范，只在同一模块中选用一种 UI 控件库，避免混用导致冲突；

手动调整控件属性（如 BackColor、FlatStyle、Font）以融合控件之间的视觉风格；

通过合理设置控件层级（BringToFront() / SendToBack()）和容器嵌套方式，解决控件覆盖问题；

在调试阶段使用控件可视化调试工具，定位显示异常的具体控件并逐一修复。

#### 5.2 全局皮肤设置对其他界面的干扰

在项目整合阶段，我的管理员界面受到了另一个同学负责的界面的全局皮肤的干扰，以提升界面一致性。但实际测试中发现，部分全局样式设定会自动作用于整个程序的所有窗体，导致部分学生端界面被非预期更改，如按钮颜色异常、字体失真等问题。

解决方法：

将皮肤加载与应用逻辑封装为单独方法，仅在需要的窗体中调用，避免全局注册；

对不希望被统一样式影响的界面，手动禁用皮肤机制或设置局部覆盖样式；

在不同模块之间设定样式隔离机制，确保各角色界面风格清晰分离、互不干扰。

#### 5.3 窗体最大化后控件与字体无法自适应

在设计界面最大化功能时，我发现虽然窗体尺寸变化成功，但内部控件（如 TextBox、Label、DataGridView 等）仍保持原始尺寸和字体大小，导致界面出现留白或控件分布不协调的问题。

解决方法：

使用控件的 Anchor 和 Dock 属性，使控件能够随着窗体缩放自动调整大小与位置；

对于包含大量控件的页面，引入 TableLayoutPanel 或 FlowLayoutPanel 等布局容器，提升整体自适应性；

针对字体大小问题，使用事件监听 Form.Resize 方法，在窗体尺寸变化时动态调整控件字体（例如：label1.Font = new Font("微软雅黑", this.Width / 100);）；

在美化库支持的前提下，借助控件库自带的缩放特性（如 SunnyUI 的 DPI 缩放）进一步优化。

## **六、后续发展与优化展望**

本项目已初步实现了教学管理流程中的多个关键模块，满足日常的学生、教师、课程数据管理、作业布置与考试信息展示等基本功能需求。在现有功能框架和系统架构的基础上，未来仍有大量优化与拓展的空间，可从以下几个方面进行深入提升：

#### 6.1 功能模块拓展

作业与考试模块增强：目前系统主要提供课程与学生信息的管理展示，后续将进一步开发作业布置、提交、批改与成绩统计等模块；同时为考试安排提供试卷录入、自动判卷（针对客观题）、成绩反馈等功能，打造更完整的教学流程闭环。

消息通知与公告系统：为提升系统的交互性与提醒机制，计划新增消息推送模块。包括管理员公告发布、作业/考试截止提醒、系统通知等，采用浮窗提示或邮件/短信通知等方式提升用户响应效率。

数据统计与可视化：针对教师端、管理员端引入数据可视化组件，展示课程成绩分布、学生出勤情况、作业提交进度等，助力教学过程精细化管理。

#### 6.2 用户体验优化

响应式界面与自适应布局：考虑当前系统主要为固定分辨率开发，后续可引入更灵活的布局控件或控件容器（如 TableLayoutPanel 或 FlowLayoutPanel），以适应不同设备分辨率和窗口大小，提升兼容性。

界面主题个性化设置：在保留基础的统一风格外，为用户提供“夜间模式”“个性配色”等皮肤切换选项，增强系统的人性化设计与使用舒适度。

#### 6.3 数据安全与权限控制强化

角色权限管理细化：目前系统基于“管理员/学生”角色划分基本权限，后续可通过 RBAC（基于角色的访问控制）模型，进一步细化操作权限（如只读/可写、模块访问控制等），提升系统安全性与多用户协作能力。

密码安全加强：在现有 SHA256 哈希加密的基础上，引入“加盐机制”与登录行为风控策略（如多次登录失败锁定账号），进一步提升系统对暴力破解与信息泄露的抵抗能力。

日志审计系统：为提升系统可追溯性与管理透明度，后续将完善操作日志模块，记录用户登录、数据修改、权限变更等关键操作，并支持管理员导出与审计分析。

#### 6.4 系统智能化升级

引入人工智能技术：在后续迭代中，可探索引入 AI 算法模型，辅助完成如下智能功能：

作业/试卷的自动评分与反馈；

学情分析与个性化学习建议；

教务辅助决策（如课程容量调整建议、教师教学绩效评估等）；

智能推荐机制：基于学生的历史选课记录与成绩表现，为其推荐合适课程，辅助学生制定合理学习计划。

#### 6.5小结

本系统作为一个以实际教学管理需求为导向的综合性平台，具有良好的扩展潜力和升级空间。通过持续的技术优化与功能完善，未来将能够服务更大规模的用户群体，提升学校教学与管理的效率，实现智慧教育的信息化转型。

**七、开发体会**

通过本次“教学作业考试一体化管理平台”的开发实践，我不仅巩固了 C# 编程语言及 Windows Forms 桌面应用开发的相关知识，还深刻体会到软件工程中各模块协作、设计思想以及编程规范的重要性。

首先，在实际动手开发之前进行完整的系统需求分析与功能规划，是保障项目有序推进的前提。在分析不同用户角色（管理员、教师、学生）需求的基础上，我明确了各界面的功能边界，提前设计好数据库表结构，确保数据之间的关系清晰，避免了后期的频繁返工。

其次，在界面开发过程中，我充分感受到UI 体验对软件使用感受的重要性。传统的 WinForms 控件虽然功能丰富，但在美观性上略显不足，因此我主动调研并集成了 SunnyUI 和 MaterialSkin 两种第三方库。在实际使用中，不仅美化了按钮与卡片控件，也让用户的操作体验更加舒适自然，这在提升软件质量方面起到了明显作用。

在编码实现阶段，我特别注重代码的规范性与可读性，将数据访问逻辑封装在 DAL 层，界面事件逻辑则集中在界面类中，实现了功能分离。同时，我首次深入使用了 LINQ 技术处理集合数据，相较于传统 foreach 遍历，写法更简洁、效率更高，也便于维护和理解。

另外，我深刻体会到多线程的重要性。起初在加载大量数据时出现界面卡顿现象，通过查阅资料并引入 Task.Run() + this.Invoke() 的异步更新模式，成功实现了“后台处理 + 主线程刷新”的流畅逻辑，提升了系统响应性能，也让我更深入理解了 UI 线程与后台线程之间的关系与同步方式。

在数据输入与系统运行稳定性方面，我引入了正则表达式进行格式校验，有效防止了错误数据进入数据库。同时，每一个可能导致异常的操作点都使用 try-catch 进行保护，增强了系统的健壮性。

最后，在文件操作、系统调试与测试过程中，我深刻理解到“细节决定成败”。功能是否稳定、逻辑是否严谨、提示是否友好，都会直接影响用户的使用感受与评价。

本次实验让我实现了从“课堂学习”到“实践开发”的质的飞跃，全面锻炼了我在需求分析、UI 设计、数据库设计、程序编写与调试方面的综合能力，同时也让我更加明确了自己在今后学习和职业发展中的方向。

**教师评语评分**

|  |  |  |  |
| --- | --- | --- | --- |
| **课程目标** | **应得分** | **实得分** | **评分说明** |
| 课程目标1： |  |  |  |
| 课程目标2： |  |  |  |
| 课程目标3： |  |  |  |
| 总分 |  |  |  |

评阅人：

年 月 日