

# 中国科学技术大学计算机学院

## 《数据库系统实验报告》



实验题目：学籍管理系统

学生姓名：牛庆源

学生学号：PB21111733

完成时间：2024年6月20日

# 需求分析

设计一个学籍管理系统，以及登录界面，实现管理员进行所有操作，学生进行部分操作。

其中管理员可以新增学生，课程等，也可以删除，更改，以及查询所有的内容，同时可以重置数据库。

学生可以查询学校开的课程，奖励和惩罚，自己的包括成绩，奖惩以及个人信息，总学分，平均成绩的信息。

# 总体设计

## 系统模块结构

前端使用Python的PyQt6，后端是基于Python的数据库调用和用户操作逻辑，数据库是Mysq。

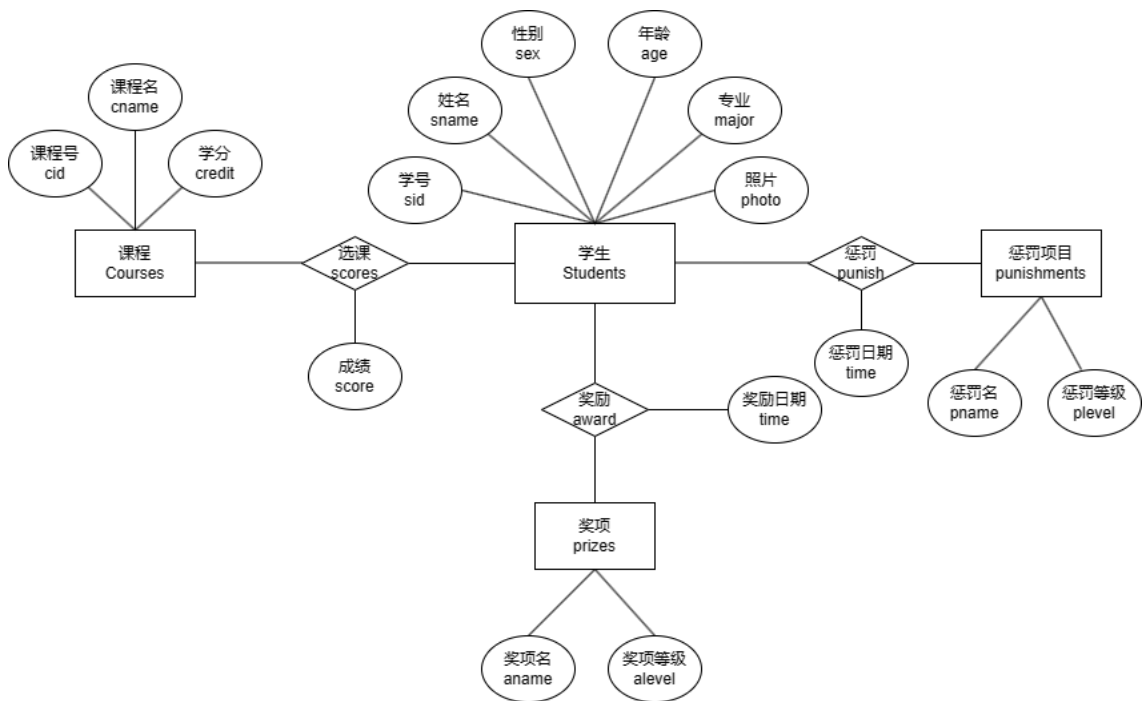
整体使用C/S架构。

## 系统工作流程

用户通过登录界面登录为学生/管理员，使用按钮进行查询等操作。操作逻辑通过Python调用，注入sql代码实现数据库内容的增删查改，可以调用存储过程和函数。

## 数据库设计

- ER 图如下：



- 模式分解为：

Students(sid, sname, sex, age, major, photo) -- sid为主码

Courses(cid, cname, credit) -- cid为主码

Prizes(aname, alevel) -- aname为主码

Punishments(pname, plevel) -- pname为主码

Scores(sid, cid, scores) -- sid,cid为主码，分别外键约束于Students和Courses的对应属性

award(sid, aname, time) -- sid,aname分别外键约束于Students和Prizes的对应属性

punish(sid, pname, time) -- sid,pname分别外键约束于Students和Punishments的对应属性

StuLogin(sid, password) -- sid为主码，外键约束于Students的对应属性

注意 time 格式为 Date , photo 格式为 blob , age , credit 和 scores 格式为 int , 其余都为 varchar 。

- 主要设计的 procedure 为修改学生ID和修改课程ID，实现修改ID时，外键约束对应的属性为修改后的内容。

`function` 为计算学生的总学分和加权平均成绩。

`trigger` 为在删除学生时删除学生有关的成绩、奖励情况和惩罚情况；在新增学号时，设置学生的默认登录密码；修改学生学号时，自动更新学生登录信息。

### 核心代码解析（可改名为对应模块，如后端实现）

### 仓库地址

Github地址: [USTC-CS-COURSES/数据库/lab/lab2/src](https://github.com/USTC-CS-COURSES/数据库/lab/lab2/src) at master · n1uf/USTC-CS-COURSES (github.com)

## 目录

|                  |                      |
|------------------|----------------------|
| db_functions.py  | ----- sql函数          |
| db_procedures.py | ----- sql存储过程        |
| db_triggers.py   | ----- sql触发器         |
| display.py       | ----- 用户界面定义         |
| functions.py     | ----- 用户功能函数         |
| inittable.py     | ----- 注入sql实现数据库初始化  |
| main.py          | ----- 主函数（运行该函数）     |
| temp_image.jpg   | ----- 经过尺寸处理后的图像（缓存） |

## 数据库的建模

参照ER图以及分解出的模式，通过使用Python的 `pymysql` 库实现从Python注入sql代码，实现数据库的建模

## 功能实现

首先确定自己需要的基本功能，包括增删查改等，然后根据需求设计函数（如新增学生等），再根据进一步的需求设计sql函数（如查询总学分、平均成绩），sql存储过程（如修改学生ID、课程ID），以及功能实现时考虑触发器（删除时同步删、新增时新增默认密码）

这里给出一些存储过程、函数和触发器的定义以及解释:

- 存储过程（修改学生ID）：

用以下代码去掉外键约束

```
// db_procedures.py
DROP FOREIGN KEY Scores_sforeign;
```

同步修改ID

```
UPDATE Scores
SET sid = new_id
WHERE sid = old_id;
```

之后后加上外键约束

```
ADD CONSTRAINT Scores_sforeign FOREIGN KEY (sid) REFERENCES Students (sid);
```

其余同步修改同理实现即可。

- 函数（查询某个学生的平均成绩）

主要实现方式：

计算每一个成绩与学分的乘积的和

```
// db_functions.py
SELECT SUM(Scores.score * Courses.credit) INTO total_score
FROM Scores, Courses
WHERE Scores.cid = Courses.cid AND Scores.sid = stuid;
```

计算学分和

```
SELECT SUM(Courses.credit) INTO total_credits
FROM Courses
WHERE cid IN (SELECT cid FROM Scores WHERE Scores.sid = stuid);
```

返回加权平均成绩

```
RETURN total_score / total_credits;
```

- 触发器（新增学生时自动赋默认登录密码）

```
// db_triggers.py
AFTER INSERT ON Students
FOR EACH ROW
BEGIN
    INSERT INTO StuLogin (sid, password) VALUES (NEW.sid, '123456');
END
```

functions.py 内设计了很多函数用来被 display.py 调用，这些函数通过注入sql代码的方式实现对数据库内容的增删查改等操作。

例如：

- 查询某个学生的所有成绩：

```
// db functions.py
cursor.execute("""SELECT scores.cid, cname, score, credit
                FROM Scores, Courses, Students
                WHERE scores.cid = courses.cid
                AND scores.sid = students.sid
                AND scores.sid = %s""", (sid,))
```

通过连接表实现各项数据的查询和显示。

- 查询某个学生的平均成绩：

```
// db functions.py
cursor.execute("SELECT GetAverageScore(%s) as AverageScore", (sid,))
```

通过编写的sql函数实现。

- 修改学号：

```
// db functions.py
cursor.execute("CALL updateStudentID(%s, %s)", (old_id, new_id))
```

通过调用sql存储过程实现。

这里需要注意，对于各种查询的返回，`fetchall` 和 `fetchone` 完全不同，需要根据后面的 `display.py` 修改，如果需要全部返回就 `fetchall`，如果需要返回一个头指针用于遍历，就 `fetchone`

## 前端实现

使用 PyQt6 实现。

主要逻辑在 `display.py` 中，每一个类都创建一个用户界面。

每一个类中都包含有以下大体内容：初始化（用于传参），布局设置（包括显示），内部函数设置。

**先说明类与类之间的调用以及类调用 `functions.py` 中的函数（例为登录操作）**

`Main_window` 是初始界面，设置布局来为用户提供登录方式，不同登录方式连接不同的内部函数，内部函数实现逻辑并按照需求调用另一个类。

如学生登录，则调用内部函数 `open_student_window`，这个函数内调用了 `StudentLogin` 类来唤起新窗口。

类 `StudentLogin` 内同样设置布局来引导用户输入信息，按确认按钮。确认按钮连接了内部函数 `get_text`，其中包含了登录的逻辑以及唤起类 `Student_window` 界面（学生操作界面）。

这里登录的逻辑使用了 `functions.py` 中的相关函数来判断信息是否输入合法，通过 `flag` 来标记合法的数量，达到一定数量则通过登录，唤起 `Student_window`，然后关闭自身。

## 这里给出管理员界面(Admin\_Window)的一些布局, 函数

使用 QTabWidget 创建一些选项卡, 归类放置功能

```
// display.py
tab_widget = QTabWidget()
tab_widget.addTab(self.create_add_box(), '添加操作')
tab_widget.addTab(self.create_query_box(), '查询操作')
# ...
```

在每一个选项卡内使用网格布局 QGridLayout 来布置按钮。

```
// display.py
buttons = [
    ('新增学生', self.add_student),
    ('新增课程', self.add_class),
    # ...
]
for i, (text, slot) in enumerate(buttons):
    btn = QPushButton(text)
    btn.setMinimumSize(100, 50) # 设置按钮最小尺寸
    btn.clicked.connect(slot)
    layout.addWidget(btn, i // 2, i % 2) # 2列布局
```

按钮功能为 self.add\_student 等, 调用内部函数。

其余的选项卡同理。

内部函数: 可以唤起新的类(界面), 将界面输入的内容作为参数传回该内部函数, 然后根据需求调用 functions.py 的函数修改或者查询数据库, 或者使用传回的参数实现逻辑的判断与处理。如果需要, 可以调用其他函数(重置数据库时需要确认界面以防止误触)。

以下是一个内部函数, 实现学生奖励的删除。(对应现实中奖励的撤销)

```
def delete_prize_for_student(self):
    self.deletewindow = DeletePAWindow(self.db, self.cursor, 0)
    if self.deletewindow.exec():
        newID = self.deletewindow.get_entered_ID()
        newName = self.deletewindow.get_entered_Name()
        functions.del_award(self.db, self.cursor, newID, newName)
        widget = QWidget()
        QMessageBox.information(widget, '信息', '删除成功')
```

首先唤起 DeletePAWindow 并传入相关参数, kind 设置为 0 来在类内使用相关的功能。

唤起成功后读取类传回该内部函数的参数, 然后通过调用 functions.py 内的 del\_award 函数对数据库进行操作, 从而实现删除学生奖励的功能, 最后返回一条删除成功的信息。(这里删除失败的信息会在 DeletePAWindow 中实现, 如果参数读入(输入)正确则不会出现删除失败的信息)

## 主函数

使用 `pymysql` 输入对应参数打开数据库连接。

调用 `display.py` 中的 `Main_window` 打开主界面即可。

## 其他

图片的输入和读取：使用 `QtWidgets` 中的 `QFileDialog` 来读取图片，然后使用 `PIL` 库的 `Image` 来规定（调整）图片大小尺寸（防止查看时图片尺寸过大破坏布局），注意转换图片格式。然后转为字节数据，二进制读取并存入图片。

## 实验与测试

---

### 依赖

需要安装 `pymysql`，`PyQt6` 和 `PIL` 库

### 部署

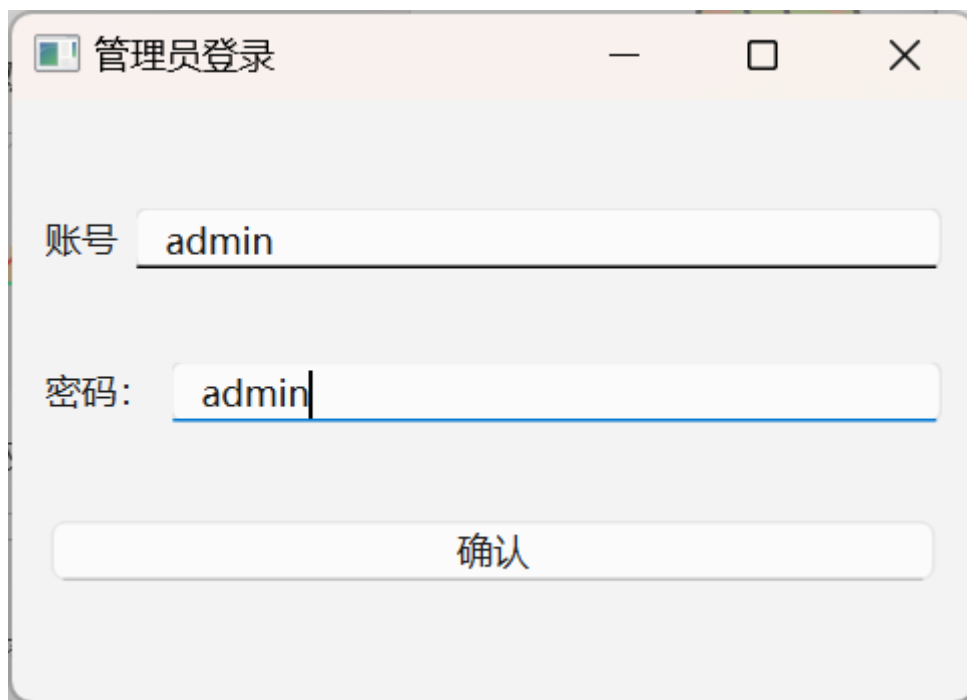
使用 `python` 运行 `main.py`。

### 实验结果

运行程序，出现登录界面，可以选择管理员登录和学生登录，分别可以使用不同的功能。

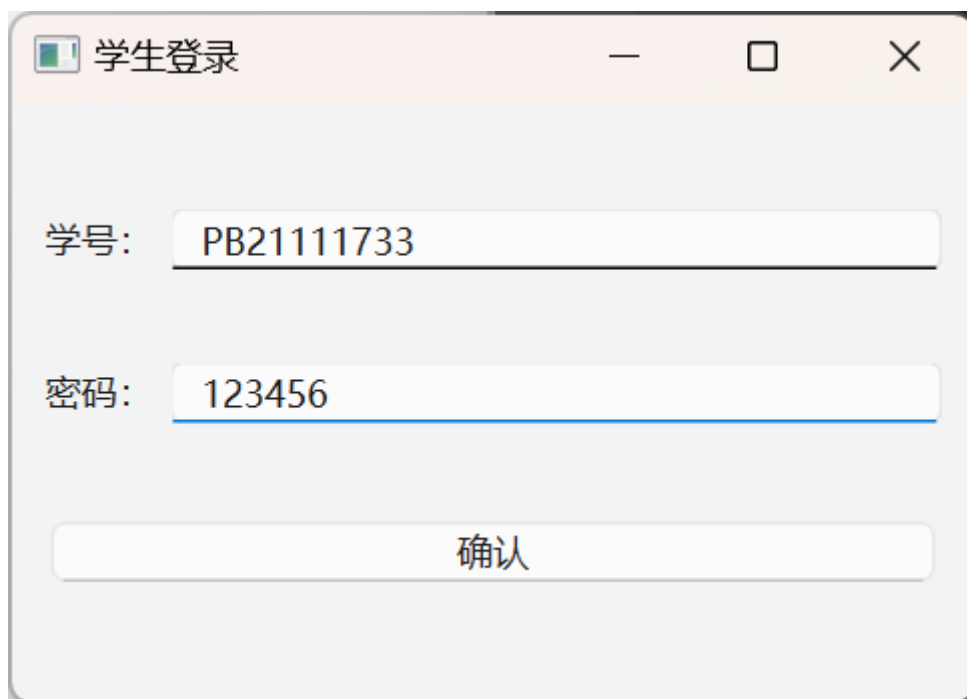


管理员登录账号为 `admin`，密码也为 `admin`。



A screenshot of a Windows-style application window titled "管理员登录" (Administrator Login). The window has a light gray background and a title bar with standard minimize, maximize, and close buttons. It contains two text input fields: the first is labeled "账号" (Account) and contains the text "admin"; the second is labeled "密码:" (Password:) and contains the text "admin". Below these fields is a single button labeled "确认" (Confirm).

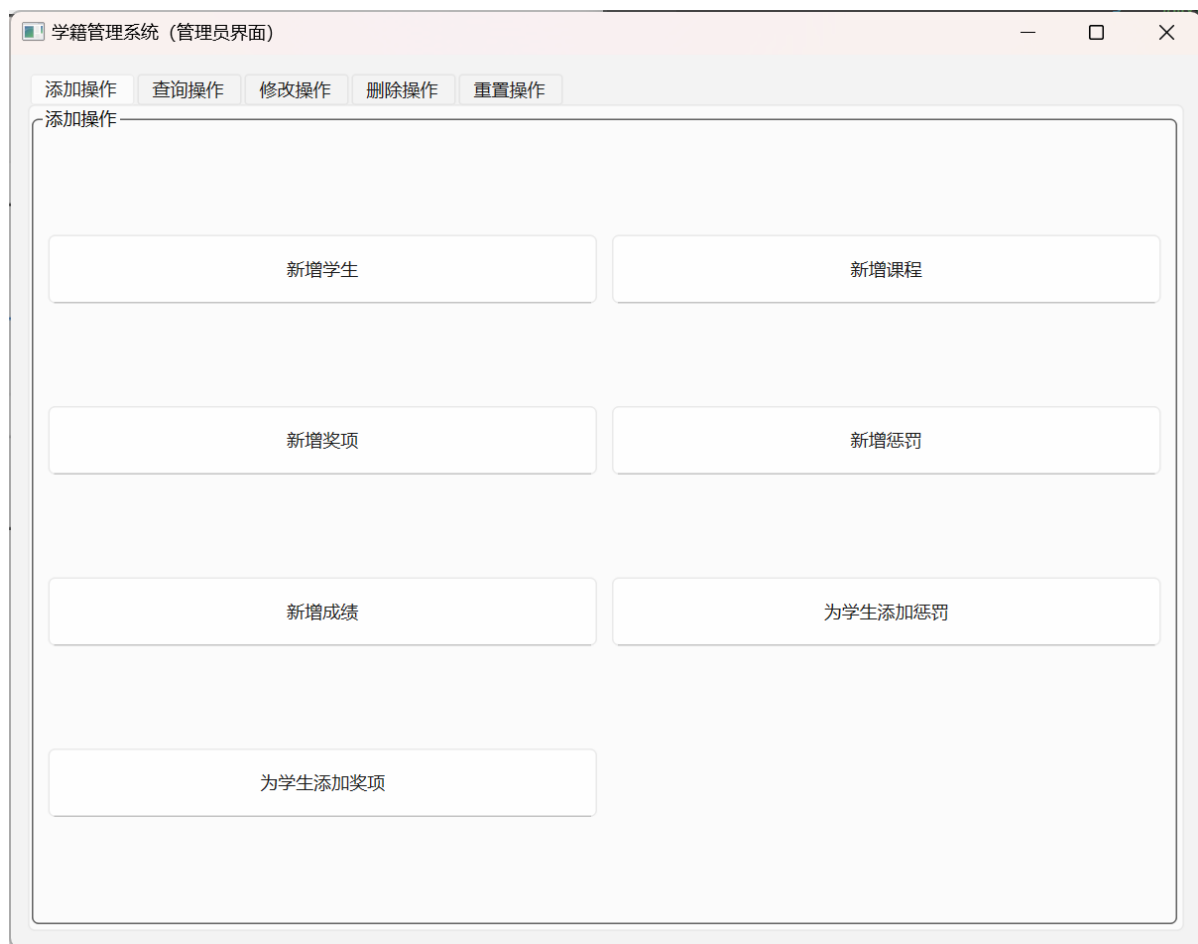
学生登录账号为学号，密码默认为'123456'，可以在登录后修改。



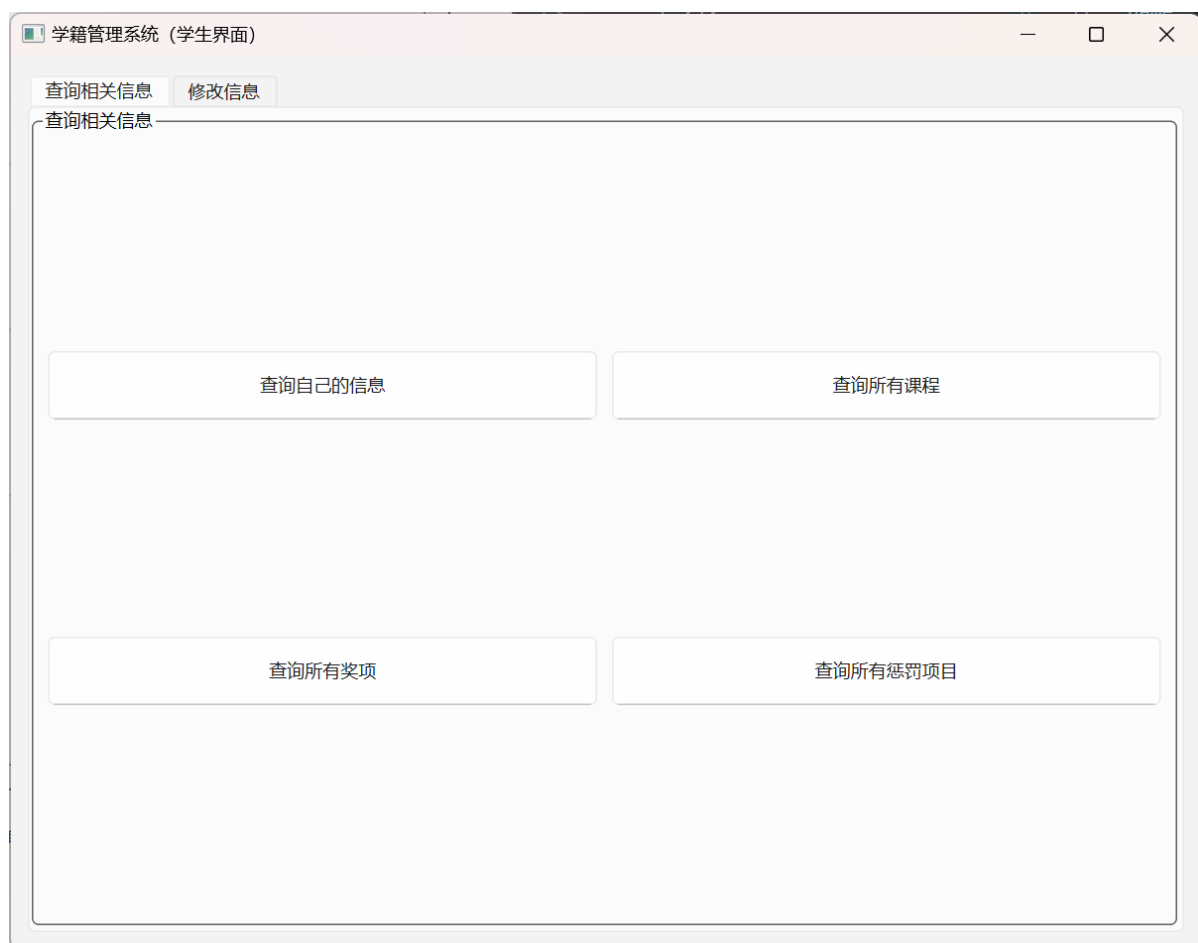
A screenshot of a Windows-style application window titled "学生登录" (Student Login). The window has a light gray background and a title bar with standard minimize, maximize, and close buttons. It contains two text input fields: the first is labeled "学号:" (Student ID:) and contains the text "PB21111733"; the second is labeled "密码:" (Password:) and contains the text "123456". Below these fields is a single button labeled "确认" (Confirm).

管理员界面如下：设计多个选项卡以使得界面美观，并且功能直观

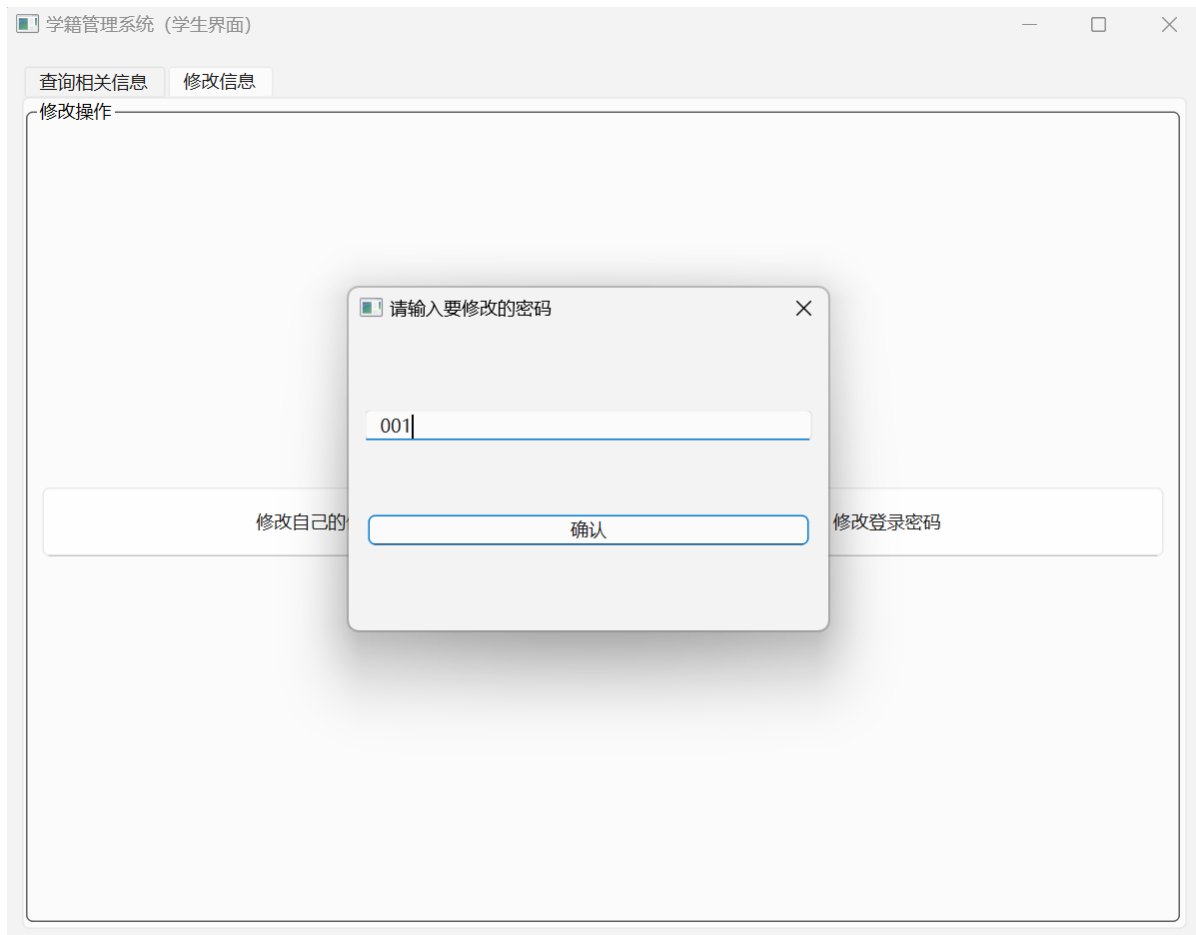




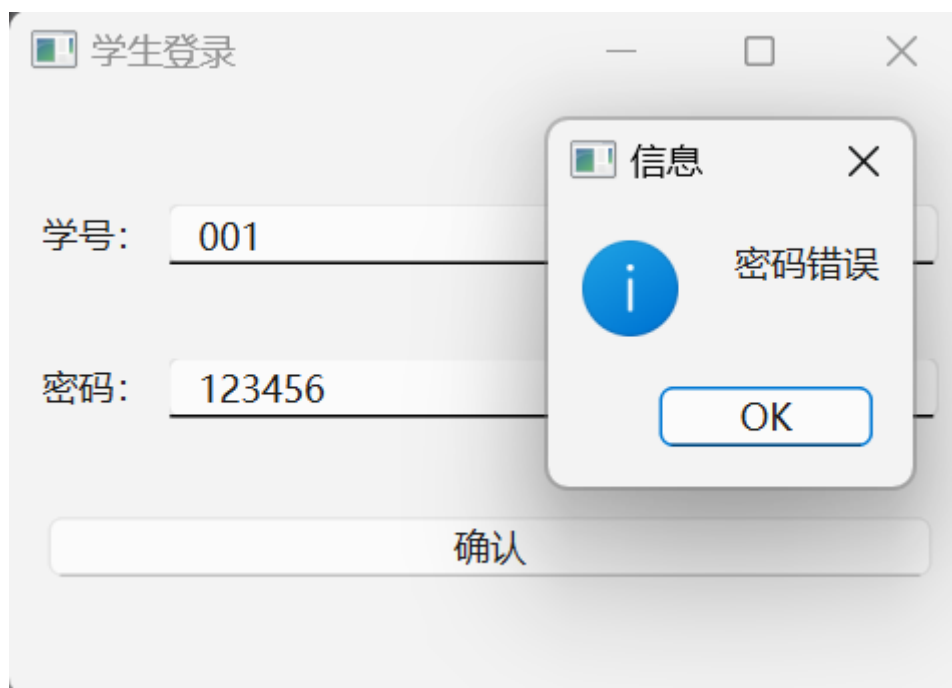
学生界面如下：比管理员少很多功能，提供了基本的信息查询以及信息修改包括修改密码。



修改密码：



修改后使用之前密码登录会错误:



一些增删查改操作:

查看学生名单:

| 学生名单  |            |     |    |    |           |
|-------|------------|-----|----|----|-----------|
|       | 学号         | 姓名  | 性别 | 年龄 | 专业        |
| 1     | 001        | ab  | M  | 23 | CS        |
| 2     | 002        | cd  | F  | 20 | EN        |
| 3     | 003        | ab  | F  | 24 | CH        |
| 4     | 111        | abc | F  | 26 | CS        |
| 5     | PB21111733 | 牛庆源 | M  | 21 | 计算机科学与... |
| Close |            |     |    |    |           |

查看课程名单：

| 课程    |     |        |     |
|-------|-----|--------|-----|
|       | 课程号 | 课程名    | 学分  |
| 1     | 001 | 数据库    | 2.5 |
| 2     | 002 | 人工智能   | 3.0 |
| 3     | 003 | 算法基础   | 3.0 |
| 4     | 004 | 数字图像处理 | 2.5 |
| 5     | 005 | 自然语言处理 | 2.0 |
| Close |     |        |     |

新增学生：（冲突，之后写学号为123）

输入新学生信息

学号: 001

姓名: niuf

性别: F

年龄: 25

专业: CS

选择要上传的照片

确认

信息

i

该学号已存在

OK

修改后的学生名单:

| 学生名单  |            |      |    |    |           |
|-------|------------|------|----|----|-----------|
|       | 学号         | 姓名   | 性别 | 年龄 | 专业        |
| 1     | 001        | ab   | M  | 23 | CS        |
| 2     | 002        | cd   | F  | 20 | EN        |
| 3     | 003        | ab   | F  | 24 | CH        |
| 4     | 111        | abc  | F  | 26 | CS        |
| 5     | 123        | niuf | F  | 25 | CS        |
| 6     | PB21111733 | 牛庆源  | M  | 21 | 计算机科学与... |
| Close |            |      |    |    |           |

添加学生成绩：

输入学生的学号和课程号

×

学生学号:

123

课程号:

003

成绩:

79

确认

添加惩罚:

输入受惩学生的学号和惩罚信息

×

学生学号:

123

惩罚名称:

001

时间:

2023.1.9

确认

查询学生具体信息 (添加了两个成绩一个惩罚一个奖项):

学生详细信息

学号: 123      姓名: niuf      性别: F

年龄: 25      专业: CS

当前已获总学分: 5.5      当前平均成绩: 83.5455



|   | 课程号 | 课程名  | 成绩 | 学分  |
|---|-----|------|----|-----|
| 1 | 001 | 数据库  | 89 | 2.5 |
| 2 | 003 | 算法基础 | 79 | 3.0 |

|   | 惩罚项目 | 惩罚等级 | 时间         |
|---|------|------|------------|
| 1 | 001  | B    | 2023-01-09 |

|   | 奖项  | 奖励等级 | 时间         |
|---|-----|------|------------|
| 1 | 002 | B    | 2024-06-20 |

Close

修改学生信息:



 请输入要修改的内容，不需要修改则留空

×

从123修改为:

在此输入新的学号...

从niuf修改为:

fuin

从F修改为:

在此输入新的性别(M/F)...

从25修改为:

在此输入新的年龄...


从CS修改为:

在此输入新的专业...

选择要上传的照片

确认

删除学生奖项:

 输入要删除的学号和奖项/惩罚

×

学号:

123

奖项/惩罚:

002

确认

删除学生成绩:

输入要删除的学生学号和课程号

×

学生学号:

123

课程号:

003

确认

修改后的学生信息：可以看到姓名，课程，奖项都发生改变，学分和平均成绩也改变了。

学生详细信息

学号: 123      姓名: fuin      性别: F

年龄: 25      专业: CS

当前已获总学分: 2.5      当前平均成绩: 89.0



|   | 课程号 | 课程名 | 成绩 | 学分  |
|---|-----|-----|----|-----|
| 1 | 001 | 数据库 | 89 | 2.5 |

|   | 惩罚项目 | 惩罚等级 | 时间         |
|---|------|------|------------|
| 1 | 001  | B    | 2023-01-09 |

| 奖项 | 奖励等级 | 时间 |
|----|------|----|
|----|------|----|

Close

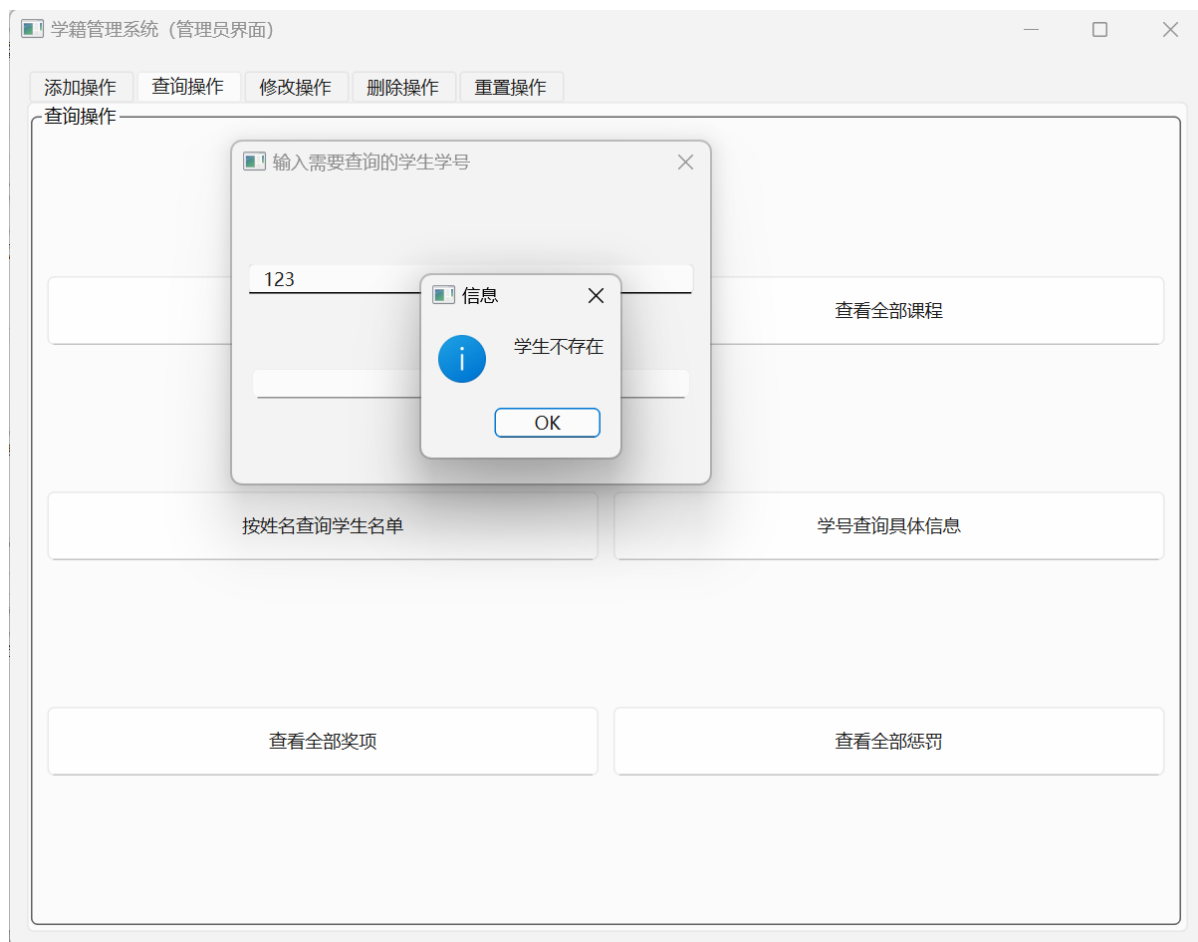
删除学生:

删除后查询学生名单:

删除后查询学生名单:

| 学生名单  |            |     |    |    |           |
|-------|------------|-----|----|----|-----------|
|       | 学号         | 姓名  | 性别 | 年龄 | 专业        |
| 1     | 001        | ab  | M  | 23 | CS        |
| 2     | 002        | cd  | F  | 20 | EN        |
| 3     | 003        | ab  | F  | 24 | CH        |
| 4     | 111        | abc | F  | 26 | CS        |
| 5     | PB21111733 | 牛庆源 | M  | 21 | 计算机科学与... |
| Close |            |     |    |    |           |

删除后查询学生具体信息：



## 参考

页面布局使用 `QtWidgets` 的一些默认布局函数来创建网格视图和标签。