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：朱炜荣

学生学号：PB21111681

完成时间：2024年6月16日

需求分析

1. 数据需求分析
    - 学校内部有很多在读的同学，每位同学可以用唯一的学号来确认，同时需要收集同学的相关个人信息，比如：名字、性别、入学年份、专业号、联系电话、个人邮箱等。其中除联系电话和个人邮箱外要求其余属性值非空。
    - 不同的专业有不同的专业号，由专业号唯一确定，同时还包括属性专业名、所属学院的学院名，属性值均要求不能为空。
    - 每个学生都有不同的选课记录。每名同学的每个课程可以由学号与课程号唯一的确定，需要包括学生成绩的属性，要求除学生成绩外属性值非空。
    - 不同的课程有不同的课程号，可以由课程号唯一的确定，同时还需要收集课程名、课程学分、课程类别的属性，要求均非空。
    - 不同的学生也会有不同的奖惩情况，可以由学号唯一确定，还需要收集信息的类别（奖/惩）、具体的信息，如果存在记录则属性值均不为空。
    - 不同的学生在上完课之后对课程也会有不同的评价，这里我选择将所有课程的评价分为不同的几个固定类别，对课程的评价可以由学号和类别唯一确定，评价可以如下定义：（学号，类别，课程），如（李华，成绩最高的课，数学分析）。所有属性值要求均不为空。
2. 功能需求分析
    - 课程管理：学生可以进行选课、退课、放弃成绩的操作
    - 专业变更：学生可以在合适的时间申请转专业
    - 学生管理：包括学生信息的增删改查
    - 课程管理：主要包括课程信息的增删改查
    - 奖惩管理：主要包括奖励惩罚记录的增删查
    - 学院管理：学院之间管理所属专业的增删改查
    - 成绩管理：学生查询课程的成绩

- 评价管理：学生对不同的评价类别选择、修改对应课程

总体设计

系统模块结构

本次实验采用的是B/S架构，采用实现Web网页的编程语言为python + django框架。依托django框架自带的安全性验证和管理员系统简化了数据库设计中的部分逻辑。

1. 前端设计：

   前端借助django框架自带的admin系统，主要负责展示查询结果，发送增删改查的条件信息回后端，以及用户登陆的相关控制。在前端不同的用户等级和用户信息会借助后端传回不同的界面信息。管理员可以看到数据库中除了学生对课程评价的信息之外的所有信息。并同时拥有对它们进行增删改查的权限。

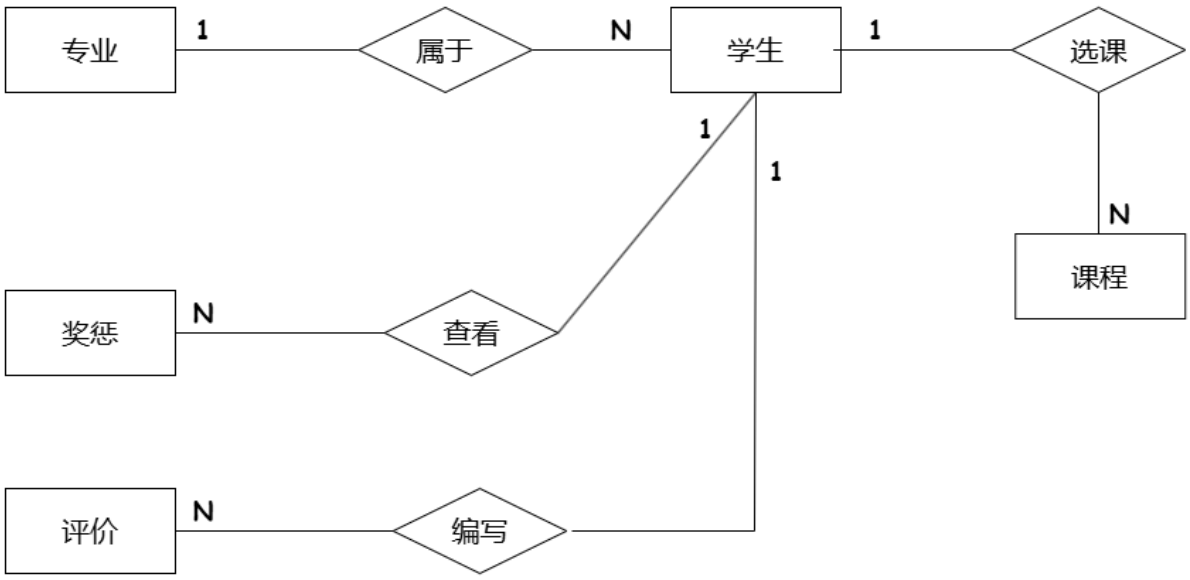
   而普通用户，也就是学生则只能看到跟自己相关的信息。学生可以修改的信息主要包括：增加放弃课程的选修和增加修改删除自己对不同课程的评价。

2. 后端设计：

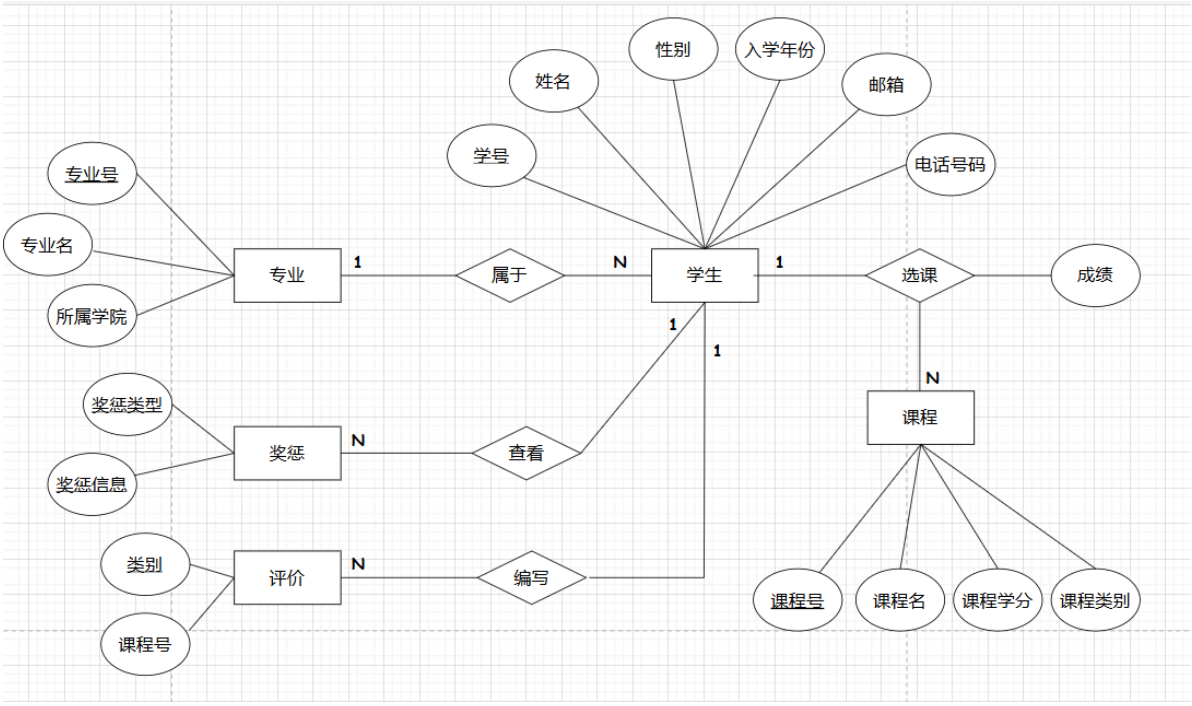
   后端主要是通过前端传递的信息进行对应的操作，主要是表单相关的提交和信息处理。借助不同分网页的提交的不同的表单名字来区别不同的执行操作。由于已经在前端进行过权限限制，所以在后端不做额外处理。同时根据前端需要展示的信息传递需要展示的数据库信息。

数据库设计

概要设计ER图如下所示：



全部属性的ER图如下所示：



初始的ER图翻译为关系模式如下：

学生（学号，姓名，性别，入学年份，联系电话，个人邮箱）

课程（课程号，课程名，课程学分，课程类别）

专业（专业号，专业名，所属学院）

奖惩（类别，奖惩信息）

评价（评价类别，课程号）

然后加入1对N的关系联系处理

学生（学号，姓名，性别，入学年份，联系电话，个人邮箱，专业号）

课程（课程号，课程名，课程学分，课程类别）

专业（专业号，专业名，所属学院）

奖惩（学号，类别，奖惩信息）

评价（学号，评价类别，课程号）

最后加入关系属性

选课（学号，课程号，成绩）

完整关系模式如下：

学生（学号，姓名，性别，入学年份，联系电话，个人邮箱，专业号）

课程（课程号，课程名，课程学分，课程类别）

专业（专业号，专业名，所属学院）

奖惩（学号，类别，奖惩信息）

评价（学号，评价类别，课程号）

选课（学号，课程号，成绩）

很明显，在每个子关系模式中，属性值不可能包含表，满足了1NF的要求；

同样的每个子关系中的非主属性都不部分依赖于主属性，满足了2NF的要求；

每个子关系中非主属性也不存在着传递依赖，从而满足了3NF的实验要求。

核心代码解析（可改名为对应模块，如后端实现）

仓库地址

> 0auv0/database2024: USTC 2024 Spring Database System (github.com)

目录

> —database
>
> ——|doc（主要为文档相关，不做过多展示）
>
> ——|src（代码）
>
> ————|——python（python相关代码）
>
> ——————————|——venv（python依赖的虚拟环境）
>
> ——————————|——StuManagementSystem
>
> ————————————|——basedb（Django定义的基础数据库APP）
>
> ————————————|——login（Django定义的登陆APP）
>
> ————————————|——statics（网页渲染用到的css js img）
>
> ————————————|——StuManagementSystem（Django定义的本体）
>
> ————————————|——templates（网页模板用于生成前端）
>
> ————————————|——_ _ init _ _.py
>
> ————————————|——manage.py
>
> ————————————|——models
>
> ————|——sql（sql相关代码）
>
> ——————————|——Course_Change.sql（有关课程的增删改查）
>
> ——————————|—— Drop.sql（删除存储过程、表）
> ——————————|——Eval_Change.sql（有关课程评价的增删改查）
> ——————————|——For_test.sql（函数sql）
> ——————————|——Info_Change.sql（有关奖惩信息的增删改查）
> ——————————|——Init_table.sql（初始化表）
> ——————————|——Pro_Change.sql（有关专业的增删改查）
> ——————————|——SC_Change.sql（有关选课情况的增删改查）
> ——————————|——Stu_Change.sql（有关学生的增删改查）
> ——————————|——test_insert.sql（测试样例插入）
> ——————————|——trigger.sql（定义触发器）

标题1

> 分为若干个标题，以一定逻辑顺序讲解代码，如按ER图实体、关系讲解数据库建模代码，按存储过程、触发器分类讲解代码、按前端实现、接口等讲解前端代码
>
> 注意不要无脑粘贴代码，选取具有代表性的代码讲解，并注明文件来源，如首行注释: // backend\models.py

basedb的设计

Django框架中支持文件结构中加入自己定义的新APP，借助这一点我们可以把数据库的相关定义从外层的管理系统中抽出来，方便我们后续的设计管理。basedb的文件夹结构十分简单，甚至可以说不涉及到任何的前后端交互，只是把我们自定义的数据库与Django框架下的数据库分开来，方便导入和处理。

与新建的APP文件夹结构唯一不同的地方在于：models.py

```
from django.db import models

# Create your models here.
class Course(models.Model):···


class Evaluation(models.Model):···


class Info(models.Model):···


class Profession(models.Model):···


class Sc(models.Model):···


class Student(models.Model):···


class Image(models.Model):
    img = models.ImageField(upload_to='img/')  # 设置图片上传路径
```

分别从链接的MySQL数据库中获得了定义的Course Evaluation Info Profession Sc Student基本表。

同时我们额外加入了Image的定义，但是我的处理是并不将其存储在数据库中，而是将其通过后端处理保存在static的img文件夹下，主要原因是MySQL数据库的不支持存储图像格式。将其转换为二进制存储又显得有点怪怪的，所以直接将其保存在本地了。后端处理相关逻辑的代码主要为StuManagementSystem中的view.py下的定义函数：

```
            if 'add_img' in request.POST:
                form = ImgForm(request.POST, request.FILES)
                if form.is_valid():
                    new_name = '%s.jpg' % request.user.username
                    where = '%s/img/%s' % (settings.MEDIA_ROOT, new_name)
                    with open(where, 'wb+') as destination:
                        print(where)
                        for chunk in request.FILES['img'].chunks():
                            destination.write(chunk)
                    messages.success(request, 'Image uploaded successfully.')
                else:
                    messages.error(request, 'Error in image form.')
```

将用户上传的图片用其用户名为名字，保存jpg图片到statics/img/路径下。

login的设计

主要的设计思想与basedb类似，创建了一个专门处理用户登陆逻辑的login APP，在这个文件夹系统下，主要涉及到了用户登陆页面的渲染和后端函数处理。

login/views.py代码主要逻辑如下：

```
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home')  # 登录成功后重定向到主页
        else:
            messages.error(request, '用户名或密码错误')
```

函数主要逻辑为通过POST方法获取用户输入的username和password，并进行Django自带的用户登陆验证。在这个数据库设计实现的过程中，对报错以及非法输入修改的处理一直都很不用户友好。不是直接出现404界面，就是出现Django框架报错的黄色背景网页。最终我选择了一个较为折中的方案：通过在执行之前就对其进行合法性判断，如果合法则正常进行下面的操作；否则我们将不执行修改数据库等的底层相关操作，并借助messages返回我们想要在网页上显示的信息。这也体现了数据库实验中对于事务性编程的相关要求。

```
{% if messages %}
    <div class="error ">
        {% for message in messages %}
        <div class="alert alert-danger alert-dismissible fade show">
            <button type="button" class="btn-close" data-bs-dismiss="alert">
</button>
            <strong>{{ message }}</strong>
        </div>
        {% endfor %}
    </div>
{% endif %}
```

StuManagementSystem的设计

针对普通用户的管理设计

每位学生在进行过登陆之后应该能看到自己的个人相关信息，但是不能修改；

每位同学在专业界面只能看到目前学校的所有专业，但是没有管理权限；

每位同学在成绩界面可以看到所有已经出分的科目和成绩，以及自己的GPA，但是没有管理权限；

每位同学可以在选课界面看到自己的课程，同时可以选课和退课；

每位同学可以在奖惩界面看到自己的奖惩情况，但是没有管理权限；

每位同学可以在课程评价界面看到自己对课程的评价，拥有增加修改删除的权限。

针对管理员的管理设计

管理员可以在信息页面看到所有人的个人信息，具有修改增加删除的权限；

管理员可以在专业页面看到所有专业的信息，具有修改增加删除的权限；

管理员可以在开课页面看到所有开课的信息，具有修改增加删除的权限；

管理员可以在选课页面看到所有选课的信息，具有修改增加删除的权限；

管理员可以在个人奖惩页面看到所有奖惩的信息，具有修改增加删除的权限；

前端的网页设计

主要借助了boostrap 5进行主要渲染加以强制性CSS进行修改，并使用Django的模板网页结构进行生成，基础模板base.html如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>学籍管理系统</title>
</head>
<body>
    <!-- templates/navbar.html -->

<nav id="nav" class="navbar navbar-expand-sm bg-primary navbar-dark sticky-top">
            <div class="container-fluid">
                <img class="navbar-brand ms-4"
src="https://email.ustc.edu.cn/coremail/s?
func=lp:getImg&org_id=&img_id=logo_001" width="200px">

                <div class=" collapse navbar-collapse" id="navbarCollapse">
                    <div class=" col-sm-3 pb-1 navbar-nav">
                        <div class="col-1"></div>
                        <span class="my-auto" style="text-align: center; font-
size:1.125rem;"><a href="" style="text-decoration: none; color: #FFFFFF;"
target="_blank">
                                        学籍管理系统</a></span>
                    </div>
                    <div class="  col-sm-9 pb-1 navbar-nav">
                        <div class="col-sm-2"></div>
                        <div class="container-fluid navbar-collapse col-sm-6 d-
flex justify-content-between" id="nav-list">
                            <a class="nav-link nav-item dropdown text-center"
target="_blank" href="../">个人信息</a>
                            <a class="nav-link nav-item dropdown text-center"
target="_blank" href="../profession">专业情况</a>
                            <a class="nav-link nav-item dropdown text-center"
target="_blank" href="../course">选课情况</a>
                            <a class="nav-link nav-item dropdown text-center"
target="_blank" href="../score">个人成绩</a>
                            <a class="nav-link nav-item dropdown text-center"
target="_blank" href="../info">个人奖惩</a>
                            {% if user.is_superuser %}
                                <a class="nav-link nav-item dropdown text-
center" target="_blank" href="../admin">管理员</a>
                            {% else %}
                                <a class="nav-link nav-item dropdown text-
center" target="_blank" href="../evaluation">课程评价</a>
                            {% endif %}
                        </div>
                        <div class="col-sm-1"></div>
                    </div>
                </div>
            </div>
```

```
        </nav>

    <div class="container-fuild pt-3 row">
        <div class="col-2"></div>
        <div class="col-8">
            <div class="text-end">
    <h1>欢迎来到学籍管理系统</h1>
</div>
    {% if user.is_authenticated %}
        <div class="text-end h5">
            <p>你好, {{ user.username }}! <a href="{% url 'login' %}">登出</a></p>
        </div>
            {% block mainbody %}
       <p>original</p>
    {% endblock %}
    {% else %}
        <div class="text-center">
            <p>你尚未登录，请 <a href="{% url 'login' %}">登录</a></p>
        </div>
            {% endif %}
        </div>
        <div class="col-2"></div>
    </div>
</body>
<link rel="stylesheet" href="/static/css/bootstrap.min.css">
<script src="/static/js/bootstrap.bundle.js"></script>
<style>
    .row{
        margin: 0px!important;
    }
    body{
        background: url(../static/img/bg.gif) repeat;
    }
    #pic{
        width: 100px;
        height: 100px;
        border-radius: 50%;
    }
</style>
</html>
```

各个分页面基本与差别不大，最大的区别为不同的数据库表需要不同类型的数值和不同数量的输入，需要对不同的提交按钮命不同的名字，同时修改不同用户在同一个页面上的内容的渲染方法。

与上述网页不同的是，register.html并没有使用模板渲染的方法，而是单独做了一个完整的页面，这样方便于对其他网页调用登陆时提供接口与渲染，具体的代码可以见GitHub仓库。

SQL部分的设计

SQL部分主要定义了各种所需要使用的表，插入测试数据，定义了一些必要的存储过程，以及函数和触发器

存储过程主要是因为，导入到Django框架下的models很难借助form的形式对多个属性作为主键的表进行查询，所以不如直接调用存储过程进行增删改查。

触发器主要是有关挂科的奖惩信息添加，当一个学生累计有三门及以上的挂科科目时，便会在这个学生的奖惩信息中加入存在挂科情况的警告信息，而当挂科科目减少到三门以下时，便会自动把这个挂科情况删掉。

函数主要定义的是GPA的计算，通过查询一个学生所有已经学习过并且出分的科目，并将对应的分数转化为绩点并进行计算。

实验与测试

依赖

> python环境中需要包含：
>
> asgiref 3.8.1
> cffi 1.16.0
> cryptography 42.0.7
> Django 4.2.13
> mysqlclient 2.2.4
> pillow 10.3.0
> pip 22.3.1
> pycparser 2.22
> PyMySQL 1.1.1
> setuptools 65.5.1
> sqlparse 0.5.0
> typing_extensions 4.11.0
> tzdata 2024.1
> wheel 0.38.4
>
> python所需的外部库已经生成了对应的虚拟环境，下载之后即可使用

> 前端渲染需要依赖boostrap5
>
> 所需的外部库已经做好了引入，无需手动添加

部署

> 代码运行只需要打开src/python/StuManagementSystem目录下，执行指令：python manage.py runserver
>
> 执行成功之后访问网址http://127.0.0.1:8000/便可以查看生成的网页

实验结果

PS以下的所有操作为有序的操作序列，是从头到尾一步一步的演示

首先我们运行启动部署的指令，在数据库后端执行插入测试样例的相关SQL语句后

首先看到的是登陆界面

我们首先以PB21111681的用户身份，12345^&*()为密码登陆用户，可以看到我们的个人信息界面：

在选择完上传新头像之后，可以看到头像会发生变化：

在点击专业情况之后，可以看到所有专业的情况：

在选课情况界面，我们可以增加几个新的选课：

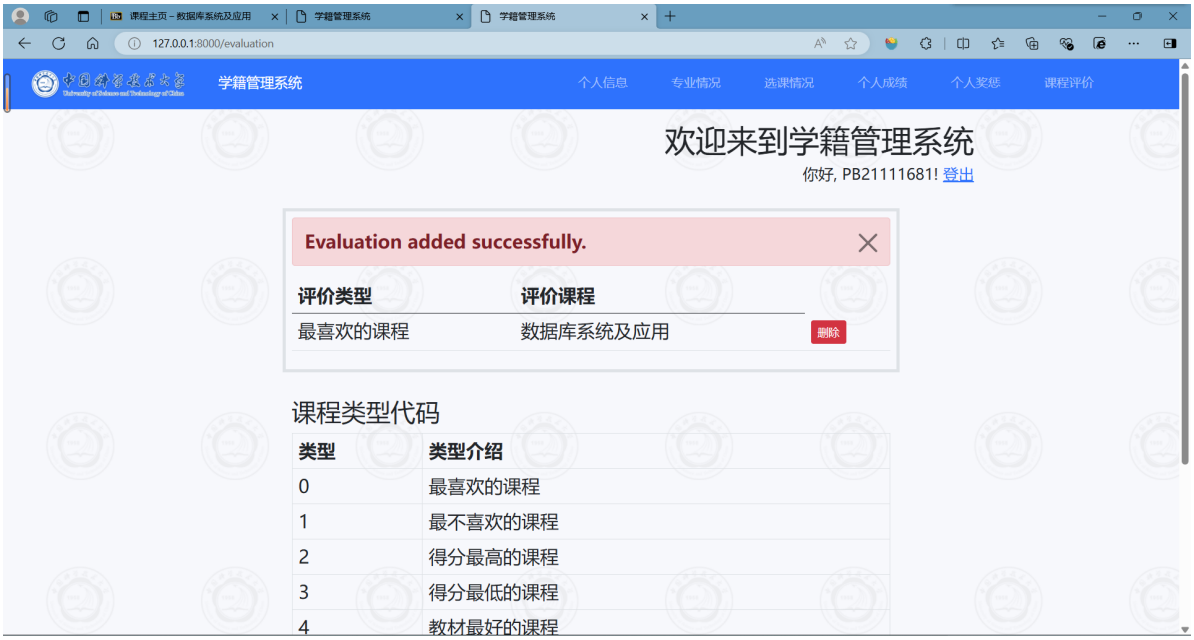
初始情况：

新选完课之后的情况：

在个人成绩界面，由于我们新选的课还没有给分，所以还不会加入到GPA的计算中来，同时由于GPA属于个人隐私的一部分，所以我们需要将鼠标悬浮在已隐藏上才会显示出我们的GPA

然后展示的是课程评价界面，我们可以新增一门最喜欢的课，比如数据库：

新增后

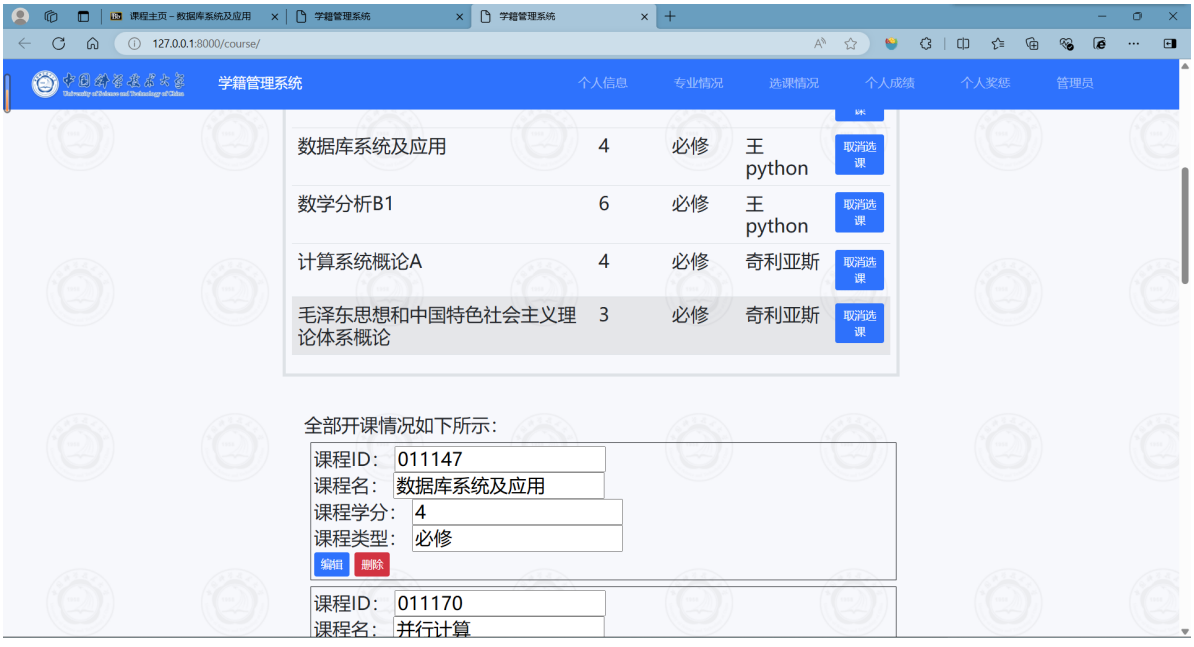


然后我们登陆管理员账号，我这里为6，密码为123456

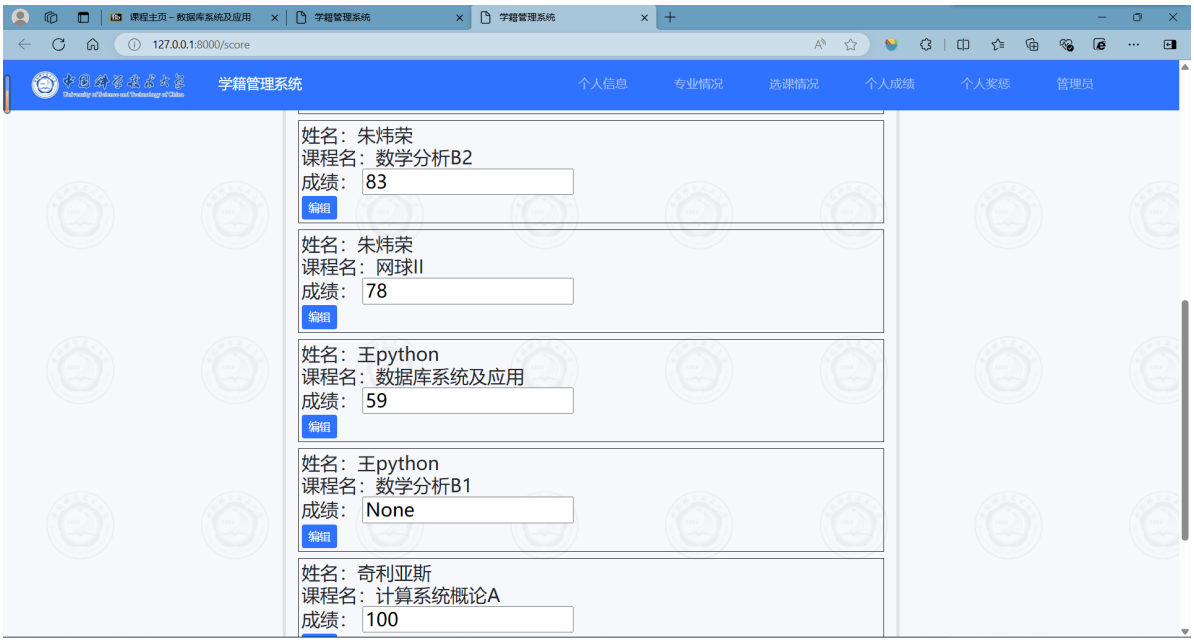
展示界面为：

在这个界面我们可以修改增加删除学生的各种信息。

来到选课情况，我们可以看到所有的开课情况和学生选课情况，修改的方法与学生信息的修改方式类似

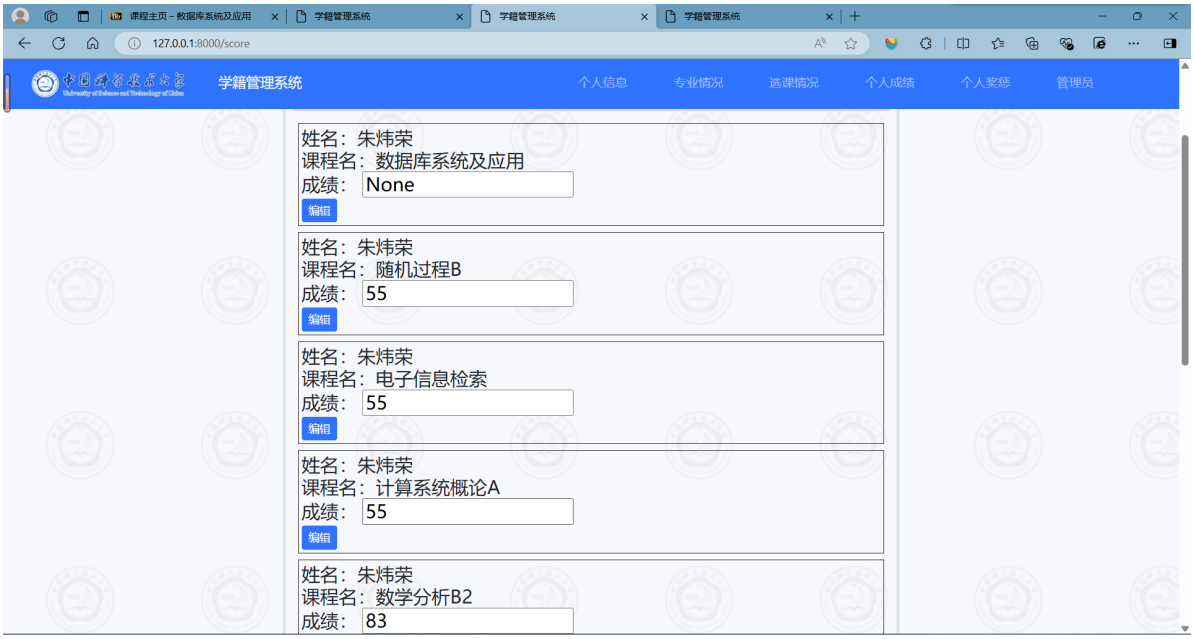


而在个人成绩方面我们就可以给学生的各种选课进行成绩赋予，如下所示



然后是个人奖惩展示：

如果我们将朱炜荣的三科任意成绩修改为不及格，那么我们就能看到触发器被成功调用，增加了朱炜荣相关的警告的挂科现象：



最后我们检验删除专业的相关删除，我们删除掉自动化专业

这时我们再浏览任何一个分页面，朱炜荣的相关信息也跟着一起消失了，例如奖惩：

参考

> 前端中的导航栏参考了本人在2022年IGEM比赛中团队使用网页前端的导航栏。