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：张延

学生学号：PB21111698

完成时间：2024年6月5日

需求分析

1.学生管理

- 学生信息包括学号（primary）、姓名、密码、头像、性别、所属专业、奖项、惩罚、学业状态、重修机会、生日和入学时间。其中学业状态取决于挂科次数，大于等于3次学业状态为“警告”，小于3次为“学业正常”，设计触发器在每次更新选课表时更新学业状态。重修机会初始默认为两次，放弃成绩或退课会使用重修机会，对已修课程进行重修暂时不会使用重修机会，直到更新成绩时会将其减一。
- 管理员能够对学生信息进行增加、删除、修改和查询操作。其中包含了对学生照片也即图像的管理，具体实现为在数据库中存储图像在本地服务器的地址，而图像实际存储在指定文件夹下，在删除学生信息时需先删除文件夹下的对应图片。对学生信息的修改可能会需要修改相应的学生选课信息，比如当修改学生学号或删除学生信息时。
- 学生可以查看自己的基本信息，包括头像、个人资料，成绩以及选课信息等。

2.课程管理

- 课程信息包括课程编号（primary）、课程名称、教师、学分、所属专业和学时。
- 管理员能够对课程进行增加、删除、修改和查询操作。同样，当修改课程编号或删除课程信息时会需要修改相应的学生选课信息。
- 学生也能查看所有的课程信息，但是分为两部分，一部分是已选课程，另一部分是未选课程，这是为了方便后续的选课和退课操作。

3.选课管理

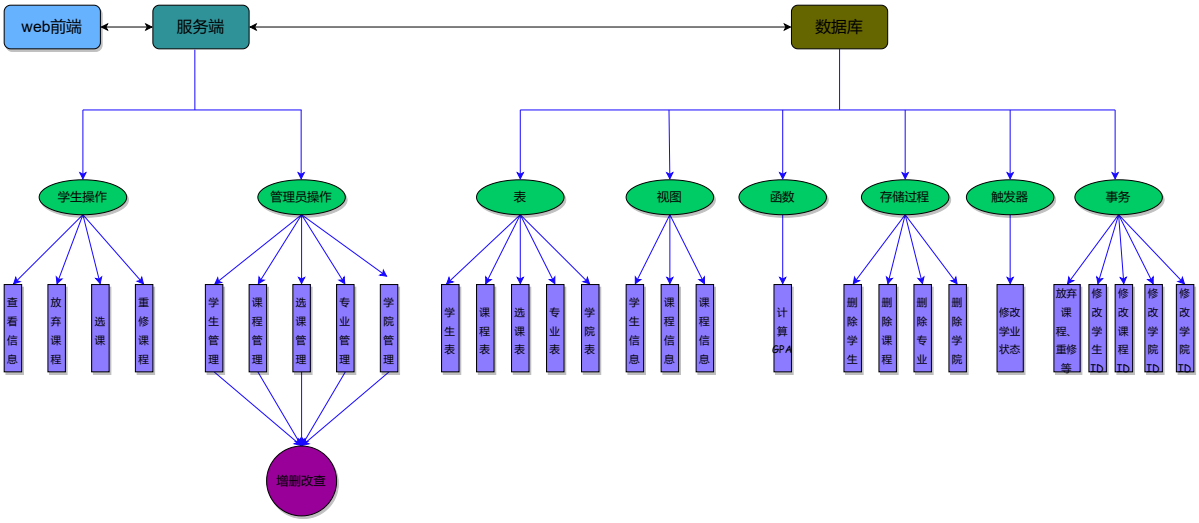
- 选课信息包括（课程编号，学号）（primary），成绩（可以为空），以及该学生是否正在重修该课程。
- 管理员能够对选课信息进行增加、删除、修改和查询操作。其中修改操作只能针对成绩和是否正在重修两项进行，因为课程编号和学号依赖于课程信息和学生信息，若要修改需要在课程管理和学生管理中修改。
- 学生能够登陆查看和修改自己的课程信息，包括课程成绩和选课信息。修改主要体现为能够放弃课程，包括已有成绩和未有成绩的课程，两种情况均需使用重修机会；可以重修课程，此操作会将选课信息的是否正在重修该课程置为true，且暂时不会使用重修机会直到更新成绩；此外还能进行选课，即能选择该学生未在选课信息中出现过的课程，退课和选课成功会增删相应的选课信息。

4.其他

- 除了上述三种信息外，系统还包括了专业信息和学院信息，并支持了对二者的增删改查，删除时同样会删除相应的学生、课程与选课信息。
- 需要能够计算学生的平均成绩进而得出其gpa（算术平均），并显示在学生信息的视图中。
- 在修改表中的外键时需要使用“事务”以保持信息的一致。而删除外键时可以不使用，因为只要按照相应的参照关系的顺序删除就不会影响数据的完整性和一致性。

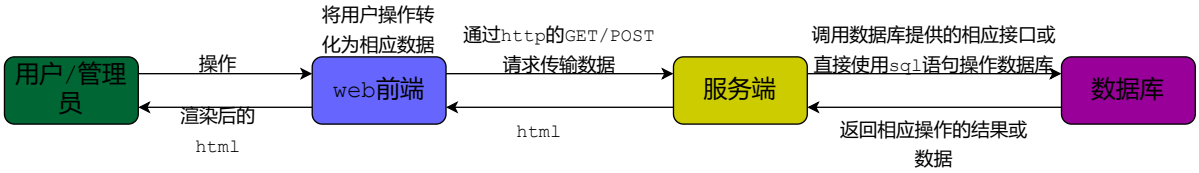
总体设计

系统模块结构



- web前端实现与用户的交互界面，提供相关操作接口。
- 服务端获取用户的相关操作信息并作出反应，通过与数据库交互实现上图中学生操作和管理员操作下面的共九个相关模块，修改数据库中的五个基本表。
- 数据库实现视图、函数、存储过程、触发器、事务等以提供接口给服务端，是之能够对数据库进行增删改查。

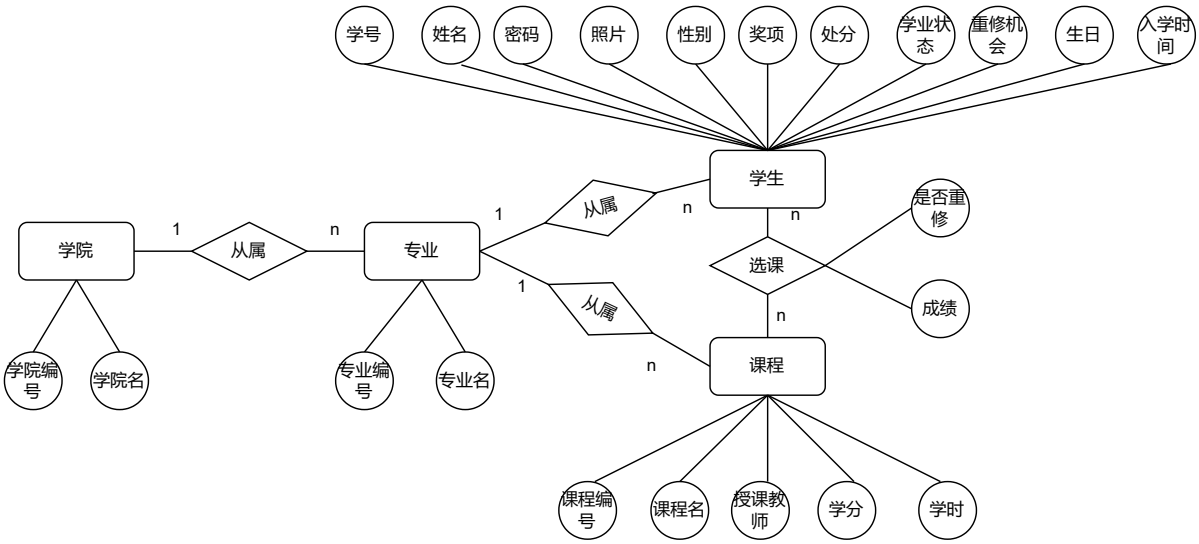
系统工作流程



- 用户或管理员通过web前端提供的结构进行操作。
- web前端获取用户或管理员的操作信息并转化为相应数据，通过http的GET或POST请求将数据传输到服务端。
- 服务端根据从前端接收到的数据调用数据库接口或直接使用sql语句操作数据库。
- 数据库返回相应操作的结果或数据。
- 服务端接收到数据库的数据，生成相应html文件交给web前端。
- 前端返回渲染后的html文件给用户或管理员。

数据库设计

ER图



关系模式任意实体的任意属性均唯一且不可再分，满足1NF；任意非主属性都完全依赖于主码，满足2NF；任意非主属性均不传递依赖于主码，满足3NF。综上关系模式范式为3NF。

建表（代码均位于[mysite/app/models.py](mysite/app/models.py)）

- 使用django建表，运行`python manage.py makemigrations`和`python manage.py migrate`迁移到数据库。以学生表为例。

```
class Stu(models.Model):
    id = models.CharField(verbose_name='学号',max_length=10, primary_key=True)
    password = models.CharField(verbose_name='密码',max_length=16, null=False)
    sname = models.CharField(verbose_name='姓名',max_length=32, null=False)
    img = models.ImageField(verbose_name='头像',max_length=32, null=True, upload_to='photos')
    gender_choices = (
        (True, "男"),
        (False, "女"),
    )
    gender = models.BooleanField(verbose_name='性别',default=True, choices=gender_choices)
    major = models.ForeignKey(to="major", to_field="id", on_delete=models.DO_NOTHING)
    prize = models.CharField(verbose_name='奖项',max_length=256, null=True, default='')
    punishment = models.CharField(verbose_name='惩罚',max_length=256, null=True,default='')
    status_choices = (
        (0, "学业正常"),
        (1, "警告"),
        (2, "已毕业"),
    )
    status = models.SmallIntegerField(verbose_name='学业状态',default=0, choices=status_choices, null=False)
    retake_chances = models.SmallIntegerField(verbose_name='重修机会',default=2, null=False)
    birthday = models.DateField(verbose_name='生日',null=False)
```

```
    data_in = models.DateField(verbose_name='入学时间',null=False)
```

视图（代码均位于mysite/app/models.py和lab2/db/view.sql）

- 在MySQL中建立视图，以便在查询时返回完整信息。同时在django中建立相应视图映射到MySQL中的视图。以学生信息为例

```
-- lab2/db/view.sql
drop view if exists stu_info;
create view stu_info as
    (select app_stu.id, password, img, sname as name, gender, prize, punishment, status, retake_chances, birthday,
    data_in, mname as major, dname as department, compute_gpa(app_stu.id) as avg_gpa
    from app_stu, app_major, app_department
    where major_id = app_major.id and department_id = app_department.id order by app_stu.id);
```

```
# mysite/app/models.py
class stu_info(models.Model):
    id = models.CharField(verbose_name='学号',max_length=10, primary_key=True)
    password = models.CharField(verbose_name='密码',max_length=16, null=False)
    img = models.ImageField(verbose_name='照片',max_length=32, null=True, upload_to='photos')
    name = models.CharField(verbose_name='姓名',max_length=32, null=False)
    gender = models.BooleanField(verbose_name='性别',default=True)
    prize = models.CharField(verbose_name='奖项',max_length=256, null=True)
    punishment = models.CharField(verbose_name='惩罚',max_length=256, null=True)
    status = models.SmallIntegerField(verbose_name='学业状态',default=0, null=False)
    retake_chances = models.SmallIntegerField(verbose_name='重修机会',default=2, null=False)
    birthday = models.DateField(verbose_name='生日',null=False)
    data_in = models.DateField(verbose_name='入学时间',null=False)
    major = models.CharField(verbose_name='专业',max_length=64, null=False)
    department = models.CharField(verbose_name='学院',max_length=64, null=False)
    avg_gpa = models.FloatField(verbose_name='绩点',null=True)
    class Meta:
        managed = False
        db_table = 'stu_info'
        verbose_name = '学生信息'
```

存储过程（代码均位于lab2/db/procedure.sql）

- 删除学生信息和课程信息。均需先删除选课表中的相应选课信息。

```
DELIMITER //
DROP procedure IF EXISTS delete_stu//
```

```
CREATE procedure delete_stu(
    in sid varchar(10)
)
BEGIN
    delete from app_sl where stu_id = sid;
    delete from app_stu where id = sid;
END //
DELIMITER ;

DELIMITER //
DROP procedure IF EXISTS delete_les//
CREATE procedure delete_les(
    in lid varchar(10)
)
BEGIN
    delete from app_sl where lesson_id = lid;
    delete from app_lesson where id = lid;
END //
DELIMITER ;
```

- 删除专业信息。先根据专业id获取隶属与该专业的学生和课程id，再据此调用删除学生信息和课程信息的存储过程，最后删除该专业。

```
DELIMITER //
DROP PROCEDURE IF EXISTS delete_major//
CREATE PROCEDURE delete_major(
    IN mid VARCHAR(10)
)
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE student_id VARCHAR(10);
    DECLARE lesson_id VARCHAR(10);

    -- 声明一个游标，用于获取特定 major_id 的学生 id
    DECLARE cur_stu CURSOR FOR
        SELECT id FROM app_stu WHERE major_id = mid;

    -- 声明一个游标，用于获取特定 major_id 的课程 id
    DECLARE cur_les CURSOR FOR
        SELECT id FROM app_lesson WHERE major_id = mid;

    -- 定义一个继续处理游标的条件
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur_stu;
    read_student_loop: LOOP
        FETCH cur_stu INTO student_id;
        IF done THEN
            LEAVE read_student_loop; -- 如果学生游标没有更多数据，则跳出循环
        END IF;
        CALL delete_stu(student_id); -- 调用删除学生记录的存储过程
    END LOOP;

    CLOSE cur_stu; -- 关闭学生游标

```

```
    -- 重置 done 变量，用于处理课程游标
    SET done = 0;

    OPEN cur_les;
    read_lesson_loop: LOOP
        FETCH cur_les INTO lesson_id;
        IF done THEN
            LEAVE read_lesson_loop; -- 如果课程游标没有更多数据，则跳出循环
        END IF;
        CALL delete_les(lesson_id); -- 调用删除课程记录的存储过程
    END LOOP;

    CLOSE cur_les; -- 关闭课程游标
    delete from app_major where id = mid;
END //

DELIMITER ;
```

- 删除学院信息。先根据学院id获取隶属于学院的专业信息，再据此调用删除专业的存储过程，最后删除该学院。

```
DELIMITER //
DROP PROCEDURE IF EXISTS delete_dep//
CREATE PROCEDURE delete_dep(
    IN did VARCHAR(10)
)
begin
    DECLARE done INT DEFAULT 0;
    DECLARE major_id VARCHAR(10);
    DECLARE cur_major CURSOR FOR
        SELECT id FROM app_major WHERE department_id = did;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN cur_major;
    read_loop: LOOP
        FETCH cur_major INTO major_id;
        IF done THEN
            LEAVE read_loop;
        END IF;
        CALL delete_major(major_id);
    END LOOP;
    CLOSE cur_major; -- 关闭课程游标
    delete from app_department where id = did;
end//
DELIMITER ;
```

函数（代码位于lab2/db/function）

- 计算学生gpa。根据学生选课表中的课程成绩，计算算数平均值，再根据对应区间输出其gpa。

触发器（代码位于lab2/db/trigger.sql）

- 增删改选课表各自需要一个触发器，选课表更新时触发，统计相应学生不及格次数，设置相应学业状态。以删除为例，增、改类似。

```
DELIMITER //
```

```
CREATE TRIGGER sl_delete
AFTER delete ON app_sl
FOR EACH ROW
BEGIN
    declare fail_cnt int default 0;
    select count(*) into fail_cnt from app_sl where stu_id = old.stu_id and
grade < 60;
    if fail_cnt >= 3 then
        update app_stu set status = 1 where id = old.stu_id;
    else
        update app_stu set status = 0 where id = old.stu_id;
    end if;
END;
//
DELIMITER ;
```

事务（代码位于[lab2/db/transaction.sql](lab2/db/transaction.sql)）

- 管理重修与放弃课程。is_quit为true表示时是放弃课程，反之则是重修更新成绩操作。首先查询学生此课程的成绩和是否正在重修，再查询正在重修的课程数量和剩余重修机会。若is_quit为true，则放弃成绩从选课表中删除相应信息；否则若新成绩更高，则更新成绩并设置选课信息的重修标志（不论之前是否正在重修，更新成绩后重修标志都应为false）。最后，若重修机会大于正在重修的课程数，或者二者相等，但当前课程就是一门正在重修的课程，则commit，并将state设为0，否则rollback，将state设为-1000。

```
DELIMITER //
DROP procedure IF EXISTS retake//
CREATE procedure retake(
    in sid varchar(10),
    in lid varchar(10),
    in is_quit bool,
    in new_grade smallint
)
BEGIN
    declare old_grade, cnt, chances, isretaking, retaking_cnt int default 0;
    select grade, count(*), is_retaking into old_grade, cnt, isretaking from
app_sl
    where sid = stu_id and lid = lesson_id group by grade, is_retaking;

    select count(*) into retaking_cnt from app_sl where sid = stu_id and
is_retaking = 1;

    select retake_chances into chances from app_stu where sid = id;
    start transaction;
    if cnt > 0 then
        if is_quit then
            delete from app_sl where sid = stu_id and lid = lesson_id;
        elseif new_grade > old_grade and isretaking = 1 then
            update app_sl set grade = new_grade, is_retaking = 0 where sid =
stu_id and lid = lesson_id;
        elseif new_grade > old_grade then
            update app_sl set grade = new_grade where sid = stu_id and lid =
lesson_id;
        end if;
    end if;
```

```
    if ((chances > retaking_cnt) or (chances = retaking_cnt and isretaking))
 and cnt > 0 then
        update app_stu set retake_chances = retake_chances - 1 where sid =
id;
        set @state = 0;
        commit;
    else
        set @state = -1000;
        rollback;
    end if;
END //
DELIMITER ;
```

- 修改学生id等被参照的键时需使用事务同时修改相应参照表中的外键，且需要先关闭外键检查后，修改完成后开启。以修改学生id为例，修改课程，专业，学院id类似。

```
DELIMITER //
DROP procedure IF EXISTS edit_stu_id//
CREATE procedure edit_stu_id(
    in old_id varchar(10),
    in new_id varchar(10)
)
BEGIN
    start transaction;
    SET FOREIGN_KEY_CHECKS = 0;
    update app_stu set id = new_id where id = old_id;
    update app_sl set stu_id = new_id where stu_id = old_id;
    SET FOREIGN_KEY_CHECKS = 1;
    set @state = 0;
    commit;
END //
DELIMITER ;
```

核心代码解析

仓库地址

https://github.com/rpbxszhc/database-lab/tree/master/lab2

目录（tree.txt）

```
卷 Windows 的文件夹 PATH 列表
卷序列号为 780E-653F
C:.                                                                    ---
----根目录
|   2_db-lab02.pptx                                                    ---
----实验文档
|   e-r.drawio                                                         ---
----e-r作图文件
|   e-r.svg                                                            ---
----e-r矢量图
|   report.md                                                          ---
----实验报告markdown文件
```

```
|   report.pdf                                                    ----实验报告pdf
|   tree.txt                                                      ----文件结构
|   ~$2_db-lab02.pptx
|   学籍管理系统.pdf                                               ----需求分析和e-r图pdf
|   系统工作流程.drawio                                            ----系统工作流程作图文件
|   系统工作流程.svg                                               ----系统工作流程矢量图
|   系统模块结构.drawio                                            ----系统模块结构作图文件
|   系统模块结构.svg                                               ----系统模块结构矢量图
|   需求分析和e-r.md                                               ----需求分析和e-r图markdown文件
|
+---img                                                           ----实验报告引用的图片，主要是实验结果
|       dep_manage_info.png
|       filter_grade.png
|       les_manage_info.png
|       login.png
|       logo.png
|       major_manage_info.png
|       quit_les.png
|       quit_les_grade.png
|       quit_les_retake_cnt.png
|       retake_les.png
|       retake_les_grade.png
|       retake_les_retake.png
|       select_les.png
|       select_les_grade.png
|       select_les_select.png
|       stules_manage_post_edit_fail.png
|       stules_manage_post_edit_ing.png
|       stules_manage_post_edit_success.png
|       stules_manage_pre_edit.png
|       stules_manage_status_info.png
|       stules_manage_status_stules.png
|       stu_manage_create_ing.png
|       stu_manage_delete_ing.png
|       stu_manage_info.png
|       stu_manage_post_create_img.png
|       stu_manage_post_create_info.png
|       stu_manage_post_delete_info.png
|       stu_manage_post_delete_stules.png
|       stu_manage_post_edit_info.png
|       stu_manage_pre_delete_stules.png
|       user_info.png
|
+---lab2                                                          ----workbench导入的相关sql语句
|   \---db
```

```
|           create_table.sql                                          ----无用
|           datainsert.sql                                            ----插入实验测试数据
|           function.sql                                              ----所有sql函数
|           procedure.sql                                             ----所有存储过程
|           tmp.sql                                                   ----在workbench中临时调用的sql语句，无用
|           transaction.sql                                           ----所有事务
|           trigger.sql                                               ----所有触发器
|           view.sql                                                  ----所有视图
|
\---mysite                                                            ----django项目文件夹
    |   format.txt                                                    ----项目实现流程
    |   manage.py
    |
    +---.idea
    |       .gitignore
    |       modules.xml
    |       mysite.iml
    |       vcs.xml
    |       workspace.xml
    |
    +---app
    |   |   admin.py
    |   |   apps.py
    |   |   models.py                                                 ----包含所有数据库表与视图
    |   |   tests.py
    |   |   views.py                                                  ----所有后端逻辑实现
    |   |   __init__.py
    |   |
    |   +---migrations                                                ----实验过程中在django中对数据库的修改
    |   |   |   0001_initial.py
    |   |   |   0002_department_major_stu_lesson_sl.py
    |   |   |   0003_sl_info_alter_les_info_table.py
    |   |   |   0004_alter_stu_img.py
    |   |   |   __init__.py
    |   |   |
    |   |   \---__pycache__
    |   |           0001_initial.cpython-39.pyc
    |   |           0002_department_major_stu_lesson_sl.cpython-39.pyc
    |   |           0003_sl_info_alter_les_info_table.cpython-39.pyc
    |   |           0004_alter_stu_img.cpython-39.pyc
    |   |           __init__.cpython-39.pyc
    |   |
```

```
    |   +---templates                                                 ---
----前端html文件
    |   |       department.html
    |   |       lesson.html
    |   |       login.html
    |   |       major.html
    |   |       root.html
    |   |       stules.html
    |   |       user_info.html
    |   |
    |   \---__pycache__
    |           admin.cpython-39.pyc
    |           apps.cpython-39.pyc
    |           models.cpython-39.pyc
    |           views.cpython-39.pyc
    |           __init__.cpython-39.pyc
    |
    +---media                                                         ---
----用于图片管理的文件，向此文件夹插入图片
    |   \---photos
    +---mysite
    |   |   asgi.py
    |   |   settings.py                                             ---
----django配置文件
    |   |   urls.py                                                 ---
----前端url与后端view中函数的映射文件
    |   |   wsgi.py
    |   |   __init__.py
    |   |
    |   \---__pycache__
    |           settings.cpython-39.pyc
    |           urls.cpython-39.pyc
    |           wsgi.cpython-39.pyc
    |           __init__.cpython-39.pyc
    |
    \---testimg                                                       ---
----提供测试图片管理的图片
            boy.jpg
            girl.jpg
```

实验代码

url与视图函数对应关系（mysite/mysite/url.py）

最后static部分是为了存储图片到本地添加的路径。

```
urlpatterns = [
    # path("admin/", admin.site.urls),
    path('login/', login),
    path('userinfo/', user_info),
    path('root/', root),
    path('lesson/', les),
    path('stules/', stules),
    path('major/', mjr),
    path('dep/', dep),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

登陆界面（前后端代码分别位于mysite/app/templates/login.html和mysite/app/view.py）

- 用户登录。登录url为login/，对应login.html文件，获取用户提交的账户密码，提交到user_info函数中校验，错误则重定向到login/，并返回错误信息request.session['error_msg']，否则获取用户信息并返回user_info.html文件

```
# mysite/app/view.py
def login(request):
    if 'error_msg' in request.session:
        error_msg = request.session['error_msg']
        del request.session['error_msg']
        return render(request, "login.html", {"error_msg":error_msg, 'title':'用户', 'action':'/userinfo/'})
    else:
        return render(request, "login.html", {'title':'用户', 'action':'/userinfo/'})


def user_info(request):
    if request.method == "GET":
        return redirect('/login/')

    """
    略去无关逻辑
    """

    id = request.POST.get("id")
    password = request.POST.get("password")
    valid = len(Stu.objects.filter(id = id, password = password))
    if not valid:
        request.session['error_msg']='用户名或密码错误'
        return redirect('/login/')
    sinfo = stu_info.objects.filter(id = id).first()
    ginfo = grade_info.objects.filter(id = id)
    with connection.cursor() as cursor:
        cursor.execute(sql,[id])
        linfo = cursor.fetchall()
        cursor.execute("select * from grade_info where id = %s and grade is not null and is_retaking = 0",[id])
        rinfo = cursor.fetchall()
```

```
    return render(request, "user_info.html", {"sinfo":sinfo, "ginfo":ginfo, "linfo":linfo, "rinfo":rinfo})
```

- 管理员登陆。登录url为root/，对应login.html文件，获取管理员提交的账户密码，提交到root函数中校验，错误则返回login.html文件，并返回错误信息error_msg，否则获取所有用户信息并返回root.html文件

```
# mysite/app/view.py
def root(request):
    if request.method == 'GET':
        return render(request, "login.html", {'title':'管理员', 'action':'/root/'})
    id = request.POST.get("id")
    password = request.POST.get("password")
    emptyform = editForm()
    """
    略去无关逻辑
    """
    if(id == "root" and password == "123456"):
        info = stu_info.objects.all()
        return render(request, 'root.html',{'info':info, 'form':emptyform})
    else:
        return render(request, "login.html", {'title':'管理员', 'action':'/root/', 'error_msg':'用户名或密码错误'})
```

- 登录页面login.html。使用form表单提交post数据，此外，页面内还有django后端传入的form表单的提交地址action以及可能的错误信息

```
<!--mysite/app/templates/login.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{title}}登录</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .account{
            width: 400px;
            border: 1px solid #dddddd;
            border-radius: 5px;
            box-shadow: 5 5 20px #aaa;
            margin-top: 100px;
            margin-left: auto;
            margin-right: auto;
            padding: 20px 40px;
        }
        .account h2{
            text-align: center;
            margin-top: 10px;
        }
    </style>
</head>
```

```
<body>
    <div class="account">
    <h2>{{title}}登录</h2>
        <form method = "post" action = {{action}}>
            {% comment %} # csrf_token验证 {% endcomment %}
            {% csrf_token %}
            <div class="form-group">
                <label>学号：</label>
                <input type="text" class="form-control" placeholder="id"
aria-label="id" aria-describedby="basic-addon1" name="id">
            </div>
            <div class="form-group">
                <label>密码：</label>
                <input type="password" class="form-control"
placeholder="password" aria-label="password" aria-describedby="basic-addon1"
name="password">
            </div>
            <input type="submit" value="登录" class = "btn btn-primary">
            <span style="color:red">{{error_msg}}</span>
        </form>
    <div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

用户界面（前后端代码分别位于mysite/app/templates/userinfo.html和mysite/app/view.py）

其余界面功能和实现逻辑均与用户界面类似，故完整讲解用户界面之后，将不再完整地叙述逻辑，只展示部分细节

- 后端逻辑（mysite/app/view.py）
    - retake（重修课程）。只能选修已有成绩且不是正在重修，即is_retaking=0的课程。需要将用户选择的课程在选课表中的重修标志is_retaking置为1，前提是要满足正在重修课程不能超过剩余重修机会。
    - delete（放弃课程）。可以放弃所有已选课程，即选课表中有的课程，无论是否有成绩。主要调用前已实现的retake事务，此处均将输入参is_quit用以表示放弃课程。放弃成功则从选课表中删除该课程。
    - select（选课）。选择未出现在选课表中的课程，选课成功则在选课表中插入一条信息。
    - 最后均需从数据库中获取最新数据传递到userinfo.html文件中

```
# mysite/app/view.py
def user_info(request):
    if request.method == "GET":
        return redirect('/login/')
    if request.POST.get("action") == "retake":
        with connection.cursor() as cursor:
            cursor.execute("select retake_chances from app_stu where id =
%s", [request.POST.get("retake_sid")])
            retake_chances = cursor.fetchone()[0]
```

```
            cursor.execute("select count(*) from app_sl where stu_id = %s and is_retaking = 1", [request.POST.get("retake_sid")])
            retaking = cursor.fetchone()[0]
            if retake_chances <= retaking:
                msg = "重修机会不足"
            else:
                cursor.execute("update app_sl set is_retaking = 1 where stu_id = %s and lesson_id = %s", [request.POST.get("retake_sid"), request.POST.get("retake_lid")])
                msg = "重修成功"
            sinfo = stu_info.objects.filter(id = request.POST.get("retake_sid")).first()
            ginfo = grade_info.objects.filter(id = request.POST.get("retake_sid"))
            cursor.execute(sql,[request.POST.get("retake_sid")])
            linfo = cursor.fetchall()
            cursor.execute("select * from grade_info where id = %s and grade is not null and is_retaking = 0", [request.POST.get("retake_sid")])
            rinfo = cursor.fetchall()
            return render(request, "user_info.html", {"sinfo":sinfo, "ginfo":ginfo, "linfo":linfo, "rinfo":rinfo, "msg":msg})
    if request.POST.get("action") == "delete":
        with connection.cursor() as cursor:
            cursor.callproc('retake', [request.POST.get("delete_sid"), request.POST.get("delete_lid"), 1, 0])
            cursor.execute("SELECT @state;")
            state = cursor.fetchone()[0]
            sinfo = stu_info.objects.filter(id = request.POST.get("delete_sid")).first()
            ginfo = grade_info.objects.filter(id = request.POST.get("delete_sid"))
            cursor.execute(sql,[request.POST.get("delete_sid")])
            linfo = cursor.fetchall()
            cursor.execute("select * from grade_info where id = %s and grade is not null and is_retaking = 0", [request.POST.get("delete_sid")])
            rinfo = cursor.fetchall()
            if state == 0:
                return render(request, "user_info.html", {"sinfo":sinfo, "ginfo":ginfo, "linfo":linfo, "rinfo":rinfo, "msg":"放弃课程成功"})
            elif state == -1000:
                return render(request, "user_info.html", {"sinfo":sinfo, "ginfo":ginfo, "linfo":linfo, "rinfo":rinfo, "msg":"重修机会不足"})
            else:
                return render(request, "user_info.html", {"sinfo":sinfo, "ginfo":ginfo, "linfo":linfo, "rinfo":rinfo, "msg":"意料之外的错误"})
    if request.POST.get("action") == "select":
        with connection.cursor() as cursor:
            try:
                cursor.execute('insert into app_sl(stu_id, lesson_id, is_retaking, grade) values(%s, %s, 0, null)', [request.POST.get("select_sid"), request.POST.get("select_lid")])
                msg = '选课成功'
            except Exception as e:
                msg = '选课失败'
            sinfo = stu_info.objects.filter(id = request.POST.get("select_sid")).first()
```

```
            ginfo = grade_info.objects.filter(id = request.POST.get("select_sid"))
            cursor.execute(sql,[request.POST.get("select_sid")])
            linfo = cursor.fetchall()
            cursor.execute("select * from grade_info where id = %s and grade is not null and is_retaking = 0", [request.POST.get("select_sid")])
            rinfo = cursor.fetchall()
            return render(request, "user_info.html", {"sinfo":sinfo, "ginfo":ginfo, "linfo":linfo, "rinfo":rinfo, "msg":msg})

"""
略去无关逻辑，此部分登陆界面已覆盖
"""
```

- 前端页面（mysite/app/templates/userinfo.html）
    - 导航部分。此处"我的课程"、"进入选课"、"可重修课程"三个按钮分别对应后端的delete、select、retake各自会唤起相应的模态框，模态框中是form表单用于提交post数据。

```
<!--mysite/app/templates/userinfo.html-->
    <nav class="navbar navbar-expand-lg bg-body-tertiary">
        <div class="container-fluid" style="background-color: lightskyblue ;height: 100px">
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li class="nav-item col-sm-4 offset-sm-2">
                        <a class="btn btn-danger" aria-current="page" href="/login/">退出登录</a>
                    </li>
                    <li class="nav-item col-sm-4 offset-sm-2">
                        <a class="btn btn-secondary" href="/root/">管理员登陆</a>
                    </li>
                    <li class="nav-item col-sm-4 offset-sm-2">
                        <button type="button" class="btn btn-info" data-bs-toggle="modal" data-bs-target="#grade_info_modal">
                            我的课程
                        </button>
                    </li>
                    <li class="nav-item col-sm-4 offset-sm-2">
                        <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#lesson_info_modal">
                            进入选课
                        </button>
                    </li>
                    <li class="nav-item col-sm-4 offset-sm-2">
```

```
                        <button type="button" class="btn
btn-success" data-bs-toggle="modal" data-bs-
target="#retake_info_modal">
                            可重修课程
                        </button>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
```

- 学生信息视图的展示。其中照片，奖项，处分点击相应查看按钮通过jQuery弹出模态框展示相应信息。其中由于照片在数据库中存储的是本地地址，因此需要先根据地址找到对应文件再展示

```
<div class="row account" >
        <table class="table table-striped table-hover table-
bordered border-primary">
            <thead>
                <tr>
                  <th scope="col">学号</th>
                  <th scope="col">密码</th>
                  <th scope="col">照片</th>
                  <th scope="col">姓名</th>
                  <th scope="col">性别</th>
                  <th scope="col">状态</th>
                  <th scope="col">剩余重修机会</th>
                  <th scope="col">生日</th>
                  <th scope="col">入学时间</th>
                  <th scope="col">专业</th>
                  <th scope="col">学院</th>
                  <th scope="col">算术平均gpa</th>
                  <th scope="col">获奖情况</th>
                  <th scope="col">处分情况</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                  <th scope="row">{{sinfo.id}}</th>
                  <td>{{sinfo.password}}</td>
                  <td id = 'img'>
                    <button type="button" class="btn btn-
info btn-sm" onclick='getimg(this, "{{sinfo.img}}");'>
                        查看
                    </button>
                  </td>
                  <td>{{sinfo.name}}</td>
                  <td id = 'gender'>{{sinfo.gender}}</td>
                  <td id = 'status'>{{sinfo.status}}</td>
                  <td>{{sinfo.retake_chances}}</td>
                  <td>{{sinfo.birthday}}</td>
                  <td>{{sinfo.data_in}}</td>
                  <td>{{sinfo.major}}</td>
                  <td>{{sinfo.department}}</td>
                  <td>{{sinfo.avg_gpa}}</td>
                  <td>
```

```
                    <button type="button" class="btn btn-info btn-sm" onclick='getinfo(this, "{{sinfo.prize}}");'>
                        查看
                    </button>
                </td>
                <td>
                    <button type="button" class="btn btn-info btn-sm" onclick='getinfo(this, "{{sinfo.punishment}}");'>
                        查看
                    </button>
                </td>
            </tr>
        </tbody>
    </table>
</div>

<!--略去无关逻辑-->

<script>
    function getimg(self, img){
        $('#imgbody').html("<img src='../media/"+img+"' alt='img' class='img-fluid img-thumbnail' style = 'width: 300px; height: 300px'>" );
        $('#imgmodal').modal('show');
    }
</script>
```

- 放弃课程的模态框。其中含有一个隐藏的form表单有用提交学生和课程id，以及相应操作到后端。选课和重修课程模态框逻辑与此相同。三个模态框均有一个筛选框根据用户输入采用文本匹配的方式筛选消息列表中的元组。此外，在放弃课程等操作完成或失败后会弹出模态框提示相应消息。

```
<div class="modal fade modal-lg" id="grade_info_modal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">我的成绩</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                <div class="row" >
                    <table class="table table-striped table-hover table-bordered border-primary">
                        <thead>
                        <tr>
                            <th scope="col">学号</th>
                            <th scope="col">姓名</th>
                            <th scope="col">课程编号</th>
                            <th scope="col">课程名</th>
                            <th scope="col">成绩</th>
```

```
                                <th scope="col">是否正在重修</th>
                                <th scope="col"></th>
                            </tr>
                            </thead>
                            <tbody id = 'tbodygrade'>
                            <input type="text" id="filtergrade" class="form-control" placeholder="输入关键词筛选">
                            {% for i in ginfo %}
                                <tr>
                                    <th scope="row">{{i.id}}</th>
                                    <td>{{i.sname}}</td>
                                    <td>{{i.lid}}</td>
                                    <td>{{i.lname}}</td>
                                    <td>{{i.grade}}</td>
                                    <td>{{i.is_retaking}}</td>
                                    <td>
                                        <form method = 'post' action='/userinfo/'>
                                            {% csrf_token %}
                                            <div class="form-group d-none">
                                                <label>学号:</label>
                                                <input type="text" value = '{{i.id}}' name = 'delete_sid'>
                                            </div>
                                            <div class="form-group d-none">
                                                <label>课程号:</label>
                                                <input type="text" value = '{{i.lid}}' name = 'delete_lid'>
                                            </div>
                                            <div class="form-group d-none">
                                                <label>操作:</label>
                                                <input type="text" value='delete', name = 'action'>
                                            </div>
                                            <input type="submit" value="放弃课程" class = "btn btn-danger">
                                        </form>
                                    </td>
                                </tr>
                            {% endfor %}
                            </tbody>
                        </table>
                    </div>
                </div>
                <div class="modal-footer">
```

```
                    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">关闭</button>
                </div>
            </div>
        </div>
    </div>
```

- 筛选框。根据用户输入采用文本匹配的方式筛选消息列表中的元组，不区分大小写。具体实现为jQuery实时监听筛选框中输入文本，将其转换为小写，与列表中每个元组匹配，匹配成功则显示，否则隐藏。项目中所有table表都有一个对应的筛选框，后不再赘述。

```
<!--模态框中相应输入框-->
<input type="text" id="filtergrade" class="form-control" placeholder="输入关键词筛选">


<!--略去中间无关逻辑-->


<!--jQuery代码-->
<script>
        $(document).ready(function() {
            $('#filtergrade').on('input', function() {
              var filterValue = $(this).val().toLowerCase(); // 获取筛选框的值并转换为小写

              $('#tbodygrade tr').each(function() {
                var text = $(this).text().toLowerCase(); // 获取当前列表项的文本并转换为小写

                if (text.indexOf(filterValue) > -1) {
                  $(this).show(); // 匹配到关键词则显示该列表项
                } else {
                  $(this).hide(); // 不匹配则隐藏该列表项
                }
              });
            });
          });
    </script>
```

管理员界面-学生管理（前后端代码分别位于mysite/app/templates/root.html和mysite/app/view.py）

- 后端逻辑（mysite/app/view.py）
    - 采用Form类校验用户输入是否合法

```
class editForm(forms.Form):
    old_id_ = forms.CharField(label='旧学号', widget=forms.TextInput(attrs={'readonly':'readonly'}), required=False)
    id_ = forms.CharField(label='学号', max_length=10, min_length=10,
                          error_messages={"min_length": "id必须为10位", "required": "该字段不能为空!"},
```

```
                            widget=forms.TextInput(attrs=
{'class':'form-control'}))
    password_ = forms.CharField(label='密码', max_length=16,
min_length=6,
                                    error_messages={"min_length":
"长度必须不能小于6", "max_length": "长度必须不能大于16",
"required": "该字段不能为空!"},

widget=forms.PasswordInput(attrs={'class':'form-control'}))
    img_ = forms.ImageField(label='照片', max_length=32,
required=False,
                            error_messages={"max_length": "输入太
长了"},
                            widget=forms.FileInput(attrs=
{'class':'form-control'}))
    sname_ = forms.CharField(label='姓名', max_length=32,
                               error_messages={"max_length": "名
字太长了", "required": "该字段不能为空!"},
                               widget=forms.TextInput(attrs=
{'class':'form-control'}))
    gender_ = forms.BooleanField(label='性别',
required=False,

 widget=forms.RadioSelect(choices=((True, "男"), (False,
"女"))))
    status_ = forms.IntegerField(label='学业状态',
error_messages={"required": "该字段不能为空!"},
                                 widget=forms.Select(choices=
((0, "学业正常"), (1, "警告"), (2, "已毕业")),attrs=
{'class':'form-control'}))
    retake_chances_ = forms.IntegerField(label='重修机会',
error_messages={"required": "该字段不能为空!"},

 widget=forms.NumberInput(attrs=
{'type':'number','class':'form-control'}))
    birthday_ = forms.DateField(label='生日', error_messages=
{"required": "该字段不能为空!"},
                                widget=forms.DateInput(attrs=
{'type':'date','class':'form-control'}))
    data_in_ = forms.DateField(label='入学时间',
error_messages={"required": "该字段不能为空!"},
                               widget=forms.DateInput(attrs=
{'type':'date','class':'form-control'}))
    major_id_ = forms.CharField(label='专业', max_length=3,
                                error_messages={"max_length":
"长度必须不能大于3","required": "该字段不能为空!"},
                                widget=forms.TextInput(attrs=
{'class':'form-control'}))
    prize_ = forms.CharField(label='奖项', max_length=256,
required=False,
                             error_messages={"max_length": "输
入太长了"},
                             widget=forms.TextInput(attrs=
{'class':'form-control'}))
    punishment_ = forms.CharField(label='惩罚',
max_length=256, required=False,
```

```
                                    error_messages=
{"max_length": "输入太长了"},

widget=forms.TextInput(attrs={'class':'form-control'}))
    action_ = forms.CharField(label='操作', max_length=10,
                              widget=forms.TextInput(attrs=
{'readonly':'readonly'}))
```

- 三种操作：增、删、改（查已实现）。其中增添最简单，确保输入学生id不与已有id冲突即可，并将输入图片存入本地，数据库中存储地址即可。删除主要注意要先删除用户的在本地的照片，再调用删除学生信息的存储过程delete_stu删除选课表中相应信息和学生信息。修改需要主要注意要先删除用户的在本地的照片，再将新的照片存入本地，然后调用修改学生id的事务edit_stu_id修改选课表中相应信息和学生id。下方只展示修改，增删类似可得。

```
form = editForm(request.POST, request.FILES)

if request.POST.get("action_") == "edit":
        if form.is_valid():
            try:
                cleaned_data = form.cleaned_data
                data = cleaned_data
                info =
Stu.objects.filter(id=data['old_id_']).first()
                if info.img.name is not None and
len(info.img.name) > 0 and
os.path.exists('media/'+info.img.name):
                    os.remove('media/'+info.img.name)
                for key in data:
                    if key != 'old_id_' and key != 'action_'
and key != 'id_':
                        info.__dict__[key[0:-1]] = data[key]
                info.__dict__['img'] = data['img_']
                info.save()
                with connection.cursor() as cursor:
                    cursor.callproc('edit_stu_id',
[data['old_id_'], data['id_']])
                info = stu_info.objects.all()
                return render(request, 'root.html',
{'info':info, 'msg':"编辑成功", 'form':form})
            except Exception as e:
                info = stu_info.objects.all()
                return render(request, 'root.html',
{'info':info, 'msg':"编辑失败请检查输入参数与当前数据是否冲突",
'form':form})
        info = stu_info.objects.all()
        return render(request, 'root.html',{'info':info,
'msg':"编辑失败请检查输入参数", 'form':form})
```

- 前端页面（mysite/app/templates/root.html）
  - 导航页面与用户界面结构相同，后者是唤起模态框，此处改为跳转到相应的url。
  - 学生信息展示与用户个人信息的展示相同，后者是一条信息，此处for循环展示所有用户信息

- 增删改对应的模态框结构与用户界面中放弃课程的结构类似，都是form提交需要的数据。

管理员界面-课程管理（前后端代码分别位于mysite/app/templates/lesson.html和mysite/app/view.py）

- 后端逻辑（mysite/app/view.py）
  - 采用Form类校验用户输入是否合法。
  - 三种操作：增、删、改（查已实现）。基本逻辑与学生管理相同，删除和修改课程信息时需调用存储过程delete_les或事务edit_les_id同步操作选课表。
- 前端页面（mysite/app/templates/lesson.html）
  - 导航页面、课程信息、增删改与学生管理基本相同。

管理员界面-选课管理（前后端代码分别位于mysite/app/templates/stules.html和mysite/app/view.py）

不能直接更改选课表中的学生和课程id，再新增时也要检验外键是否合法。其余与学生和课程管理基本相同且更为简单，略去。

管理员界面-专业管理（前后端代码分别位于mysite/app/templates/major.html和mysite/app/view.py）

- 后端逻辑（mysite/app/view.py）
  - 删除时需调用存储过程delete_major同时删除关联的学生、课程、选课信息。且在此之前需要先删除存储在本地的相应学生的照片。
  - 修改时需调用事务edit_major_id同时修改关联的学生、课程信息。
  - 其余前述信息的管理类似，下只展示删除逻辑。其中s[3]中是学生信息中的照片地址。

```
def mjr(request):
    if request.method == "GET":
        return redirect('/root/')
    if request.POST.get("action") == "delete":
        try:
            with connection.cursor() as cursor:
                cursor.execute("select * from app_stu where major_id = %s", [request.POST.get("id")])
                sinfo = cursor.fetchall()
                for s in sinfo:
                    if s[3] is not None and len(s[3]) > 0 and os.path.exists('media/'+s[3]):
                        os.remove('media/'+s[3])
                cursor.callproc('delete_major', [request.POST.get("id")])
            info = major.objects.all()
            return render(request, 'major.html', {'info':info, 'msg':"删除成功", 'form':majorform()})
        except Exception as e:
            info = major.objects.all()
            return render(request, 'major.html', {'info':info, 'msg':"删除失败", 'form':majorform()})
```

- 前端页面（mysite/app/templates/major.html）略去

管理员界面-学院管理（前后端代码分别位于mysite/app/templates/department.html和mysite/app/view.py）

- 后端逻辑（mysite/app/view.py）
    - 删除时需调用存储过程delete_dep同时删除关联的专业、学生、课程、选课信息。且在此之前需要先删除存储在本地的相应学生的照片。
    - 修改时需调用事务edit_dep_id同时修改关联的专业信息。
    - 其余前述信息的管理类似，下只展示删除逻辑。其中s[3]中是学生信息中的照片地址。

```
def dep(request):
    if request.method == "GET":
        return redirect('/root/')
    if request.POST.get('action') == 'delete':
        try:
            with connection.cursor() as cursor:
                cursor.execute("select * from app_major where department_id = %s", [request.POST.get("id")])
                minfo = cursor.fetchall()
                for m in minfo:
                    cursor.execute("select * from app_stu where major_id = %s", [m[0]])
                    sinfo = cursor.fetchall()
                    for s in sinfo:
                        if s[3] is not None and len(s[3]) > 0 and os.path.exists('media/'+s[3]):
                            os.remove('media/'+s[3])
                cursor.callproc('delete_dep', [request.POST.get("id")])
            info = department.objects.all()
            return render(request, 'department.html', {'info':info, 'msg':"删除成功", 'form':depform()})
        except Exception as e:
            info = department.objects.all()
            return render(request, 'department.html', {'info':info, 'msg':"删除失败", 'form':depform()})
```

- 前端页面（mysite/app/templates/department.html）略去

实验与测试

依赖

- django--4.2.13
- python--3.9.18
- bootstrap--5.3.0
- jQuery--3.6.4

部署

```
"""
1.在mysite/mysite/settings.py中修改DATABASES设置与数据库建立关联。设置存储学生照片的文件夹MEDIA_ROOT和MEDIA_URL
"""
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": 'lab2',
        "USER": 'root',
        "PASSWORD": 'XXXXXX',
        'HOST': 'localhost',
        'PORT': '3306'
    }
}
MEDIA_ROOT = os.path.join(BASE_DIR, 'media').replace('\\', '/')  # 设置静态文件路径为主目录下的media文件夹
MEDIA_URL = '/media/'

"""
2.下列命令将mysite/app/models.py中表项迁移到数据库
"""
python manage.py makemigrations
python manage.py migrate

"""
3.运行
"""
python manage.py runserver

"""
4.测试数据插入：lab2/db/datainsert
"""
```

实验结果

按照代码叙述顺序

登陆界面

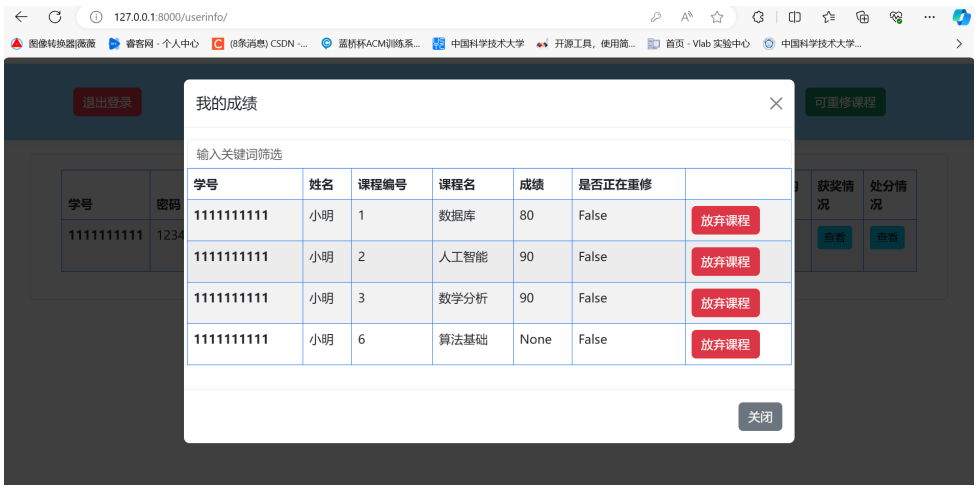
- 用户登录（管理员登陆类似）

用户界面

- 查看信息(此处算术平均gpa调用了求gpa的函数)

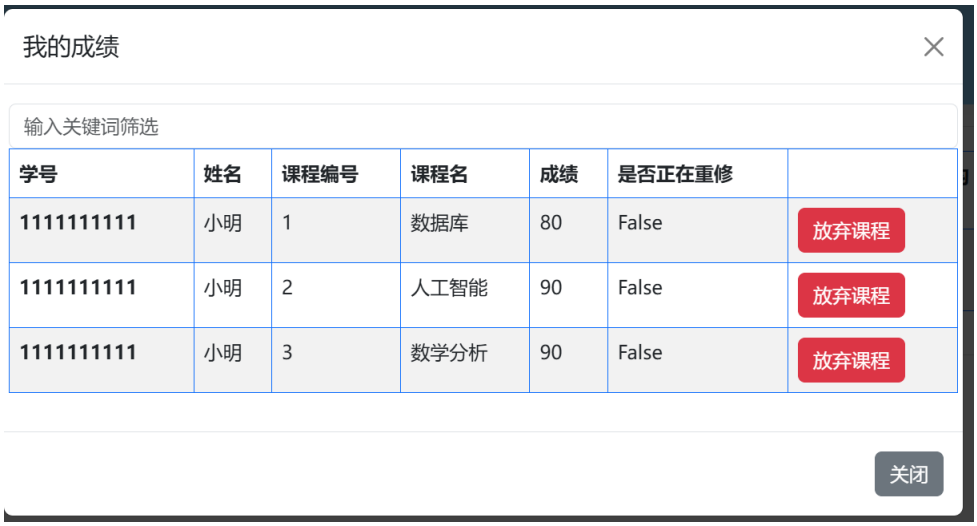
学号	密码	照片	姓名	性别	状态	剩余重修机会	生日	入学时间	专业	学院	算术平均gpa	获奖情况	处分情况
1111111111	123456	查看	小明	男	学业正常	2	Jan. 1, 2003	Sept. 1, 2021	计算机科学与技术	计算机科学与技术学院	3.7	查看	查看

- 我的课程
  - 查看

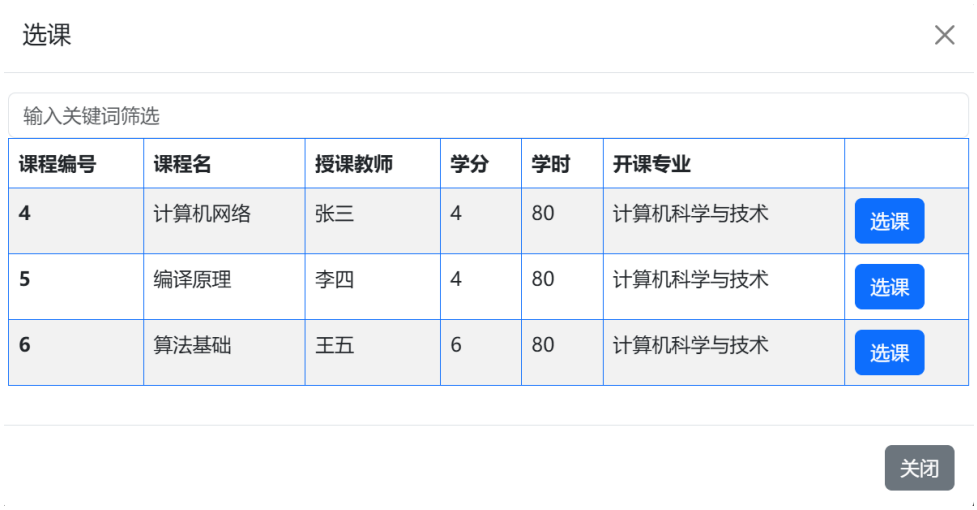


  - 放弃（放弃前重修机会剩余2，后应该为1，我的成绩中放弃算法基础，放弃后算法基础会出现在选课表中）

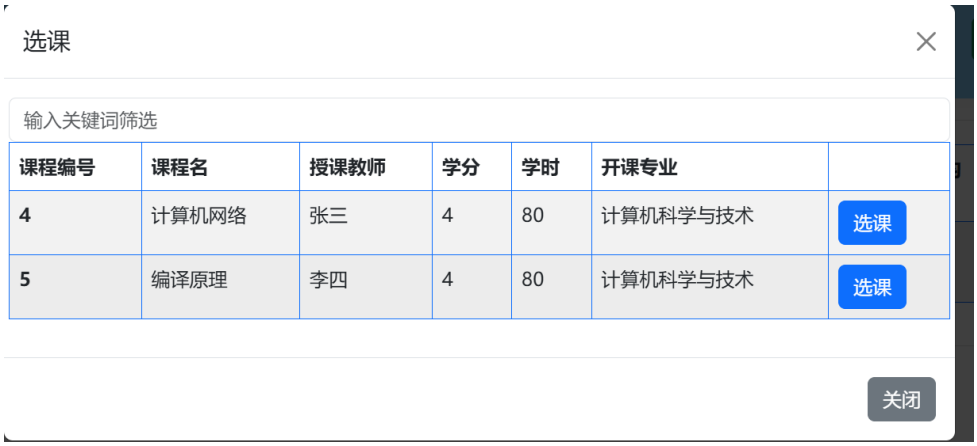
学号	密码	照片	姓名	性别	状态	剩余重修机会	生日	入学时间	专业	学院	算术平均gpa	获奖情况	处分情况
1111111111	123456	查看	小明	男	学业正常	1	Jan. 1, 2003	Sept. 1, 2021	计算机科学与技术	计算机科学与技术学院	3.7	查看	查看

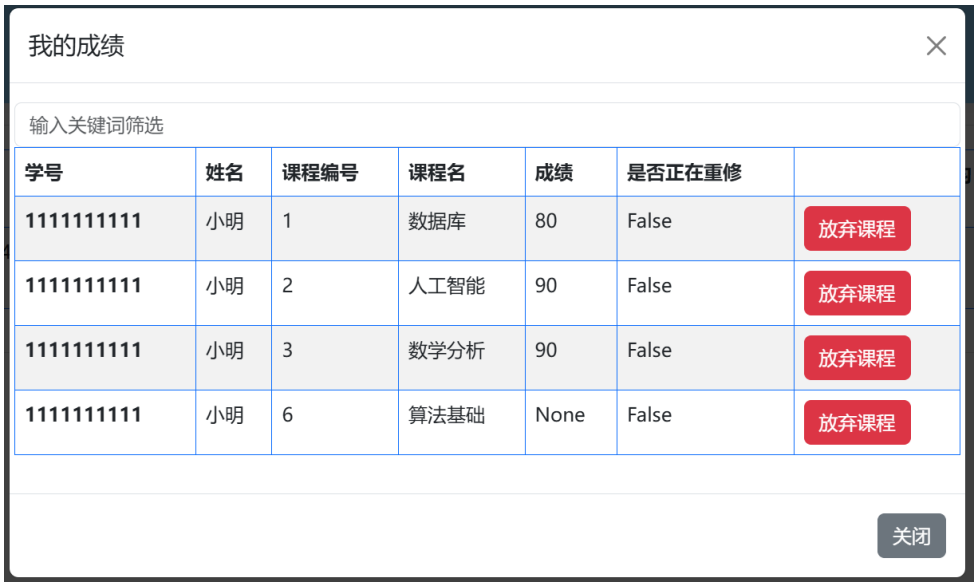


- 选课
  - 查看

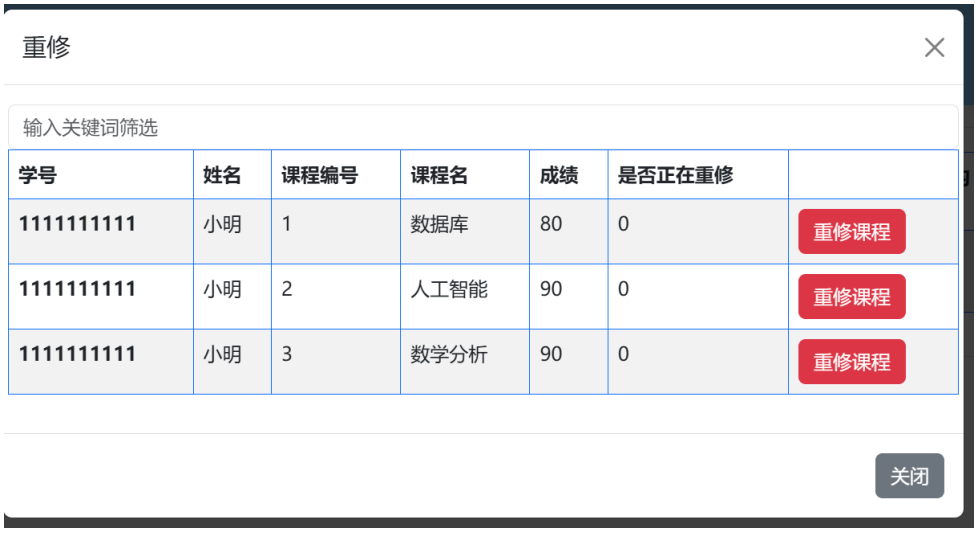


  - 选课（选课算法基础后，不再显示在选课表中，会再次出现在我的成绩，即已选课程中）





- 重修课程
  - 查看



  - 重修（重修数据库，会将是否正在重修置为1，前提是数量不超过剩余重修机会，同时该项将不显示在可重修课程中）

重修

输入关键词筛选

学号	姓名	课程编号	课程名	成绩	是否正在重修	
1111111111	小明	2	人工智能	90	0	重修课程
1111111111	小明	3	数学分析	90	0	重修课程

关闭

- 筛选框（以重修后我的成绩为例，中有四条信息，输入90，则会选出两个成绩为90的课程）

我的成绩

90

学号	姓名	课程编号	课程名	成绩	是否正在重修	
1111111111	小明	2	人工智能	90	False	放弃课程
1111111111	小明	3	数学分析	90	False	放弃课程

关闭

管理员界面-学生管理

- 查看学生信息

- 删除（调用存储过程会同时删除其关联的选课信息，后将不再展示级联删除或修改的部分）
    - 删除前

- 删除

- 删除后

- 增加
  - 增加前

    如上所示有三条学生信息
  - 增加（此处插入了图片）

- 增加后

- 查看插入照片（图片插入在mysite/media/photos下）

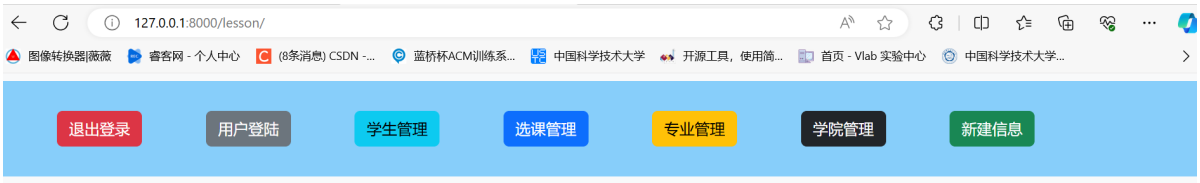
- 修改
  - 修改前如上图所示
  - 修改（第四条信息）

    页面与增加信息一致，此处主要是修改id，将4444444444改为5555555555
  - 修改后

学号	密码	照片	姓名	性别	状态	修机会	生日	入学时间	专业	学院	算术平均gpa	获奖情况	处分情况		
1111111111	123456	查看	小明	男	学业正常	1	Jan. 1, 2003	Sept. 1, 2021	计算机科学与技术	计算机科学与技术学院	3.7	查看	查看	编辑	删除
2222222222	123456	查看	小华	男	学业正常	2	Jan. 1, 2002	Sept. 1, 2020	应用物理	物理学院	2.0	查看	查看	编辑	删除
3333333333	123456	查看	小花	女	学业正常	2	Jan. 1, 2004	Sept. 1, 2022	天文学	物理学院	4.0	查看	查看	编辑	删除
5555555555	1234567	查看	小聪	男	学业正常	1	Sept. 3, 2003	Sept. 1, 2021	计算机科学与技术	计算机科学与技术学院	0.0	查看	查看	编辑	删除

管理员界面-课程管理

功能与学生管理基本对称，不多赘述。



输入关键词筛选

课程编号	课程名称	教师	学分	学时	专业	学院		
1	数据库	张三	4	80	计算机科学与技术	计算机科学与技术学院	编辑	删除
2	人工智能	李四	4	80	计算机科学与技术	计算机科学与技术学院	编辑	删除
3	数学分析	王五	6	120	天文学	物理学院	编辑	删除
4	计算机网络	张三	4	80	计算机科学与技术	计算机科学与技术学院	编辑	删除
5	编译原理	李四	4	80	计算机科学与技术	计算机科学与技术学院	编辑	删除
6	算法基础	王五	6	80	计算机科学与技术	计算机科学与技术学院	编辑	删除

管理员界面-选课管理

只展示与存储过程、事务、触发器等有关的内容

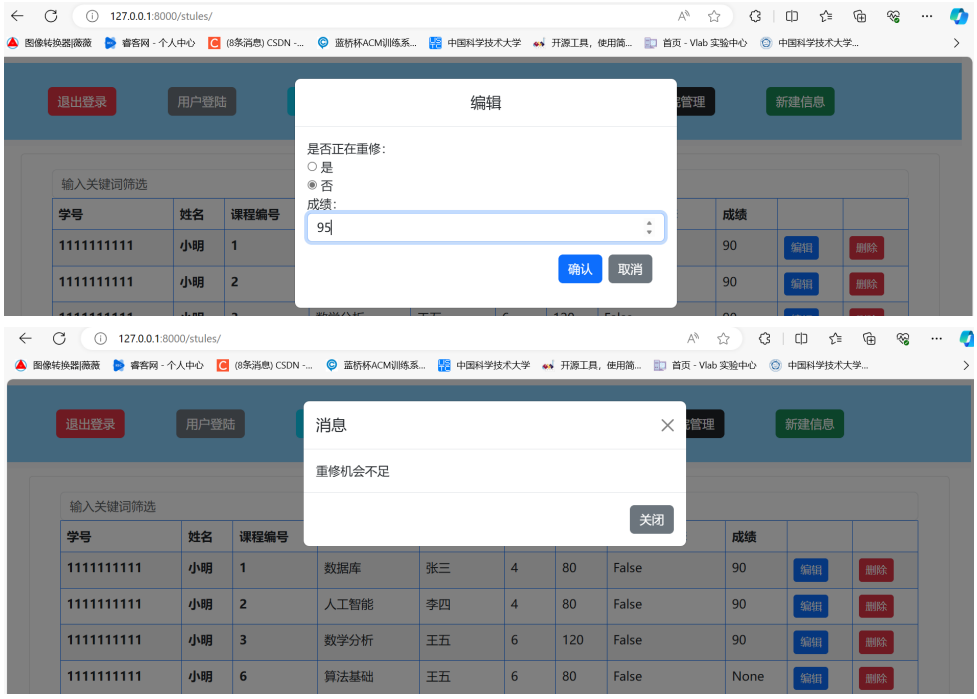
- 重修事务有关操作（对1111111111，小明）
  - 初始，剩余1次重修机会（具体可见上图）

111

学号	姓名	课程编号	课程名	授课教师	学分	学时	是否正在重修	成绩		
1111111111	小明	1	数据库	张三	4	80	True	80	编辑	删除
1111111111	小明	2	人工智能	李四	4	80	False	90	编辑	删除
1111111111	小明	3	数学分析	王五	6	120	False	90	编辑	删除
1111111111	小明	6	算法基础	王五	6	80	False	None	编辑	删除

  - 此时对课程（2，人工智能）和（3，数学分析）的成绩修改或删除是不合法的，因为已有一门重修课程（1，数据库），而对（1，数据库）和（6，算法基础）的成绩修改是合法的，因为前者是正在重修的课程，后者是还未有成绩的课程。此处将（1，数据库）改为90

- 尝试修改（2，人工智能）的成绩为95



- 学业状态触发器有关操作（对1111111111，小明）
  - 当挂科次数大于3时会触发器会将学业状态设为警告，此处将（1111111111，小明）的（6，算法基础）成绩设为50，另外再插入两条50分的成绩，最终的选课表如下所示。

  - 查看（1111111111，小明）的信息可以看到状态已设为警告。若减少挂科次数则状态会再次变成正常，不再赘述。

管理员界面-专业管理

功能与前述管理基本相同。在编辑id和删除时均会调用相应存储过程和事务以处理相关外键，实现逻辑基本相同，不多赘述。

管理员界面-学院管理

功能与前述管理基本相同。在编辑id和删除时均会调用相应存储过程和事务以处理相关外键，实现逻辑基本相同，不多赘述。

参考

- 前端引用

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/jquery@3.6.4/dist/jquery.min.js">
</script>
```

- 用于测试的图片

存放于本地mysite/testimg

- 前端引用官网

https://cdn.jsdelivr.net