学生专业学习管理

信息系统

计科1702 夏泽坤170950119

目录

[学生专业学习管理 1](#_Toc38573109)

[信息系统 1](#_Toc38573110)

[一、 项目概述 3](#_Toc38573111)

[二、 技术介绍 4](#_Toc38573112)

[三、 项目分析 4](#_Toc38573113)

[四、 项目设计 4](#_Toc38573114)

[1. 关键数据结构设计 5](#_Toc38573115)

[(1) 用户账号模型 5](#_Toc38573116)

[(2) 学生信息模型 5](#_Toc38573117)

[(3) 教师信息模型 5](#_Toc38573118)

[(4) 学生奖惩模型 5](#_Toc38573119)

[(5) 用户权限模型 5](#_Toc38573120)

[(6) 学院模型 5](#_Toc38573121)

[(7) 专业模型 5](#_Toc38573122)

[(8) 班级模型 5](#_Toc38573123)

[(9) 课程模型 5](#_Toc38573124)

[(10) 课程班级模型 5](#_Toc38573125)

[(11) 学生成绩模型 5](#_Toc38573126)

[(12) 日志模型 5](#_Toc38573127)

[2. 功能实现设计 5](#_Toc38573128)

[(1) 学生基本信息显示 5](#_Toc38573129)

[(2) 学生基本信息修改 5](#_Toc38573130)

[(3) 密码修改 5](#_Toc38573131)

[(4) 奖惩查看 5](#_Toc38573132)

[(5) 选课系统 5](#_Toc38573133)

[(6) 教师端的课程班级管理与分数登记 5](#_Toc38573134)

[(7) 图表信息生成 5](#_Toc38573135)

[(8) 学院、专业、班级、课程、课程班级、学生成绩、学生的一体式分级增删改查 5](#_Toc38573136)

[3. 界面设计 5](#_Toc38573137)

[(1) 主页内容的居中显示 5](#_Toc38573138)

[(2) 按照权限区分状态的导航栏 5](#_Toc38573139)

[五、 系统实验说明 5](#_Toc38573140)

[(1) 登录界面： 5](#_Toc38573141)

[(2) 学生、教师、管理员下的导航栏： 5](#_Toc38573142)

[(3) 学生主页： 5](#_Toc38573143)

[(4) 教师主页（课程班级管理）： 5](#_Toc38573144)

[(5) 管理员主页 5](#_Toc38573145)

[六、 小结 5](#_Toc38573146)

[七、 参考文献 5](#_Toc38573147)

# 项目概述

学生专业信息管理系统是为了提供学生在读期间，选择各自专业课程的一个平台，是一个教育单位不可缺少的部分。学生专业信息管理系统应该能够为用户提供充足的信息和快捷的查询手段。

随着科学技术的不断提高，计算机科学日渐成熟，其强大的功能己为人们深刻认识，它已进入人类社会的各个领域并发挥着越来越重要的作用。作为计算机应用的一部分，使用计算机对学生专业信息进行管理，有着检索迅速、查找方便、可靠性高、存储量大、保密性好、寿命长、成本低等优点，能够极大地提高管理的效率。

因此，开发这样一套成绩管理系统成为很有必要的事情。学生专业信息管理系统提供了强大的学生成绩管理管理功能，方便教师对学生成绩等信息的添加、修改、删除、查询等操作，同时方便学生对自己各科成绩查询，进行学习上的交流。

# 技术介绍

本系统采用了前后端分离开发的方法（如图2.1）：后端采用了MVC的框架模式的由Python写成的Web应用框架Django(3.0.4)；前端框架采用基于HTML、CSS、JavaScript的Bootstrap(4.41)和Bootstrap Table(1.16)；数据库部分使用的关系型数据库管理系统MySQL。

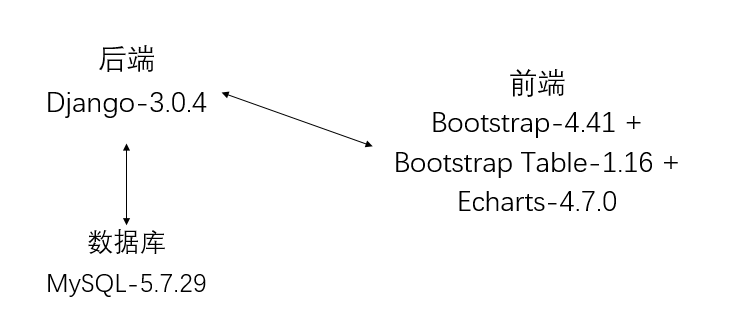


图2.1 前后端架构

本项目采用的三个关键部件均是开放源代码的项目，为日后实际部署降低了总体拥有成本(Total Cost of Ownership)。

模型(Model)-视图(View)-控制器(Controller)是Xerox PARC在二十世纪八十年代为编程语言Smalltalk－80发明的一种软件设计模式，已被广泛使用。这种设计模式有助于管理复杂的应用程序，同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。

# 项目分析

就项目本身而言，需要从各个方面进行考虑。下面从题目要求，项目完整性以及开发方式三个方向进行分析

1. 题目要求
2. 系统可以准确地记录和查询学生信息，包括学生的姓名、单位、年龄、性别以及身份证号码等。
3. 系统可以准确地记录学生地每一次奖惩情况。
4. 系统可以对学校的院系情况进行管理，包括设置学院名称、修改某学院某专业方向的名称等。
5. 系统应该可以对基础数据进行维护。
6. 系统能够对开设的课程进行管理。
7. 学生选课管理、考试(登记分数)、补考重修管理。
8. 系统还应该提供强大数据统计、查询、报表生成以及打印等功能。
9. 用户权限管理。
10. 异常处理。

以上是题目中对学生专业信息管理系统所做的要求，其中可以分为几个部分理解。

1. 要求拥有学生信息管理、奖惩管理、院系管理、课程管理、选课管理、成绩管理、权限管理这七个功能模块
2. 要求对每个功能模块都具有查、插、删和改的数据库操作
3. 要求具有数据统计、异常处理等辅助功能
4. 项目完整性

选择这个项目的理由，是因为这个项目的要求和我们大学生实际使用的教务管理系统十分接近，有实际使用过的经验。但题目中存在一些说法模糊的部分，需要自行考虑。

1. 权限管理我们分为三个类别：学生、老师和管理员。
2. 异常处理我们采用了比较完善的日志系统，可以对网站的每次访问以及数据库的所有操作分成普通(Info)、警告(Warning)、错误(Error)、调试(Debug)四个等级进行记录。
3. 大规模数据的界面采用表格的方式进行呈现，数据统计使用图表来体现。同时表格提供数据导出的功能，解决报表生成的问题。
4. 开发方式

项目实际的开发周期是三周的时间，从教学周的第5周到第8周。我们小组由两人组成，代码迭代次数多、代码量大，所以我们决定采用GitHub托管代码(<https://github.com/TTopoo/Student>)，使用QQ进行沟通，作为协同办公的平台。截至项目尾声，进行了105commit，最高的单周提交达到44次（如图3.1、图3.2）。

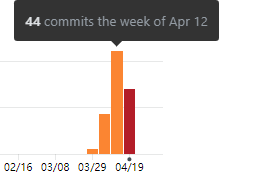


图3.1 单周提交次数分布

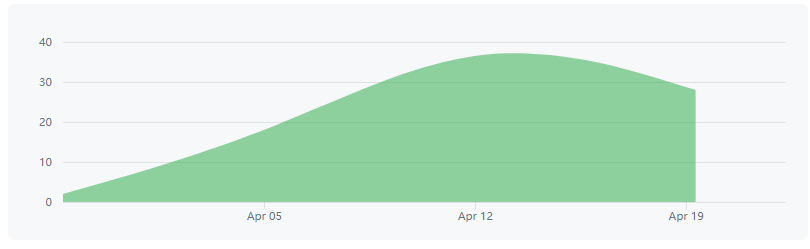


图3.2 每日提交次数分布

数据库我们采用的方案是托管在云服务器上（如图3.3），实际部署在百度云学生机。双方使用同一份数据库，避免了数据库结构的差距可能产生的问题。

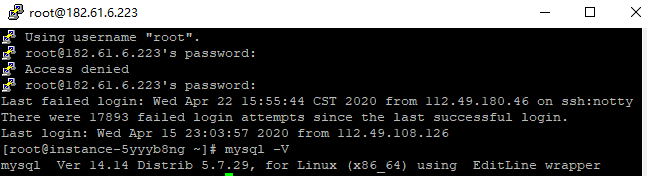


图3.3 部署在云服务器上的数据库

# 项目设计

## 关键数据结构设计

数据管理系统的数据大都存储在数据库内，采用关系模型的表结构存取数据。Django自带了ORM框架，可以将数据库中的表持久化为一种模型。Django对表的操作进行了面向对象程度较高的封装，一个模型的逻辑功能需要一个或多个表支撑，并可自动维护各个表之间的关系。因此在Django对数据结构的描述中，以模型为单位描述而不是以表为单位描述。

本系统总共设计了12种模型, 在不计算Django自带的诸如用来存储admin后台账号、模型修改记录等表的情况下, 12个模型对应了16张表。多出来的表是部分模型自建的关联表。

接下来讲解设计的12个模型

### 用户账号模型

用户账号模型存储了账号号码、密码和账号编辑时间。是用户使用该系统的必备数据。

# 用户账号模型

class User(models.Model):

    account = models.CharField(max\_length=128, unique=True, verbose\_name='账号')

    password = models.CharField(max\_length=256, verbose\_name='密码')

    edit\_time = models.DateTimeField(auto\_now\_add=True)

### 学生信息模型

学生信息模型下存储了学生的基本信息。分别存储了对应用户账号模型的外键、姓名、性别、年龄、邮箱、身份证号。

# 学生信息模型

class StudentInformationModel(models.Model):

    user\_id = models.ForeignKey('User', on\_delete=models.CASCADE)

    name = models.CharField(max\_length=30, verbose\_name='姓名', null=True)

    sex = models.CharField(max\_length=32, choices=gender, default='男')

    age = models.CharField(max\_length=20, verbose\_name='年龄', null=True)

    email = models.EmailField(verbose\_name='邮箱')

    idc = models.CharField(max\_length=20, verbose\_name='身份证', null=True)

### 教师信息模型

教师信息模型下存储了教师的基本信息。除了学生需要的基本信息之外，还存储了教师的学历和教育经历。

# 教师信息模型

class TeacherInformationModel(models.Model):

user\_id = models.ForeignKey('User', on\_delete=models.CASCADE)

    email = models.EmailField(verbose\_name='邮箱')

    name = models.CharField(max\_length=30, verbose\_name='姓名', null=True)

    sex = models.CharField(max\_length=32, choices=gender, default='男')

    idc = models.CharField(max\_length=20, verbose\_name='身份证', null=True)

    age = models.CharField(max\_length=20, verbose\_name='年龄', null=True)

    graduate\_school = models.TextField(verbose\_name='毕业学校')

    education\_experience = models.TextField(verbose\_name='教育经历')

### 学生奖惩模型

学生奖惩模型存储了学生的奖惩信息。内容有：映射学生信息模型的外键、奖惩类型、奖惩详细信息、奖惩日期。

# 学生奖惩模型

class StudentAwardsRecodeModel(models.Model):

    stu\_id = models.ForeignKey(

        'StudentInformationModel', on\_delete=models.CASCADE)

    award\_type = models.CharField(

        max\_length=5, verbose\_name='奖惩记录类别', null=True)

    award\_content = models.CharField(

        max\_length=50, verbose\_name='奖惩信息', null=True)

    award\_date = models.DateField(verbose\_name='奖惩日期', null=True)

### 用户权限模型

用户权限模型用于区分学生、教师和管理员之间的权限。存储了教师信息模型和权限的级别。

# 权限表（只记录teacher升admin的账号）

class Privilege(models.Model):

    type = models.TextField(max\_length=64)

    account = models.ForeignKey(

        'TeacherInformationModel', on\_delete=models.CASCADE)

### 学院模型

学院模型只需要记录学院名称。

# 学院模型

class CollegeModel(models.Model):

    college\_name = models.CharField(max\_length=64, verbose\_name='学院名称')

### 专业模型

专业模型存有专业名称、对应学院模型的外键和对课程的多对多映射关系。因为专业与课程之间的关系是多对多的。多对多的映射关系自带了一张关联表。因此，此模型对应两张表。

# 专业模型

class MajorModel(models.Model):

    major\_name = models.CharField(max\_length=64, verbose\_name='专业名称')

    college\_id = models.ForeignKey('CollegeModel', on\_delete=models.CASCADE)

    courses = models.ManyToManyField(CourseModel, null=True, blank=True)

### 班级模型

班级模型存有班级名称、对应专业的外键和对应学生的关联表。

# 班级模型

class ClassModel(models.Model):

    class\_name = models.CharField(max\_length=64, verbose\_name='班级名称')

    major\_id = models.ForeignKey('MajorModel', on\_delete=models.CASCADE)

    students = models.ManyToManyField(

        StudentInformationModel, null=True, blank=True)

### 课程模型

课程模型存有课程模型和其下开设的课程班级。

# 课程模型

class CourseModel(models.Model):

    course\_name = models.CharField(max\_length=64, verbose\_name='课程名称')

    courseClass = models.ManyToManyField(

        CourseClassModel, null=True, blank=True)

### 课程班级模型

课程班级模型存有对应课程的外键、对应教师的外键、课程最大人数和该课程下所有的学生成绩模型。

class CourseClassModel(models.Model):

    course = models.ForeignKey(

        'CourseModel', on\_delete=models.CASCADE, null=True)

    teacher = models.ForeignKey(

        'TeacherInformationModel', on\_delete=models.CASCADE)

    maxNum = models.CharField(max\_length=16, verbose\_name='最大人数')

    studentsScore = models.ManyToManyField(

        StudentScoreModel, null=True, blank=True)

### 学生成绩模型

学生成绩模型存有对应学生信息的外键、对应课程班级的外键、学生成绩和当前状态。

# 学生成绩模型

class StudentScoreModel(models.Model):

    student = models.ForeignKey(

        'StudentInformationModel', on\_delete=models.CASCADE)

    courseClass = models.ForeignKey(

        'CourseClassModel', on\_delete=models.CASCADE, null=True)

    score = models.CharField(max\_length=16, verbose\_name='分数', null=True)

    states = models.CharField(max\_length=16, verbose\_name='状态', null=True)

### 日志模型

日志模型存储了日志类型和操作详情

# 日志

class OperationLogs(models.Model):

    type = models.CharField(default='info', max\_length=64, verbose\_name="日志类型")

    content = models.TextField(verbose\_name="修改详情", null=True)

## 功能实现设计

该项目是合作项目，仅介绍本人参与过的功能设计。

### 学生基本信息显示

目前将学生基本信息显示作为主页。

后端通过模型获取相关数据：

        message = ''

        user\_id = request.session['user\_id']

        account = request.session['account']

        stu\_info = StudentInformationModel.objects.get(user\_id=user\_id)

        request.session['stu\_info\_id'] = stu\_info.id

        stu\_info.sex = sex\_map[stu\_info.sex]

        stu\_info.major = major\_map[stu\_info.major]

        # 学生所在班级

        classes = stu\_info.classmodel\_set.all()

        if not classes.exists():

            classname = '无'

        else:

            unify\_class = classes[0]

            # 班级所在专业

            major = unify\_class.major\_id

            # 专业所在学院

            college = major.college\_id

        return render(request, 'login/index\_student.html', locals())

在Django框架中，前端通过｛｛变量名｝｝的方式就可以获取后端传的值。

<h2 class="sub-header">{{ account }} {{ stu\_info.name }}</h2>

    <div>{{ message }}</div>

    <br>

    <br>

    <div>账号: {{ account }}</div><br>

    <div>姓名: {{ stu\_info.name }}</div><br>

    <div>性别: {{ stu\_info.sex }}</div><br>

    <div>年龄: {{ stu\_info.age }}</div><br>

    <div>邮箱: {{ stu\_info.email }}</div><br>

    <div>身份证号: {{ stu\_info.idc }}</div><br>

    <div>班级: {{ unify\_class.class\_name }}</div><br>

    <div>专业: {{ major.major\_name }}</div><br>

<div>学院: {{ college.college\_name }}</div><br>

### 学生基本信息修改

为了做到信息修改的人性化，信息修改的页面需要默认填充原信息，通过ajax获取原信息，并填充进输入框。前端通过post方式，url为getInfo来获取json信息：

$.ajax({

            cache: false,

            type: "POST",

            dataType: "json",

            headers: { "X-CSRFToken": $('[name="csrfmiddlewaretoken"]').val() },

            url: "./getInfo/",

            data: $('#alter-info-form').serialize(),

            async: false,

            success: function (data) {

                console.log(data)

                if (data.status == "success") {

                    $("#name").val(data.name);

                    $("#email").val(data.email);

                }

            },

后端将相关信息生成为json格式：

def getInfo(self, request):

        logging.info('enter stu\_alter\_info getInfo')

        stu\_info\_id = request.session['stu\_info\_id']

        stu\_info = StudentInformationModel.objects.get(id=stu\_info\_id)

        data = {

            'status': 'success',

            'name': stu\_info.name,

            'email': stu\_info.email,

        }

        return HttpResponse(json.dumps(data))

然后是信息的修改。前端通过ajax将输入框的信息传到后台，方式post，数据类型json，url：alterInfo。成功后跳转至学生主页。

$.ajax({

            cache: false,

            type: "POST",

            dataType: "json",

            url: "./alterInfo/",

            data: $('#alter-info-form').serialize(),

            async: false,

            success: function (data) {

                if (data.status == "success") {

                    toastr.success('修改信息成功');

                    window.location = '/manage/student/'

                }

后端获取信息查错并写入数据库：

        name = request.POST.get("name", None)

        if name == '':  # 姓名非空

            return HttpResponse(json.dumps({'status': 'name0'}))

        email = request.POST.get("email", None)

        if email == '':  # 邮箱非空

            return HttpResponse(json.dumps({'status': 'email0'}))

        stu\_info\_id = request.session['stu\_info\_id']

        stu\_info = StudentInformationModel.objects.get(id=stu\_info\_id)

        stu\_info.name = name

        stu\_info.email = email

        stu\_info.save()

### 密码修改

密码修改需要原密码验证和两次新密码相同验证。同样采用ajax的技术，细节不再介绍，展示一下前端的报错内容的详细程度。

    function alterPassword() {

        $.ajax({

            cache: false,

            type: "POST",

            dataType: "json",

            url: "/manage/student/info/alterPassword\_/",

            data: $('#alter-password-form').serialize(),

            async: false,

            success: function (data) {

                console.log(data);

                if (data.status == "success") {

                    toastr.success('修改密码成功');

                    window.location = '/manage/student/'

                }

                if (data.status == "0") {

                    toastr.warning('密码不能为空!');

                }

                if (data.status == "0f") {

                    toastr.warning('原密码不正确!');

                }

                if (data.status == "12f") {

                    toastr.warning('两次新密码不相同!');

                }

            },

后端密码验证的处理逻辑：

password0 = request.POST.get('password0', None)

        password1 = request.POST.get('password1', None)

        password2 = request.POST.get('password2', None)

        if password0 == '' or password1 == '' or password2 == '':

            return HttpResponse(json.dumps({'status': '0'}))

        user\_id = request.session['user\_id']

        user = User.objects.get(id=user\_id)

        if hash\_code(password0) != user.password:

            return HttpResponse(json.dumps({'status': '0f'}))

        if password1 != password2:

            return HttpResponse(json.dumps({'status': '12f'}))

        user.password = hash\_code(password1)

        user.save()

### 奖惩查看

学生的奖惩查看需要构建表格，采用BootStrapTable插件实现。

前端表格样式：

<table id="table" data-toolbar="#toolbar" data-search="true" data-show-refresh="true" data-show-toggle="true"

  data-show-fullscreen="true" data-show-columns="true" data-show-columns-toggle-all="true" data-detail-view="true"

  data-show-export="true" data-click-to-select="true" data-detail-formatter="detailFormatter"

  data-minimum-count-columns="2" data-show-pagination-switch="true" data-pagination="true" data-id-field="id"

  data-page-list="[10, 25, 50, 100, all]" data-show-footer="true" data-side-pagination="server"

  data-url="/manage/student/info/award\_" data-response-handler="responseHandler">

</table>

前端表格内容：

columns: [

        { field: 'id', title: 'award\_id', align: 'center' },

        { field: 'stu\_id\_\_name', title: 'name', align: 'center' },

        { field: 'award\_type', title: 'award\_type', align: 'center' },

        { field: 'award\_content', title: 'award\_content', align: 'center' },

        { field: 'award\_date', title: 'award\_date', align: 'center' },

      ]

后端学生奖惩数据的选择：

        # 学生信息ID

        stu\_info\_id = request.session['stu\_info\_id']

        awards = StudentAwardsRecodeModel.objects.filter(stu\_id=stu\_info\_id)

        awards = awards.values('id', 'stu\_id\_\_name',

                               'award\_type', 'award\_content', 'award\_date')

        data['total'] = awards.count()

     data['rows'] = list(awards)

return JsonResponse(data)

### 选课系统

选课的表格需要显示课程、课程班级、教师、最大人数、该学生分数、当前状态等数据。在后端，要想出简洁的课程数据筛选算法有一定难度。

因为有的课程学生选了，有的没选。所以并不能从学生的角度筛选课程。从学生信息获取到专业信息，可以得到所有的课程信息，根据Django的特性，我们能很方便的获得“课程-课程班级-学生分数”三个模型链接起来的信息，并方便的分步转换成json格式。这与手动组合json数据的方式相比有着很大的便利。但问题是刚获得的信息还含有其他学生的成绩信息，去掉该学生以外的学生记录的同时还要保证没有该学生的课程不会被删掉（如果把带有其他学生成绩的课程都删掉，那就只能剩下学生选过的课程）。

因此经过一段时间想出了一种相对简洁的方案：第一轮循环将其他同学的信息设空，这样其他同学的信息就没有了；第二轮循环去重复项目，即可每个课程保留一条空信息；第三轮循环将该学生有选过课的课程的空信息删掉。这样就可以做到选过的课程显示该学生成绩，没选过的课程显示并且只显示一次。而且不需要再去组合json格式。

去重函数：

def remove\_duplicates(self, list01):

        list02 = list()

        for i in list01:

            if i not in list02:

                list02.append(i)

    return list02

三次循环的课程筛选函数：

def course\_filter(self, courselist, stu\_info\_id):

        # 设重

        for i in courselist:

            if i['courseClass\_\_studentsScore\_\_student\_\_id'] is not None and i['courseClass\_\_studentsScore\_\_student\_\_id'] != stu\_info\_id:

                print(i['courseClass\_\_studentsScore\_\_student\_\_id'])

                i['courseClass\_\_studentsScore\_\_student\_\_id'] = None

                i['courseClass\_\_studentsScore\_\_score'] = None

                i['courseClass\_\_studentsScore\_\_states'] = None

        # 去重

        courselist = self.remove\_duplicates(courselist)

        # 去空

        for i in courselist:

            if i['courseClass\_\_studentsScore\_\_student\_\_id'] == stu\_info\_id:

                for j in courselist:

                    if j['courseClass\_\_id'] == i['courseClass\_\_id'] and j['courseClass\_\_studentsScore\_\_student\_\_id'] is None:

                        print(j['courseClass\_\_id'])

                        courselist.remove(j)

    return courselist

从学生信息获取到发送课程信息json过程：

# 获取学生专业

        user\_id = request.session['user\_id']

        stu\_info = StudentInformationModel.objects.get(user\_id=user\_id)

        unify\_class = stu\_info.classmodel\_set.all()[0]

        major = unify\_class.major\_id

        # print(user\_id, stu\_info, unify\_class, major)

        # 该专业可选的课程

        course\_set = major.courses

        data['total'] = course\_set.count()

        # 课程中该学生的成绩和状态

        course\_set\_v = course\_set.values('id', 'course\_name', 'courseClass\_\_id', 'courseClass\_\_teacher\_\_name', 'courseClass\_\_maxNum',

'courseClass\_\_studentsScore\_\_student\_\_id', 'courseClass\_\_studentsScore\_\_score','courseClass\_\_studentsScore\_\_states')

        logging.debug(course\_set\_v)

        courseList = list(course\_set\_v)

        courseList = self.course\_filter(courseList, stu\_info.id)

    data['rows'] = courseList

return JsonResponse(data)

前端选课信息的显示，每行不仅要显示相应的课程信息，还要再加上选课和退课的按钮，方便学生操作。

            columns: [

                { field: 'course\_name', title: 'Name', align: 'center' },

                { field: 'courseClass\_\_teacher\_\_name', title: 'Teacher', align: 'center' },

                { field: 'courseClass\_\_maxNum', title: 'MaxNum', align: 'center' },

                { field: 'courseClass\_\_studentsScore\_\_score', title: 'Score', align: 'center' },

                { field: 'courseClass\_\_studentsScore\_\_states', title: 'State', align: 'center' },

                {

                    field: 'operate',

                    title: 'Operate',

                    align: 'center',

                    clickToSelect: false,

                    events: window.operateEvents,

                    formatter: operateFormatter

                }

            ]

选课和退课过程需要显示一个确认按钮防止误操作，并实现选课或退课的操作。选课过程前端也需要带有误操作人性化提示的内容：

        'click .add\_course': function (e, value, row, index) {

            console.log(row);

            bootbox.confirm({

                title: "添加课程",

                size: "small",

                message: "您确定要添加课程 " + row.course\_name + " 吗？",

                buttons: {

                    confirm: {

                        label: '是',

                        className: 'btn-success'

                    },

                    cancel: {

                        label: '否',

                        className: 'btn-danger'

                    }

                },

                callback: function (result) {

                    if (result) {

                        $.ajax({

                            type: "POST",

                            dataType: "json",

                            async: false,

                            url: "./add/",

                            data: JSON.stringify([row]),

                            contentType: 'application/json',

                            type: "POST",

                            traditional: true,

                            headers: { "X-CSRFToken": $('[name="csrfmiddlewaretoken"]').val() },

                            success: function (data) {

                                console.log(data);

                                if (data.status == "success") {

                                    toastr.success('添加课程成功');

                                    $("#table").bootstrapTable('refresh');

                                }

                                if (data.status == "authority\_check0") {

                                    toastr.warning('权限不足');

                                }

                                if (data.status == "exists") {

                                    toastr.warning('您已经选择该课程下的一个班级, 请不要重复选择!');

                                }

                                if (data.status == "none") {

                                    toastr.warning('该课程下未开设班级,敬请期待!');

                                }

                            },

                            fail: function (result) {

                                toastr.warning('添加失败');

                            }

                        });

                    }

                }

            });

        },

选课过程后端需要额外判断以防止重复选择，防止选择同课程下的其他班级，课程有可能未开设班级等情况

# 添加课程

    def add(self, request):

        logging.info('enter choose\_course add\_course')

        json\_receive = json.loads(request.body)

        course\_id = json\_receive[0]['id']

        courseClass\_id = json\_receive[0]['courseClass\_\_id']

        # 获取学生信息

        user\_id = request.session['user\_id']

        stu\_info = StudentInformationModel.objects.get(user\_id=user\_id)

        # 获取课程班级

        course = CourseModel.objects.get(id=course\_id)

        # 课程有可能没开班级

        courseClasses = course.courseClass.filter(id=courseClass\_id)

        if not courseClasses.exists():

            return HttpResponse(json.dumps({'status': 'none'}))

        courseClass = courseClasses[0]

        # 防止选择同类课程

        courseClasses=course.courseClass.all()

        for i in courseClasses:

            s = i.studentsScore.filter(student=stu\_info)

            if s.exists():

                return HttpResponse(json.dumps({'status': 'exists'}))

        # 创建学生成绩模型

        stu\_score = StudentScoreModel.objects.create(

            student=stu\_info, courseClass=courseClass, score='-', states='学习中')

        courseClass.studentsScore.add(stu\_score)

退课过程前端也要有人性化提示的设计：

'click .remove\_course': function (e, value, row, index) {

    console.log(row);

    bootbox.confirm({

    title: "删除课程！",

    size: "small",

    message: "您确定删除课程 " + row.course\_name + " 吗？",

    buttons: {

     confirm: {

         label: '是',

            className: 'btn-success'

        },

        cancel: {

         label: '否',

            className: 'btn-danger'

        }

    },

    callback: function (result) {

     if (result) {

         $.ajax({

             type: "POST",

                dataType: "json",

                async: false,

                url: "./delete/",

                data: JSON.stringify([row]),

                contentType: 'application/json',

                type: "POST",

                traditional: true,

                headers: { "X-CSRFToken": $('[name="csrfmiddlewaretoken"]').val() },

                success: function (data) {

                 console.log(data);

                    if (data.status == "success") {

                     toastr.success('删除课程成功');

                        $("#table").bootstrapTable('refresh');

                    }

                    if (data.status == "authority\_check0") {

                     toastr.warning('权限不足');

                    }

                    if (data.status == "nothing") {

                     toastr.warning('您没选择该课程, 所以删除无效!');

                    }

                    if (data.status == "none") {

                     toastr.warning('该课程下未开设班级,敬请期待!');

                    }

                },

                fail: function (result) {

                toastr.warning('删除失败');

            }

        });

    }

}

退课后端防止删除未选择过的课程，防止退出不存在的课程

        # 学生信息

        user\_id = request.session['user\_id']

        stu\_info = StudentInformationModel.objects.get(user\_id=user\_id)

        # 课程移除该学生

        course = CourseModel.objects.get(id=course\_id)

        courseClasses = course.courseClass.filter(id=courseClass\_id)

        if not courseClasses.exists():

            return HttpResponse(json.dumps({'status': 'none'}))

        courseClass = courseClasses[0]

        stu\_scores = StudentScoreModel.objects.filter(

            student=stu\_info, courseClass=courseClass)

        if not stu\_scores.exists():

            return HttpResponse(json.dumps({'status': 'nothing'}))

        stu\_score = StudentScoreModel.objects.get(

            student=stu\_info, courseClass=courseClass)

        courseClass.studentsScore.remove(stu\_score)

        stu\_score.delete()

### 教师端的课程班级管理与分数登记

教师端仅允许管理该教师开设的课程班级，对班级内的学生和成绩等信息进行增删改查。

筛选出该教师开设的所有课程班级信息：

        # 该教师模型

        user\_id = request.session['user\_id']

        teacher = TeacherInformationModel.objects.get(user\_id=user\_id)

        # 该教师教的课

        courseClasses = CourseClassModel.objects.filter(teacher=teacher)

        data['total'] = courseClasses.count()

        courseClasses = courseClasses.values(

            'id', 'course\_\_course\_name', 'teacher\_\_id', 'teacher\_\_name', 'maxNum')

        data['rows'] = list(courseClasses)

学生成绩有指向学生学生信息的外键，因此在添加学生成绩模型的时候，表单中需要有一个下拉框，里面选择已存在的学生：

<div>请选择学生:</div>

<div class="form-group">

  <select data-placeholder="选择学生..." class="form-control" name="student\_id" id="student\_id" required="required">

    {% for student in students %}

    <option value="{{ student.id }}" hassubinfo="true">{{ student.name }}</option>

    {% endfor %}

  </select>

</div>

然后是填写该学生的分数和状态信息：

<div class="form-label-group">

  <input type="text" class="form-control" name="score" id="score" placeholder="分数" required="required"></input>

  <label for="lastname" class="col-sm-3 control-label">分数</label>

</div>

<div class="form-label-group">

  <input type="text" class="form-control" name="states" id="states" placeholder="状态" required="required"></input>

  <label for="lastname" class="col-sm-3 control-label">状态</label>

</div>

添加学生成绩的后端：

        teacher\_id = request.POST.get("teacher\_id", None)

        maxNum = request.POST.get("maxNum", None)

        if maxNum == '':

            return HttpResponse(json.dumps({'status': 'maxNum0'}))

        # 创建课程班级

        course\_id = request.session['course\_id']

        course = CourseModel.objects.get(id=course\_id)

        teacher = TeacherInformationModel.objects.get(id=teacher\_id)

        courseClass = CourseClassModel.objects.create(

            course=course, teacher=teacher, maxNum=maxNum)

        courseClass.save()

        # 添加进对应的课程

        course.courseClass.add(courseClass)

学生成绩信息修改，也是学生分数登记的地方，前端的学生成绩编号仅提供显示，用于确认，不可编辑。

<div class="form-label-group">

  <input type="text" class="form-control" name="id\_update" id="id\_update" placeholder="学生成绩编号" required="required"

    readonly unselectable="on"></input>

  <label for="lastname" class="col-sm-3 control-label">学生成绩编号</label>

</div>

<div>请选择学生:</div>

<div class="form-group">

  <select data-placeholder="选择学生..." class="form-control" name="student\_id\_update" id="student\_id\_update"

    required="required">

    {% for student in students %}

    <option value="{{ student.id }}" hassubinfo="true">{{ student.name }}</option>

    {% endfor %}

  </select>

</div>

<div class="form-label-group">

  <input type="text" class="form-control" name="score\_update" id="score\_update" placeholder="分数"

    required="required"></input>

  <label for="lastname" class="col-sm-3 control-label">分数</label>

</div>

<div class="form-label-group">

  <input type="text" class="form-control" name="states\_update" id="states\_update" placeholder="状态"

    required="required"></input>

  <label for="lastname" class="col-sm-3 control-label">状态</label>

</div>

学生成绩修改后端：

        courseClass\_id = request.POST.get("id\_update", None)

        teacher\_id = request.POST.get("teacher\_id\_update", None)

        maxNum = request.POST.get("maxNum\_update", None)

        if maxNum == '':

            return HttpResponse(json.dumps({'status': 'maxNum0'}))

        courseClass = CourseClassModel.objects.get(id=courseClass\_id)

        teacher = TeacherInformationModel.objects.get(id=teacher\_id)

        courseClass.teacher = teacher

        courseClass.maxNum = maxNum

        courseClass.save()

学生成绩删除前端：

$.ajax({

    type: "POST",

    dataType: "json",

    async: false,

    url: "./delete/",

    data: JSON.stringify([row]),

    contentType: 'application/json',

    type: "POST",

    traditional: true,

    headers: { "X-CSRFToken": $('[name="csrfmiddlewaretoken"]').val() },

    success: function (data) {

     console.log(data);

     if (data.status == "success") {

         toastr.success('删除数据成功');

            $("#table").bootstrapTable('refresh');

        }

        if (data.status == "authority\_check0") {

         toastr.warning('权限不足');

        }

},

后端：

        json\_receive = json.loads(request.body)

        courseClass\_id = json\_receive[0]['id']

        courseClass = CourseClassModel.objects.get(id=courseClass\_id)

        studentsScores = courseClass.studentsScore.all()

        if studentsScores.exists():

            return HttpResponse(json.dumps({'status': 'have'}))

        courseClass.delete()

### 图表信息生成

本系统为教师和管理员界面提供了可视化的图表信息设计。本人参与了图表json信息的生成部分。将该课程班级下的学生成绩分为6类：优秀（90到100）,良好（80到90），普通（70到80），及格（60到70），不及格（60以下），学习中（还没参加考试的）。各自统计并手动组装成echarts所需的json格式。

        courseClass\_id = request.session['courseClass\_id']

        courseClass = CourseClassModel.objects.get(id=courseClass\_id)

        # 学生成绩

        studentScores = courseClass.studentsScore.all()

        # 处理

        excellent = 0

        well = 0

        general = 0

        passed = 0

        fail = 0

        learning = 0

        for i in studentScores:

            print(i.score, i.states)

            if i.states == '学习中':

                learning += 1

                continue

            if int(i.score) >= 90:

                excellent += 1

            elif int(i.score) >= 80:

                well += 1

            elif int(i.score) >= 70:

                general += 1

            elif int(i.score) >= 60:

                passed += 1

            else:

                fail += 1

        data = {

            'data\_pie': [

                {'value': excellent, 'name': '优秀'+str(excellent)},

                {'value': well, 'name': '良好'+str(well)},

                {'value': general, 'name': '普通'+str(general)},

                {'value': passed, 'name': '及格'+str(passed)},

                {'value': fail, 'name': '不及格'+str(fail)},

                {'value': learning, 'name': '学习中'+str(learning)},

            ]

        }

        print(data)

        return JsonResponse(data)

### 学院、专业、班级、课程、课程班级、学生成绩、学生的一体式分级增删改查

本系统全做了学院、专业、班级、课程、课程班级、学生成绩、学生的增删改查。增删改查方式与 **(6) 教师端的课程班级管理与分数登记**基本一致，不再讲解与贴示代码，但增加了一个“进入”的功能。在学院管理界面点击进入便可进入该学院的专业管理界面，在专业管理界面点击课程可以进入该专业的课程管理界面……以此类推。

“进入”功能的前端设计（以学院进入专业的功能为例）：

'click .enter': function (e, value, row, index) {

      console.log([row.college\_name]);

      $.ajax({

        url: "./enter/",

        data: JSON.stringify(row),

        contentType: 'application/json',

        type: "POST",

        traditional: true,

        headers: { "X-CSRFToken": $('[name="csrfmiddlewaretoken"]').val() },

        success: function (result) {

          window.location = '../major/'

        },

        fail: function (result) {

        }

      });

    }

进入功能的后端设计，这里要保存session以便下一级知道自己的上一级具体是谁。这样下一级的选择操作就可以进行数据筛选。

def enter(self, request):

        logging.info("enter college enter")

        json\_receive = json.loads(request.body)

        college\_id = json\_receive['id']

        request.session['college\_id'] = college\_id

        logging.info("end college enter")

        return redirect('/manage/aadmin/major/')

特殊情况有两个。一是专业下分有班级和课程两组，所以要安排两种不同的进入函数；二是在专业下的课程界面，因为专业与课程是多对多的关系，所以课程有2种添加方式，添加现有课程和创建新的课程。但相关实现方式在前面都有所提及，就不再详细展示。

## 界面设计

### 主页内容的居中显示

在css的样式中定义类的宽度和margin设为auto即可。

<style type="text/css">

    .div1 {

        width: 300px;

        margin: auto;

    }

</style>

### 按照权限区分状态的导航栏

导航栏分为3个下拉框，分别为学生、教师和管理员。学生和教师教师只能选择对应自己的下拉框，管理员可以选择教师和管理员的下拉框。

<li class="nav-item dropdown">

{% if request.session.authority == 1 %}

<a class="nav-link dropdown-toggle" href="#" id="dropdown03" data-toggle="dropdown"aria-haspopup="true" aria-expanded="false">学生</a>

{% else %}

<a class="nav-link dropdown-toggle disabled" href="#" id="dropdown03" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">学生</a>

{% endif %}

<div class="dropdown-menu" aria-labelledby="dropdown03">

<a class="dropdown-item" href="/manage/student/index/">学生主页</a>

<a class="dropdown-item" href="/manage/student/choose\_course/">选课系统</a>

<a class="dropdown-item" href="/manage/student/info/award/">奖惩查看</a>

<a class="dropdown-item" href="/manage/student/info/">修改信息</a>

<a class="dropdown-item" href="/manage/student/info/alterPassword">修改密码</a>

</div>

</li>

<li class="nav-item dropdown">

{% if request.session.authority == 0 or request.session.authority == 9 %}

<a class="nav-link dropdown-toggle" href="#" id="dropdown03" data-toggle="dropdown"aria-haspopup="true" aria-expanded="false">教师</a>

{% else %}

<a class="nav-link dropdown-toggle disabled" href="#" id="dropdown03" data-toggle="dropdown"aria-haspopup="true" aria-expanded="false">教师</a>

{% endif %}

<div class="dropdown-menu" aria-labelledby="dropdown03">

<a class="dropdown-item" href="/manage/teacher/index/">教师主页</a>

<a class="dropdown-item" href="/manage/teacher/stu\_info/">信息管理</a>

<a class="dropdown-item" href="/manage/teacher/course\_class/">班级管理</a>

<a class="dropdown-item" href="/manage/teacher/award/">奖惩管理</a>

</div>

</li>

<li class="nav-item dropdown">

{% if request.session.authority == 9 %}

<a class="nav-link dropdown-toggle" href="#" id="dropdown03" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">管理员</a>

{% else %}

 <a class="nav-link dropdown-toggle disabled" href="#" id="dropdown03" data-toggle="dropdown"aria-haspopup="true" aria-expanded="false">管理员</a>

{% endif %}

<div class="dropdown-menu" aria-labelledby="dropdown03">

<a class="dropdown-item" href="/manage/aadmin/index/">管理员主页</a>

<a class="dropdown-item" href="/manage/aadmin/privilege/">权限管理</a>

<a class="dropdown-item" href="/manage/aadmin/college/">学院管理</a>

<a class="dropdown-item" href="/admin/">admin后台</a>

</div>

</li>

</ul>

# 系统实验说明

### 登录界面：



### 学生、教师、管理员下的导航栏：







### 学生主页：



学生信息修改页面：



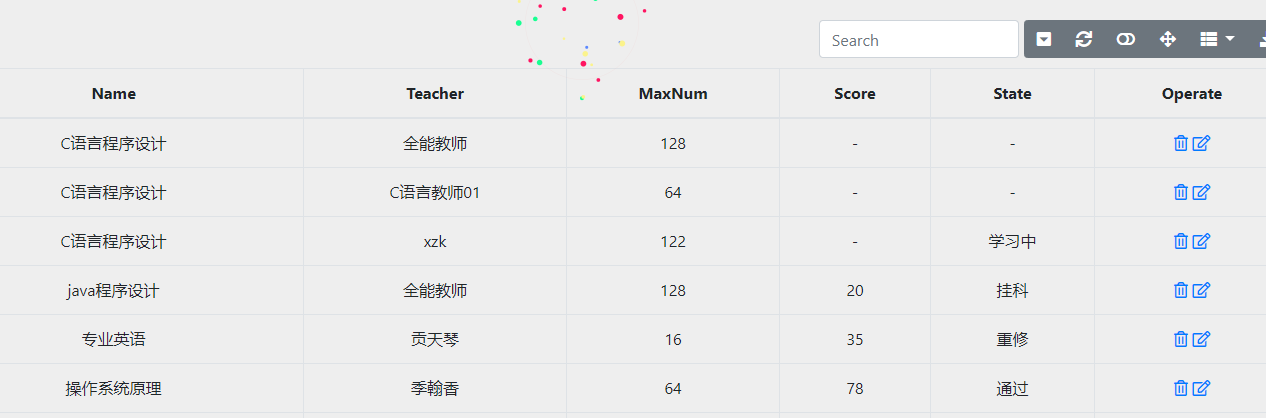
密码修改页面：

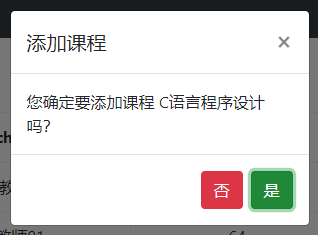


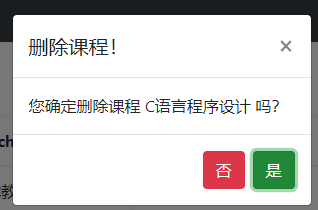
奖惩查看页面：



选课页面：







### 教师主页（课程班级管理）：



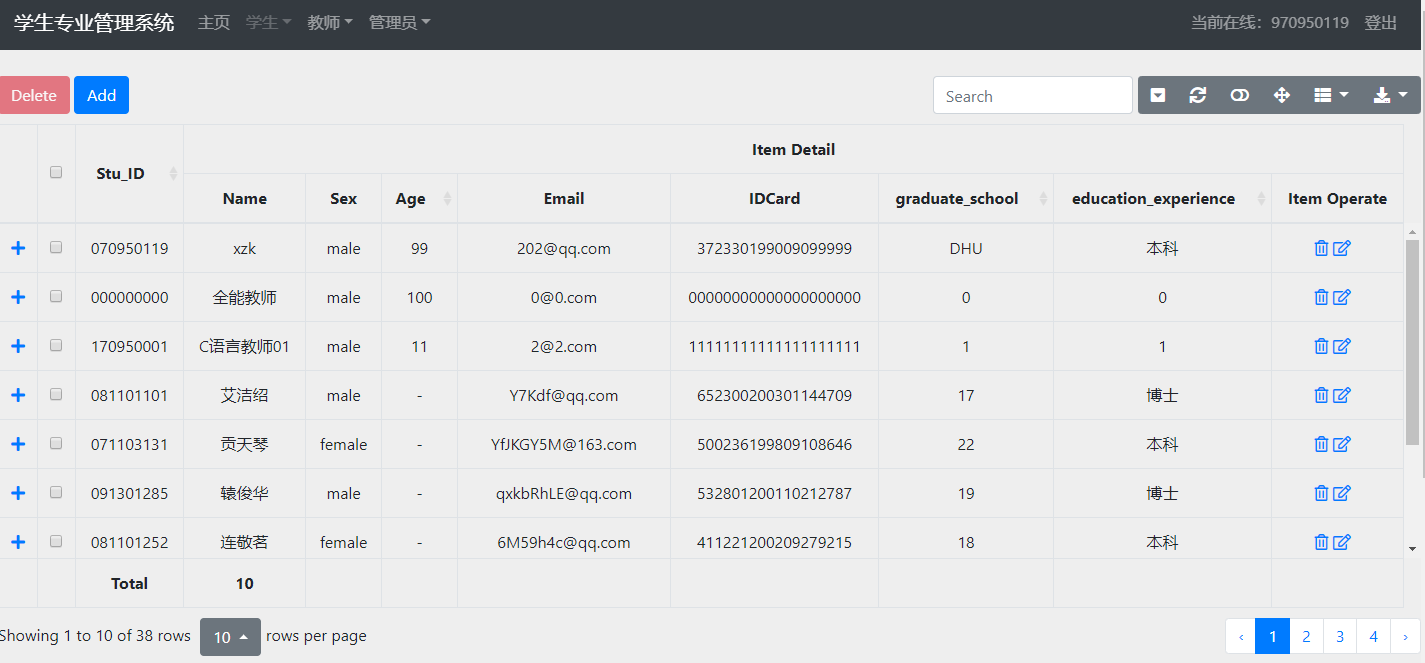
教师学生成绩管理：







### 管理员主页



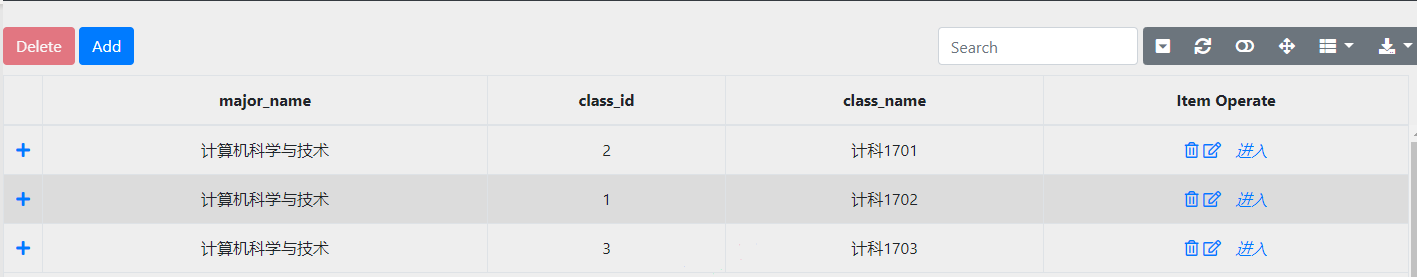
学院管理：



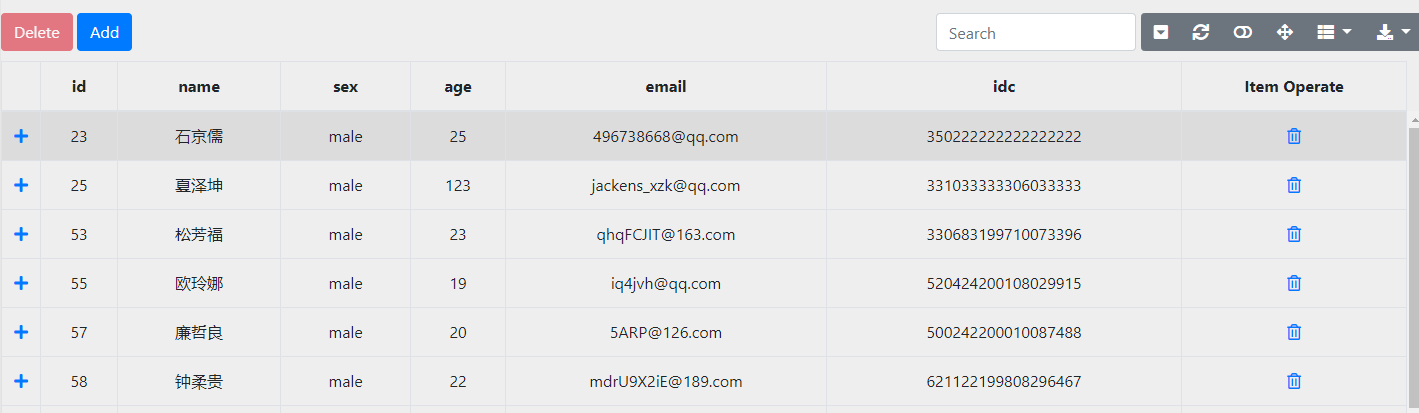
专业管理：



班级管理：



班级学生管理：



课程管理：



课程班级管理：



学生成绩管理：



# 小结

本次合作开发感触颇深，收获颇丰。开始正式使用了GitHub开协同开发软件，体会到了协同开发时GitHub的便利性，也学会了再遇到冲突时的合并方法。

在开发过程中遇到了诸多的问题，也几乎都一一解决了，之前觉得我不太行的部分在这次得到了充分的磨练，一些觉得太难而不想学的功能也逐渐学会并应用到了框架中。

Django是基于Python开发的web框架，这次开发体会到了Python的便利性，但同时也认识到了Python的部分不足，Python的部分功能不尽人意，比如运行速度较慢，报错功能相对简陋，部分数据结构功能不够完善等。

Django的模型层是一种ORM。本次开发中充分认识到了ORM框架的必要性，它把关系数据转化为面向对象的数据模型，极大的简化了对数据库数据的操作，在开发过程中给增删改查操作提供了巨大的便利。

采用了Django提供的页面模板技术，大大减少了html页面的代码量。也学会了更灵活的基于ajax的前后端数据交互技术，不再为前后端数据交互感到头疼。

# 参考文献

[1]https://wenku.baidu.com/view/63bdb7c5974bcf84b9d528ea81c758f5f71f299c.html

[2] <https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6>