微人大 beta 教务系统设计文档

一、前言

作为人大学生，微人大是我每天都在用的一个网站，它包括了与我学习相关的各项功能——选课、退课、查成绩、转专业申请、课程表、学校公告……

本次大作业，我利用新学的数据库设计知识，使用 Flask 架构编写程序，复现了微人大的部分功能。其中主要包括：选课、退课、查成绩，以及其他一系列与之相关的功能。同时，我做了用户管理系统，可以根据登录人的身份（学生、老师、教务）来开放不同的功能权限。

我使用的数据库是 Microsoft SQL Server，在程序中使用了 SQLAlchemy 来对接数据库，并用 Matplotlib 来绘制图表。最后，我用了 Bootstrap 来美化网站设计。

二、业务描述

在教务系统的设计中，我首先考虑的三项核心功能是选课、退课、成绩查询。其他一切功能都是围绕这三者展开。

根据生活经验，学校中有一些组织机构来适应以上功能。如一门课程由不同的老师讲授，开设不同的教学班给学生选择；由学生组成班级，并任命一位班主任，等等。

具体到网站设计上，首先，我希望引入用户管理系统，像微人大一样做一个登录界面。（事实上，我花了挺长时间获取和整理微人大登录界面的网站源码，试图把我的登录界面做得跟微人大一模一样，并且成功了。不过后来，为了保持整个项目的风格统一，我还是把登录界面换成了跟项目内其他网站一样的简朴风格。）先登录，再根据用户权限来开放网站的不同功能，而不是直接打开就是赤裸裸的功能界面。

我希望我的用户分两大类：学生、老师。而在老师之中，我又希望分出教务老师和其他老师，并对教务老师开放特殊的管理员权限，使其可以直接操作数据库，如添加老师、学生和课程等；而对于学生和一般的老师，则主要只能查询数据库而不能改动。（唯一可以改动的是选课记录）

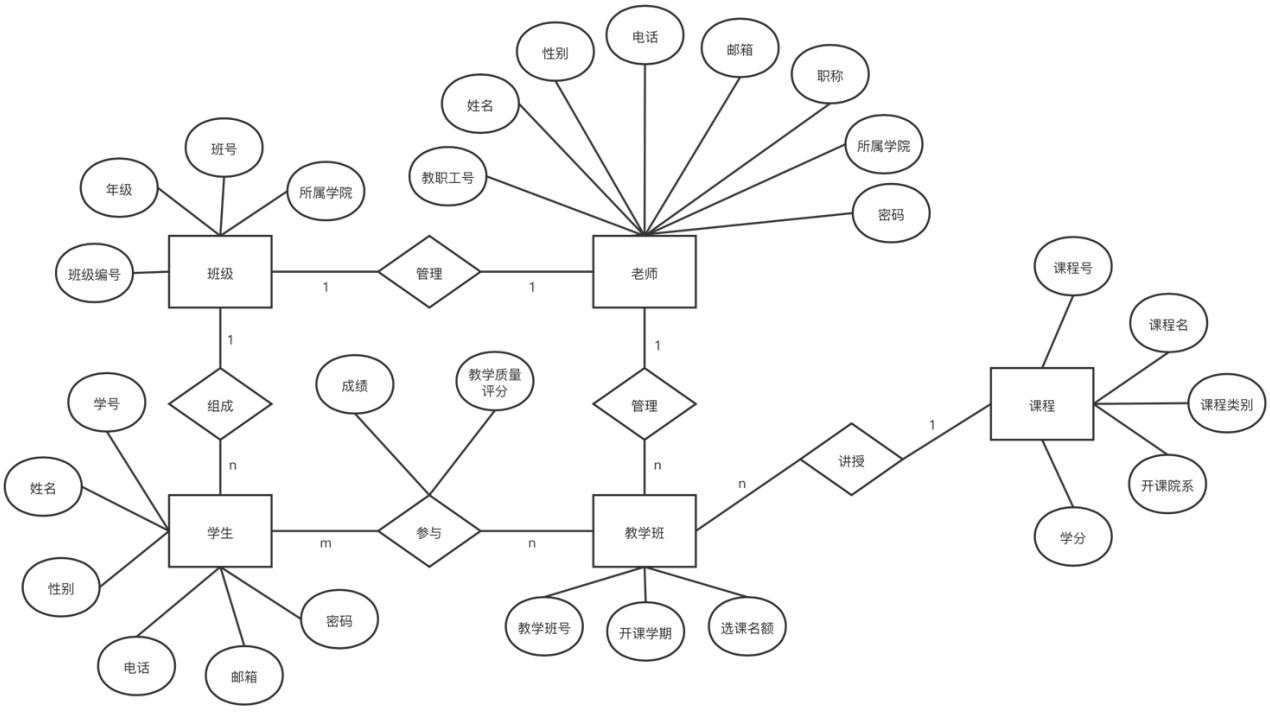
那么登录之后，不同类型的用户在主页上会看到不同的功能选项。学生应该可以选课、退课、查询已选课程、查询全校课程、查询成绩等。选课与退课时，应只能对下一学期的课程来操作，不能对以往已经上过的课程来操作；在查询已选课程和查询全校课程时，应设置一些筛选条件，根据不同的筛选条件显示不同的查询结果；在查询成绩时，也要像微人大一样，不仅可以看到每科成绩，还可以显示平均成绩，以及转化后的平均绩点（GPA），并且可以设置学期、课程类型等筛选条件来查询特定课程的成绩。

对于一般的老师，应当可以和学生一样查看全校课程，并可以注册教学班，设置名额，让学生来选。如果该老师是一个班级的班主任，则还应开放班级管理功能，显示班级内学生的一些基本信息，如 GPA 等，方便教师了解情况、针对指导。

对于教务老师，最主要的功能，应当是注册新的学生、老师和课程。这些都是学生和一般的老师没有权限做的。

以上便是主要的业务描述。根据以上对业务的设想，我开始了数据库的具体设计。

三、数据库设计



E-R 图

根据业务设想，我画出了如上的 E-R 图。

接下来，根据画出的 E-R 图，把它转换为关系数据库中的一些具体模式。

首先，出于尽量减少主体、尽量把一些主体变成属性的原则，我把学院从主体变成了课程和老师的一个属性（E-R 图已经做出相应修改）；

其次，把学生和教学班之间的多对多关系，独立出来作为一个模式，把学生和教学班的主键作为该模式的主键；

最后，把其他一对一和一对多关系转化成关系模式中的属性。

最终得到的关系模型如下：

学生（学号，姓名，性别，电话，邮箱，密码，所属班级编号）
班级（班级编号，班主任教职工号，所属学院，年级，班号）
教师（教职工号，姓名，性别，电话，邮箱，职称，所属学院，密码）
课程（课程号，课程名，课程类别，开课院系，学分）
教学班（教学班号，开课学期，选课名额，授课教师教职工号，课程号）
选课记录（学号，教学班号，成绩，教学质量评分）

四、模块设计

1. 用户管理模块

在学生和教师的模式中都加入和密码属性，在对象创建时初始化为“password1234”，后续老师和学生可以自行修改，以此作为登录的依据。

运行程序时首先显示登录界面，即把“/”和“/login”修饰器都用来修饰 login 视图函数，即让二者都对应登录页面。

登录时像微人大一样，使用学号/教职工号和密码来登录。而学号和教职工号又是作为不同模式的主键，即没有机制保证不会有重复的学号和教职工号。因此，手动规定学号由 100001 开始，教职工号由 200001 开始，在新加入对象时往后加一。这样就基本保证了不会有重复的学号和教职工号。

登录之后，进入主页，主页的导航栏上有“修改密码”栏，用户可以通过输入原密码，再二次输入新密码（两次须相同），来修改密码。

2. 选课/退课模块

首先，在教师端，开放注册教学班功能。即教师可以选择一个课程库里存在的课程，加上自己要开课的学期以及规定的选课名额，来创建一个新的教学班，并开放选课。

在学生端，在选课页面可以看到所有可选的教学班。可选的教学班包括：本学院所有类型课程的教学班，以及其他学院“跨学科专业选修”课程的教学班。并且，可选的教学班的开课学期都是下个学期，即“2020-2021 秋季学期”。

在退课页面，则可以看到所有已选的下学期的教学班。

选课、退课页面上都是通过提交一个包含教学班号的表单来选课、退课。

3. 信息查询模块

项目中包括多个专门信息查询的页面，如全校课程查询、全校教学班查询、个人选课历史查询等，以及在多个其他页面内附带了查询功能。查询功能的实现主要是抓取表单内提交的信息，并在数据库中利用 SQLAlchemy 的 query 功能（相当于帮助执行 SQL 语句）查询出想要的结果，然后以表格、统计图、统计信息等多种形式渲染到页面上。

4. 管理员模块

这个模块是专门为教务老师设计的。当登录系统检测到用户的登录 id 大于 200000（如前文所述，学生的 id 从 100001 开始，教师的 id 从 200001 开始），则辨认该用户为教师；然后在教师表中查询出改用户的职称，若职称为“教务秘书”，则辨认该用户为管理员。

管理员拥有注册学生、注册教师和注册课程的权限。具体的注册方法为，填写一张包含学生、教师或课程的各项信息的表单，并检测表单中的注册内容是否有效（例如，不能有选项为空，学生表单上的“所属班级编号”必须在数据库的班级表中存在，等等）。若有效，则注册成功，否则注册失败，重定向到原页面，要求用户重新填写。

另外，上述“注册成功”和“注册失败重新填写”的信息，都是通过 flash 信息来显示的，其效果为，当注册成功时，页面将重定向至主页，且页面上方显示一个绿色的文本框，显示“注册成功！”；而注册失败时，重定向至原页面，且页面上方显示一个红色的文本框，显示“由于……，注册失败！”。

Flash 信息在该项目的各个页面都有使用，不仅仅是在该模块中。例如在登录

时，根据登录的成功与否在页面上方显示 flash 信息，等等。

五、系统实现描述

由于整体实现比较繁杂，这里挑选两处比较具有代表性的实现作为样例。完整的实现请参阅代码。

1. 登录页面

在登录页面，主要内容是表单，以及一个登录按钮。登录成功或失败时，跳转到的页面上方会显示 flash 信息。其他很多页面的表单具体实现方式都和这个类似。

在 request.method 为 GET，即初次访问登录页面时，将表单渲染在页面上。表单的设计样式使用了 Bootstrap 的部分代码。

用户在表单内填写信息，并点击登录，此时表单通过 POST 方法把用户填写的内容发送至服务器。我们获取表单的内容，并到数据库内查询学号/教职工号是否存在，密码是否正确，等等。对于每种错误，都重定向至原登录页面，并显示对应的 flash 错误信息。

为了做到以上内容，需要在 login 视图函数中判断 request.method，若为 GET 则直接渲染页面，若为 POST 则获取表单内容，检测是否能登录。若登录成功，则将用户 id 储存在 session 中，这样在我们访问网站的过程中，每个网页都能方便地获得当前登录账户的信息。

在导航栏中，有“登出”选项，点击时触发 logout 视图函数，将 session 清空，并重定向至登录页面，在页面上方显示 flash 信息“您已成功登出！”。

login 和 logout 视图函数的代码如下：

```
@app.route('/')
@app.route('/登录', methods=('POST', 'GET'))
def login():
    if request.method == 'POST':
        _id = int(request.form['id'])
        password = request.form['password']

        if 100000 < _id < 200000:
            found_student = Student.query.get(_id)
            if found_student and found_student.password ==
password:
```

```
                session['user_id'] = _id
                session['user_type'] = 'student'
                flash('登录成功！')
                return redirect(url_for('index'))
        elif 200000 < _id < 300000:
            found_teacher = Teacher.query.get(_id)
            if found_teacher and found_teacher.password ==
password:
                session['user_id'] = _id
                if found_teacher.title == '教务秘书':
                    session['user_type'] = 'admin'
                else:
                    session['user_type'] = 'teacher'
                flash('登录成功！')
                return redirect(url_for('index'))

        flash('学号/教职工号或密码错误，请重新登录！')
        return redirect(url_for('login'))

    if 'user_id' in session:
        return redirect(url_for('index'))

    return render_template('登录.html')


@app.route('/登出')
def logout():
    if 'user_id' in session:
        session.pop('user_id')
        session.pop('user_type')
        flash('您已成功登出！')

    return redirect(url_for('login'))
```

2. 选课历史页面

选课历史页面可以作为所有查询类页面的一个代表。它即有查询的表单，也有表格、统计图表等多种方式显示的查询结果。

选课历史页面中，我们首先检测用户是否登录，即 session 中是否有内容。若没有，即重定向至登录界面，显示 flash 信息“请先登录！”

若有用户登录，则进一步检测该用户的类型，若不是学生账户，则显示 flash

信息“您没有权限访问！”并重定向至主页。

接下来便是功能的正文。同样要对 request.method 进行判断，不同的是，无论是 GET 还是 POST，都要渲染表格和统计图表；只是 GET 时（初次访问时）传给 html templates 的参数是默认参数，POST 时根据用户在表单中填写的信息来查询数据库，改变参数。

参数即为要在页面上显示的变量信息。Flask 内置 Jinja2，因此可以把传入的参数渲染进 html 模板，甚至可以在 html 里写循环、判断的类 Python 语句，大大方便了页面的设计。

而 png 格式的统计图表要如何在 html 页面中显示呢？它并不可以作为参数传入，因此我们要把它通过 plt.savefig(filepath)函数储存在项目文件内某路径下，然后在 html 中通过以下代码来访问：

```
<img src="{{ url_for('static', filename='images/图片名.png') }}?t={{ random }}" class="rounded mx-auto d-block" alt="图片名">
```

其中，src 代表图片源，页面渲染时会到对应路径下寻找相应图片并显示出来。

这里有个小细节，其实`{{ url_for('static', filename='images/图片名.png') }}`已经代表了图片路径，`?t={{ random }}`有什么用？

用处在于，如果不加后者，由于浏览器的缓存设置，当 src 不变时，页面会自动采用上次的图片，不会随着路径下的图片更新而更新。加入了一个随机数到路径中后，每次浏览器都会重新调取图片，达到实时更新的效果。

course_register_history 视图函数的代码如下：

```
@app.route('/主页/选课历史', methods=('POST', 'GET'))
def course_register_history():
    if 'user_id' not in session:
        flash('请先登录！')
        return redirect(url_for('login'))

    user_type = session['user_type']
    if user_type == 'student':
        user = Student.query.get(session['user_id'])
        SCs = list(user.SCs)

        if len(SCs) > 0:
            categories = [sc.teaching_class.course.category for sc in SCs]
```

```
            labels = ['学科基础', '专业必修', '专业选修', '跨学科专业选修
']
            counts = [label_count(categories, label) for label in
labels]

            plt.title('各类课程选修数量')
            plt.bar(range(len(labels)), counts)
            plt.xticks(range(len(labels)), labels)
            plt.savefig('./Application/static/images/各类课程选修数
量.png', dpi=120)
            plt.clf()

            have_fig = True
        else:
            have_fig = False

        if request.method == 'POST':
            semester = request.form['semester']
            category = request.form['category']

            if semester == '本学期':
                for i in reversed(range(len(SCs))):
                    if SCs[i].teaching_class.semester != '2019-2020
春季学期':
                        SCs.pop(i)
            elif semester == '本学年':
                for i in reversed(range(len(SCs))):
                    if SCs[i].teaching_class.semester not in ['2019-
2020 秋季学期', '2019-2020 春季学期']:
                        SCs.pop(i)

            if category != '全部':
                for i in reversed(range(len(SCs))):
                    if SCs[i].teaching_class.course.category !=
category:
                        SCs.pop(i)

            if len(SCs) > 0:
                categories = [sc.teaching_class.course.category for
sc in SCs]
                labels = ['学科基础', '专业必修', '专业选修', '跨学科专业
选修']
                counts = [label_count(categories, label) for label
in labels]
```

```
                plt.title('各类课程选修数量')
                plt.bar(range(len(labels)), counts)
                plt.xticks(range(len(labels)), labels)
                plt.savefig('./Application/static/images/各类课程选修
数量.png', dpi=120)
                plt.clf()

                have_fig = True
            else:
                have_fig = False

            return render_template('选课历史.html', SCs=SCs,
options=[semester, category], have_fig=have_fig,
random=random.random())
        else:
            return render_template('选课历史.html', SCs=SCs,
options=['全部', '全部'], have_fig=have_fig,
random=random.random())
    else:
        flash('您没有权限访问！')
        return redirect(url_for('index'))
```

六、代码运行指南

- 在 requirements.txt 中有项目所需的包，请提前安装好。
- 在 config.py 中请将 SQLALCHEMY_DATABASE_URI 项做本地化修改，将数据库该为自己建立的新数据库，将密码改为自己数据库管理软件的密码。
- 首次运行，可以在 routes.py 的 login 视图函数中，将注释的一段代码取消注释，然后运行 wsgi.py。运行后，数据库中应该已经创建好了所需的表，以及插入了少量种子数据。之后请及时将那段代码重新注释，因为数据无法重复插入，会出现主键重复的冲突。
- 建议不要仅适用种子数据进行后续测试，因为种子数据的量比较少，很多查询的结果都比较少量。可以在网页中插入数据，通过登录教务的账号来注册新的学生、教师和课程，通过登录一般教师的账号来注册教学班，通过登录学生的账号来增加选课记录……数据量多起来后，查询结果才会比较丰富。也顺便检验了这些功能的运行状况。

七、总结

总体来看，我的项目有一定完善度，用户管理功能比较完善，对不同类型的用户开放不同的功能；查询方面大体成熟，可以在同一个页面上显示表单、表格、统计图表和统计信息，并在表单中上传查询条件来改变查询结果，让表格、统计图表和统计信息都相应变化；页面设计也做了一定的美化，虽然肯定还远远不足，但至少不显得丑陋。

同时，我的项目也有一些明显的功能缺失，离一个闭合完善的系统还有一定距离。例如，注册班级功能目前还缺失，只能通过手动添加来注册新的班级；虽然选课记录中有分数和教学质量评分，但在学生端还没有做“给教师进行教学质量评分”功能，教师端也没有做“给学生打分”功能，这两项属性都只能手动添加。如果以上这些功能得到完善，这个项目将更加完整，几乎全部功能都不需要回到代码里去做操作，而只需在页面上进行。

八、参考文献

本项目主要参考了几个博客、YouTube 系列教程和官方文档，以及 Stack Overflow、CSDN 等论坛。下面列出几个主要参考的博客和官方文档：

[1] Building a Python App in Flask https://hackersandslackers.com/your-first-flask-application/

[2] Flask 入门教程 https://read.helloflask.com

[3] Flask 官方文档 https://flask.palletsprojects.com/en/1.1.x/

[4] Flask-SQLAlchemy 官方文档 https://flask-sqlalchemy.palletsprojects.com/en/2.x/#

[5] Learn Flask https://www.tutorialspoint.com/flask/index.htm

[6] Bootstrap 官方文档 https://getbootstrap.com/docs/4.5/getting-started/introduction/