

**本科毕业设计(论文)**

**FINAL PROJECT/THESIS OF UNDERGRADUATE**

**(2019届)**

**大学生信用评价系统分析与设计**

**Analysis and Design of Credit Evaluation System for University Students**

|  |  |
| --- | --- |
| **学　　院** | 管理学院 |
| **专　　业** | 信息管理与信息系统 |
| **学生姓名** | 杨逸辉 |
| **学　　号** | 1513491329 |
| **指导教师** | 朱小栋 副教授 |
| **完成日期** | 2019年5月 |

# 承诺书

本人郑重承诺：所呈交的毕业论文《大学生信用评价系统分析与设计》是在导师的指导下，严格按照学校和学院的有关规定由本人独立完成。文中所引用的观点和参考资料均已标注并加以注释。论文研究过程中不存在抄袭他人研究成果和伪造相关数据等行为。如若出现任何侵犯他人知识产权等问题，本人愿意承担相关法律责任。

承诺人(签名)：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

日期： 年 月 日

# 摘 要

在国内，有很大一部分人，如果不涉及贷款买房买车，很多时候不会认识到信用的重要性。大学生，作为即将步入社会的一大社会群体，往往对征信缺乏科学的认识。本文将整合一些易于得到的信息，借鉴Fico系统的思想，开发一套在大学校园内较为可行的征信系统，以供用人单位了解大学毕业生，便于大学生走向社会，接触真正的征信系统。

本文主要介绍大学生信用评估信息系统的开发环境，需求分析和整体思路。本文采用结构化的系统法系统与设计方法，通过业务流程，数据流程，E-R图等等展现系统的基本数据及数据流动关系。

**关键词**：大学生 征信 Fico 系统分析与设计

# ABSTRACT

In China, a large number of people rarely recognize the importance of credit, if they do not involve loans to buy a house to buy a car. College students, as a large social group that is about to enter the society, often lack scientific understanding of credit reporting. This article will integrate some easy-to-obtain information, draw on the ideas of the Fico system, and develop a more feasible credit information system on the university campus for the employer to understand the university graduates, to facilitate the college students to go to the society, and to contact the real credit information system.

This paper mainly introduces the development environment, demand analysis and overall thinking of college students' credit evaluation information system. This paper uses a structured system of systems and design methods to present the basic data and data flow relationships of the system through business processes, data flows, E-R diagrams, and so on.

**Key words**: College students, credit, Fico, MIS

# 目 录

[摘 要 5](#_Toc9614770)

[ABSTRACT 7](#_Toc9614771)

[目 录 i](#_Toc9614772)

[第一章 绪论 1](#_Toc9614773)

[1.1系统开发背景 1](#_Toc9614774)

[1.2 芝麻信用 1](#_Toc9614775)

[1.2.1芝麻信用分 1](#_Toc9614776)

[1.2.2芝麻信用分的局限性 3](#_Toc9614777)

[1.3 FICO系统 4](#_Toc9614778)

[1.3.1 Fico评分系统 4](#_Toc9614779)

[1.3.2 Fico评分的模型 4](#_Toc9614780)

[1.4 中国当前征信行业的机遇与挑战 4](#_Toc9614781)

[1.4.1互联网金融的机遇和挑战 4](#_Toc9614782)

[1.4.2大数据对人才市场的优势 5](#_Toc9614783)

[1.5研究的目的和意义 5](#_Toc9614784)

[1.5.1研究目的 5](#_Toc9614785)

[1.5.2研究意义 5](#_Toc9614786)

[1.6研究难点 5](#_Toc9614787)

[第二章 开发方法和技术 7](#_Toc9614788)

[2.1开发方法理论 7](#_Toc9614789)

[2.1.1原型法 8](#_Toc9614790)

[2.1.2面向对象开发方法 8](#_Toc9614791)

[2.1.3结构化开发方法 8](#_Toc9614792)

[2.2开发环境 9](#_Toc9614793)

[2.2.1 Python 9](#_Toc9614794)

[2.2.2 Django 9](#_Toc9614795)

[2.2.3 MySQL 11](#_Toc9614796)

[第三章 大学生信用评估系统规划 13](#_Toc9614797)

[3.1 大学生信息评估系统需求分析 13](#_Toc9614798)

[3.1.1需求分析 13](#_Toc9614799)

[3.1.2 公式 14](#_Toc9614800)

[3.1.3 分值对照 14](#_Toc9614801)

[3.2大学生信用评价体系 UC矩阵 19](#_Toc9614802)

[3.2.1教务子系统 19](#_Toc9614803)

[3.2.2二手商品交易子系统 20](#_Toc9614804)

[3.2.3学生个人认证子系统 20](#_Toc9614805)

[3.2.4图书子馆系统 20](#_Toc9614806)

[3.2.5信用评分子系统 21](#_Toc9614807)

[3.2.6其他 21](#_Toc9614808)

[第四章 大学生信用评分系统业务流程分析 23](#_Toc9614809)

[4.1泳道图 23](#_Toc9614810)

[4.2图例 23](#_Toc9614811)

[4.3子系统业务流程图 23](#_Toc9614812)

[4.3.1教务系统流程图 23](#_Toc9614813)

[4.3.2二手商品子系统业务流程图 25](#_Toc9614814)

[4.3.3学生个人申请子系统业务流程图 26](#_Toc9614815)

[4.3.4图书馆子系统业务流程图 27](#_Toc9614816)

[4.3.5信用评分子系统业务流程图 28](#_Toc9614817)

[第五章 大学生信用评分系统数据流程分析 30](#_Toc9614818)

[5.1数据流程图 30](#_Toc9614819)

[5.1.1数据流图图例 30](#_Toc9614820)

[5.2数据流程图 30](#_Toc9614821)

[5.2.1教务子系统数据流程图 30](#_Toc9614822)

[5.2.2个人申请子系统数据流程图 31](#_Toc9614823)

[5.2.3二手商品子系统数据流程图 32](#_Toc9614824)

[5.2.4图书馆子系统数据流程图 33](#_Toc9614825)

[5.2.5信用评分子系统数据流程图 35](#_Toc9614826)

[5.3第二层数据流程图 35](#_Toc9614827)

[第六章 大学生信用评估系统ER图 37](#_Toc9614828)

[6.1 ER图 37](#_Toc9614829)

[6.1.1教务系统 37](#_Toc9614830)

[6.1.2二手商品交易 37](#_Toc9614831)

[6.1.3个人申请子系统 38](#_Toc9614832)

[6.1.4 图书馆子系统 39](#_Toc9614833)

[6.1.5 信用评分子系统 39](#_Toc9614834)

[第七章 大学生信用评估系统实践 40](#_Toc9614835)

[7.1基础界面 40](#_Toc9614836)

[7.2教务子系统实操界面 43](#_Toc9614837)

[7.3图书馆子系统界面实操 46](#_Toc9614838)

[7.4 二手商品交易子系统 49](#_Toc9614839)

[7.5个人申请子系统实操 52](#_Toc9614840)

[7.6信用评级子系统实操 55](#_Toc9614841)

[第八章 总结与展望 61](#_Toc9614842)

[参考文献 63](#_Toc9614843)

[致谢 65](#_Toc9614844)

[附录 67](#_Toc9614845)

**第一章 绪论**

**1.1系统开发背景**

随着中国社会蓬勃发展，社会不断地走向繁荣富强，经济活动也随之日益增长。在人民，毫无疑问在这些经济活动中扮演了重要的角色。很多中国人在乎的“成家立业”，在我们的眼中，家的前提，是有一个属于自己的房子，没有房子，在很多情况下一切免谈。但是很显然，能够用现金购买一套房子的人少之又少，贷款则成为很多家庭的第一选择。

一些专家对于信贷有专业的解释，“信贷”被理解为一种获取商品或服务的手段，该商品或服务面向预期的未来，这意味着它被积极看待[1]。银行可以通过央行的征信系统，进行查阅和调取个人资料。但是这套从上个世纪八十年代才开始起步的征信体系对于普通大众而言有一些遥远。首先，没有普通大众知道自己的央行信用分是多少，也鲜有人谈及此类话题。然而，我们还是有必要了解，信用分是怎么来的：

传统的信用评分是统计、运筹学和相关学科（如机器学习和数据挖掘）技术的一种应用，旨在预测与贷款相关的财务风险。它本质上是一种根据观察到的特征来区分一个群体中具有不同信用风险的群体的方法[2]。

那么在中国，有什么信用分值得国人常常提及又简单直观呢。

**1.2 芝麻信用**

**1.2.1芝麻信用分**

很多时候，让普通百姓挂在嘴边的，往往是一月更新一次的芝麻信用分。这套阿里信用评价体系，背后的具体模型不得而知，但是由于更加贴近人的生活，同时背靠阿里这棵大树，收到了一定关注。截止到目前支付宝芝麻信用-信用生活页面，信用分大于650分，就可以试穿某些天猫服饰，绑定支付宝，用户也可以享受到免押金租用充电宝、雨伞、单车等等，给人们的生活带来了一定的便利。



图1.1 芝麻信用分示意图



图1.2 芝麻信用生活示意图

可以看到，阿里的芝麻信用分玩法还是很多的，可以给用户带来的福利也很多。

结合阿里的业务体系，芝麻信用分的来源可能包括如下三个维度[3]：

（1）阿里体系内数据。 阿里巴巴体系内的数据包括电商平台，互联网金融以及娱乐业务这三块，电商平台数据来源主要为淘宝、天猫等电商平台拥有的用户信息，互联网金融数据为蚂蚁金服旗下的支付宝、余额宝、阿里小贷等金融产品所产生交易信息，娱乐业的数据则来自阿里巴巴旗下的游戏、音乐等业务。

（2）外部数据。外部数据是指与阿里巴巴达成合作协议的公共机构所提供的数据，以及一些与芝麻信用交换信息的国 内主流 P2P 平台。

（3）用户提交的数据。用户可以在阿里巴巴官网、支付宝平台提交一些个人信息证明，包括实名认证、 个人收入、资产和工作证明等等。

**1.2.2芝麻信用分的局限性**

（1）信息壁垒

大部分国内外的征信体系最大的问题在于信息壁垒。数据的不完整对于整体信用评估是非常致命了，这会直接导致整套系统失去可行度。根据马太效应，如果这家信用评分机构数据越匮乏，愿意和它合作，开放自己的数据库的企业或政府机构也就越少。对于采用信用评分的放贷机构而言，如果不同征信机构对相同的用户评级结果不一致，市场销量无疑会降低。这可能造成的“多头借贷”现象，也使国家金融风险增加。

（2）信息安全

很多互联网巨头自嘲中国人是世界上最不注重个人信息隐私的群体。这点其实无可辩驳，大量用户面对《用户条例》，毫不犹豫地选择“我已阅读以上条例”，在毫不知情的情况下将个人信息提供给独角兽企业们。不过近几年，互联网的崛起，铺天盖地的新闻，让越来越多人意识到保护个人信息的重要性。但是不少国际黑客，依然试图攻破国内企业的数据库，从而获得个人信息，得到不菲的收入。一些企业将用户的重要信息明文保存在数据库中，从而导致了信息泄露。

（3）处罚力度

大部分用户只享受到了芝麻信用分给我们带来的便捷，没有意识到使用这款产品其实也是有相应责任的。信用机构往往只考虑了用好处吸引用户使用这款产品，而不强调或者建立相应的惩罚机制。部分用户为了提升自己的信用分，不停地互相转账，从而抬高自己的信用分。这些钻空子的行为缺乏监管和惩罚力度，必然导致信用分的局限性[4]。

除了国内的征信，我们也可以参考一下国外的征信情况。

**1.3 FICO系统**

**1.3.1 Fico评分系统**

美国作为全世界最发达的资本主义国家，也是最早的信用卡发源地，个人信用评估也很早地起步了。Fico评分系统只要有Fair Isaac Company提出，由此得名。Fair Isaac前后提出来三套评分系统，三套系统分别由美国三大信用管理局使用。

同一个用户在三套系统中的得分可能不尽相同，但是相差无几。Fico评分主要是用于贷款的发放。传统通过人工审核难以避免地会有个人偏见，比如性别，宗教，肤色，国籍等等。引入Fico系统后，系统可以客观公正的评价申请人的各项指标[6]。

**1.3.2 Fico评分的模型**

FICO 评分模型有五类主要因素组成, 包括了客户信用偿还历史、信用账户数、客户使用信用的年限、 正在使用信用类型、 新开设的信用账户[1]。各部分的权重如图所示：

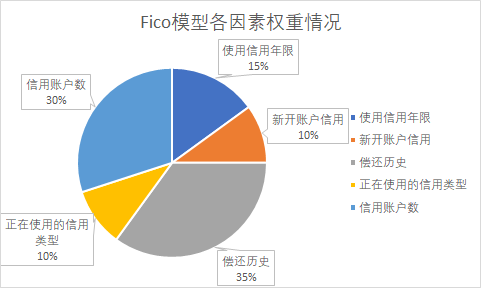


图1.1 Fico模型个因素权重比例情况

可以看出，偿还历史对于Fico模型而言是最为重要的因素，如果一个人的历史信用不佳，那么他下一次得到贷款的可能性也就降低。另外年限也是一大考量元素之一[7]。

1.4 中国当前征信行业的机遇与挑战

1.4.1互联网金融的机遇和挑战

对于P2P借贷行业，大学生无疑使一个非常庞大的经济群体，截止到2014年，我国的互联网金融消费市场已经达到了96,9亿元人民币。如果数字属实的话，那么截止到现在，这个数字应该已经超过1000亿元人民币大关[8]。同时，大学生也是思想最为前卫的群体，经济的发展和生活质量的提高都带动了年轻人的消费力，与之产生矛盾的是不匹配的消费能力，贷款则可以适当缓解这种尴尬的局面。但是事实上，一些自力雄厚的银行在很早之前就已经开始争夺大学生信用卡市场，但是由于过高的坏账率，不得不叫停。现在很多大学生信用卡美其名曰信用卡，其实额度为0，没有提前消费的能力，终究还是和借记卡一样，需要先存钱。

如果有良好的信用评级机制，银行可以提供专业的服务，学生也可以得到合理的利率。

1.4.2大数据对人才市场的优势

领英（LinkedIn）作为世界数一数二的互联网职业招募平台和职业社交平台，通过互联网实名社交将人才与企业联系起来。如今的领英，不仅仅是一个让机构发布信息，展现自我的平台，也是行业精英寻觅属于自己的高薪职业的渠道，职场小白也可以在上面学习技能，拓展人脉，所谓‘networking’。在领英上，不仅仅是毕业之后的工作情况，即便是小小的志愿者经历，或者不知名的证书，都可以在领英上进行认证。虚拟的徽章和认证，仿佛是一针强心剂，促使大量用户在领英平台上展现自己。不难想象中国大学生，也会乐此不疲。况且，如果可以连接校内系统，展示学生每个学期，每个学年，每节课的表现，以及诚信情况，奖励情况，志愿者情况，这样的系统会非常有市场。

1.5研究的目的和意义

1.5.1研究目的

近期，大学生贷款的话题一直非常热议，很多人关注与大学生超前的消费意识和与之不匹配的偿还能力，也有人批评一些灰色的借贷机构，通过文字游戏，蒙蔽缺乏社会经验，手头又不宽裕的大学生。国家出台了相应的法律法规约束这些机构，但是凡事宜疏不宜堵，如果可以客观合理地提过一个第三方评级，让正规银行进行参考，放贷的时候就有参考价值，就可以达到多赢的局面。

另外，对于用人单位，校招的时候凭借一张简历很难对人才进行筛选，如果学校方面可以通过自己的数据库，开通可以让授权的用人单位实时查询，那么将极大地提高人才招募的效率。

1.5.2研究意义

迎合2013年中国银行颁布的《征信业管理条例》（下文中简称《条例》）的精神，维护市场环境，保证大学生合法权益，开发一套完整可靠的信用评价体系非常重要。大学生信用评价系统可以通过学校的数据库，加上严格的审核制度，建立健全全面可执行的评价模型。该模型，可以提高大学生对于自己信用行为的积累意识。对于用人单位，可以通过最终的分值从不同角度了解大学生的在校行为，比简历更加全面可靠。同时，对于正规的借贷机构而言，也可以在此系统中判断是否可以发放贷款。

1.6研究难点

获取学生各种校内信息的过程较为复杂，比如图书馆借阅信息，教超消费，志愿者经历等等。其中如果稍有不慎，就会有泄露个人信息的隐患，与征信的初衷背道而驰。

同时，中国教育界还没有完全脱离以分数看高低的模式，有些同学并不重视信用分的重要性，如何在征信的过程中严肃而不失乐趣，是吸引大学生关注征信业务的重要指标。有些论文中，推荐采用一定的宣传模式，比如通过校园广播，屏幕等传统手段，或者公众号，微博等新媒体工具；除了宣传，最好也将征信纳入大学生的各类活动中，比如新生欢迎会、培训、干部竞选、就业指导等等。

最困难的莫过于指标和权重的选择。对于招聘企业和贷款方向的指标是不一样的，就像Fico的三套系统一样，对于不同需求，采用的指标和权重应该是不一样的。笔者的构想是两个系统，一个针对用人单位的在校生校内表现得分，一个是正对大学生借贷的个人收入情况评分。这个系统有共同考量的因素，也有单独考量的因素。两者分开可以保证两个系统用户都可以得到最可靠的结果。

第二章 开发方法和技术

**2.1开发方法理论**

信息系统，即信息传递交流系统，信息系统通常解释为有目的、和谐地处理信息的主要工具，它对所有形式的信息进行收集、处理、存储、传输和输出。收集：获取和收集原始数据的活动，包括手动输入和自动输入；处理：对数据进行处理，使得获得新的结构与形态或产生新的数据；存储：采用信息存储技术，将有用的信息进行存储保管，以便系统随时调用；传输：把信息从一处传到另一处，不改变信息本身的内容；输出：将系统处理后获得的有用信息按照一定要求和形式输出到某种介质上。

信息系统具有以下特征：目的性、系统要素、相关性、结构层次性、整体性、环境适应性。现如今基本将组织中的信息系统分为四个层次：运行层系统、知识处理层系统、管路控制层系统、战略层系统。不同类型的信息系统彼此相互联系，负责信息系统各个过程的分工。一切都会产生、发展、成熟和消亡。施工企业信息系统项目在更新过程中也不例外。随着生活环境的变化，在使用过程中需要不断的维护。当它不再适应时，它将被消除并采用新的系统，而不是成为旧系统，这一周期被称为信息系统的生命周期。在信息系统中经常使用生命周期法将系统建设过程分解为五个阶段，并且对每个部分的目标、活动、内容制定具体规定。五个阶段分为以下：

（1）系统规划阶段：

对组织的环境、目标和现有体系的现状进行了初步调查。根据组织目标和发展战略，确定了信息系统的发展战略，并对新系统建设的需求进行了分析和预测。考虑到新系统建设的各种制约因素，研究了新系统建设的必要性和可能性，分析了替代方案的可行性。系统建设计划和实施计划编制成系统规划报告。

（2）系统分析阶段：

根据系统规划报告确定的范围，对当前系统进行了详细的研究，描述了当前系统的业务流程，指出了当前系统的局限性和不足，确定了新系统的基本目标和逻辑功能要求，即提出了新系统的逻辑模型。系统分析阶段的成果体现在系统分析说明中。

（3）系统设计阶段：

系统设计阶段的任务是根据系统规范中规定的功能要求，结合实际情况，设计实现逻辑模型的技术方案，即设计新系统的物理模型。本阶段的技术文档是系统设计规范。

（4）系统实施阶段：

根据系统设计规范的要求，系统设计实施阶段具有多个相互关联、相互制约的任务同时进行的特点。系统的实施必须根据实施计划，认真安排，合理组织，分阶段实施。各阶段应编写实施进度报告。系统测试结束后，编写系统测试分析报告。

（5）系统运行和维护阶段：

系统投入运行后，需要定期进行维护和评估，记录系统运行情况，根据一定的规则对系统进行必要的修改，并对系统的质量和取得的效益进行评价。对于不能修改或难以修改的问题，将其记录并定期编制成新的需求建议，从而为新的信息系统项目规划做准备。

信息系统的开发是一项复杂的系统工程。它不仅涉及计算机技术，还涉及管理业务、组织和行为。这就加大了开发一个信息系统的工作量和工作难度。而掌握合适的系统开发方法对有效实现信息系统的开发会很有帮助。需要解决的问题不同，对应的信息系统规模与技术复杂度也会不同。不同类别的信息系统各有其适合的开发方法，常用的开发方法有：

**2.1.1原型法**

原型法是一种基于4GL的快速模拟方法，多被用于小型局部系统或处理过程比较简单的系统设计到实现的环节。原型开发信息系统的基本思想是首先总结用户提出的初始要求，然后构造一个合适的原型并运行它。 之后，通过系统开发人员和用户对原型操作的不断分析，修改和讨论，不断扩展和改进系统的结构和功能，直到获得满足用户要求的系统。

**2.1.2面向对象开发方法**

面向对象的方法侧重于围绕对象的系统分析和系统设计，然后使用面向对象的工具来构建系统。 这种方法可以普遍应用于各种类型的信息系统开发，但它不能参与以前的系统分析开发。 面向对象的开发过程包括五个阶段：系统调查和需求分析，问题的性质和问题分析，详细的设计问题，编程和系统测试。

**2.1.3结构化开发方法**

结构化开发方法又称为SSA&D，是自顶向下的结构化方法、工程化的系统开发和生命周期方法结合的产物，是至今为止所有开发方法中应用最广泛、最成熟的系统开发技术。宠物店系统的开发也应用了这种开发方法。

结构化系统开发方法擅长发现和整理系统调查、分析中的问题及疏漏，便于开发人员准确地了解 业务处理过程，有利于与用户一起分析新系统中适合企业业务特点的的新方法和新模型。

结构化系统开发方法的基本思想是利用系统工程思想和工程方法，根据用户至上，结构化，模块化和自上而下的原则对系统进行分析和设计。 具体而言，整个信息系统开发过程首先分为几个相对独立的阶段，如系统规划，系统分析，系统设计和系统实现。 每个阶段都有明确的任务和目标，每个阶段可以分为几个任务和步骤，逐个完成任务，以达到预期目标。

结构化方法的突出优点是，它强调系统开发过程的完整性和整体性，强调在整体优化的前提下考虑具体的分析设计问题，即自上而下的视角，减少失误和失败 系统开发 可能性。 同时，随着计算机应用的深入和计算机软硬件的快速发展，系统开发的周期要求越来越高，短时间和低成本成为系统开发的基本要求。 面对这一新趋势，强调标准化和标准化的标准化系统开发过程逐渐暴露出缺点和不足，如开发周期过长，难以适应快速变化的环境，所使用的工具落后。

毫无疑问，结构化系统开发方法存在不可避免的缺点，然而这种开发方式其严密的理论基础和系统工程方法，依然被广泛使用，且仍然是系统开发中不可缺少的一部分。结构化系统开发方法经常应用在大型、复杂的影响企业整体运作的信息系统的开发项目中，也经常应用在政府项目中。目前它仍然是一种被广泛采用的系统开发方法。

**2.2开发环境**

Python :3.7.2

Django: 2.1.0

Mysql : 5.7

**2.2.1 Python**

Python是一种计算机程序设计语言，是一种动态的，面向对象的脚本语言，最初设计用于编写自动脚本（shell），随着版本的不断更新和新的语言功能的增加，这些脚本越来越多地用于独立的大型项目的开发和维护。如今，随着人工智能进入国家的视野，Python在程序时间的地位进一步提高。由于Python社区贡献了大量高质量的人工智能和机器学习代码，越来越多的人加入了人工智能的大军，也就是Python的学习。同时，大数据为更多的商家提供商机，数据分析的能力，如Scipy，Numpy, Pandas，Tensorflow等等包变得耳熟能详。这些都促使了Python不仅仅好用，而且有人用，马太效应之下，Python正在变得语法流行。

本次开发使用的是Python 3.7.2，最新版本。基于anaconda的最新版本，优点在于anaconda可以更好的管理Python封装问题。

**2.2.2 Django**

Django是一个用Python编写的开源Web应用程序框架。 采用MVC的框架模式，即模型M，视图V和控制器C.最初开发用于管理劳伦斯出版集团的一些基于新闻内容的网站，即CMS（内容管理系统）软件。 它于2005年7月在BSD许可下发布。该框架以比利时的Gypsy爵士吉他手Django Reinhardt命名。

Django是一个基于MVC结构的框架。 但在Django中，接受用户输入的控制器部分由框架本身处理，因此Django更关注模型，模板和视图，称为MTV模式。 各自的职责如下：

|  |  |
| --- | --- |
| 表2.1 Django MVC模式图 | |
| 层次 | 职责 |
| 模型（Model），即数据存取层 | 处理与数据相关的所有事务： 如何存取、如何验证有效性、包含哪些行为以及数据之间的关系等。 |
| 模板(Template)，即表现层 | 处理与表现相关的决定： 如何在页面或其他类型文档中进行显示。 |
| 视图（View），即业务逻辑层 | 存取模型及调取恰当模板的相关逻辑。模型与模板的桥梁。 |

从上面的语句中可以看出，Django视图不处理用户输入，这部分由JavaScript和HTML完成，而只决定向用户显示哪些数据。Django模板仅确定如何呈现Django视图指定的数据。或者说，Django进一步将MVC中的视图分解为两部分：Django视图（对于views.py）和Django模板（对于templates文件夹下的HTML文件），它们确定“要显示的数据”和“如何显示”，以便Django的模板可以根据需要进行替换，而不仅仅是内置模板。

至于MVC控制器部分，它由Django框架的URLconf实现。 URLconf机制使用正则表达式匹配URL，然后调用适当的Python函数。 URLconf对URL规则没有限制。任何用户都可以将它们设计为任何URL样式，无论是传统的，RESTful的还是替代的。框架封装了控制层，没有其他任何东西与数据交互。该层是数据库表的读取，写入，删除和更新操作。在编写程序时，只需调用相应的方法，与一些传统的框架相比可以说非常简便了。程序员自动将控制层交给Django。只写很少的代码来做很多事情。因此，它比MVC框架考虑的更深一步，因为我们的大多数程序员都在编写控制层程序。现在这项工作被移交给框架，只需要少量的调用代码，这大大提高了工作效率。

在美国，很多知名的初创企业在一开始，都是用Django作为框架。由于依附于Python，Django有很强的后台数据处理能力。例如著名社交网站Instagram（照片墙）。

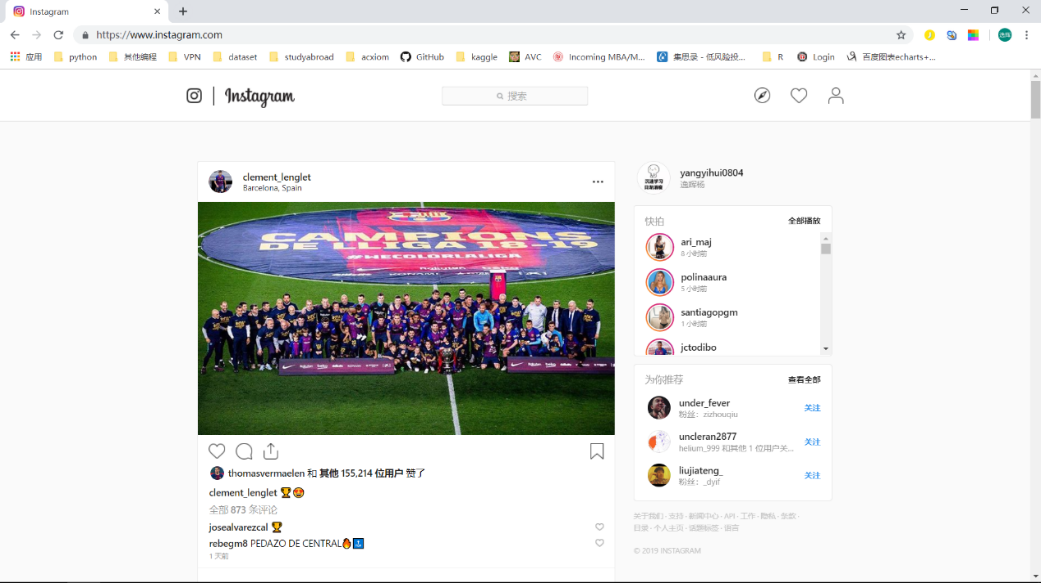


图2.1 Instagram首页

以及美国的大众点评Yelp。

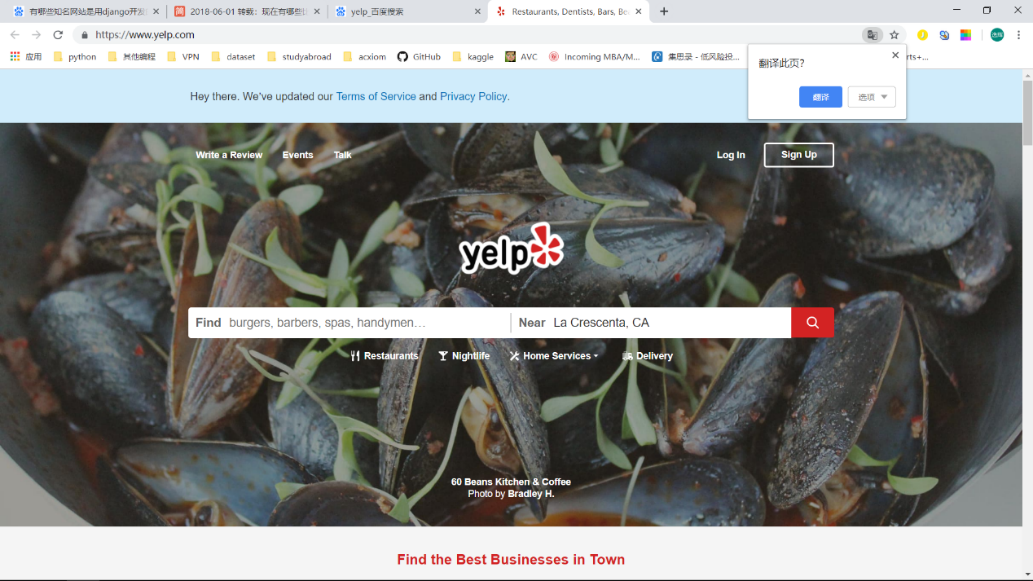


图2.2 Yelp首页

本次使用的django版本是django2.1.1版本。通过在anaconda中输入pip install Django,就可以下载并安装最新的django包，可以直接调用。

**2.2.3 MySQL**

MySQL是最受欢迎的关系数据库管理系统。 MySQL是WEB应用程序中最好的RDBMS（关系数据库管理系统）应用程序之一。 MySQL是一个真正的多用户，多线程SQL数据库服务器。 SQL（结构化查询语言）是世界上最流行和标准化的数据库语言，可以更轻松地存储，更新和访问信息。 MySQL是一种客户端/服务器体系结构实现，由服务器守护程序，mysqld和许多不同的客户端程序和库组成。

MySQL的主要目标是快速，健壮且易于使用。最初是因为MySQL的创始人需要这样一个SQL服务器，它可以处理大型数据库，供应商在任何廉价的硬件平台上提供数据库，但是以更快的速度开发了MySQL。

MySQL提供C，C ++，Java（JDBC），Perl，Python，PHP和TCL API接口;多平台支持，包括Solaris，SunOS，BSDI，SGI IRIX，AIX，DEC UNIX，Linux，FreeBSD，SCO OpenServer，NetBSD，OpenBSD，HPUX，Win9x和NT;各种数据类型，包括/ unsigned 1,2,3,4,8字节整数，FLOAT，DOUBLE，CHAR，VARCHAR，TEXT，BLOB，DATE，DATETIME，YEAR，SET，ENUM;非常灵活安全的权限系统，密码加密;用于Windows的ODBC接口，可以通过Access连接，而第三方开发人员提供各种ODBC驱动程序。但是，为了速度，健壮性和易用性，MySQL必须牺牲一些灵活和强大的功能，如子查询，存储过程和触发器，外键，事务处理等。因此，MySQL在某些应用程序中缺乏灵活性，但它也使其比其他数据库服务器快2到3倍。发器，外键，事物处理等。因而，MySQL在某些应用中缺乏灵活性，但这也使它对数据的处理速度较其它数据库服务器至少快2〜3倍。

在Mysql官网，可以看到使用Mysql服务的知名企业不计其数：

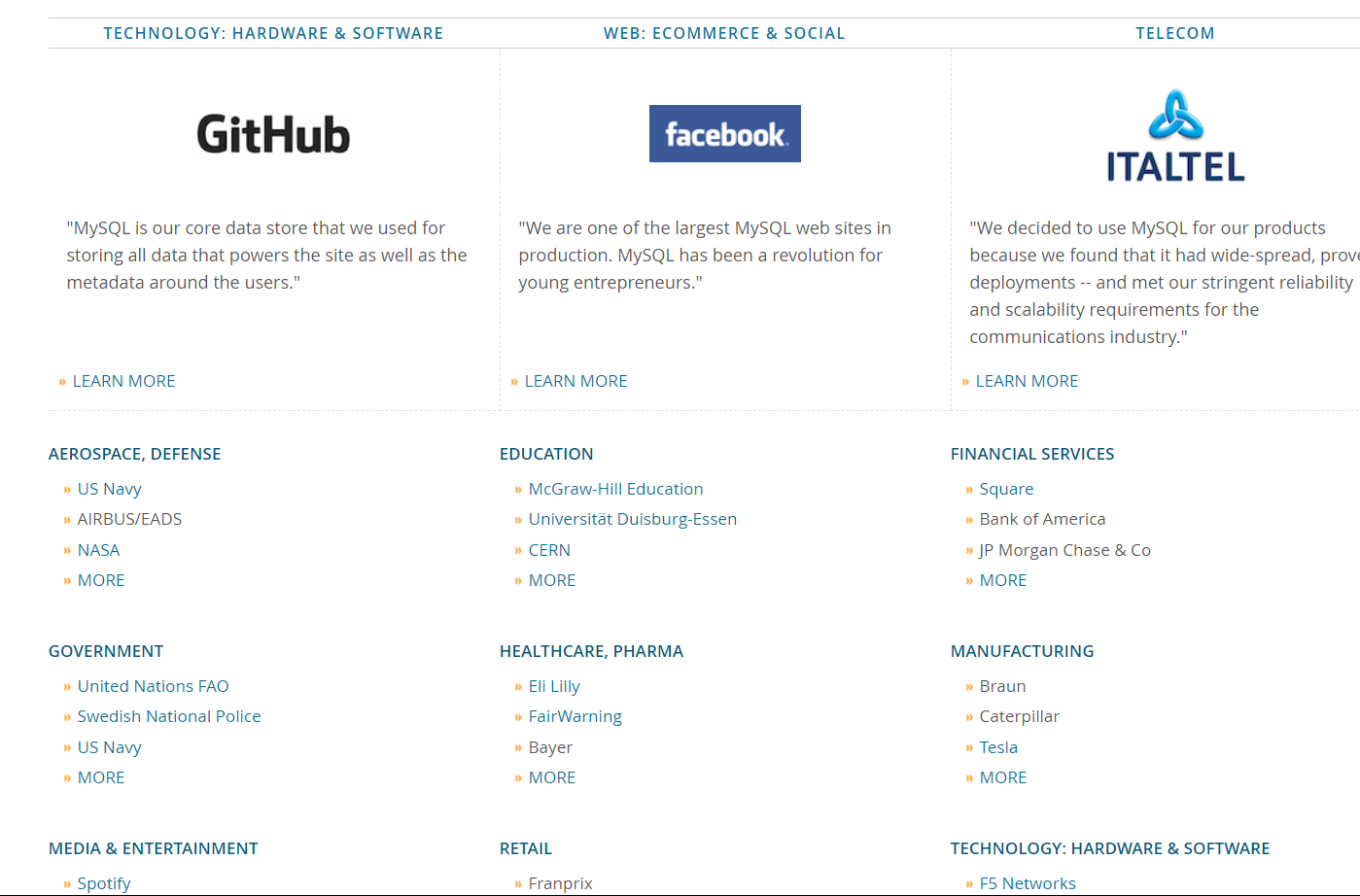


图2.3 Mysql首页

里面包含了不少耳熟能详的企业，如Github，Facebook，NASA，JP Morgan等企业。

本次使用的Mysql是Mysql5.7。

**第三章 大学生信用评估系统规划**

**3.1 大学生信息评估系统需求分析**

**3.1.1需求分析**

既然是以大学生信用为主题的系统，大学生就是避不开的话题。虽然现在唯分数论的思想已经不再可靠了，但是学习成绩的好坏依然在一定程度上反应了一个人的信用程度，所以成绩毫无疑问是必须纳入考虑的一大因素之一。为了便于查询成绩的来源，可靠地反应学生的学术水平，该系统也会包含一套简易的选课系统。教师可以为学生打分，每个学期的算术平均绩点就是奖学金的依据。辅导员可以根据学生每个学期的绩点，决定奖学金名单，学生在系统中确认申请。这套流程是模仿了上海理工大学的奖学金申请流程。

同时，为了倡导环保和循环利用，加入了二手商品交易这个子系统。旨在倡导学生，热心环保，注意回收，也可以为学生提供一个收回部分成本的机会。如今国内教材的价格远远低于国外，但是对于学生而言，能够循环利用书籍，收回一部分成本，总是一件值得庆幸的事情。更加重要的是，在这个过程中，学生会因为商品相符程度，回复态度，回复速度，交易的便捷性等等得到买家相应的分数，这些分数会相应计入人才招募和贷款发放的信用分中[9]。

信用的另一个维度是履约能力。在校园内，能够非常显著表现履约能力的就是图书馆借阅归还情况。在图书馆借书，可以体现一个学生对于知识的渴望，及时归还则反应了该学生履约的能力。对于不及时归还的，按照一定的规则进行扣分和罚款；对于长期不归还的，则当作丢书处理；如果主动承认丢书，则可以相应减免责罚。当然，对于一些学生因为客观原因不能及时归还的情况，向老师发起申诉，只要老师认可，则可以不予惩罚。这样也体现了客观公正，又不失人情[10]。

在国际知名招聘和职业社交网站LinkedIn中，技能认证是一项非常有趣的功能。在敬佩产品经理才思敏捷的同时，本系统也可以加入类似的功能。在领英中，技能认证是通过个人发布，得到上下级或者同事的认可，技能旁边的数字就会相应加一。在学校中，想要获得同学之间的认可太过简单，结果也有失公允。而且根据马太效应，一些朋友比较多的同学认可也比较多，因此认可的来源不太应该源自于学生之间。根据学校的课程，不难发现有些课程比较简单，但是有些课程可以扎扎实实学到有用的知识。有些同学也会抱怨在学校里没法集中注意力，没有学习的动力，学的东西得不到相应的证明，那么如果根据课程的不同，学生学习的结果不同，在征信平台上进行认证，那么无疑会使多赢的局面。例如，对于某些课程，如数据库。如果学生的成绩大于90分，则可以申请数据库技术认证。得到对应教师批准之后，相应的得到加分。

除了校内的成绩外，学生多半会参与一些课外活动丰富自己的课余生活。为了鼓励大家多参与志愿者活动和学校学生会工作[11]，将这两项加入模型中，是无可厚非的。当然这两项每种活动会有对应的分数，参与的次数也有限制，否则分值就会失去可信度。另外，很多学生对于学术研究有独到的兴趣，发表的论文数量和对于期刊的质量都会纳入信用分的考量。另外如果学生获得一些国家国际奖项，也可以进行申报，从而获得一些分数上的奖励[12]。

所得到的奖学金可以算入学生收入，从而让第三方贷款方得到学生的收支情况。当然只有收肯定不可靠的，所以比如从银行账户的调取学生每个学期的银行流水。

实习往往是学生通向职场的第一步，学生在实习期间的情况和收入情况也可以纳入其中。人才招募的信用分可以将实习的工种纳入考量，贷款发放的时候收入的水平可以视作贷款发放的重要水平。例如，一个有技术含量的实习得到的分数要比没有技术含量的得到的分数要高；实习超过一年的实习生，肯定要比刚刚开始实习的实习生要高；单日收入税后达到180元的学生得到的分数也要比日收100元的学生得分要高。

为了避免学生在考试中违纪，很多学校实行严格的校纪校规，对于严重违纪的学生不惜开除也要显示考试违纪是学生不可逾越的红线[13]。然而，不到毕业很多用人单位并不会调用学生档案，对于学生这方面的成绩了解不少非常透彻，因此可以在评分系统中体现出来。

**3.1.2 公式**

本着避免重复计算的原则，削弱马太效应对于分值的影响，人才招募和贷款发放的函数如下：

y\_人才招募 = F(学生年级，绩点，二手商品交易历史，技能，学生工作&论文，学生罚款清单，学生违纪情况）

y\_贷款发放 = G(学生年级，奖学金，银行流水，二手商品交易历史，校外工作证明，学生罚款清单，学生违纪情况)

**3.1.3 分值对照**

为了更好地明确每个指标衡量的标准和对应的分值，本文将使用Excel表格，对分支进行详细的描述。

（1）学生年级：

不同的年级可以得到的加分不同，越接近毕业，分数越高。但是在工作和放贷两种不同的场景中，各自的加分应当是不同的。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.2 学生年级对应加分情况 | | | |
|  |  | 人才招募 | 贷款发放 |
| 学生年级 | 大一 | 2 | 1 |
| 大二 | 4 | 2 |
| 大三 | 6 | 3 |
| 大四 | 8 | 4 |

（2）绩点：

学习成绩体现了学生的学习能力，在工作中，学习能力也是极其重要的因素，但是在贷款方面 似乎没有那么要紧，所以在贷款申请评分的时候，权重相应降低。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.3 绩点加分对应表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 绩点 | >4.0 | 10 | 5 |
| >3.8 | 8 | 3 |
| >3.5 | 6 |
| >3.0 | 4 | 1 |
| <3.0 | 2 |

（3）技能：

技能体现了学生在大学期间对于技术性课程的重视程度，不仅仅是学习能力的体现，也是考验学生在有限时间里如何对课程进行取舍。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.4 技能加分对应表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 技能 | >3 | 10 | 5 |
| >1 | 5 | 2 |
| 0 | 0 | 0 |

（4）银行流水

银行流水查看学生的消费能力和生活水平，在发放贷款是非常必要。同时，很多现实中的场合都需要查看银行流水，房贷车贷，办理护照签证等。对于学生而言，生活费是非常常见的资金来源，可以以此作为银行流水的参考，结合大学生每月生活费的水平，得到如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.5 银行流水加分对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 银行平均流水 | >2500 |  | 25 |
| >2000 | 20 |
| >1500 | 15 |
| >1000 | 10 |
| <500 | 5 |

（5）学生工作/论文/校外奖励：

学生工作和奖励非常多而复杂，经过一系列资料的查阅，考虑到某些奖励高不可及，也进行了一些整理和合并。同时也考虑到越是接触的人越多的学生，越是讲究自己的信用，所以曝光度是贷款发放考量的因素之一，相应的分值与人才招募相比也会增涨。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.5 学生工作/论文/校外奖励加分对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 学生工作 | 学生会干事 | 5 | 5 |
| 学生主席 | 10 | 10 |
| 学生副主席 | 8 | 8 |
| 志愿者每小时 | 1 | 0 |
| A类期刊 | 30 | 5 |
| B类期刊 | 25 | 3 |
| 国际期刊 | 50 | 10 |
| 市级奖励 | 10 | 10 |
| 国家级奖励 | 15 | 15 |

（6）学生罚款清单：

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.6学分罚款规则对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 学生罚款清单 | 逾期一次 | 10 | 20 |
|  | 丢书一次 | 15 | 25 |

所谓有赏必有罚，赏罚分明的制度才是最能让人信服的制度。人才招募的过程中要了解大学生在校的守信情况，从而对人才做出判断。贷款涉及真金白银，更加不能儿戏，因此贷款的扣分幅度更大。

（7）奖学金：

奖学金也是一个衡量学生经济水平的要素。很多学生可以通过一年的学习得到不菲的奖学金，以一定的比例算入大学生借贷评价体系很有必要。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.7 奖学金奖励规则对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 学生罚款清单 | 累计每100元 |  | 1 |

（7）二手商品交易历史：

上手商品交易的评价是由两个维度组成的，分别是卖家维度和买家维度。这里的考虑方式是卖家维度较为重要因为很多情况下只有买家会为卖家打分，卖家不愿意在打分，而必须有一个默认分值。因此初步结果如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.8二手交易评价规则对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 二手商品交易 | 买家评价 | 1 | 2 |
| 卖家评价 | 2 | 4 |

（8）校外工作：

校外工作主要就是实习工作，公司的大小固然由一定的关系但是非常难以统计，这里暂时用工作年限为考量依据。为了避免重复计算，贷款发放的时候相应分值会略有下降。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.9 校外工作加分对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 学生校外工作 | 累计工作12个月 | 20 | 10 |
| 累计工作9个月 | 15 | 8 |
| 累计工作6个月 | 10 | 6 |
| 累计工作3个月 | 5 | 4 |

（9）违纪

考试违纪作弊是绝对不可以容忍的问题，因此必须通过为了让用人单位及时知晓相关的情况[14]。

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.10 违纪作弊扣分对照表 | | | |
|  |  | 人才招募 | 贷款发放 |
| 学生校外工作 | 夹带旁抄 | 100 | 100 |
| 偷看 | 80 | 80 |
| 协助作弊 | 120 | 120 |

为了得到一个完整的分数，用思维导图的方式展示每个维度在所占的百分比：

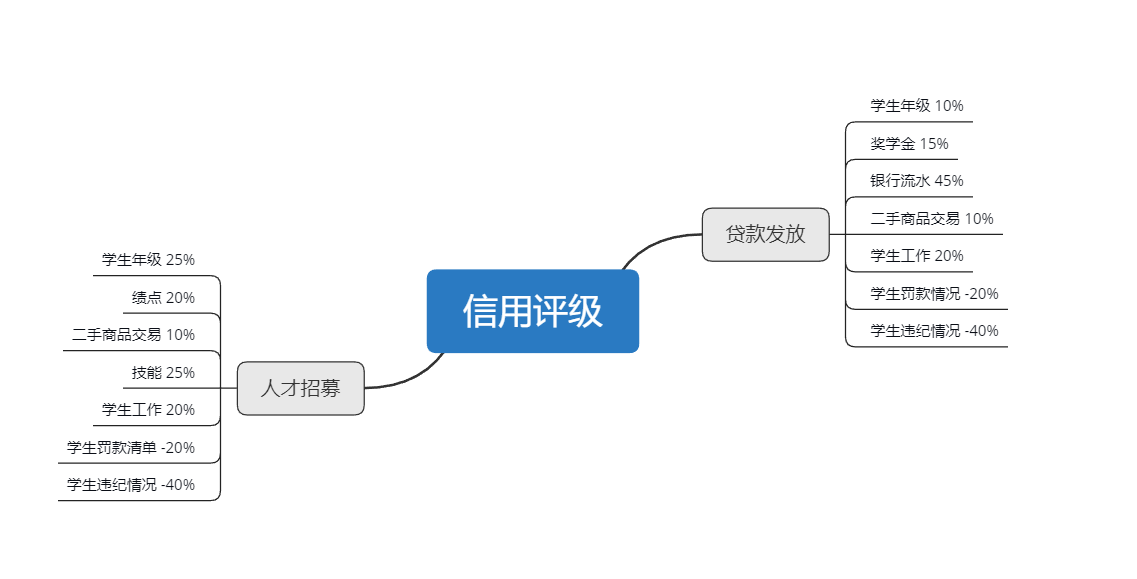


图2.1信用评级思维导图

|  |  |  |  |
| --- | --- | --- | --- |
| 表2.11信用评级规则对照表 | | | |
| 信用评级 | 贷款发放 | 学生年级 | 10% |
| 奖学金 | 15% |
| 银行流水 | 45% |
| 二手商品 | 10% |
| 学生工作 | 20% |
| 学生罚款情况 | -20% |
| 学生违纪情况 | -40% |
| 人才招募 | 学生年级 | 25% |
| 绩点 | 20% |
| 二手商品交易 | 10% |
| 技能 | 25% |
| 学生工作 | 20% |
| 学生罚款清单 | -20% |
| 学生违纪情况 | -40% |

**3.2大学生信用评价体系 UC矩阵**

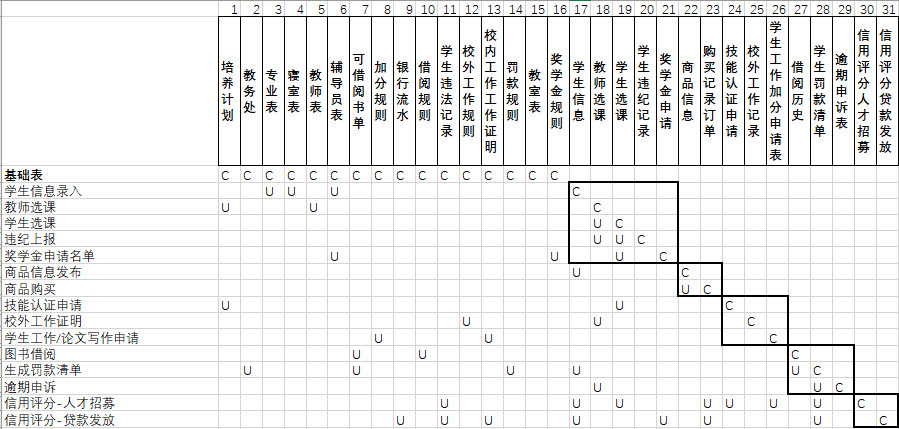


图3.1大学生信用评级体系UC矩阵

从UC矩阵可以看出，整个系统根据需求分析被划分为5个部分，分别是教务子系统，二手商品交易子系统，学生个人认证子系统，图书子馆系统，信用评分子系统。

**3.2.1教务子系统**

教务系统是每一个涉及学生的系统不可避免的部分。教务处方面讲学生的各项信息录入数据库，包括编号、姓名、密码、入学年份等等。学生通过账号密码登录后可以进行一系列操作。

首先是教务处对培养计划进行修改，看看是否需要修改开课数量，如果需要则更新有关课程上限。

之后教师进行选课。教师根据培养计划，选择课程同时生产教师选课表。每节课都会有相应的教师选课上限，如果高于这个上限，教师则不可选课。学生根据教师选课的情况自行选择课程。教务处根据每节课的选择情况安排教室，如果选课的人过少，则取消该课程。通过教务处确认的课程，学生可以查看到自己的课程安排情况，在何处上课，什么时候上课等等。每个学期中期，教务处会提醒任课教师安排期末考核方式，例如上传试卷，随后进行审卷等操作，最后安排考试教室。教务处会根据教室考试学生的容量安排教师，保证学生与学生之间的间隔。

考试期间，如果有学生违纪作弊，一经发现，教师通过系统上传相应原因，取消其资格等等操作。

考试结束之后，教师根据一定的规则打分，输入学生选课表，生成成绩单。对于成绩大于90分且课程可以参与技能认证的课程，学生可以进行技能申请。如果教师同意了，那么可以得到相应的标识和加分。

下个学期开始时候，辅导员根据学生绩点的算术平均数对学生的成绩进行排名，一定比例的学生获得奖学金申请资格。学生点击申请之后，可以获得相应的记录和金额，金额会在学生账户中体现。

**3.2.2二手商品交易子系统**

电子商务在中国迅速发展，网络将商品和顾客快速简单的联系起来。更加重要的就是提供了一个交换的平台。那么在校园内，提供一个交换的平台，可以极大的促进循环利用，降低学生开销。如果可以进一步开发和畅想的话，完全可以像早期的Facebook一样做成校内网，帮助同学间互相了解，互相认识，甚至共同创业。

学生可以发布商品，例如书籍，将部分页拍照时上传，让对方判断新旧以及是否有笔记，这是模仿先咸鱼的流程。辅导员则扮演了审核的角色，如果照片和商品实际不符合，贩卖商品违法违规，及时阻止。作为买家的学生，可以浏览被辅导员审核过的商品，根据预定的交易地点进行交易。之后改变各自账户中的金额。买家卖家的评价默认值为满分，用户选择的评价会是最终的结果，计入交易记录。

**3.2.3学生个人认证子系统**

奖学金申请是辅导员发起的，所以不会放在学生申请子系统中。该系统包括学生技能认证的申请与批准，校外工作证明的申请与批准，学生工作、论文发表证明的申请与批准。

学生根据自己的实际情况发起申请，附上相关证明的电子版或者书面版本当面认证。认证之后可以该项加分可以算入总分当中。以技能认证为例，学生可以在自己的界面中看到分数大于90分，且可以参与认证的项目。提交申请后，教师可以根据该学生的表现选择是否通过申请。申请通过后，学生会得到相应的认证。

**3.2.4图书子馆系统**

图书借阅是学生授信模块分数的主要来源，如果做好这块的系统尤为关键。学生可以通过系统查看到可以借阅的书籍，点击借阅后，改变这条书籍的属性为不可借阅。同时生成根据学生的学历，在借阅历史中插入一条借阅数据。例如本科生的借阅时长为3个月，研究生可以达到5个月。在借阅期限之内，学生可以前往图书馆，由图书馆管理员进行归还操作。但是如果没有按时归还，则会在罚款系统中生成一条记录，如果超过双倍的期限内没有归还，则强制执行罚款。当然考虑到学生对书籍的需求可能是长期的，也有可能是暂时遗忘，并非故意，所以可以想老师提出申请，如果认可老师同意了，则不用执行该罚款。学生也可以主动表示自己丢书了，进行相应的惩罚。

**3.2.5信用评分子系统**

由上文可知，评分系统是由人才招募评分系统和银行放贷系统组成。在每个学期结算的时候，由辅导员发起，结算该学期所有相应的分数模块，得到最终的分数。以人才招募为例，

y\_人才招募 = F(学生年级，绩点，二手商品交易历史，技能，学生工作&论文，学生罚款清单，学生违纪情况）

在学期结束，辅导员发起结算，调用学生信息表，学生选课表，二手商品信息评价，技能认证表，学生工作和学生论文发表情况表，学生罚款清单，学生违纪情况，学生违法情况表数张表，根据上文 中所描述的比例每个学生会得到一个分数。校外的社团体，可以通过一定的验证过程，得到查看学生分数的权限，从而做出判断。这个权限也可以当作营收的一部分，在未来，企业也可以通过支付一定的广告费用，增长曝光度，增加学生投递的比例。

**3.2.6其他**

受到数据来源的限制，学生的银行流水和公安违法记录暂时无法获取，本系统中将这两块部分直接按照满分计算，但是接口依然会留给后面的同学进行开发。

**第四章 大学生信用评分系统业务流程分析**

**4.1泳道图**

泳道图是将模型中的活动按照职责组织起来。这种分配可以通过将活动组织成用线分开的不同区域来表示。由于它们的外观的缘故，这些区域被称作泳道。它可以方便的描述企业的各种业务流程，能够直观地描述系统的各活动之间的逻辑关系，利于用户理解业务逻辑。

**4.2图例**



图4.1 开始图例

表示开始和结束



图4.2文件图例

表示文件



图4.3动作图例

表示动作

图4.4业务流图例

表示业务流

**4.3子系统业务流程图**

**4.3.1教务系统流程图**

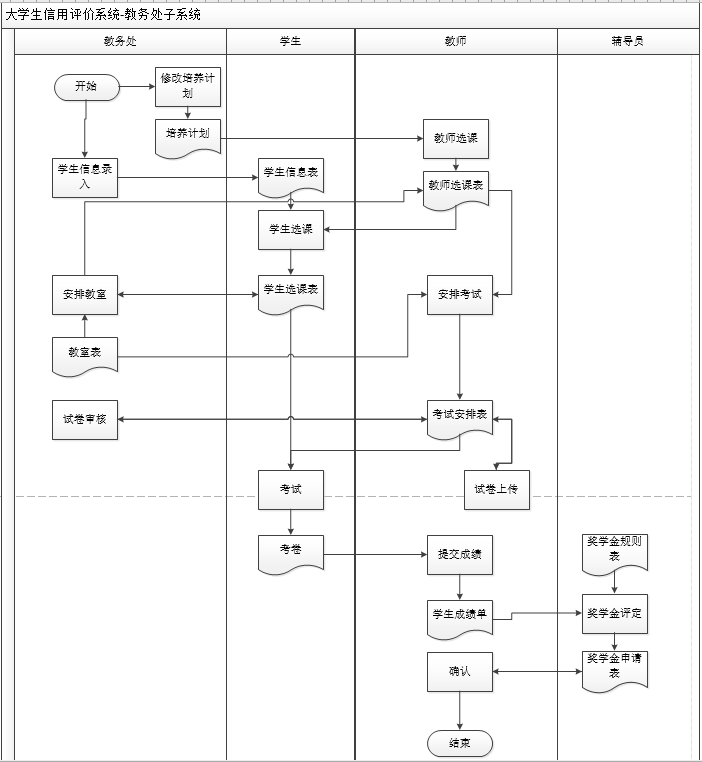


图4.5教务子系统泳道图

系统从教务处开始。教务处根据实际情况，选择是否要增开课程，如果需要就改变课程数量，则可以改变培养计划中各项课程的上限，从而限制教师选课。教师选课的时候，则根据培养计划和培养计划课程相对于的数量进行选择，如果当前数量已经等于上限，则教师页面将不会看到可选课程。教师课程选择完毕后，学生可以通过查看教师选课的结果从而生成学生选课表。但是由于教室尚未安排，所以教室这个属性是null。教务处根据教室表，即教室的容量，安排教室。安排完成后改变教室的可用属性。如果选课人数过少，则取消这节课。

学期中旬，教务处安排教师上传考卷。教师上传试卷后，教务处会进行试卷审核。审核通过之后，安排学生进行考试。根据选课的人数安排考试教室，由于考试的座位和普通上课的作为有所不同，调用教室表属性的时候也不是同一个属性。安排妥当之后学生可以查看到自己的考试安排。

考试结束后，教师可以录入成绩。如果考试期间学生有作弊行为，老师也可以录入学生的作弊记录，记录在案。相应地，这么门课的成绩会变成0分。

学期结束之后，辅导员根据奖学金规则表对优秀学生进行评比。学生可以在自己的界面查看自己是否可以申请奖学金。点击确定后，奖学金会发放到各自的学生账户。

**4.3.2二手商品子系统业务流程图**

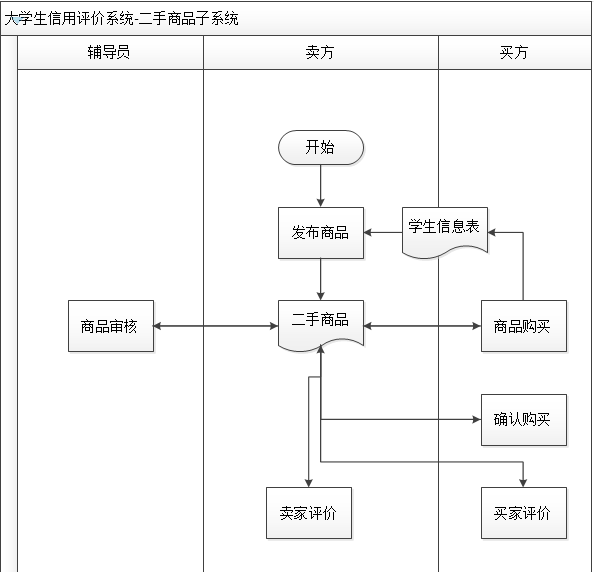


图4.6二手商品子系统业务流程图

二手商品子系统是一个别具一格的创新设计，考虑到学生偶尔会冲动消费，通过类似于咸鱼的二手平台收回一定的成本，一定程度上缓和“剁手”带来的不利影响，同时，对于教科书之类的商品，循环利用，促进绿色发展。

从流程上，作为卖方的学生，发布商品，发布的内容包括但不限于商品名称，商品名称，商品原价（标明的价格），商品现价，交易地点，交易时间，联系方式，商品照片等等。为了保证商品的合法有效，必须通过辅导员审核这一环节。辅导员需要审核商品是否合法，时间是否合理，交易地点是否安全等操作，辅导员审核通过之后，方可交易。

作为买方的学生，在自己的界面可以看到被辅导员审核过的商品。点击相应的按钮后，款项从自己的金额中扣除。但是相应的款项不会直接打到卖方的账户。只有在交易按照原定计划进行的情况下，买房再次点击完成交易后，款项才会相应出现在卖方账户。之后，双方需要进行打分操作。卖方买方都需要为对方的行为进行评价，且短时间内互相不知道对方的打分。

交易结束之后，作为卖方和买方的评分都会计入各自的个人信用评级中。

**4.3.3学生个人申请子系统业务流程图**

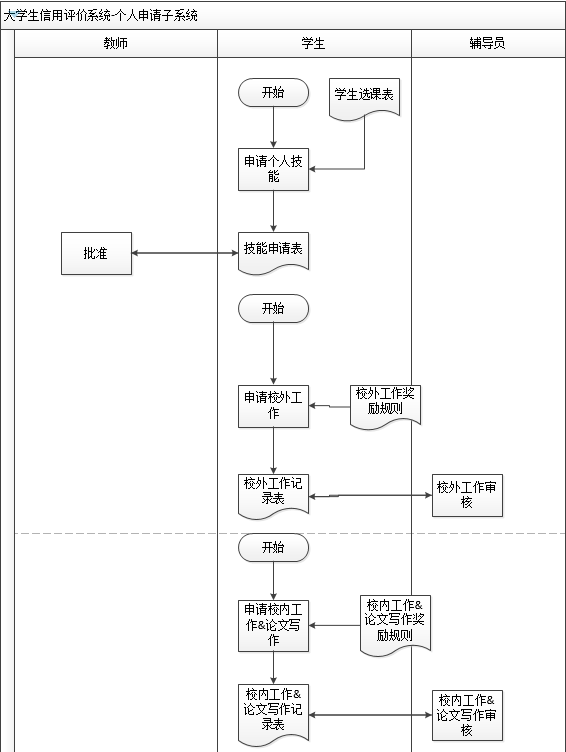


图4.7学生个人申请子系统业务流程图

从流程上而言，个人申请系统是比较简单的，不外乎调用规则表，申请，批准申请而言，但是犹豫申请的种类繁多，信息入口也比较繁杂，所以被单独做成一个子系统。

学生可以在自己的选课记录中，选择分数大于90分且在在培养计划中可以被认证的课程。选择后，在教室界面会出现相应的记录。教师可以根据对学生的印象进行确认或取消。教师确认之后，学生可以在自己的个人界面看到有关的记录。在加分系统中也会有所体现。

外校工作申请，需要外部文件的输入，例如实习证明。同时调用申请规则表，辅导员会对该条记录进行审核。审核通过后，学生也会获得相应的加分。如果审核不通过，学生也可以对该记录进行修改。

校内工作和论文发表同样调取相应的规则表，上传有关证明。这里的规则表可能比较复杂，例如学生会会长需要院团委的证明，A级论文则需要出版社的有关证明。同样都交给辅导员进行审核。审核通过后，学生也会获得相应的加分[15]。

**4.3.4图书馆子系统业务流程图**

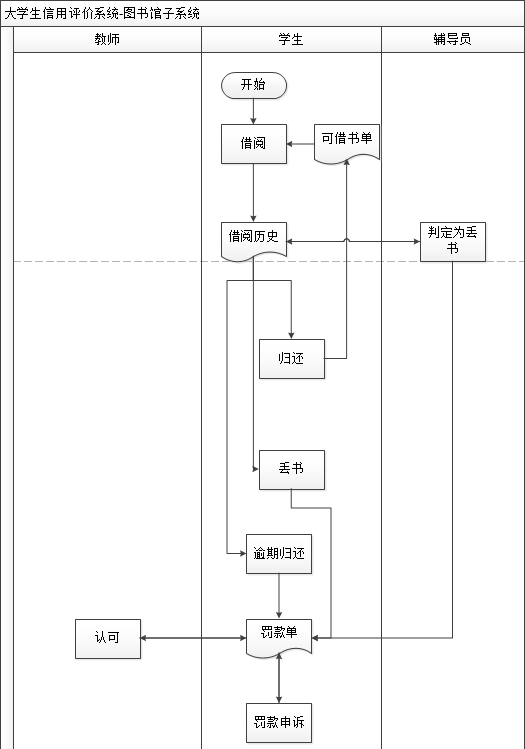


图4.8图书馆子系统业务流程图

图书馆系统，作为整套系统中最为关键的部分，是信用分的主要来源。良好的读书习惯可以增强人的涵养，但是忘记归还或者丢书，就是非常不应该的事情了。

图书管理子系统从学生调用可借图书表单开始，生成借阅历史。如果学生在规定日期内归还，则仅仅修改可借书单中的属性，从而结束该条记录。如果学生逾期归还，同样修改可借书单中的属性，但同时生成一张罚款单。

学生也可以主动承认错误，比如书丢了。那么不再修改可借书单的属性因为该书已经无法再次借阅。同时，生成一张罚款清单。如果学生选择缴纳罚款，则扣除学生名下账户相应的金额。

当然我们也要考虑到有些学生没有归还书籍是因为教学用途，这种情况下可以申请向老师申请。只要得到老师的批准，就可以免去罚款。

**4.3.5信用评分子系统业务流程图**

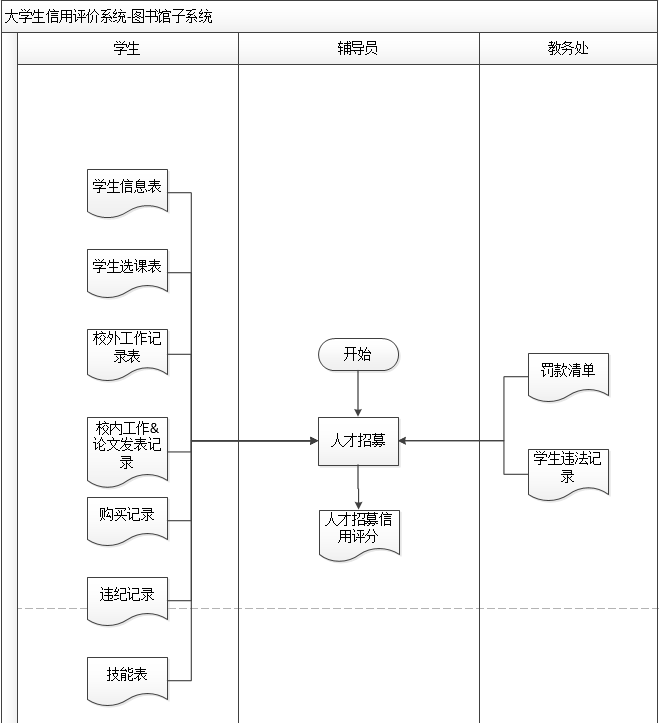


图4.9信用评分子系统业务流程图1

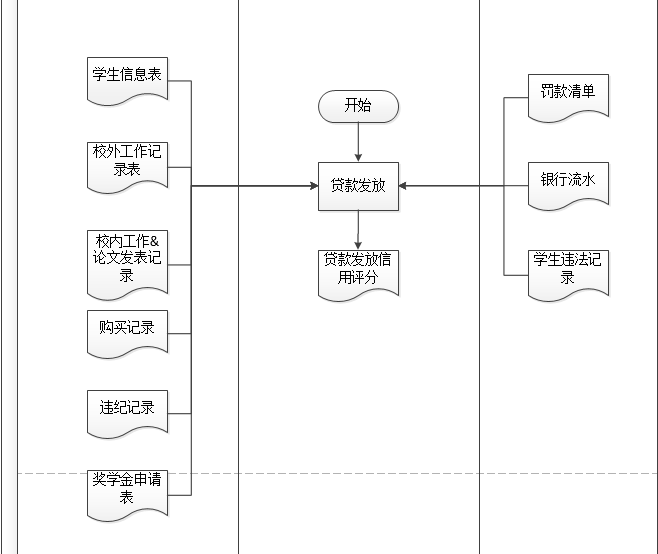


图4.10信用评分子系统业务流程图2

和奖学金一样，信用评分每个学期评价一次。综合每个学期、每个学年的表现，根据相应的规则给与评分。介于图中已经很清晰的表面需要调取的表单和生成的表单，在此不多做赘述。不过值得注意的是，两个评价体系所用的表不完全相同，在系统编写中，需要特别注意。

**第五章 大学生信用评分系统数据流程分析**

**5.1数据流程图**

**5.1.1数据流图图例**



图5.1外部实体

表示外部实体



图5.2加工过程

表示加工过程

图片

图5.3数据表单

表示数据表单

图片

图5.4数据流

表示数据流

**5.2数据流程图**

**5.2.1教务子系统数据流程图**

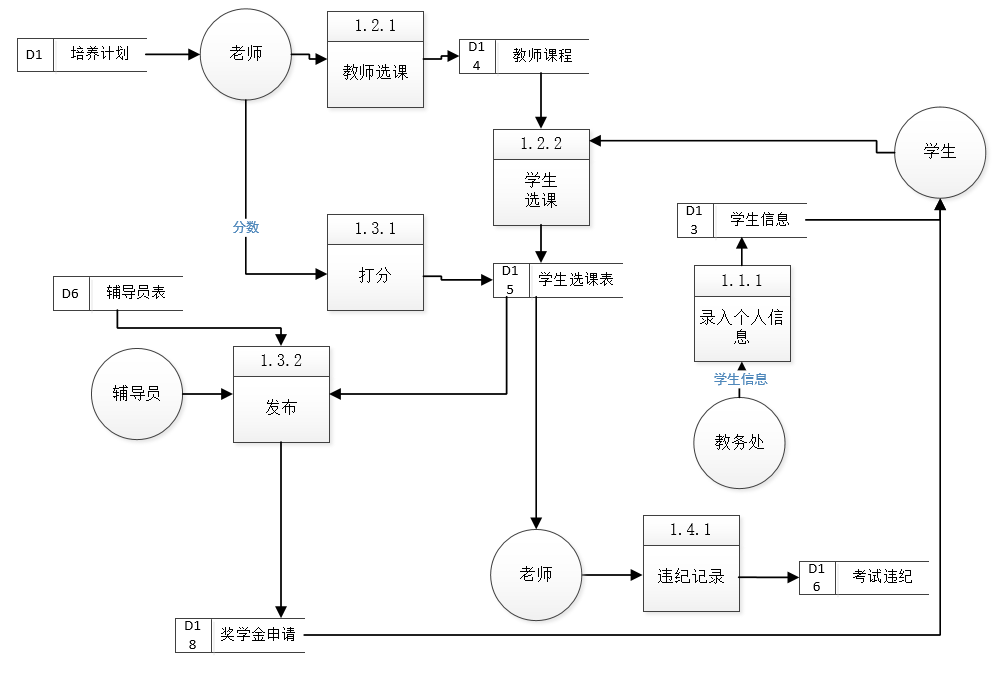


图5.5 教务子系统数据流程图

整套系统从教务处输入学生的信息开始。教务处方面，将学生信息输入数据库，包括学生姓名，密码，专业，辅导员等。教师在本系统中暂时不设计注册问题，因为这是一个信用系统，教师注册也应当属于教务处方面管辖，再次不多做重复。教师根据培养计划进行选课，教师选课完成后，轮到学生选课。学生通过账号密码登录后，根据教师选课表进行选课。选课之后生成学生选课表，当然学生选课表中，成绩一栏是空着的。学期结束后，教师对学生进行打分。如果考试期间有作弊行为，则学生成绩变为0分，并且将违纪记录计入考试违纪表。学期结束后，辅导员根据学生该学期的绩点的加权平均数，前10%的同学可以获得奖学金申请资格。学生可以点击申请，随后资金会注入学生的个人账户中。奖学金申请记录就会保留下来。

**5.2.2个人申请子系统数据流程图**

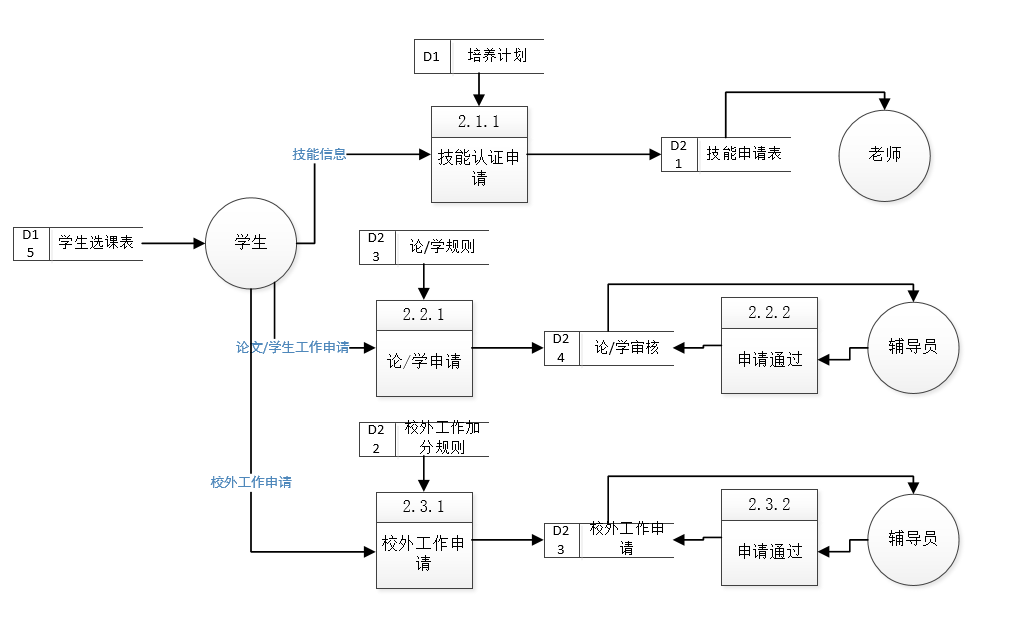


图5.6 个人申请子系统数据流程图

个人申请子系统就相对比较简单。技能申请子系统，根据学生的实际得分情况和课程技能申请情况，向老师发起个人技能申请。老师可以根据实际对于学生的了解，确认学生的申请。学生可以在个人界面看到自己技能的标签。

学生工作和校内工作，学生根据规则表选择自己的加分项目，生成论/学规则表。辅导员则要调用对应的表，根据实际情况，选择是否通过。

校外工作申请，则需要学生提交电子版的实习证明，填写对应的时间。辅导员判断两者是否相符，如果符合就点击确定。

这三大部分都会作为信用评分的重要组成部分，但是具体的得分情况还需要通过实际的数据再做打算。

**5.2.3二手商品子系统数据流程图**

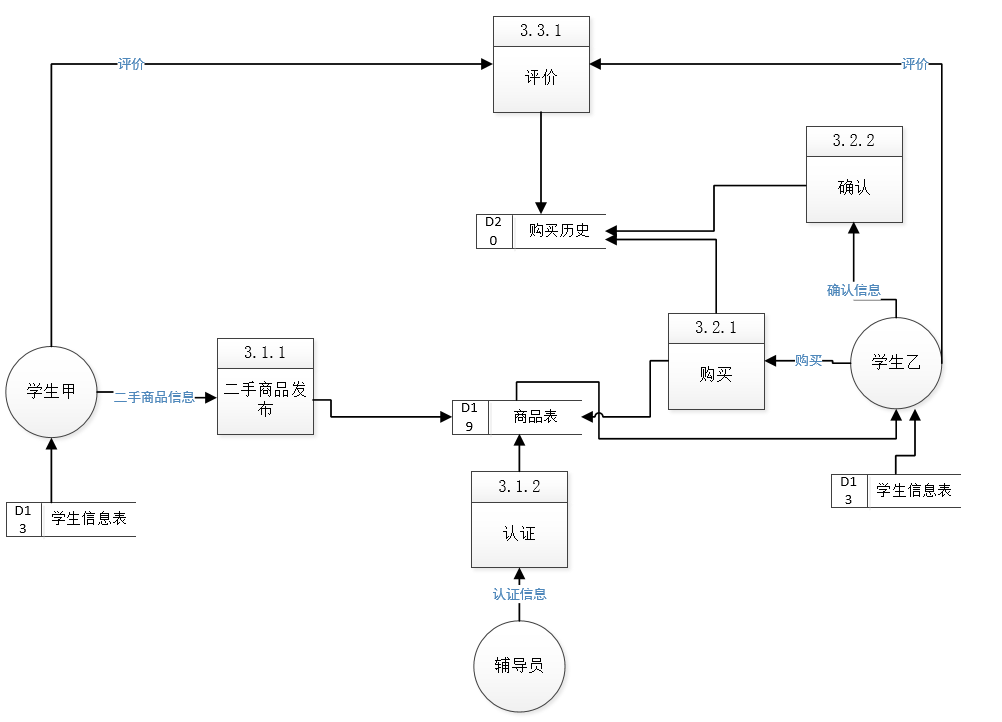


图5.7 二手商品子系统数据流程图

二手商品子系统参照了著名二手商品交易平台咸鱼的运作模式。卖家上传商品，上传的时候进行，对于商品进行限制，从而达到商品合法合理的目的。在此系统内中，辅导员代替了其中审核的机制。首先作为卖家的学生发布自己的商品，当然也需要填写自己的联系方式，交易地点，原价和现价等等。等到辅导员审核之后，作为买家的用户可以看到可以购买的商品。点击购买后，买房的金额会减少，但是卖方的金额不会改变。等买方确认收货后，卖方才会收到相应的款项。完成支付后，买卖双方可以互相评价，如果不评价，分值则默认为5分。如果评价，则相应改变对应的分值。在此系统中，暂时不考虑交易不成功的可能性。

**5.2.4图书馆子系统数据流程图**

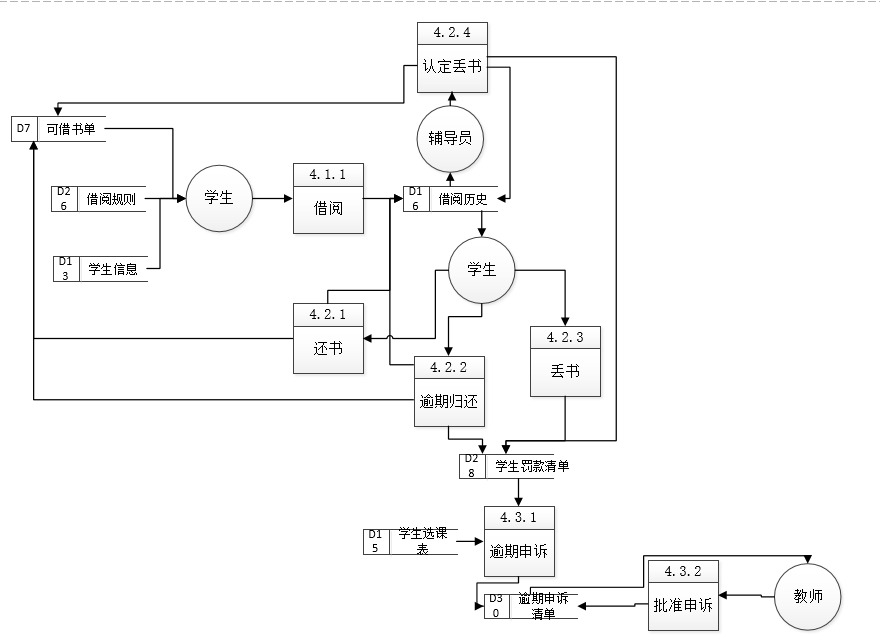


图5.8 图书馆子系统数据流程图

图书馆子系统的难点在于书籍归还的时候有不同的情况。

图书馆子系统从书籍的借阅开始。学生根据自己的学历调用借阅规则和可以借阅的书籍，点击借阅后，改变改书籍的状态同时生成借阅记录。

这其中有对于的开始借阅时间和最后归还时间。如果在最后归还日期之前前往图书馆归还，则改变对于借阅记录的归还情况，然后将可借阅数据的属性相应变为可借。

但是如果学生没有在规定的时间内还书，则使用逾期归还功能，生成学生罚款清单。同时，既然书籍已经归还，则改变书籍借阅历史中的相关属性，将书籍变为可借。当然学生可以对于逾期有异议，比如由于使用时间过长，只要向自己的认可老师，例如这里调用学生选课表中的老师，向他进行申诉。如果申诉通过，则不需要缴纳罚款，如果申诉不通过，依然不得不缴纳罚款。

如果逾期时间过长，或者学生主动承认，可以将书籍变为丢书。丢书之后，书籍属性不改变，借阅历史更变为已还。

如果书籍判定为丢书，则会直接扣除学生的金额。如果判定为逾期，由于还需要进行申诉，所以不会立即扣除。

**5.2.5信用评分子系统数据流程图**

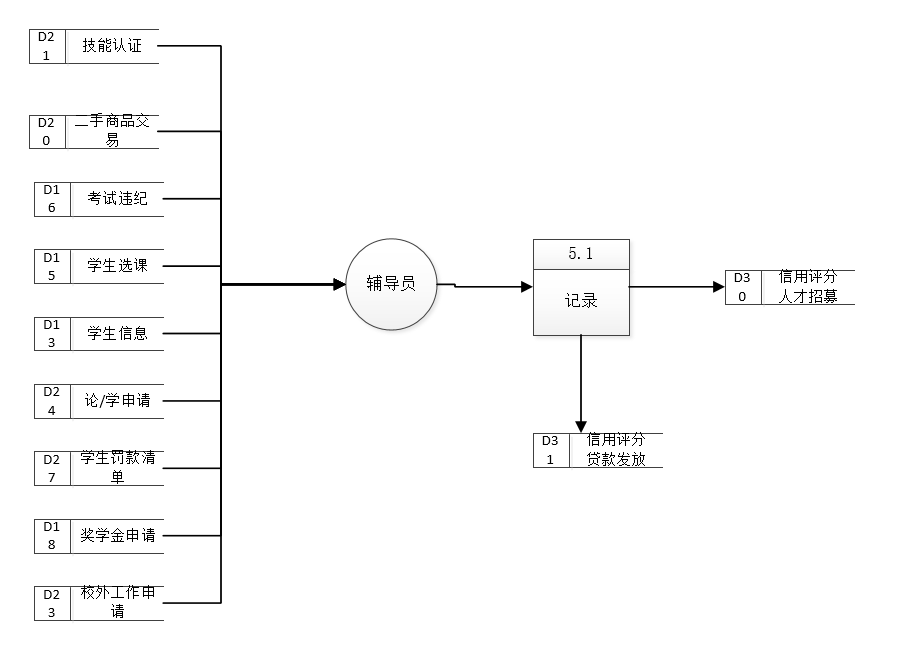


图5.9 信用评分子系统数据流程图

最后，是信用评分子系统数据流程图，不过这部分的图是最为简单的。九张表，对应不同的维度，通过辅导员，生成贷款发放信用评分和人才招募信用评分。

**5.3第二层数据流程图**

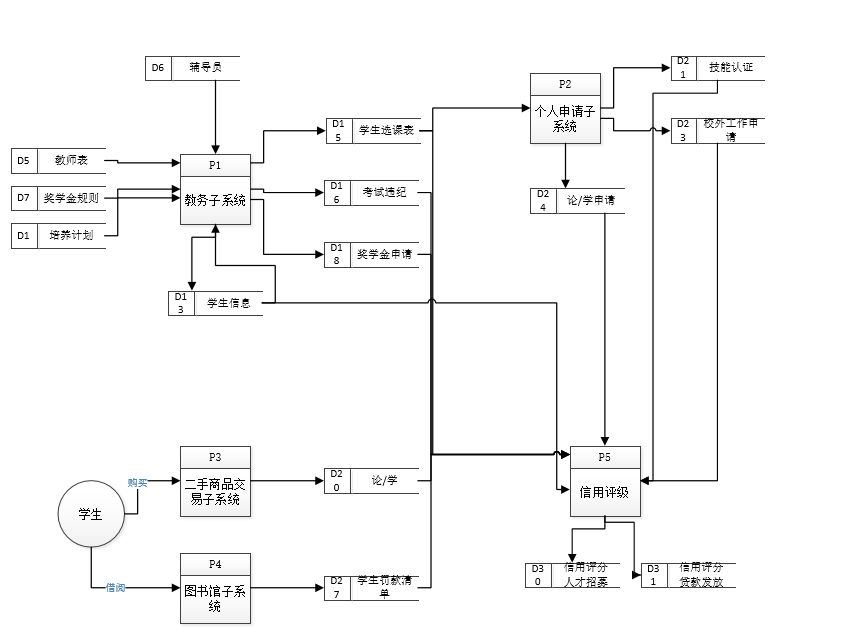


图5.10 大学生信用评估系统第二层数据流程图

第二层数据流程图和可以体现子系统之间互相的关系，清楚这几者之间的关系有助于我们理清整套系统的合理性。

**第六章 大学生信用评估系统ER图**

**6.1 ER图**

**6.1.1教务系统**

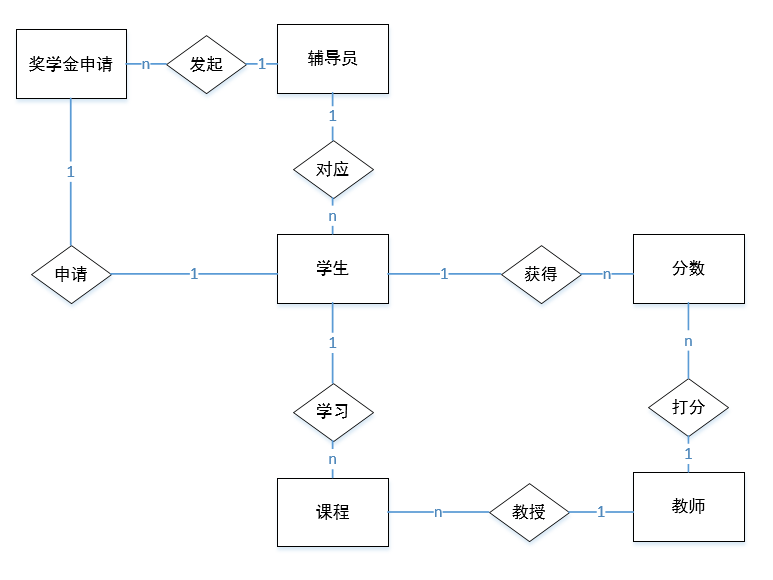


图6.1 教务系统ER图

一个辅导员对应多个学生，一个学生可以学习多门课程，一个老师教授多门课程，一个老师给予多个分数，一个学生获得多个分数。一个辅导员可以发起多个奖学金申请，一个学生只能通过一个奖学金申请。

**6.1.2二手商品交易**

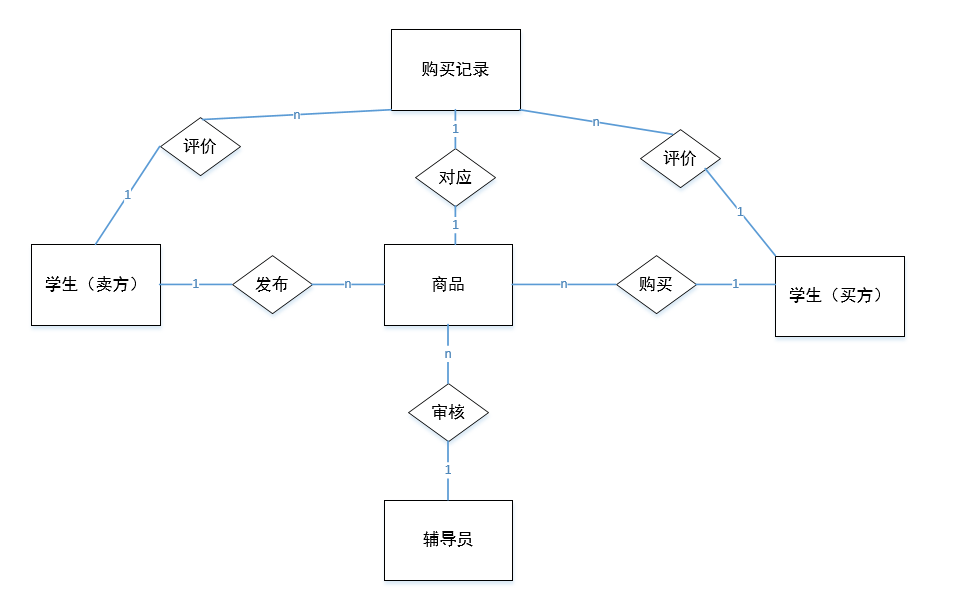


图6.2 二手商品交易ER图

作为卖家的学生可以发布多个商品，一个辅导员可以审核多个商品，一个商品可以产生一条记录，作为买方的学生可以购买多个商品，购买之后，学生可以评价多条购买记录。

**6.1.3个人申请子系统**

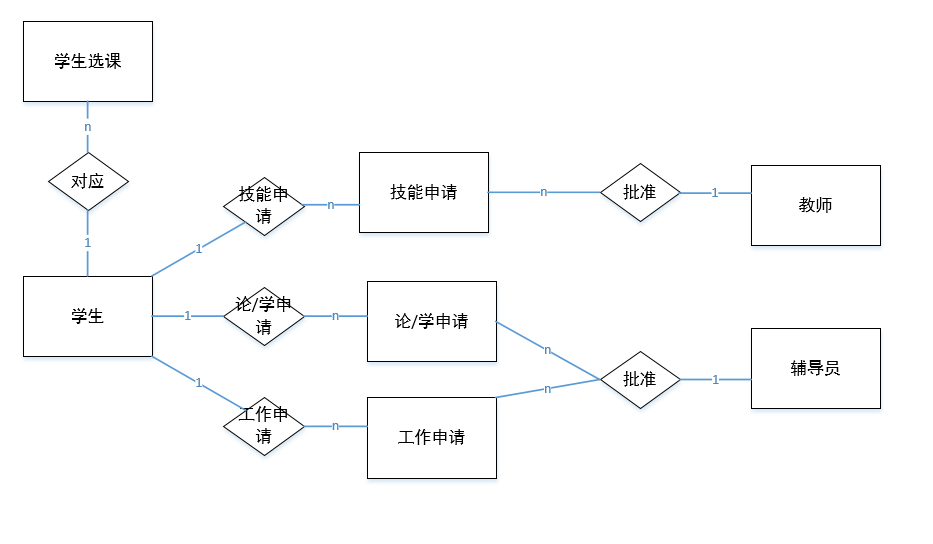


图6.3 个人申请子系统ER图

一个学生对应多条学生选课记录，一个学生可以申请多个技能申请，一个老师可以批准多条技能申请。同样，一个学生可以进行多条论文学生工作申请和校外工作申请，一个辅导员可以进行多个申请的批准。

**6.1.4 图书馆子系统**

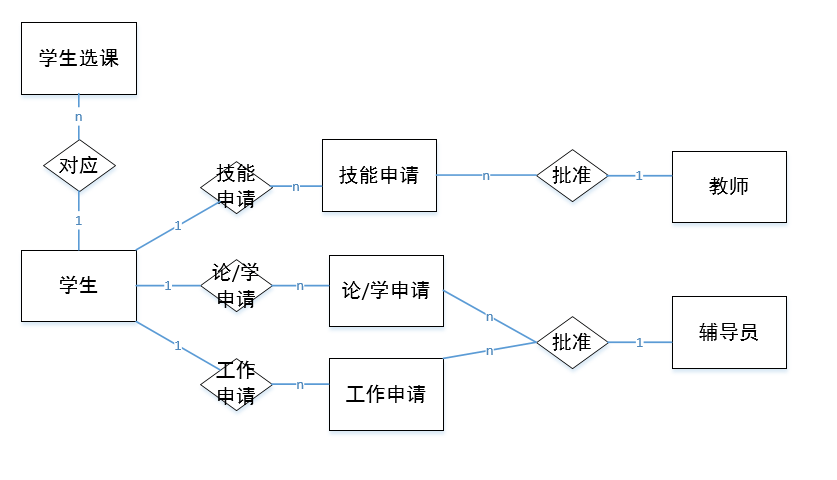


图6.4 图书馆子系统ER图

这里都是一对多的关系。学生可以选择多本图书，可以归还多本图书，可以逾期归还多本图书，也可以承认丢失多本图书。一个学生可以根据借阅情况，生成多个借书记录。学生会因为逾期而提出多个申诉。一个教师也可以通过多个申诉。

**6.1.5 信用评分子系统**

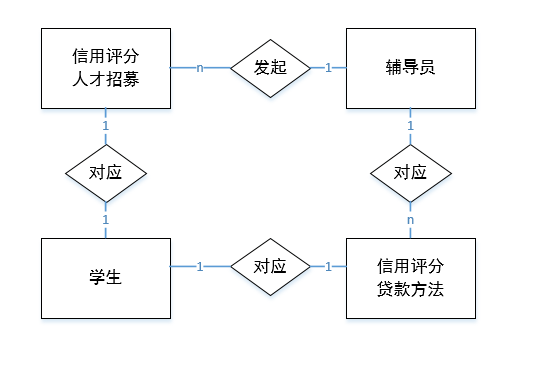


图6.6 信用评分子系统ER图

信用评分子系统的ER图相对比较简单，一个学生对应一个人才招募的信用评分，一个学生也对应这一个贷款方法的信用评分。一个辅导员可以发起多个信用评分。

**第七章 大学生信用评估系统实践**

**7.1基础界面**

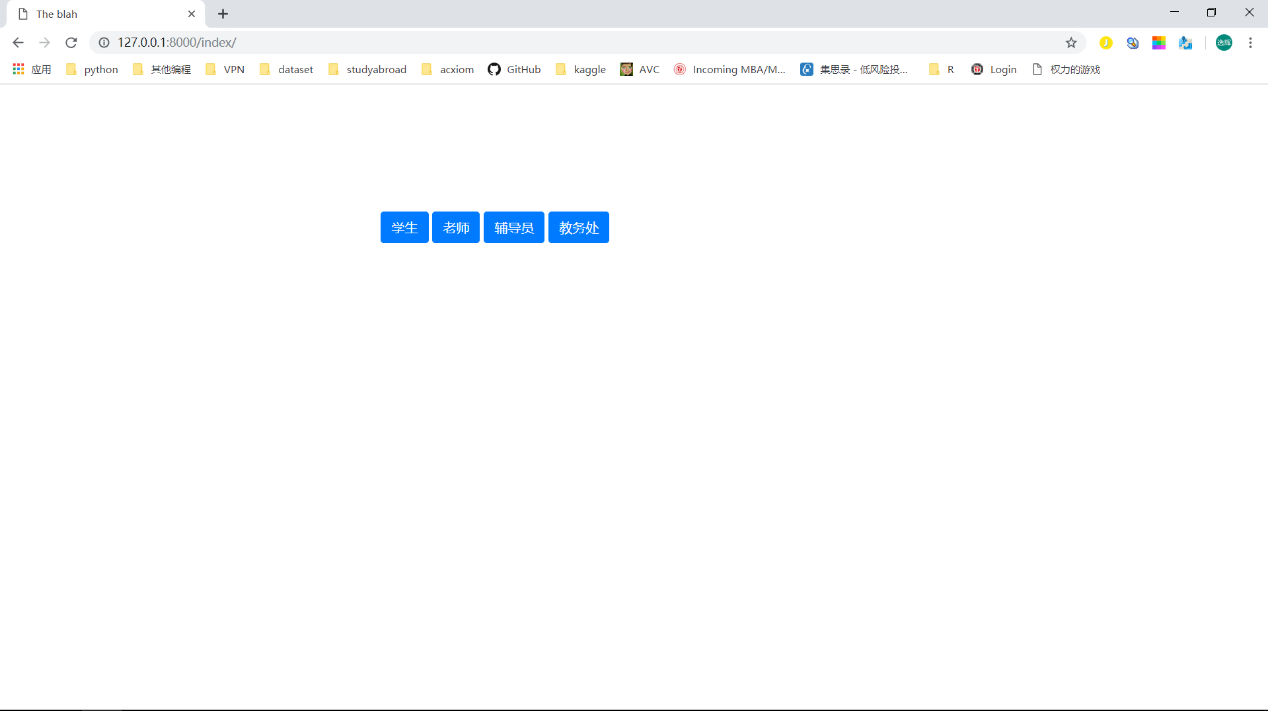


图7.1 基础界面

这是系统的index界面，在这里可以选择登录的身份。

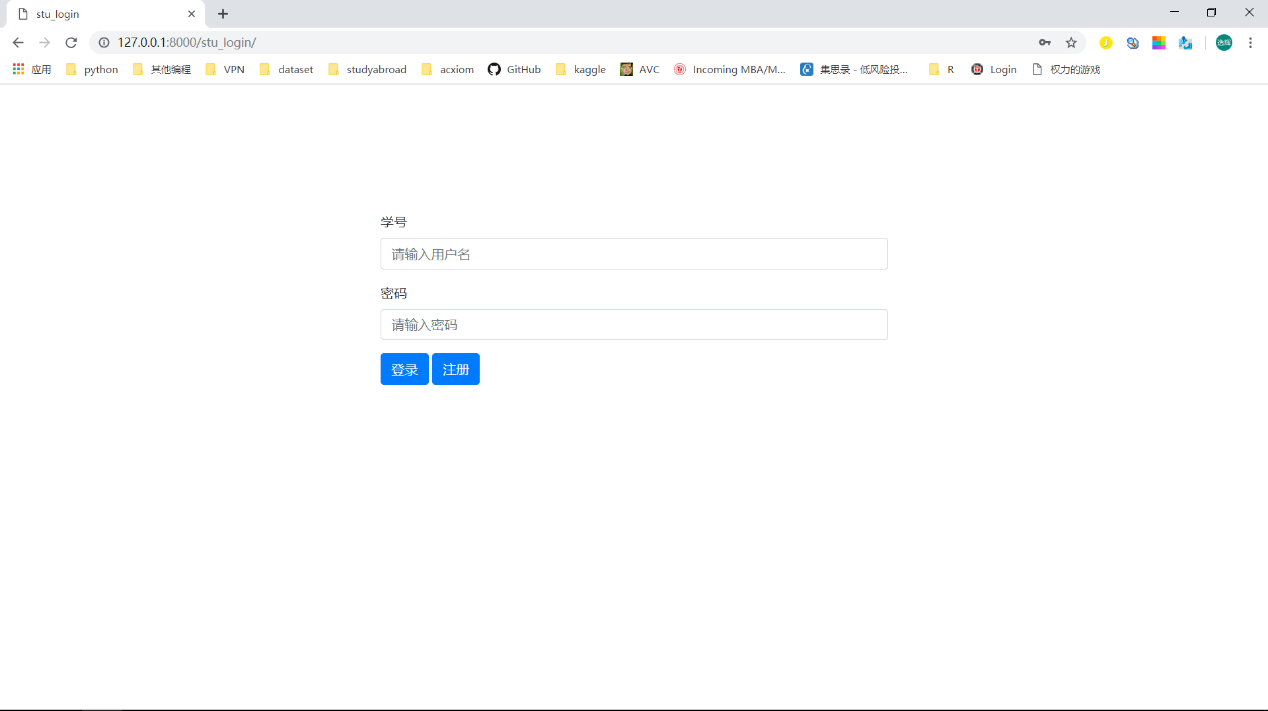


图7.2学生登录界面

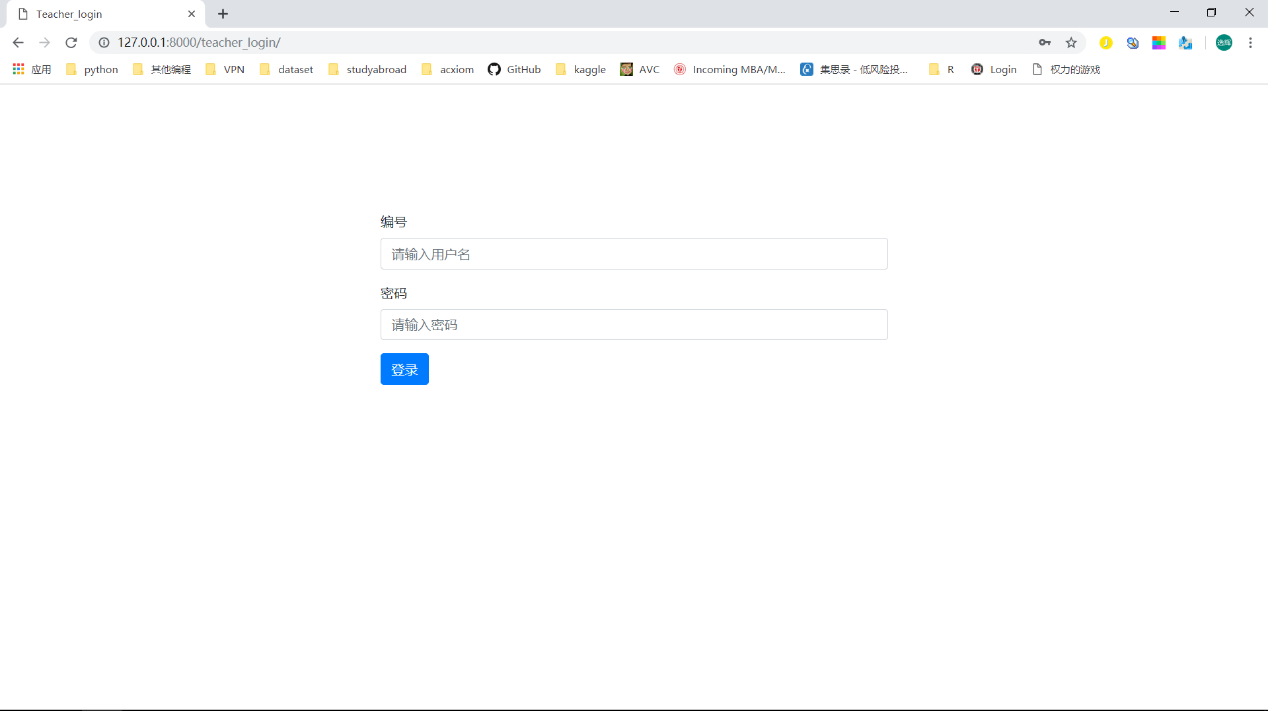


图7.3教师登录界面

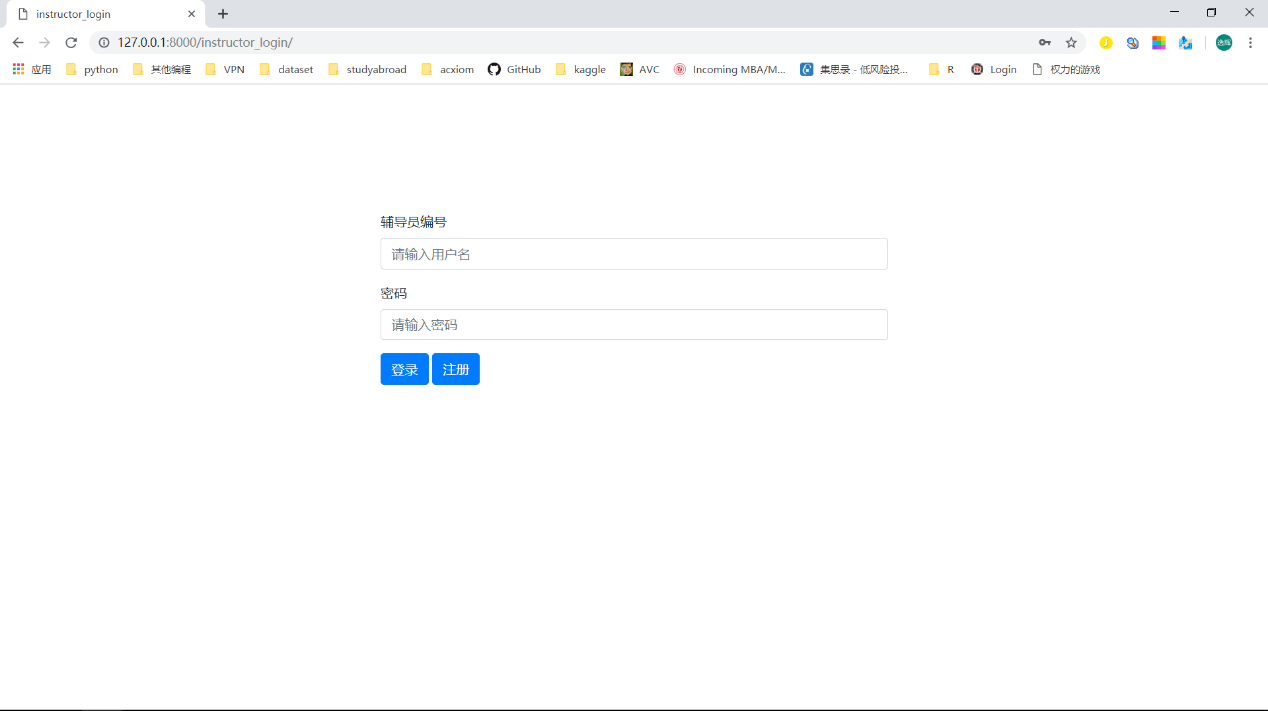


图7.3辅导员登录界面

学生、老师、辅导员可以输入编号和密码进行登录。

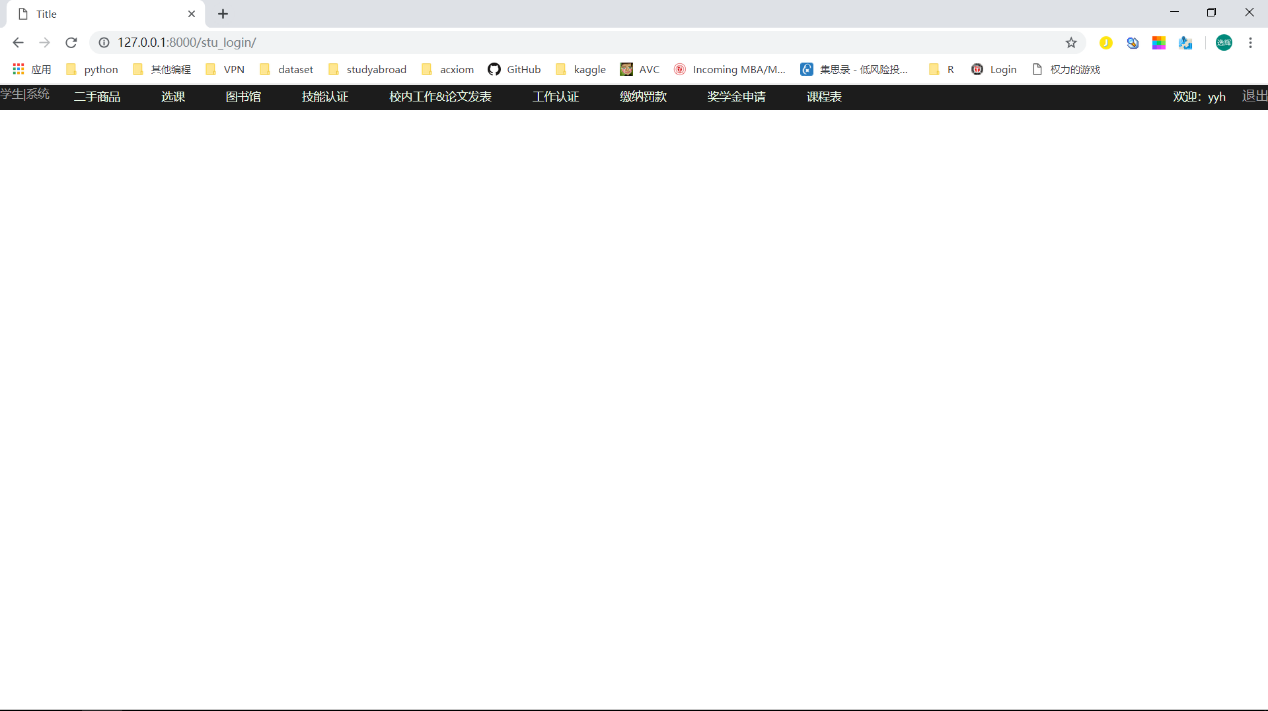


图7.4 学习登录后界面

学生登录后的页面

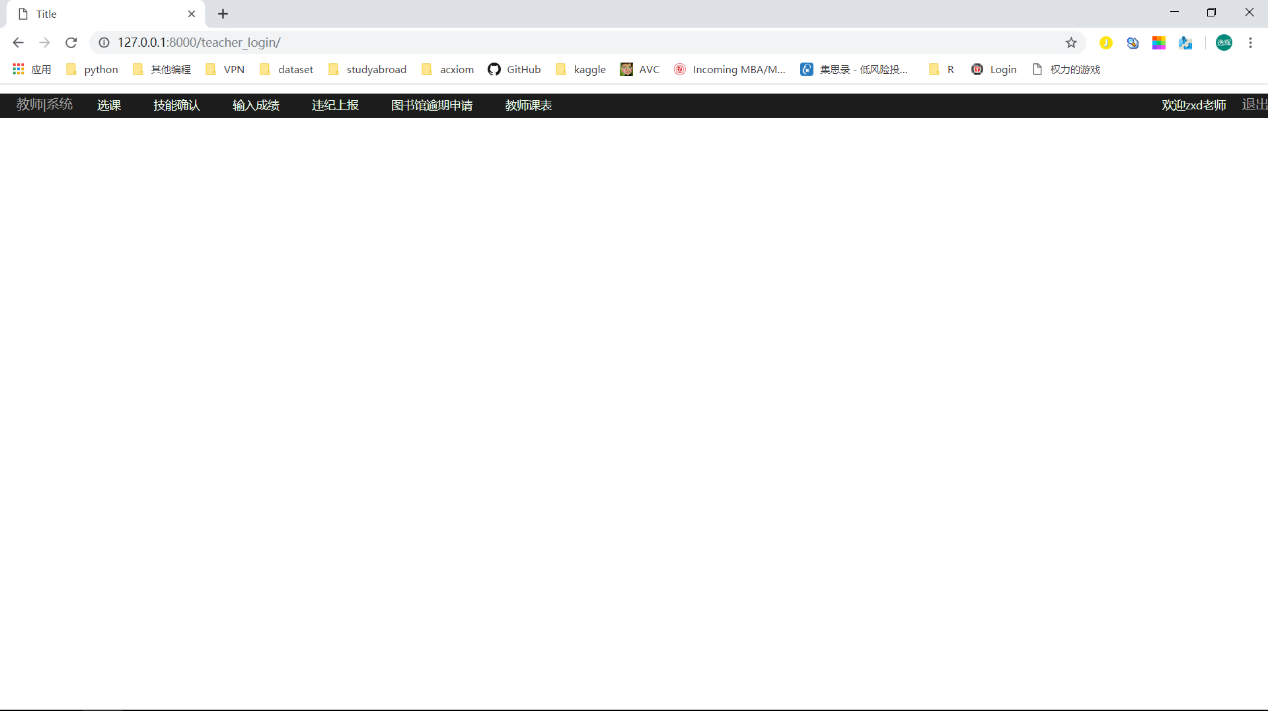


图7.5教师登录后界面



图7.6 辅导员登录后的界面

辅导员登录后的界面。

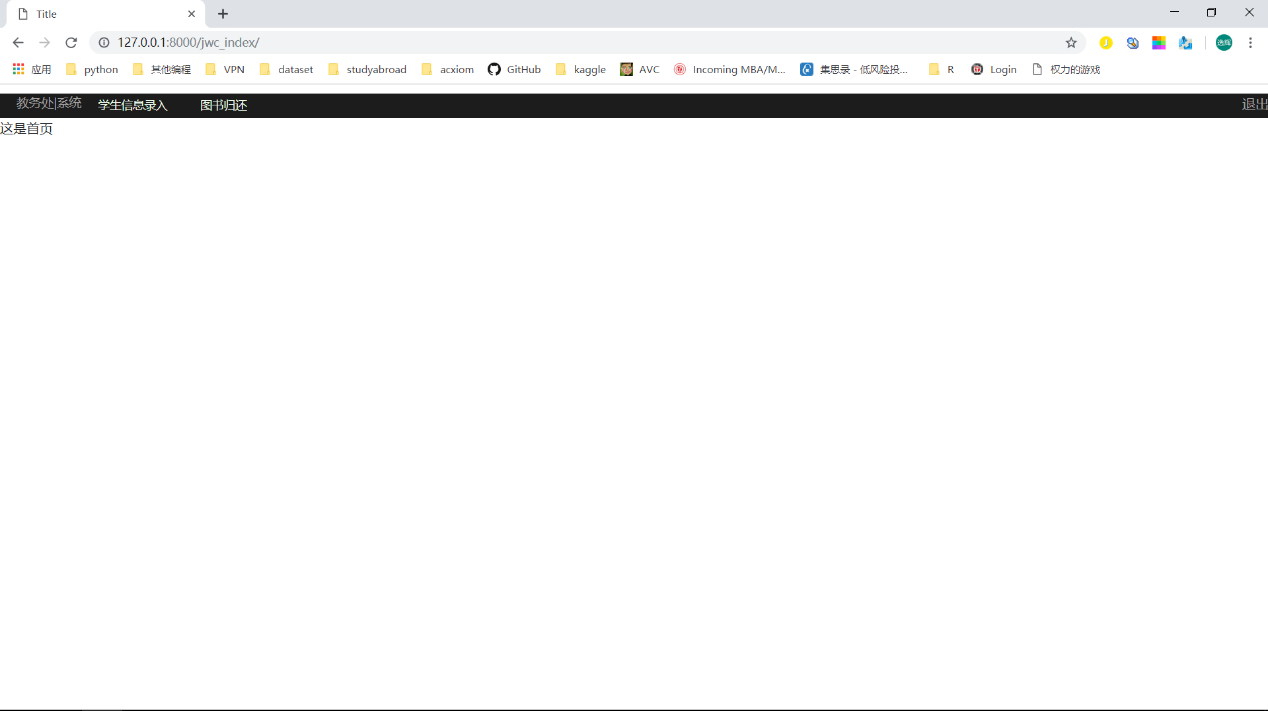


图7.7 教务处首页

教务处首页。

**7.2教务子系统实操界面**

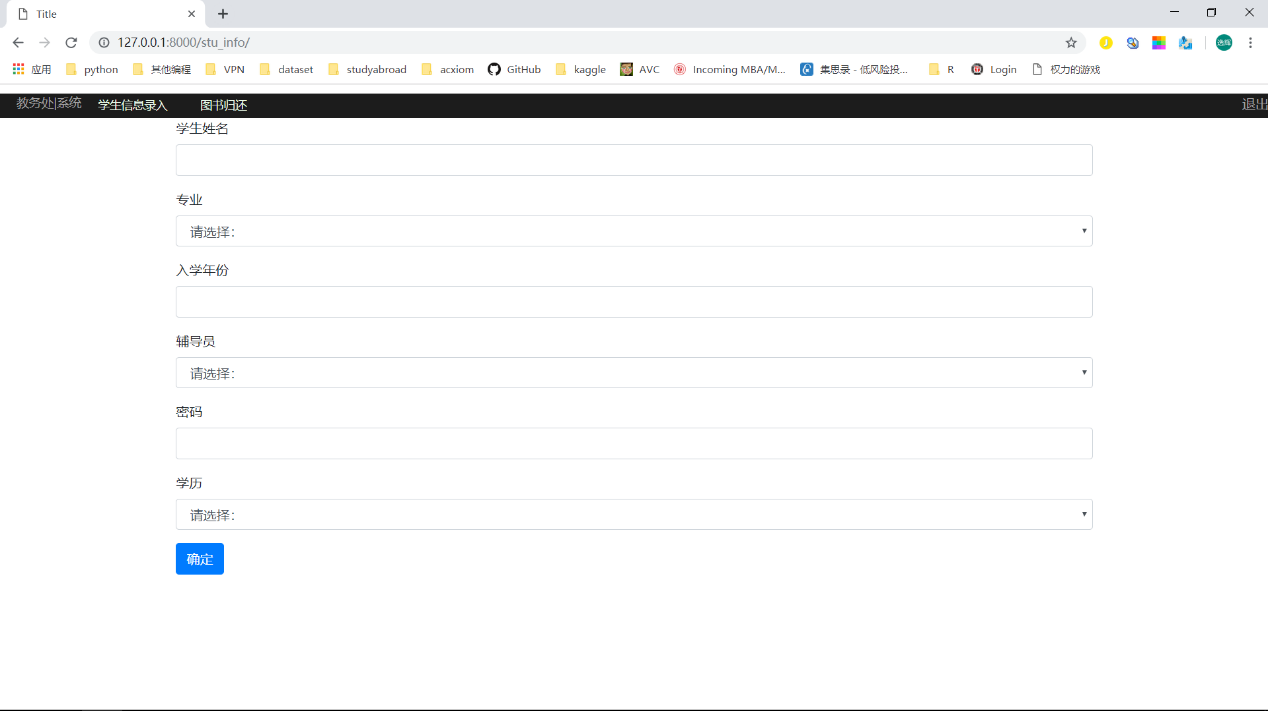


图7.8 学生信息录入界面

教务处进行学生信息录入的工作，输入学生的姓名，专业，入学年份，密码，学历等等。学生金额默认为100元。

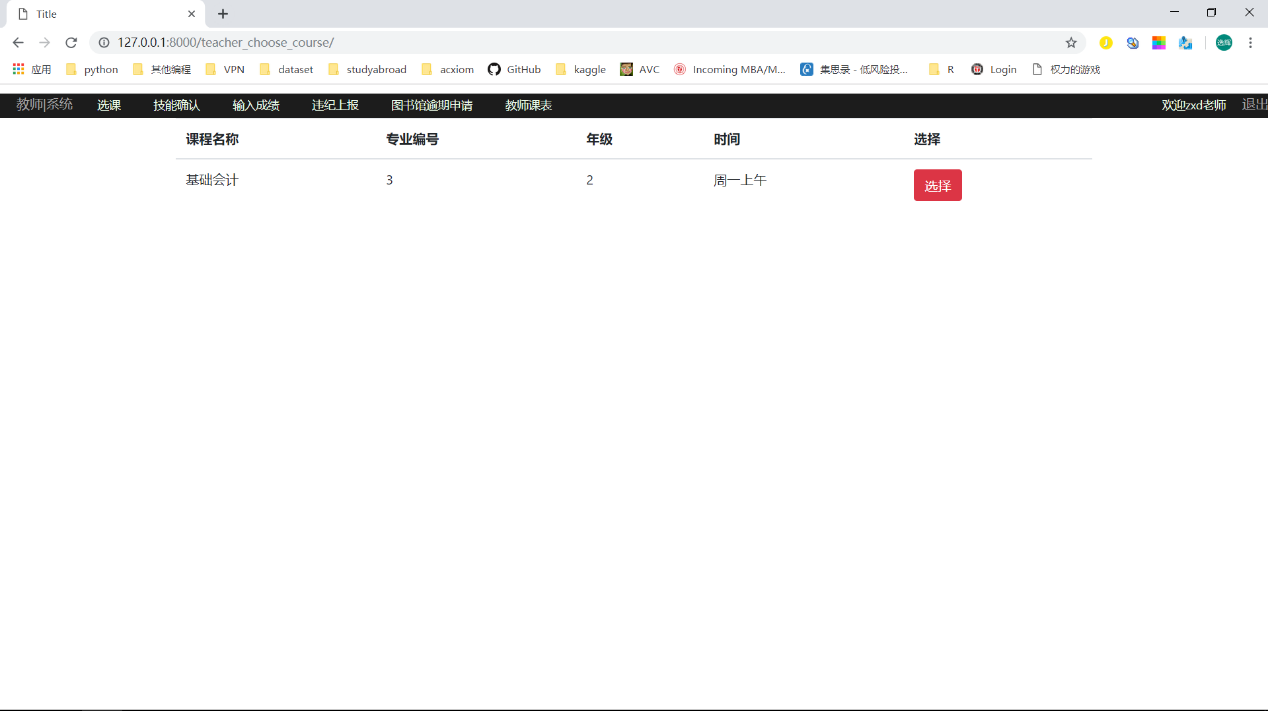


图7.9 教师选课界面

在教师选课界面，教师可以点击选择，进行选课，如果超过上限，就不会出现在选课界面。可以看到这位老师只剩下一门课可以选择。

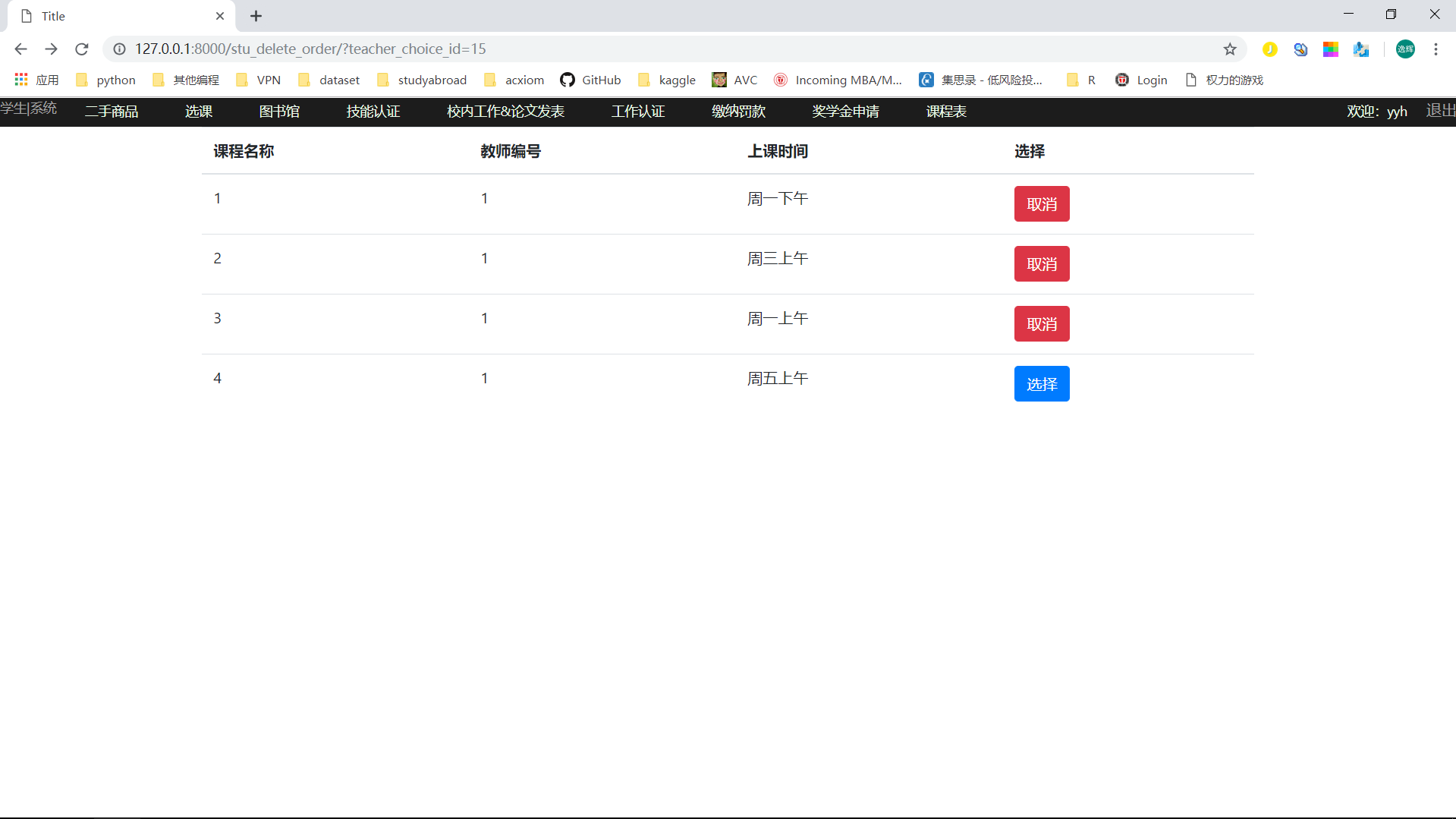


图7.10 学生选课界面

学生选课界面，学生可以通过选择和取消进行选课。

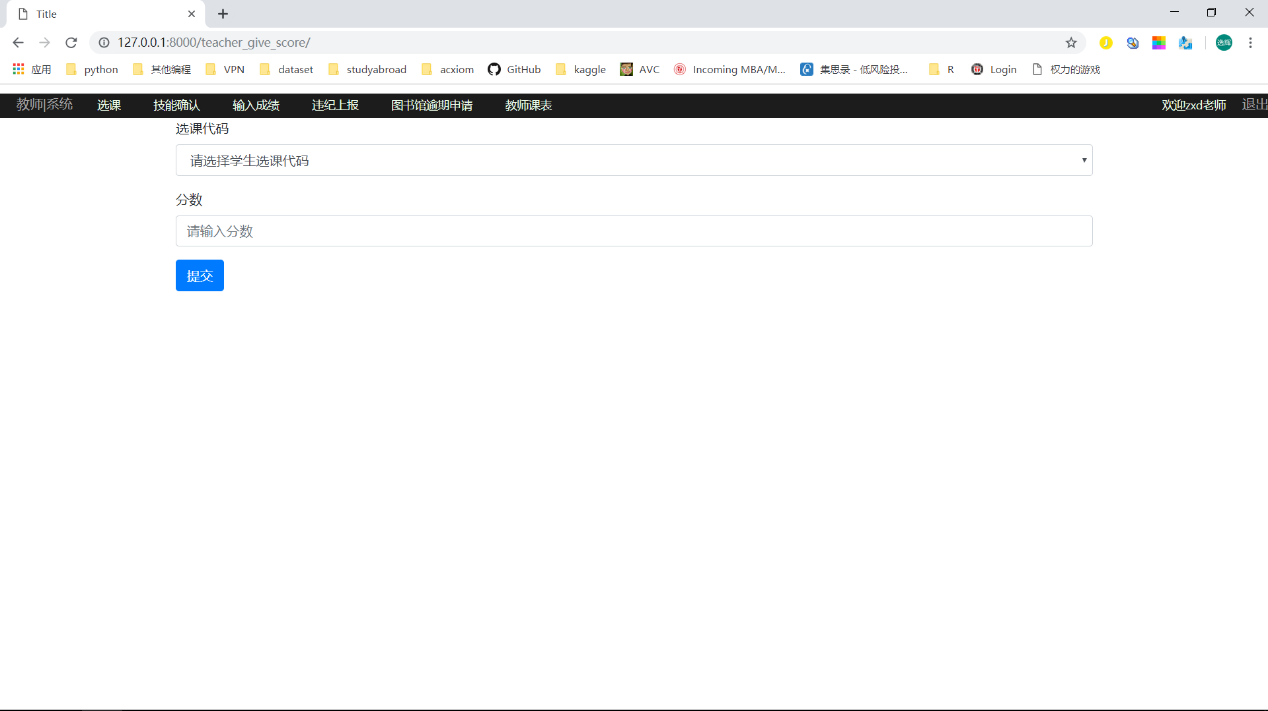


图7.11录入学生成绩界面

教师录入学生成绩界面。教师可以根据学生的选课编号，对学生进行打分。

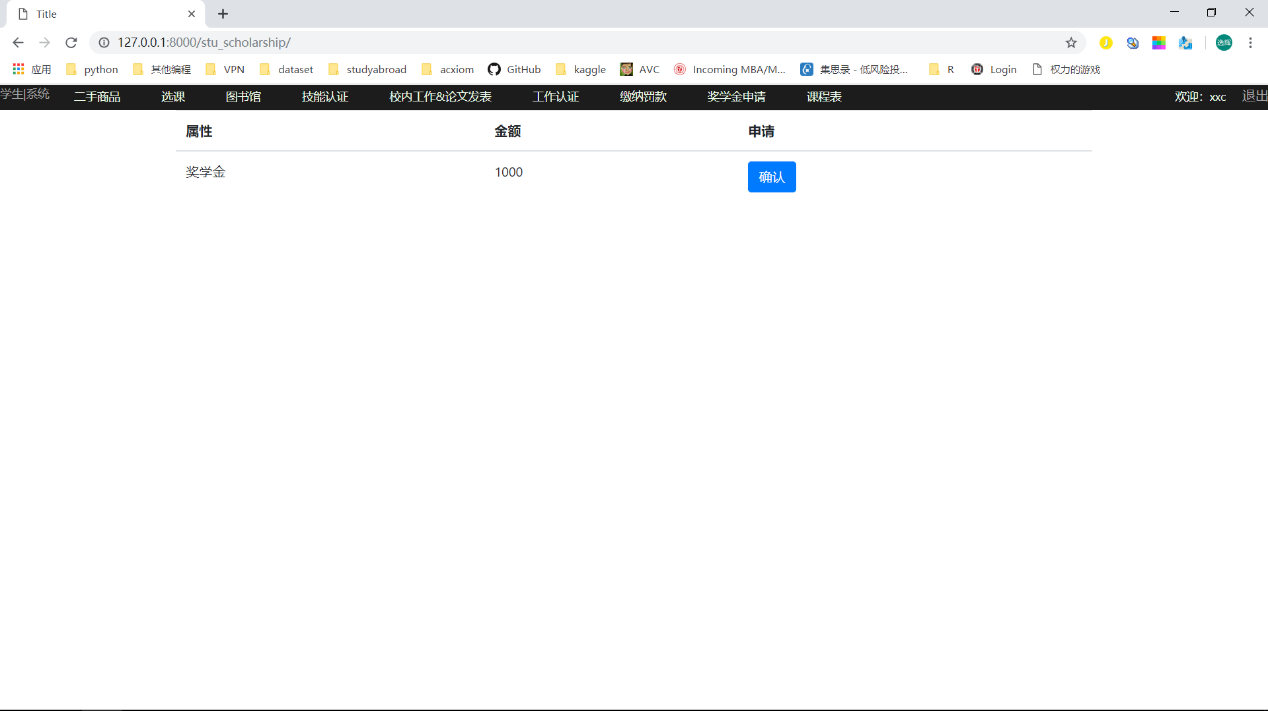


图7.12奖学金申请界面

学期结束后，辅导员会触发奖学金更新表的程序，随后学生会在自己的界面看到奖学金申领提示。

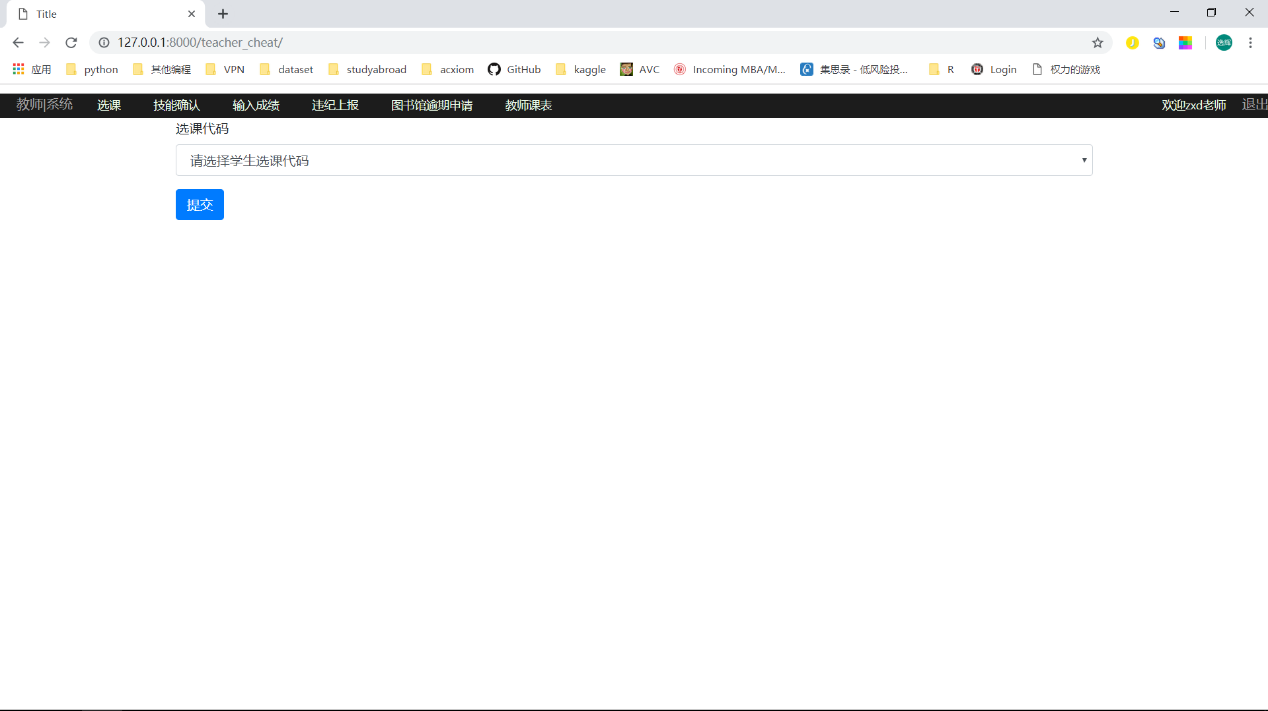


图7.13违纪上报界面

教师可以在这个界面中上报学生考试中的违纪情况

**7.3图书馆子系统界面实操**

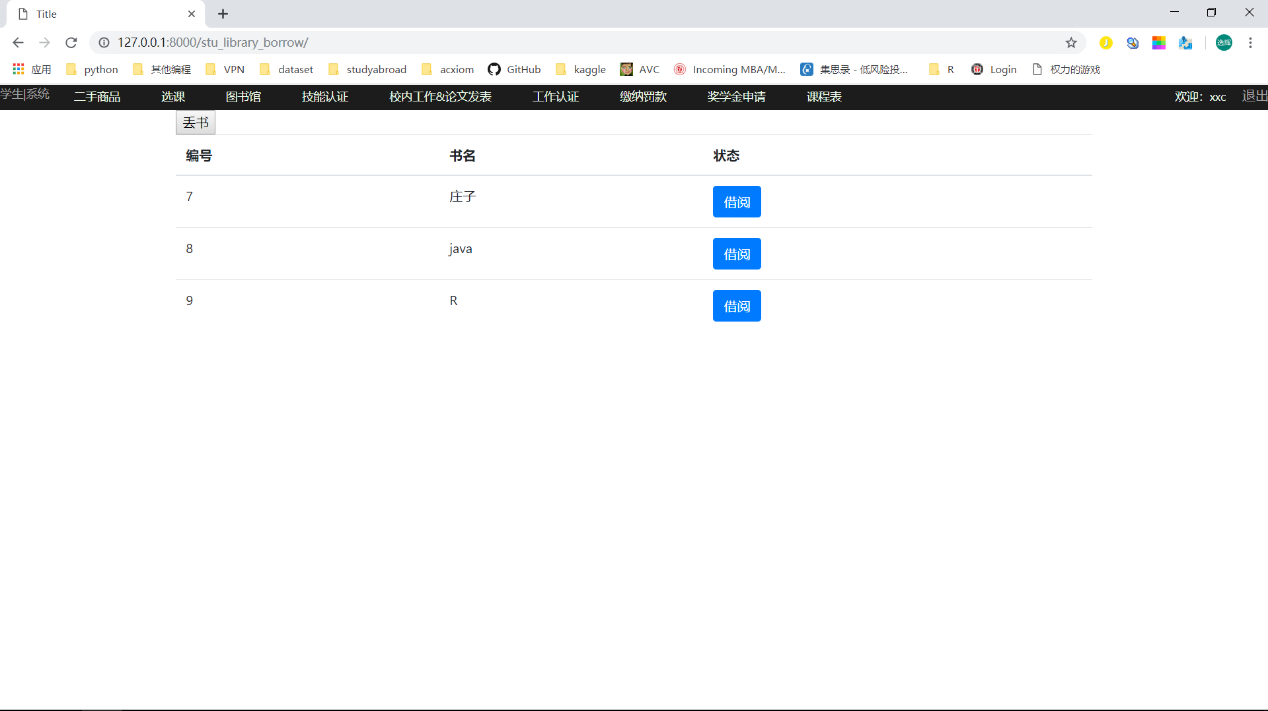


图7.14学生借阅界面

学生可以点击借阅，对书籍进行借阅。

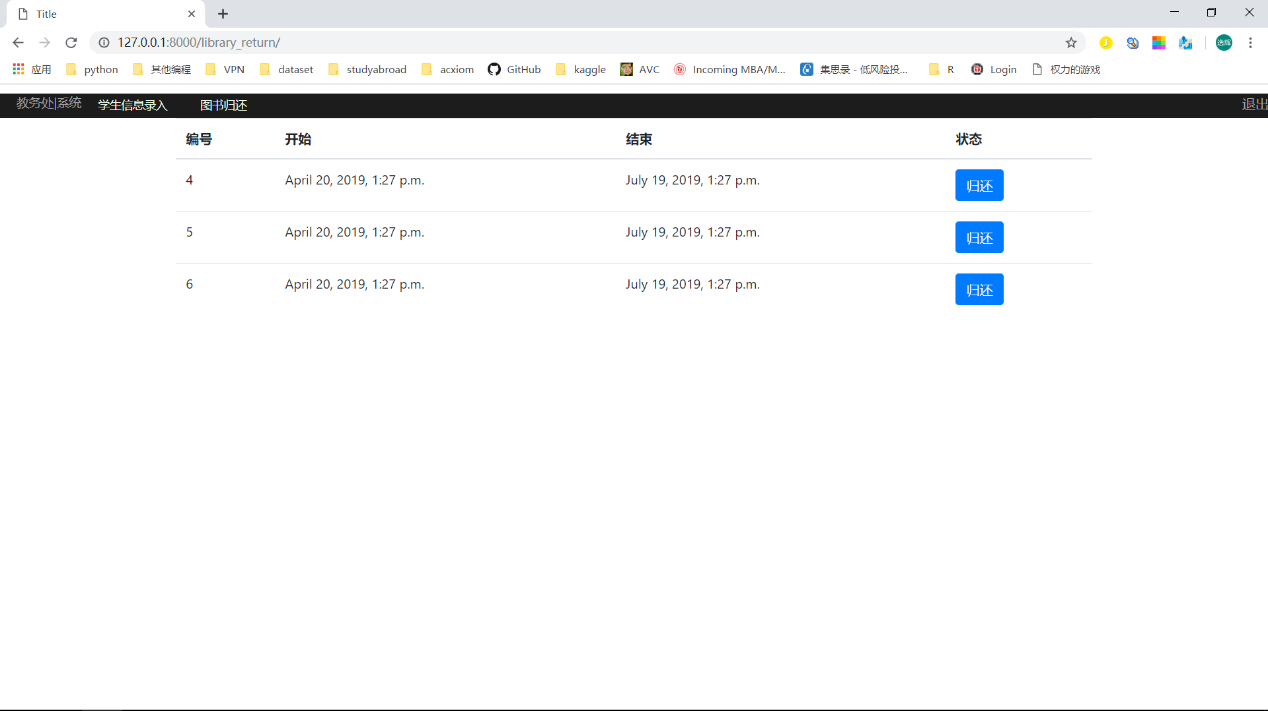


图7.15图书归还界面

为了保证客观性，所以归还不能通过学生界面进行。因为图书归还在图书馆界面，这里为了便于设置权限，放在教务处类目下。

如果图书逾期，归还后，则会相应记录则会出现在对应学生的逾期罚款清单界面中。

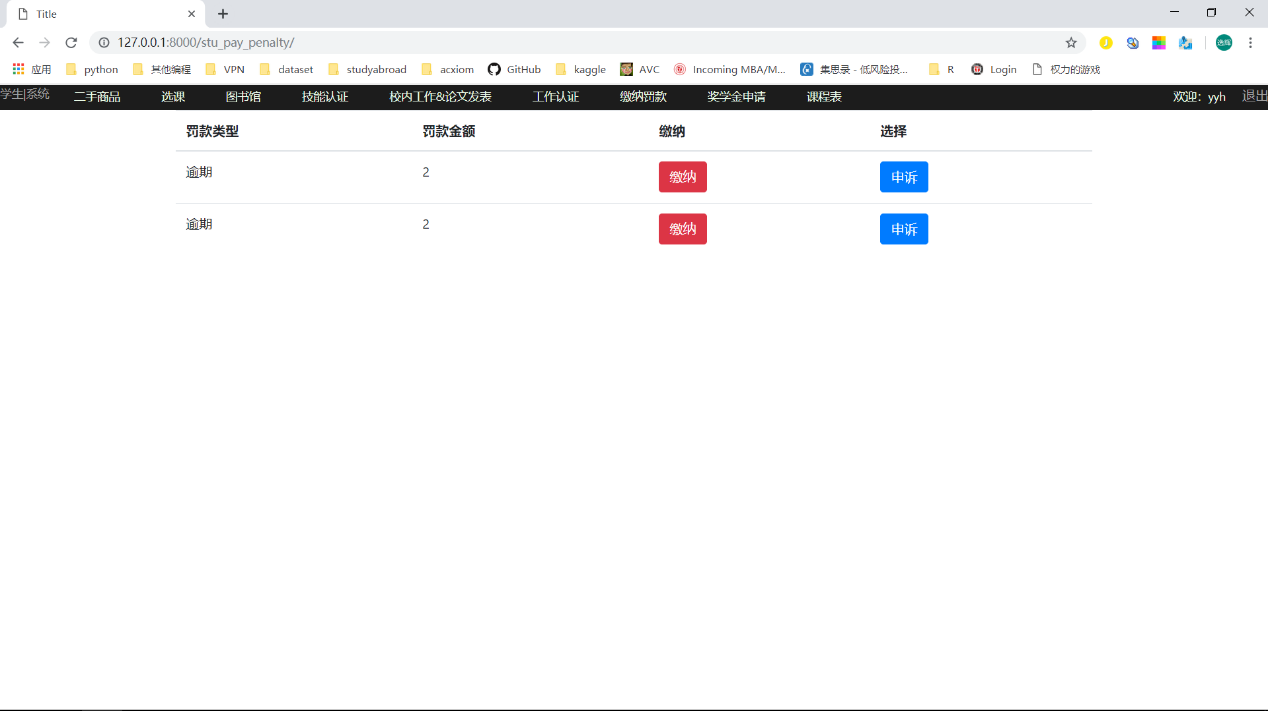


图7.16逾期罚款界面

罚款金额和逾期天数有关。

学生可以点击罚款，随后金额会相应从自己的账户中扣除。当然也可以选择向老师进行申诉。

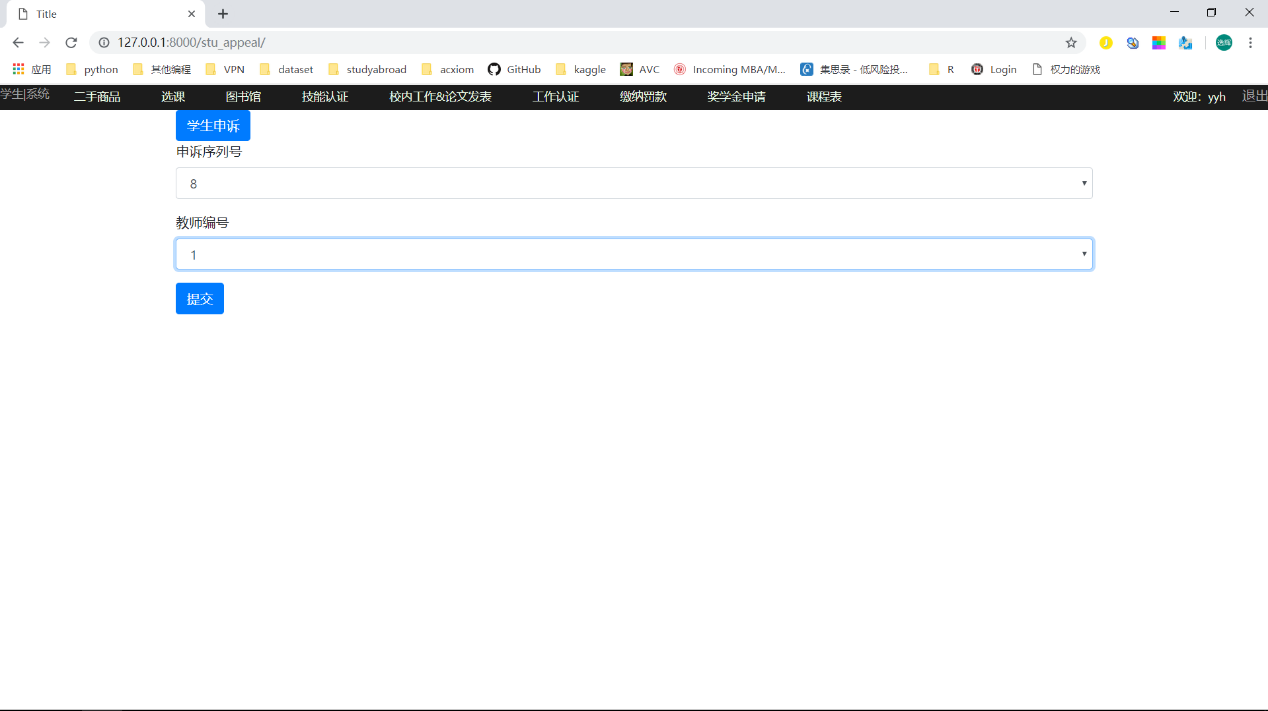


图7.17 学生申诉界面

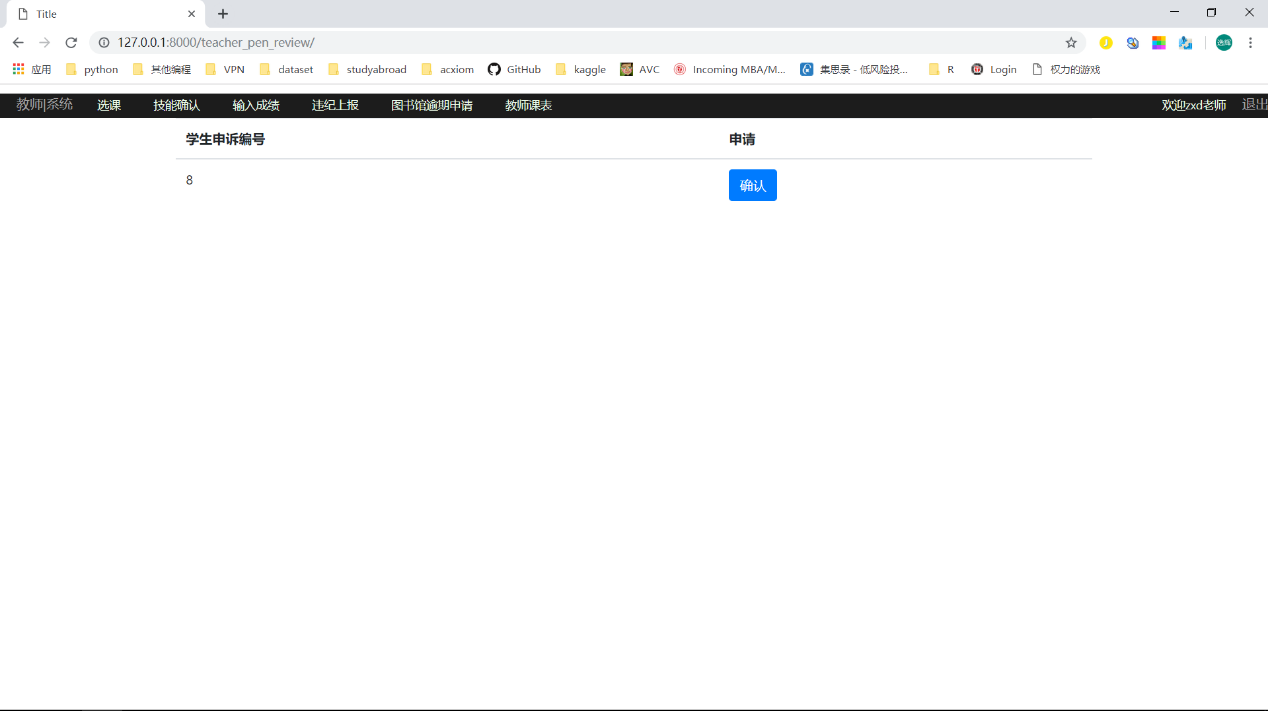


图7.18 教师申诉界面

教师也可以看到相应的申诉。

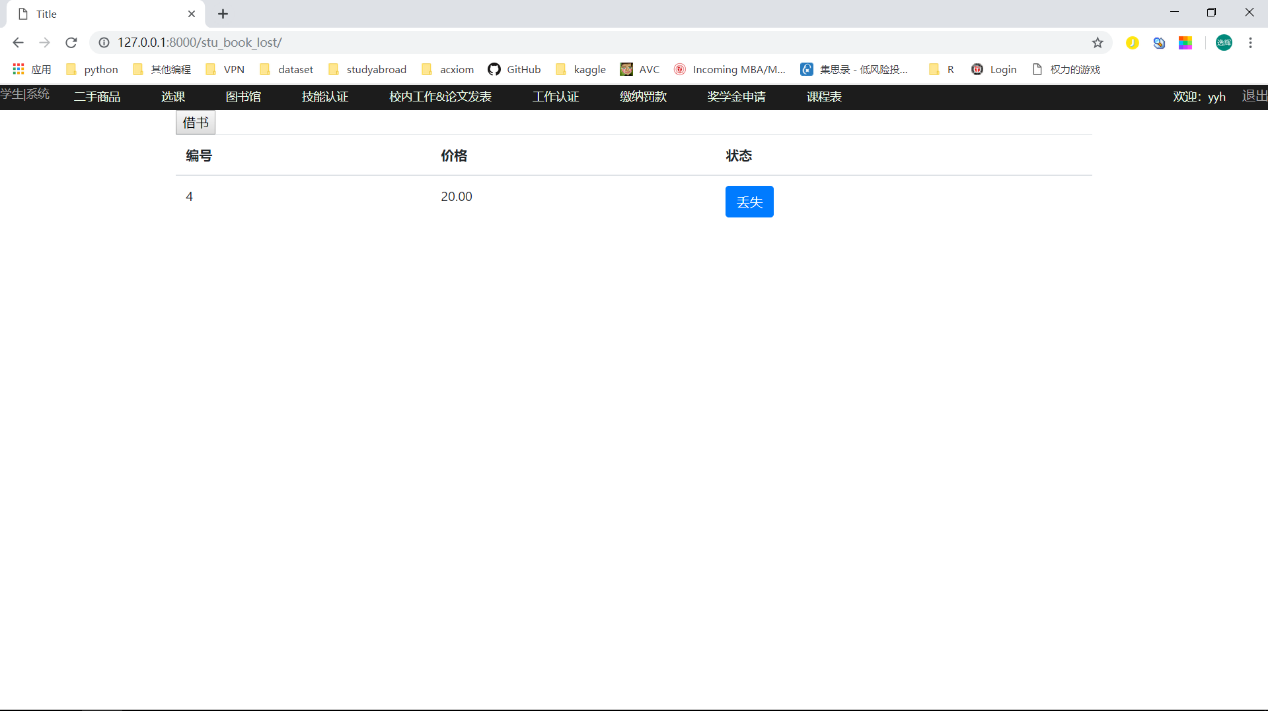


图7.19 学生丢书界面

学生可以承认自己丢书，点击丢失，自动扣除相应的价格，书籍不再出现在借阅界面。

**7.4 二手商品交易子系统**

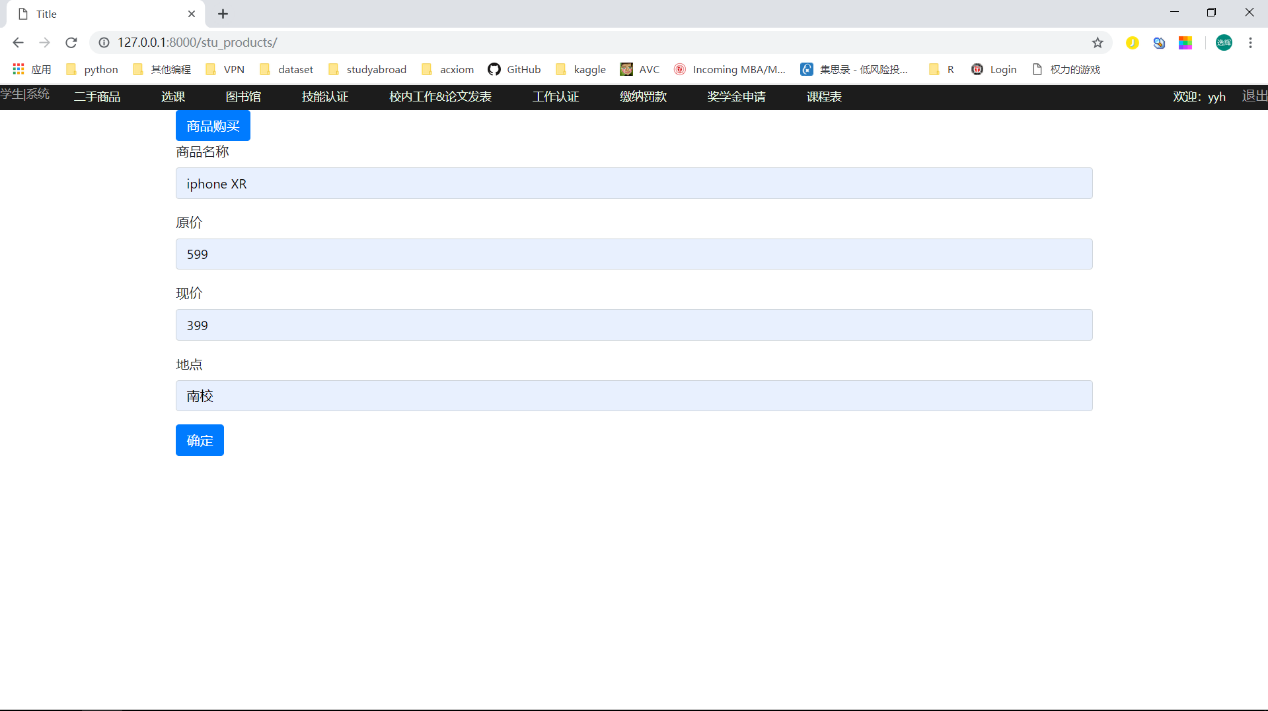


图7.20 学生商品发布界面

发布商品界面，学生可以发布商品名称，价格，原价，交易地点。



图7.21 辅导员商品审核界面

辅导员商品审核界面中，辅导员点击确认，则商品可以被购买。

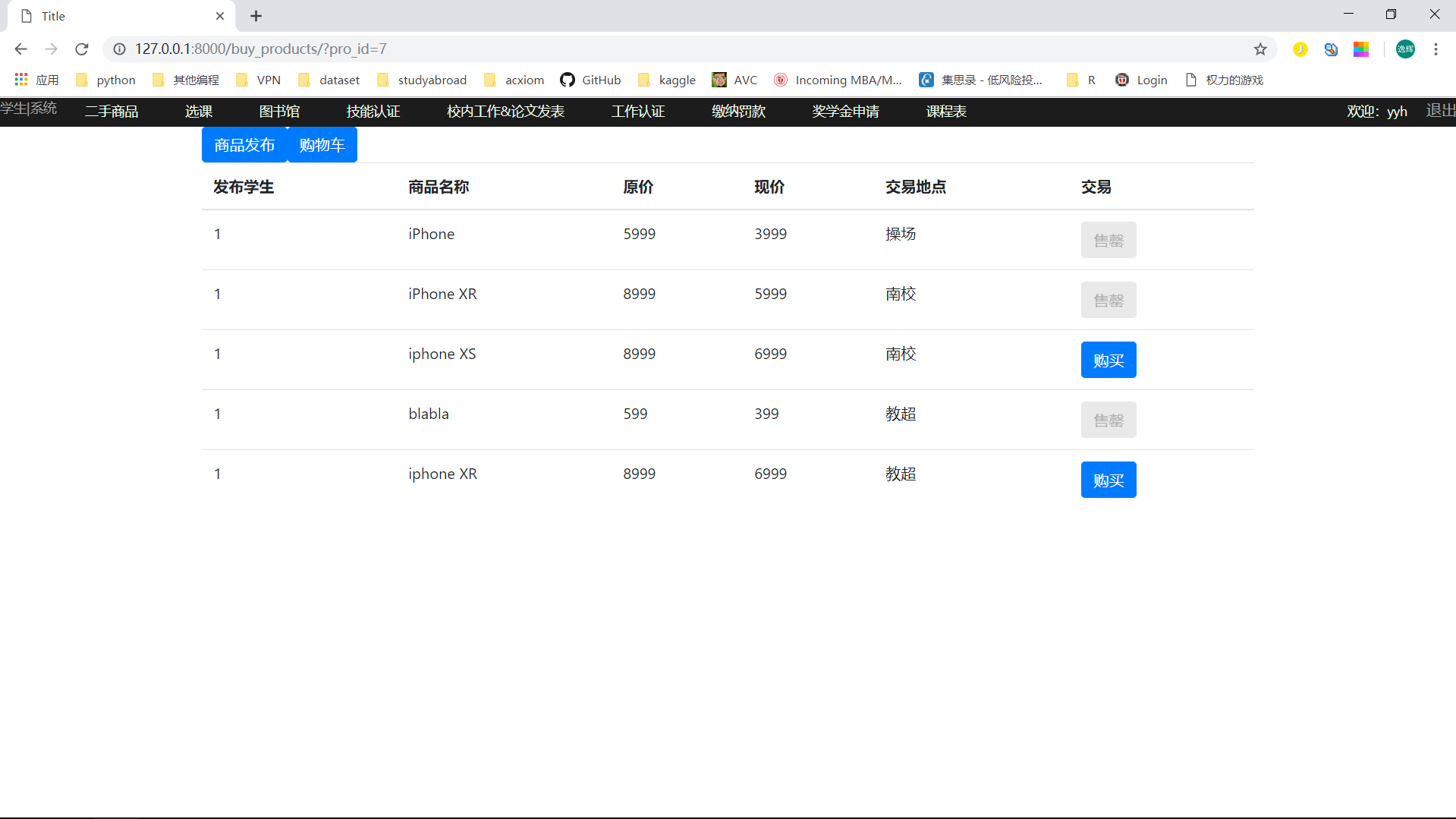


图7.22商品交易界面

在交易界面，作为买方的学生可以查看到辅导员审核之后的商品，点击购买，则商品变为售罄。

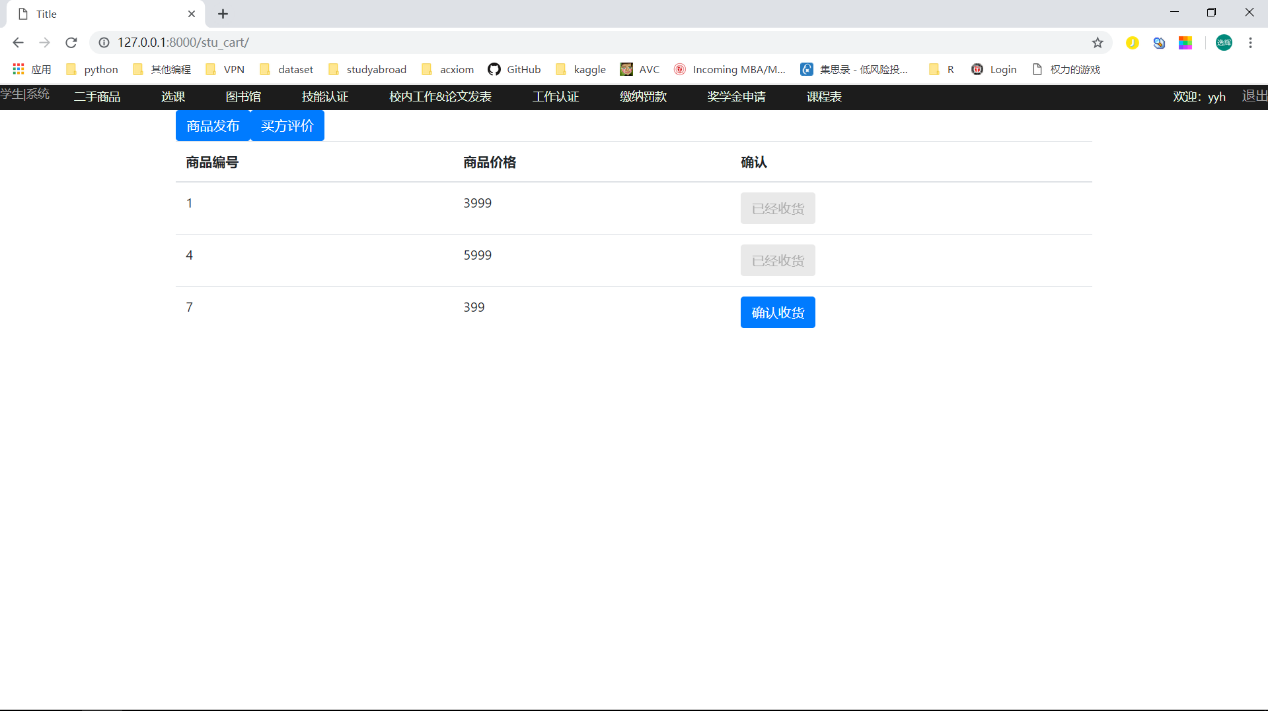


图7.23确认收货界面

在线下交易后，买方点击确认收货，金额才可以打入卖方账户。

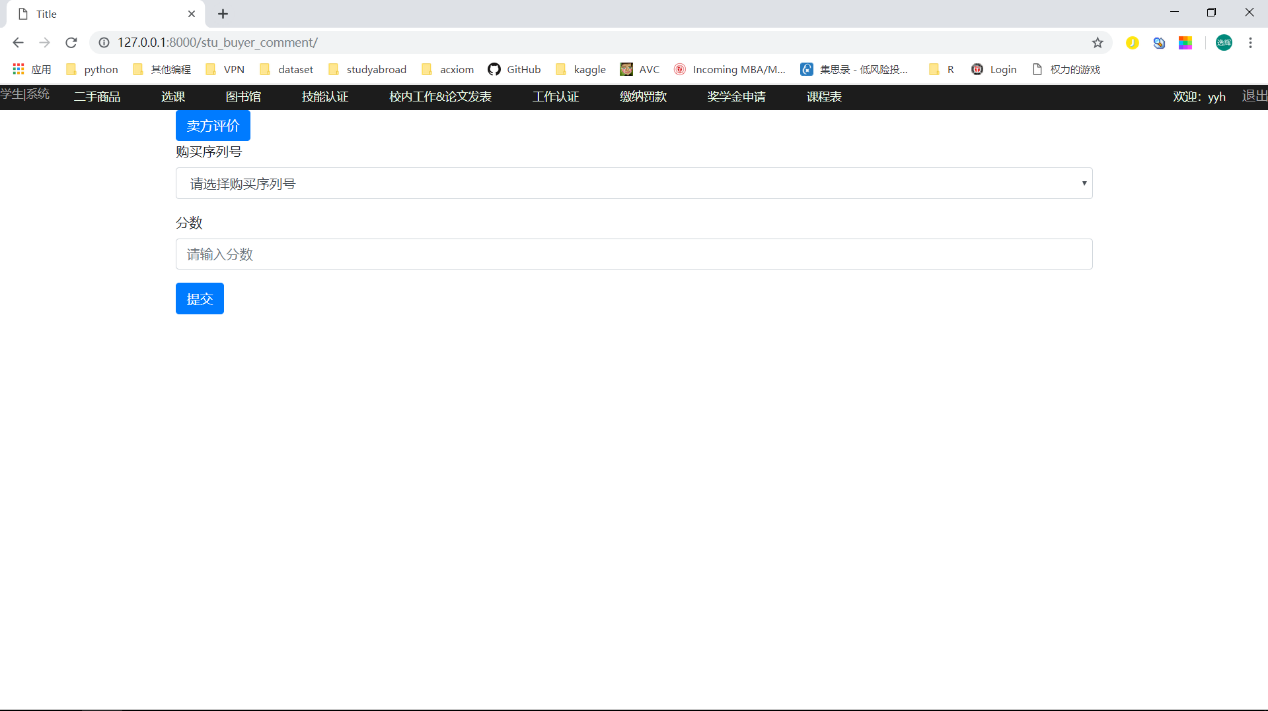


图7.24买方评价

交易结束之后，买方可以对已经支付的订单进行评价。

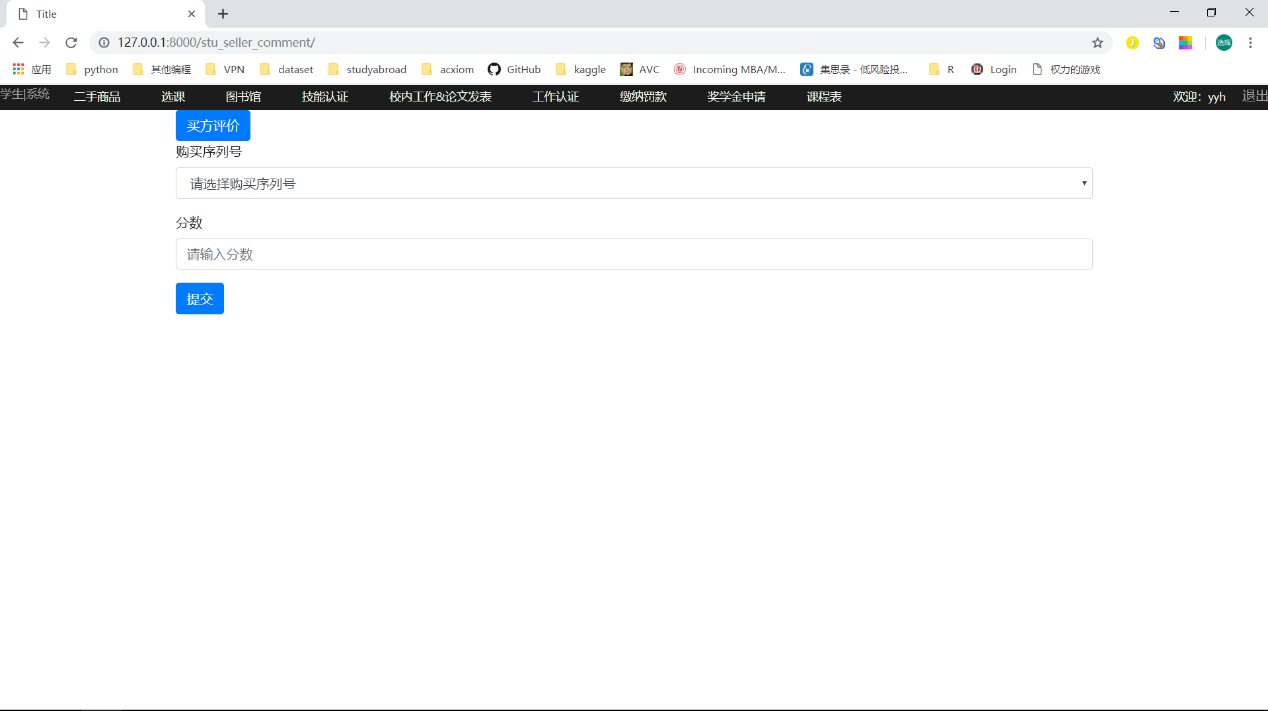


图7.25卖方评价

交易结束之后，卖方可以对已经支付的订单进行评价。

**7.5个人申请子系统实操**

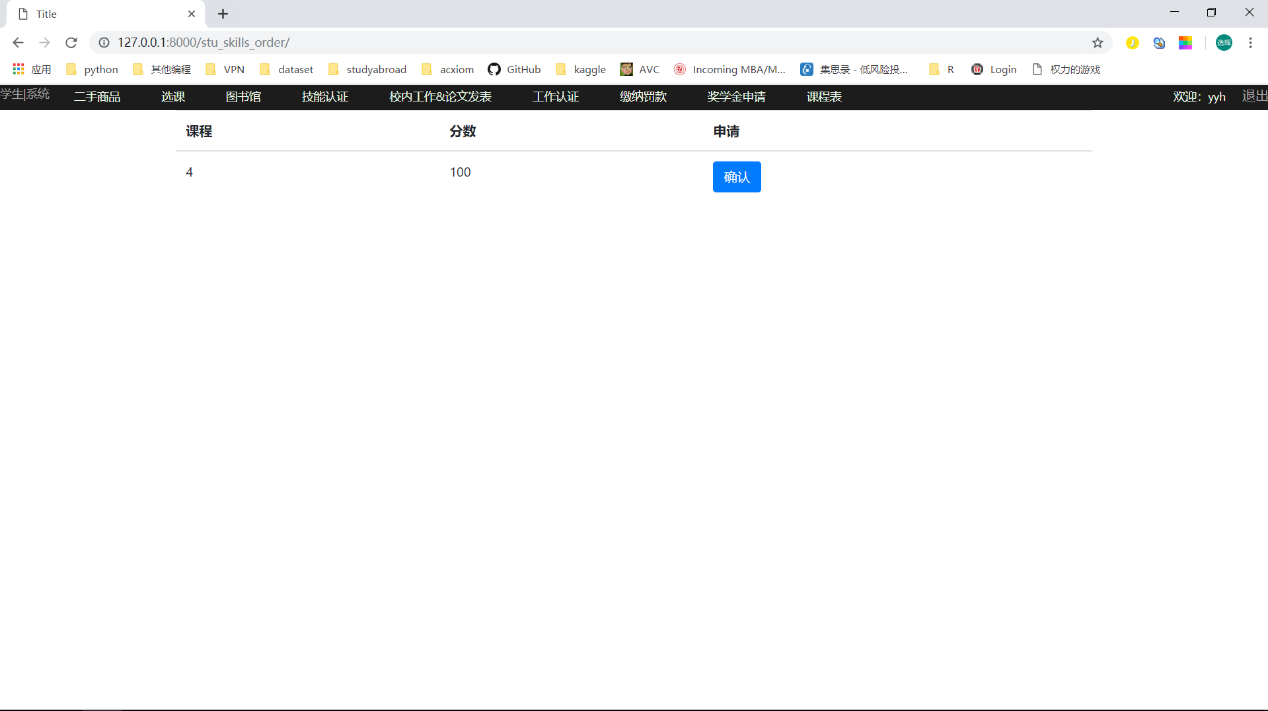


图7.26学生申请技能界面

学生可以发起自己的技能申请，前提是分数大于90分，且该课程可以被申请。

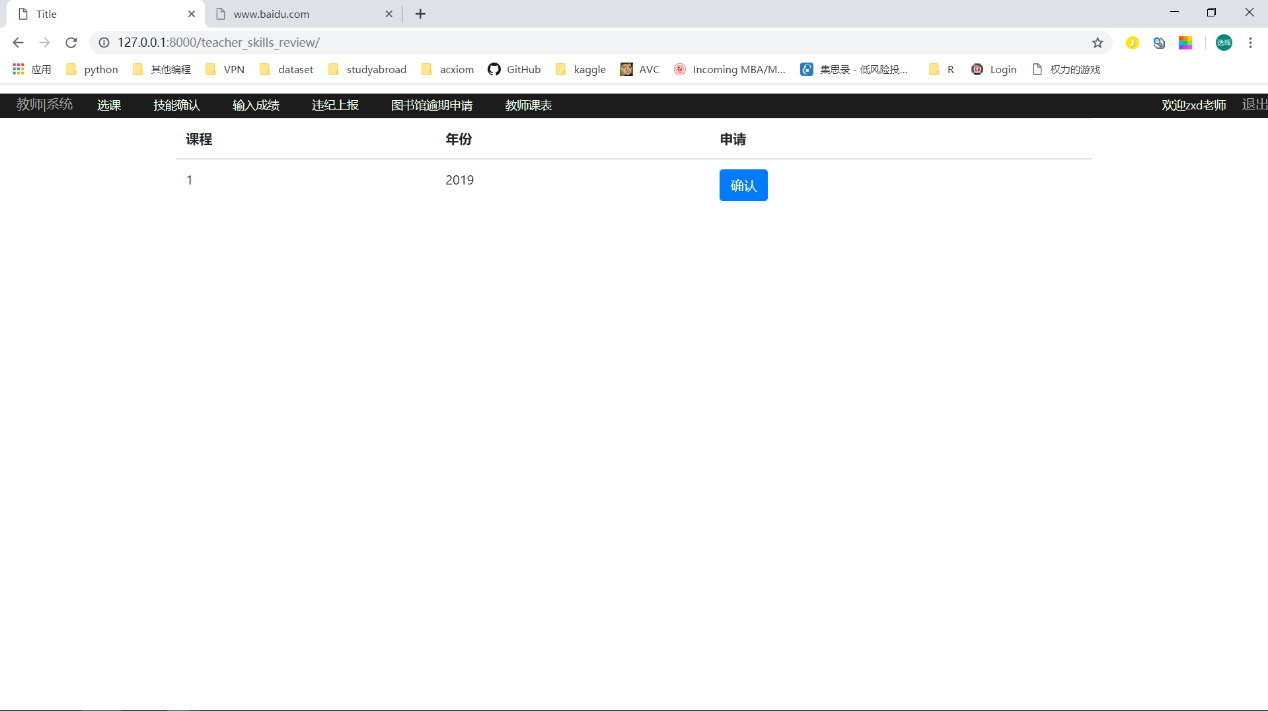


图7.27学生申请技能界面

在教师界面，教师可以对发起的申请进行确认。

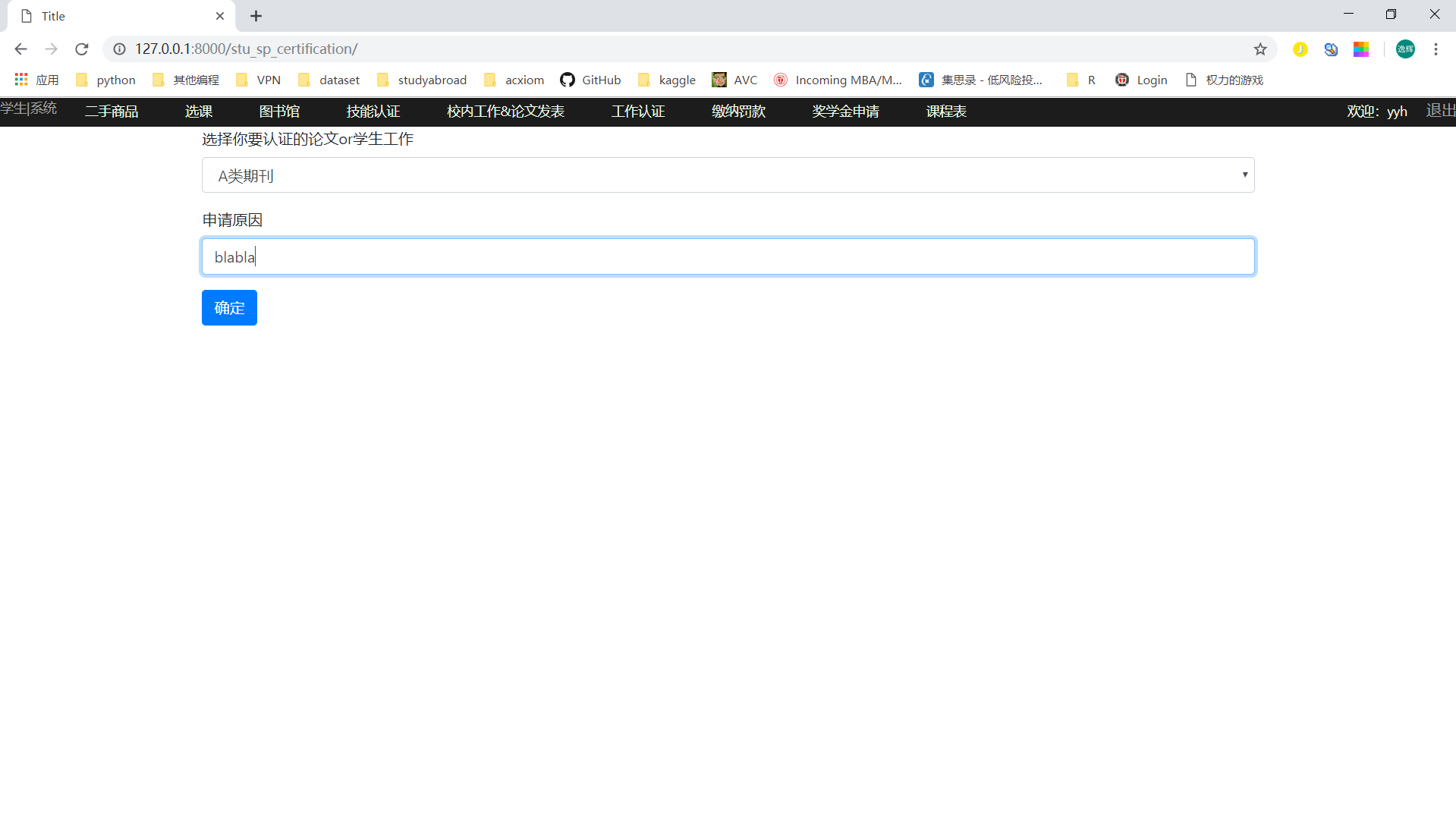


图7.28学生论文和学生工作认证清单

论文和学生校内工作认证，需要学生从加分规则表中选择。

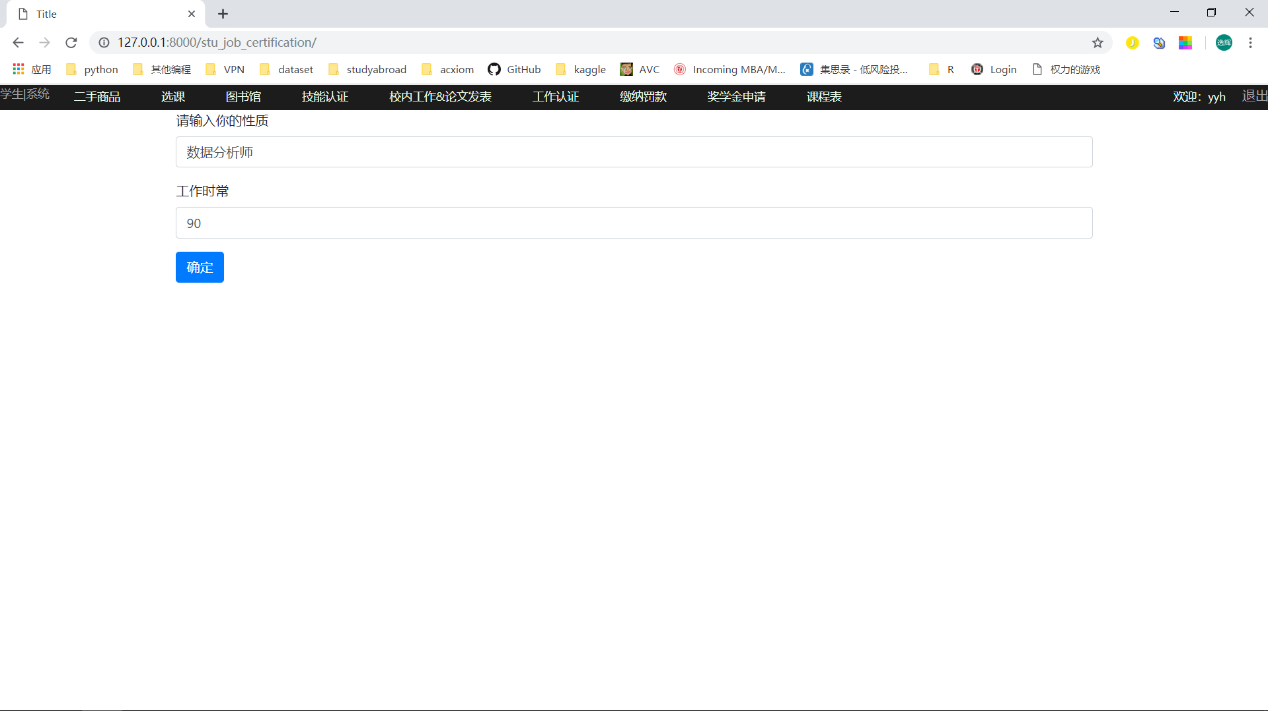


图7.29学生论文和学生工作认证界面

学生可以发布自己的工作情况，等待辅导员确认。

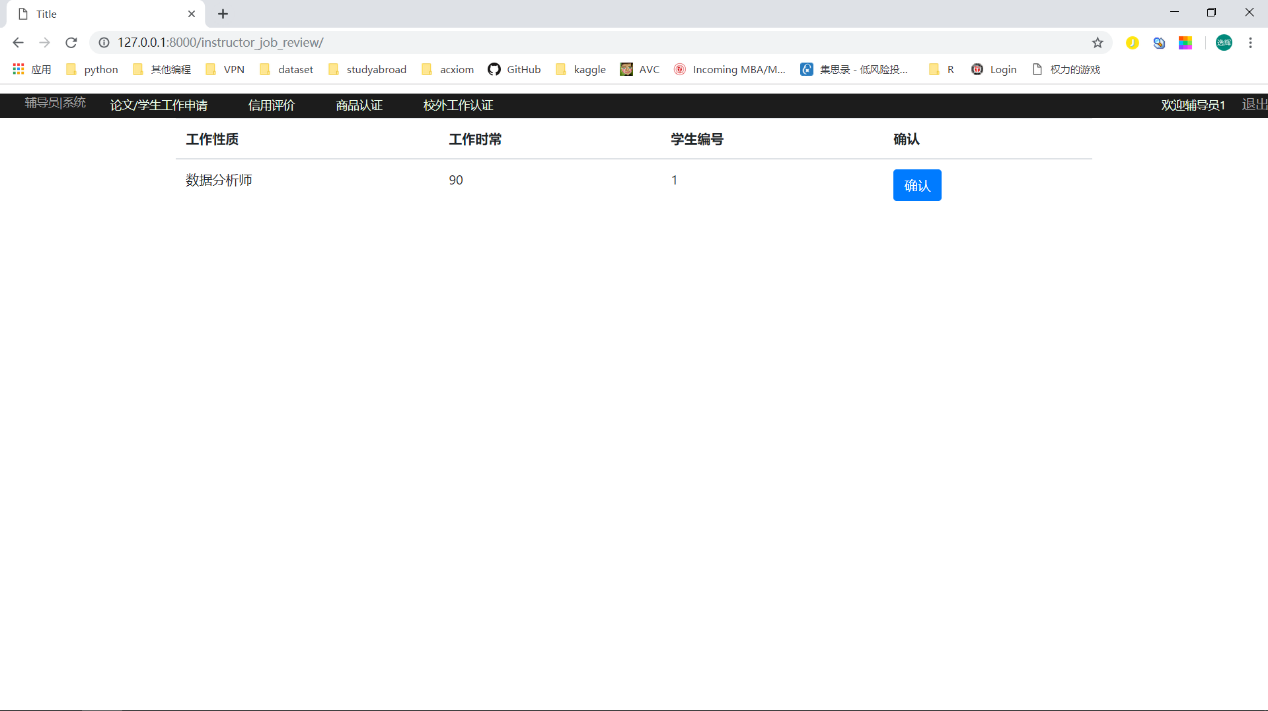


图7.30学生工作通过界面

辅导员论文、学生工作申请批准界面，辅导员点击确定就可以更新确认属性。

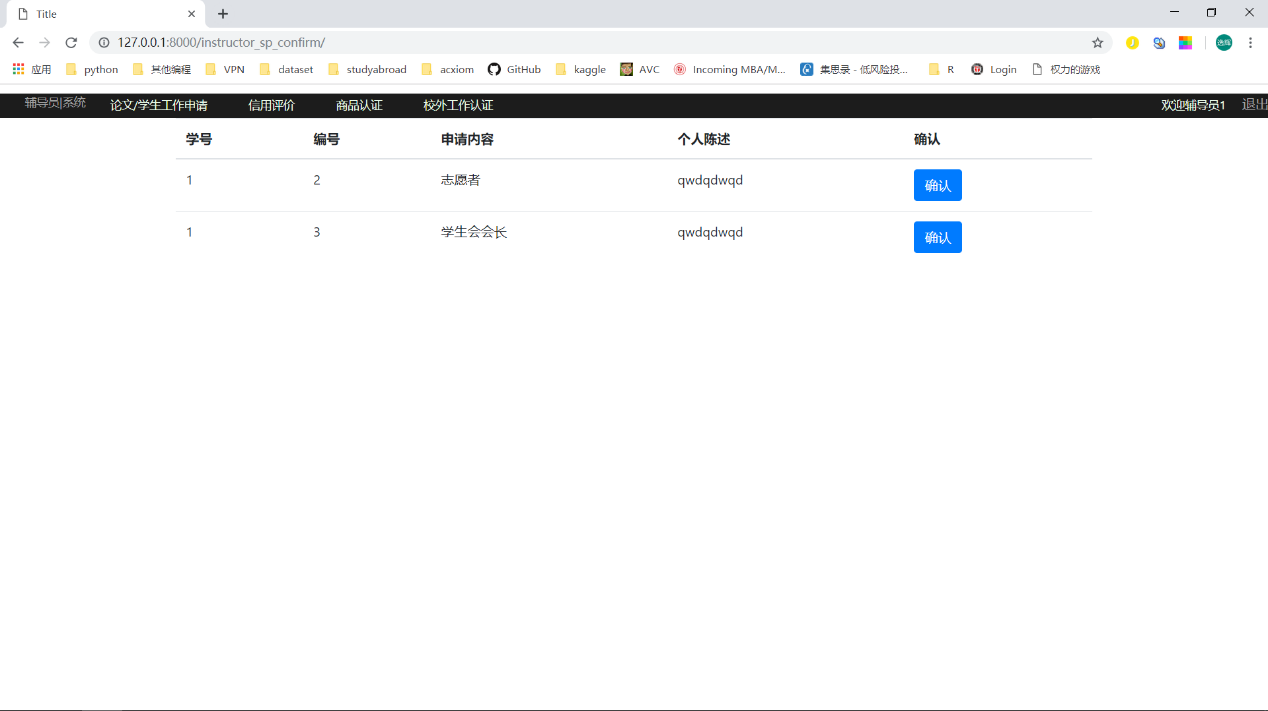


图7.31论文和学生校内工作通过界面

学生工作审核清单，只有点击确认之后，相应的分数才会计入最后的评价。

**7.6信用评级子系统实操**

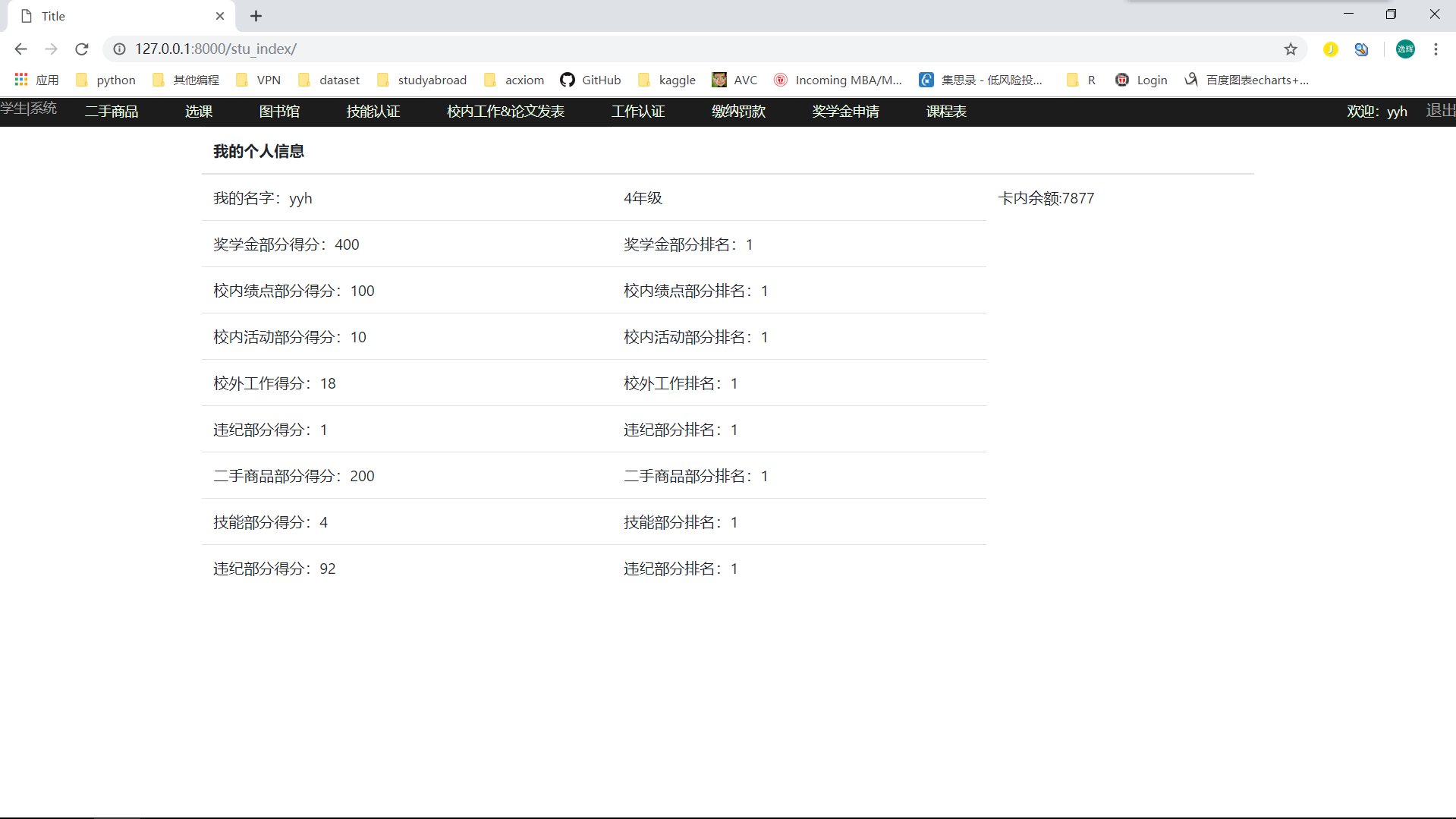


图7.32 信用评级子系统实操1

如果改变该学生的某项得分，学期结束通过验证后，相应的数据和排名也会改变。

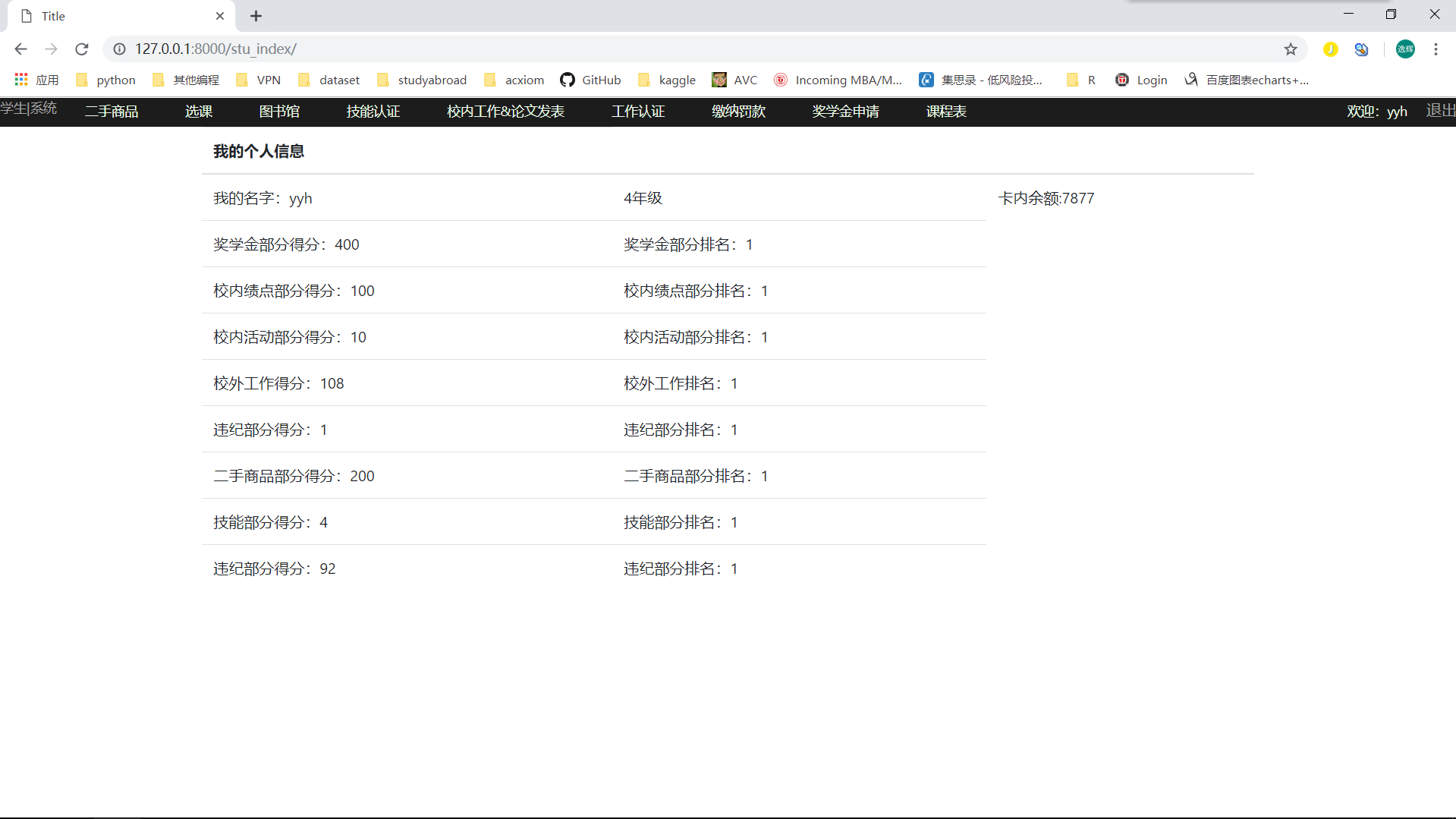


图7.32 信用评级子系统实操2

对于非学生而言，即需要查看学生成绩的用人单位和贷款发放机构，他们看到的界面是这样的：

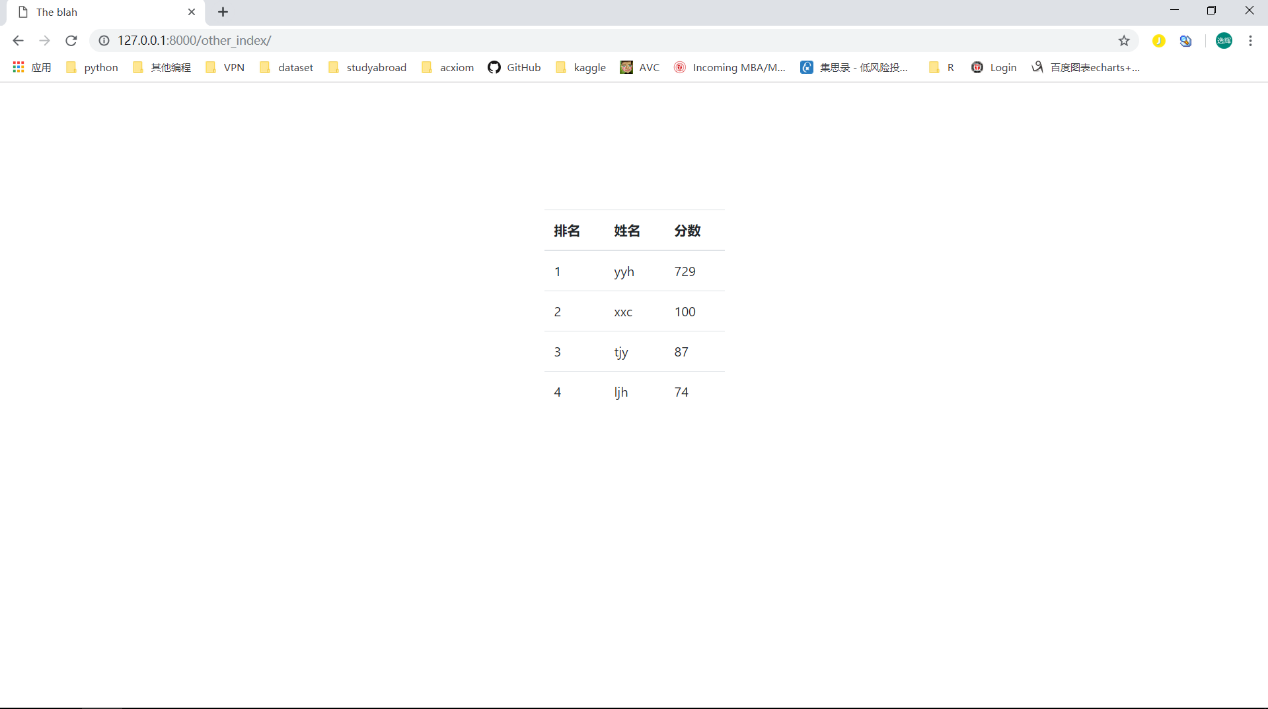


图7.33信用评级子系统实操3

考虑到实际的效果，笔者将学生的总分归类放在一个界面，供校外人员参考和使用，并且在学生界面加入雷达图，便于学生理解和弥补自身的不足，同时，在5月1日期间，笔者修改了网站的菜单栏，用dropdown样式，将不同的内容归类起来，便于操作，也更加有利于理解网站逻辑。

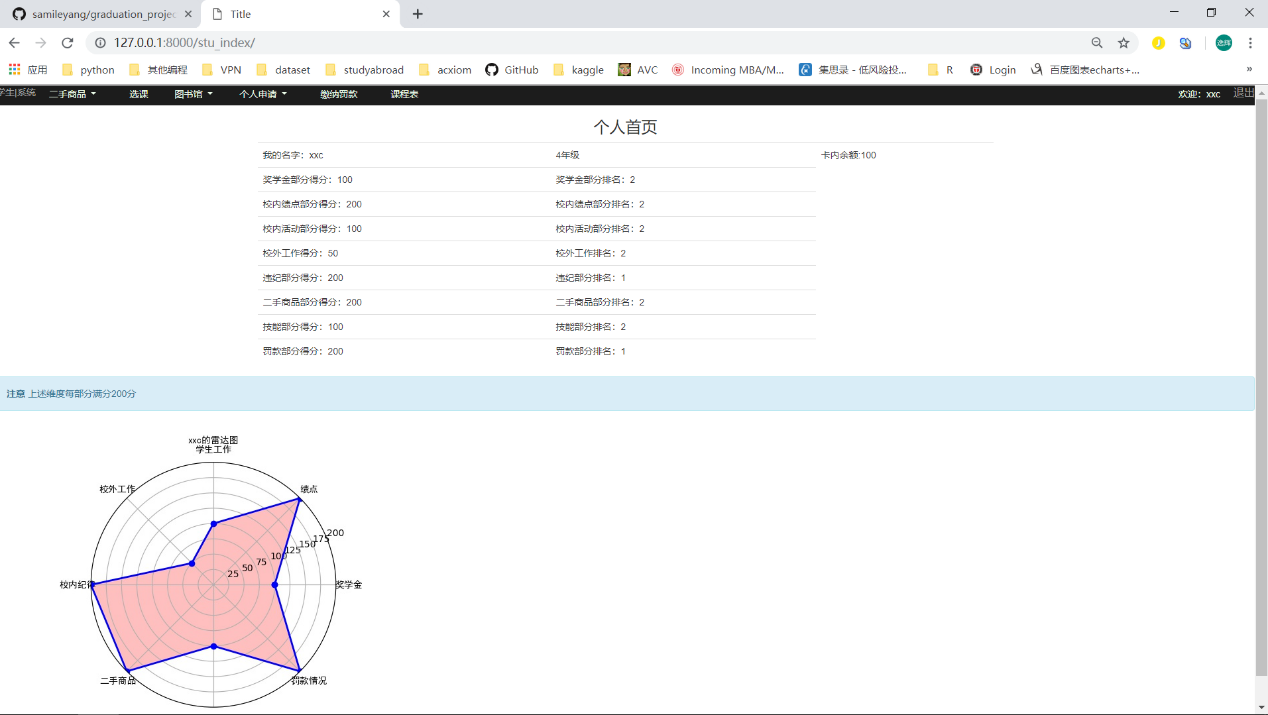


图7.34信用评级子系统实操4

可以看到学生2，xxc同学在绩点和学生罚款情况方面表现良好，但是校外工作等方面需要加强。如此，学生就比较容易理解。

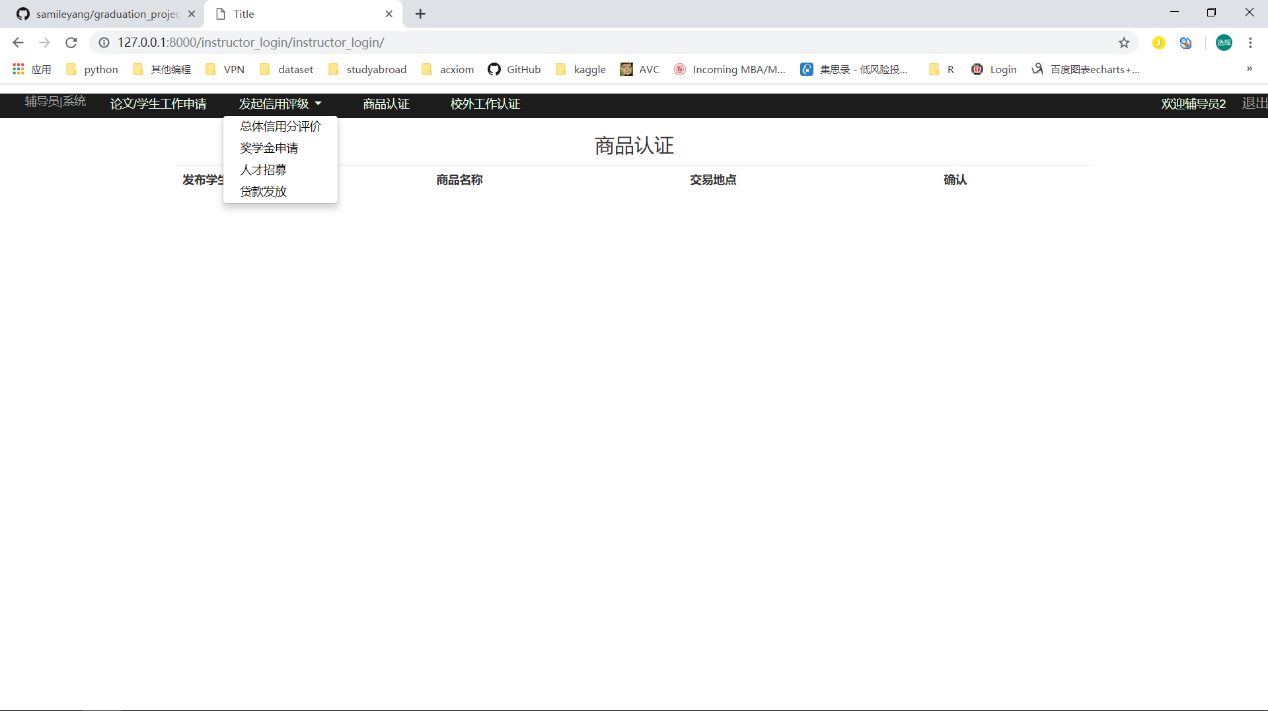
同时，辅导员触发学期信用分结算的按钮也加入了辅导员界面。

图7.35信用评级子系统实操5

辅导员可以发起总体评价、奖学金申请、人才招募信用分、贷款发放信用分这四种。

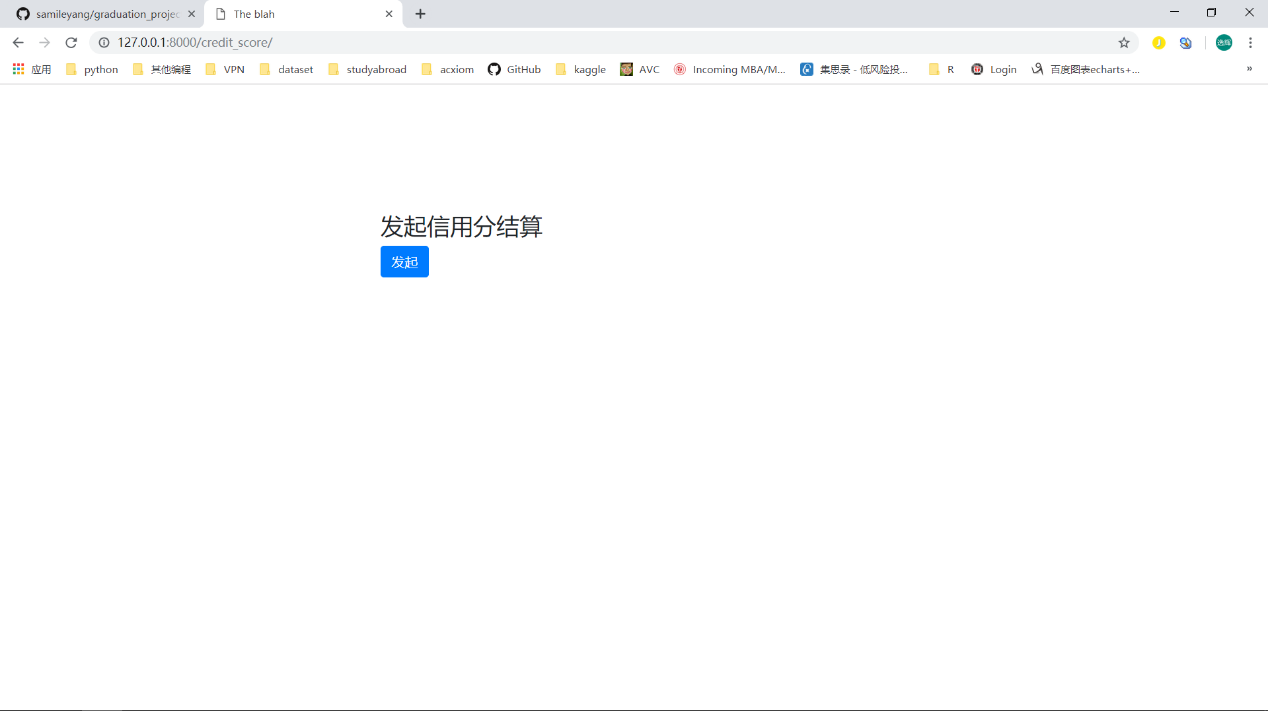


图7.36信用评级子系统实操6

其他四类界面相同，在此就不再重复了。

外部人员看到的界面，则是如下。



图7.37信用评级子系统实操7

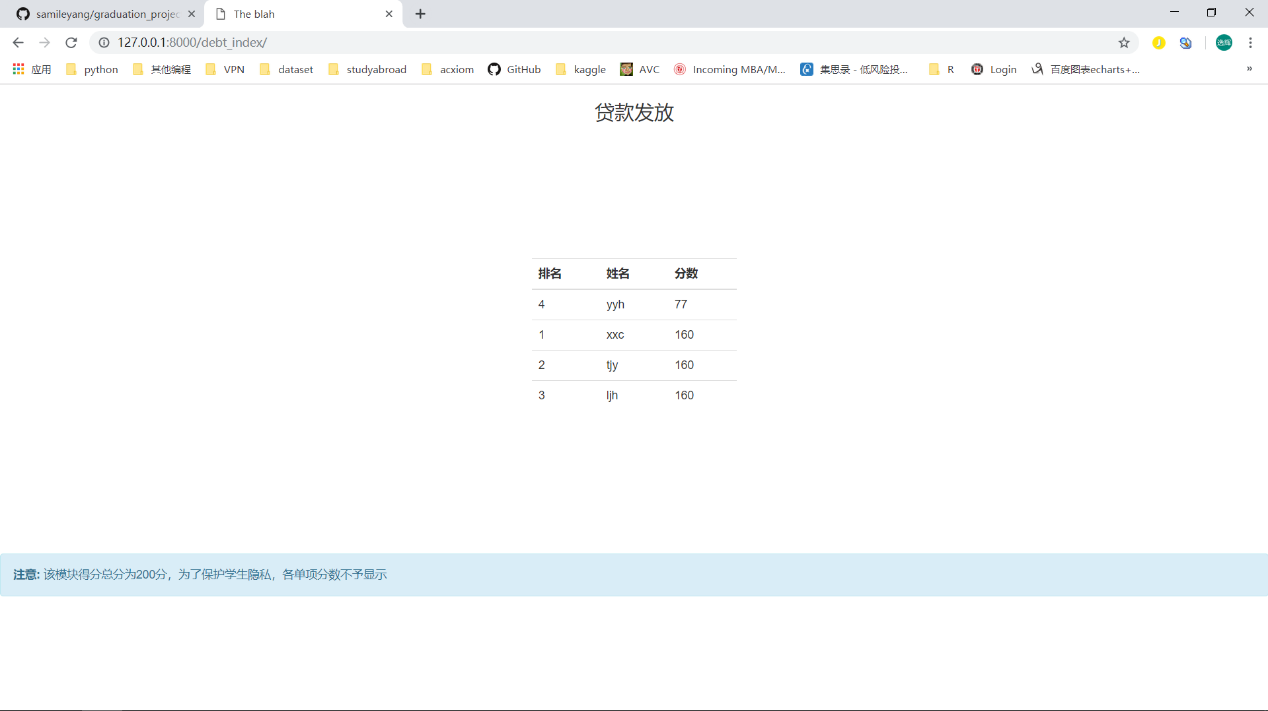


图7.38信用评级子系统实操8

可以发现，这两个维度中，同一个学生的得分都是不同的。这是因为，这其中的计算逻辑有所不同。在上文第三章已经讲述了详细的算分逻辑，但是由于实际操作难度极大，偏离了系统的本质，所以提出以下的算法。

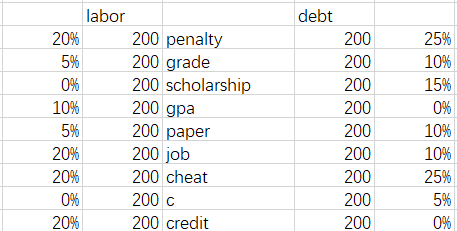


图7.39 实际分数分布

每个小维度都是200分的总分，每个维度可以被学生个人查看，但是不能被校外的人查看，是为了保护学生的隐私。

两侧的百分比，则是在总分中占据的百分比，在人才招募评分中（左），奖学金不计入评分，历史工作的占比达到20%，诚信也是最为重要的维度。

**第八章 总结与展望**

虽然完成了整套系统的设计与实施，但是值得改进的地方还很多。例如以蚂蚁花呗为参考，整套信用评分的关键还只是集中在信息收集方面，例如教务子系统，图书管理子系统等等，但是学生能够通过信用评分子系统具体得到什么好处，还尚未可知。蚂蚁花呗可以信用借伞，租车，然而笔者的系统却无法有类似的功能。在未来，这个系统可以往实用性的角度发展。

所以，我认为，未来蚂蚁花呗可以大肆进入校园，和学生的校内表现结合起来。当然笔者也很希望看到专业的征信公司，将目标瞄向在校大学生，不仅可以提升大学生的授信意识，更重要的是让大学生可以从较高的信用分中得到乐趣，得到一定的便利。否则，枯燥的征信体系往往不能达到想要的效果。

**参考文献**

1. 杜淼淼. 美国个人信用评分系统及其启示[J]. 南方金融,2008(08):63-66.
2. 姜琳.美国FICO评分系统述评[J].商业研究,2006(20):81-84.
3. 鞠传霄. 互联网时代的个人信用评估机制研究——以芝麻信用为例[J]. 现代管理科学,2018(05):109-111.
4. 阮玉芳.个人信用报告展示问题研究——对《征信业管理条例》的思考[J].征信,2018,36(09):28-31.
5. 钱颖懿,钮珏,金妍乐,田益坤.松江大学城学生个人信用系统设计与实现[J].科技创新导报,2018,15(11):136-142.
6. None. Economic Anthropology[J]. 2018, 5(1):2-5.
7. Dey S . Credit limit management using action-effect models[C]// Financial Theory and Engineering (ICFTE), 2010 International Conference on. IEEE, 2010.
8. 刘新海, 丁伟. 大数据征信应用与启示——以美国互联网金融公司ZestFinance为例[J]. 清华金融评论, 2014(10):93-98.
9. 沈小袷. 论胜任力理论在大学生综合素质评价中的应用[J]. 湘潮(下半月)(理论),2008(06):12-13.
10. 范君晖. 应用型高校大学生学习动力激励机制[J]. 高教论坛,2012(09):77-78+100.
11. 杨一,李梦雪,朱沛东,高翔. 大学生理财认知及行为的调查分析——以南京大学生为例[J]. 新经济,2014(17):17-18.
12. 何绚. 关于大学生义工活动的思考[J]. 科教文汇(上旬刊),2013(09):177-178.
13. 赵晓兵，刘京娟.开展网络运用道德教育培养大学生守法精神[J].才智，2011(31):288.
14. 陆建香,张文元. 高校个人信用档案建立及管理[J]. 档案时空(业务版),2006(10):38-39.
15. 杨秀春. 浅谈建立个人信用档案的紧迫性[J]. 内蒙古师范大学学报(哲学社会科学版),2002(S2):164-168.

**致谢**

感谢这几个月内朱小栋副教授对我的悉心指导和帮助，提供了之前研究的方法。同时，该项研究其实从上个学期就开始了，我与吴靓学长一起完成的对于南航和上理部分学生的信用评分可视化操作，包括了雷达图，各自排名等等。

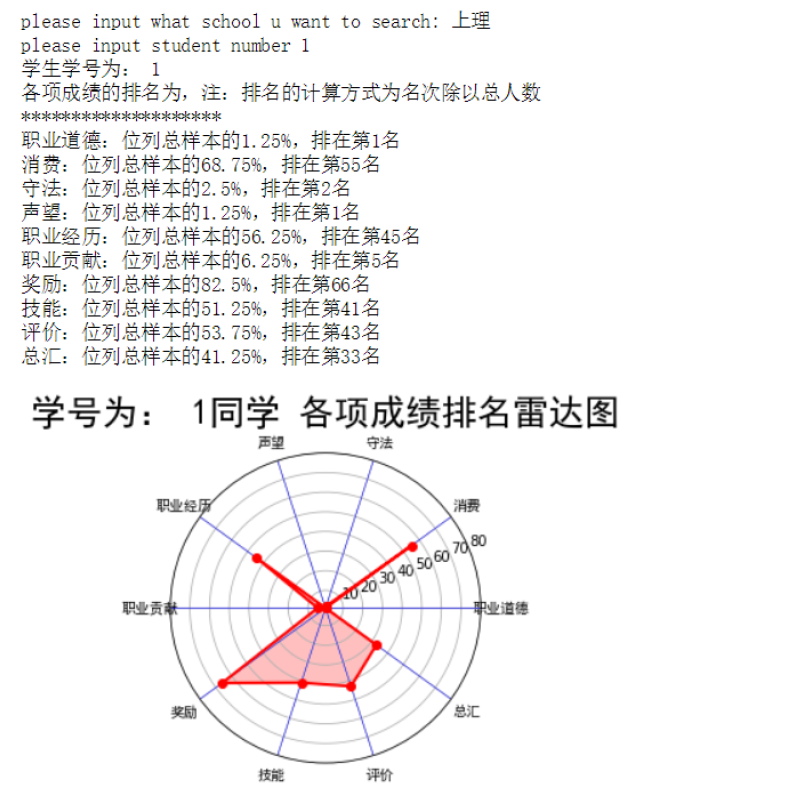


图9.1 之前的研究成果1

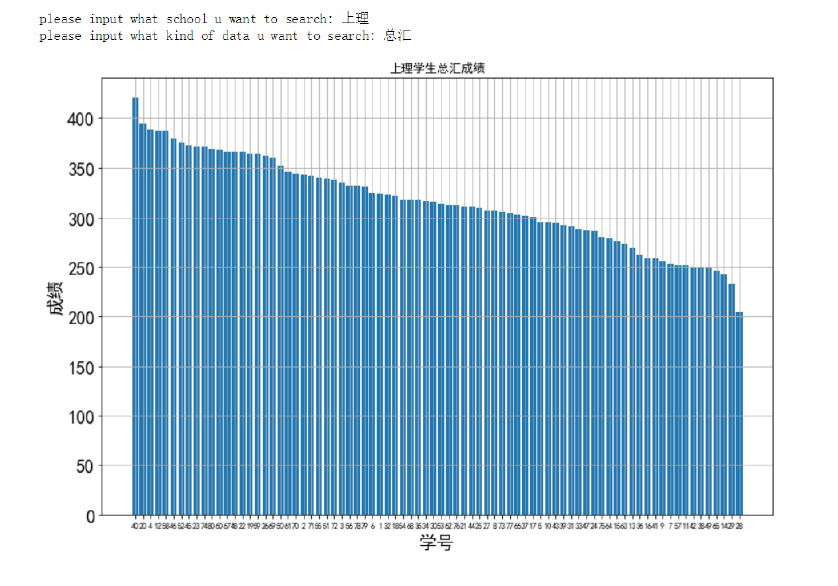


图9.2之前的研究成果2

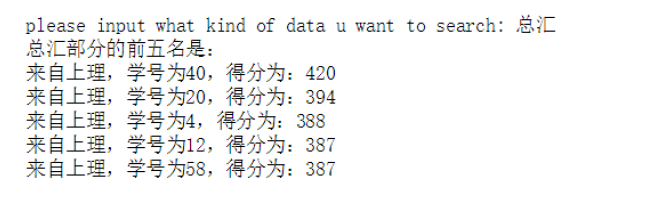


图9.3之前的研究成果3

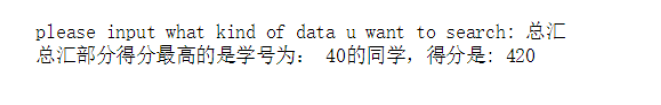


图9.4之前的研究成果4

可以看出，研究的维度还是很多的，但是由于这是Django框架的系统，所以有些无法快速的、实现。如果有机会，那么还是会把该系统更加的完善。

**附录**

下述代码为views.py 文件，即方法文件中的完整代码：

from django.shortcuts import render,redirect,HttpResponse

from mybishe import models

from django.db.models import F

import datetime

from django.db.models import Sum,Count,Max,Min,Avg

import matplotlib.pyplot as plt

import numpy as np

def stu\_info(request):

if request.method == 'GET':

majors = models.Major.objects.all()

educations = models.BorrowRule.objects.all()

instructors = models.Instructor.objects.all()

return render(request,'jwc/stu\_info.html',{'majors' : majors,'educations': educations,'instructors': instructors})

if request.method == 'POST':

majors = models.Major.objects.all()

educations = models.BorrowRule.objects.all()

instructors = models.Instructor.objects.all()

stu\_pwd = request.POST.get('stu\_pwd')

stu\_name = request.POST.get('stu\_name')

year = request.POST.get('year')

major\_name = request.POST.get('major\_id')

money = 100

stu\_edu = request.POST.get('stu\_edu')

instructor\_id = request.POST.get('instructor\_id')

students = models.Student(stu\_pwd = stu\_pwd,stu\_name=stu\_name,stu\_year=year,major\_name=major\_name,stu\_edu=stu\_edu,instructor\_id=instructor\_id,money = money)

students.save()

return render(request,'jwc/stu\_info.html',{'majors' : majors,'educations': educations,'instructors': instructors})

def stu\_login(request):

if request.method == 'GET':

return render(request, 'student/stu\_login.html')

if request.method == 'POST':

user = models.Student.objects.get(stu\_id = request.POST['stu\_id'])

request.session['stu\_name'] = user.stu\_name

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

return redirect('/stu\_index/')

def stu\_library\_borrow(request):

borrows = models.Borrow.objects.filter(status ='0')

return render(request,'student/stu\_library\_borrow.html',{'borrows':borrows})

def stu\_borrow\_order(request):

mybook = models.Borrow.objects.get(book\_id = request.GET.get('book\_id',None))

stu\_id = request.session.get('stu\_id',None)

book\_id = request.GET.get('book\_id',None)

now = datetime.datetime.now()

delta = datetime.timedelta(days=90)

n\_days = now + delta

price = mybook.price

borrow\_order = models.BorrowOrder(stu\_id = stu\_id,book\_id=book\_id,startdate=now,supposedate = n\_days,return\_status = 0,price = price)

mybook.status = '1'

mybook.save()

borrow\_order.save()

borrows = models.Borrow.objects.filter(status ='0')

return render(request,'student/stu\_library\_borrow.html',{'borrows':borrows})

def library\_return(request):

orders = models.BorrowOrder.objects.filter(return\_status = '0')

return render(request,'jwc/library\_return.html',{'orders':orders})

def stu\_book\_return(request):

book\_id = request.GET.get('book\_id',None)

borrow\_order = models.BorrowOrder.objects.get(return\_status = '0', book\_id = book\_id)

models.Borrow.objects.filter(book\_id = book\_id).update(status = '0')

now = datetime.datetime.now()

borrow\_order.return\_status = 1

borrow\_order.actdate = datetime.datetime.now()

if datetime.datetime.now() > borrow\_order.supposedate:

pen\_money = (datetime.datetime.now() - borrow\_order.supposedate).days

borrow\_id = borrow\_order.borrow\_id

stu\_id = request.session.get('stu\_id')

penatly = models.Penalty(borrow\_id = borrow\_id,pen\_money = pen\_money,stu\_id = stu\_id,pen\_type="逾期",paid = 0, appeal = 0)

penatly.save()

else:

pass

borrow\_order.save()

orders = models.BorrowOrder.objects.filter(return\_status = '0')

return render(request,'jwc/library\_return.html',{'orders':orders,'now':now})

def stu\_sp\_certification(request):

if request.method =='GET':

certifications = models.Rules.objects.all()

return render(request,'student/stu\_sp\_certification.html',{'certifications':certifications})

if request.method =='POST':

rule\_name = request.POST.get('rule\_name')

print(rule\_name)

stu\_id = request.session.get('stu\_id')

comment = request.POST.get('comments')

print(comment)

getrule = models.Rules.objects.get(rule\_name = rule\_name)

rule\_id = getrule.rule\_id

score = getrule.score

addorder = models.AddCreditOrder(stu\_id = stu\_id,rule\_id = rule\_id,rule\_name = rule\_name,score = score,status = 0,comment = comment)

addorder.save()

certifications = models.Rules.objects.all()

return render(request,'student/stu\_sp\_certification.html',{'certifications':certifications})

def stu\_products(request):

if request.method =='POST':

pro\_name = request.POST.get('product\_name')

price\_org = request.POST.get('price\_org')

price\_cur = request.POST.get('price\_cur')

location = request.POST.get('location')

seller\_id = request.session.get('stu\_id',None)

status = 0

products = models.Products(pro\_name = pro\_name,price\_org = price\_org,price\_cur = price\_cur,seller\_id = seller\_id,location = location,status = status,paid =0)

products.save()

return render(request,'student/stu\_products.html')

if request.method =="GET":

return render(request,'student/stu\_products.html')

def instructor\_pro\_review(request):

products = models.Products.objects.filter(status=0)

return render(request,'instructor/instructor\_pro\_review.html',{'products':products})

def stu\_pro\_purchase(request):

products = models.Products.objects.filter(status=1)

return render(request,'student/stu\_pro\_purchase.html',{'products':products})

def buy\_products(request):

pro\_id = request.GET.get('pro\_id',None)

buyer\_id = request.session.get('stu\_id')

myproduct = models.Products.objects.get(pro\_id = pro\_id)

price\_cur = myproduct.price\_cur

seller\_id = myproduct.seller\_id

myproduct.paid = 1

myproduct.save()

moneychange = models.Student.objects.get(stu\_id= buyer\_id)

moneychange.money -= price\_cur

moneychange.save()

addproduct = models.Orders(product\_id = pro\_id,buyer\_id = buyer\_id,price = price\_cur,received = 0,seller\_id = seller\_id)

addproduct.save()

products = models.Products.objects.filter(status=1)

return render(request,'student/stu\_pro\_purchase.html',{'products':products})

def teacher\_choose\_course(request):

courses = models.CultivatePlan.objects.filter(class\_field\_\_lt=F('limit'))

return render(request,'teacher/teacher\_choose\_course.html',{'courses':courses})

def stu\_choose\_course(request):

students = models.StudentChoice.objects.filter(stu\_id=request.session.get("stu\_id"))

stu\_list = []

for student in students:

stu\_list.append(student.course\_id)

courses = models.TeacherChoice.objects.all()

return render(request,'student/stu\_choose\_course.html',{'courses':courses,'stu\_list':stu\_list})

def stu\_book\_lost(request):

books = models.BorrowOrder.objects.filter(stu\_id = request.session.get('stu\_id'),return\_status = 0)

return render(request,'student/stu\_book\_lost.html',{'books':books})

def book\_lost(request):

borrow\_id = request.GET.get('borrow\_id',None)

mybook = models.BorrowOrder.objects.get(borrow\_id=borrow\_id)

mybook.actdate = datetime.datetime.now()

mybook.return\_status = 1

stu\_id = mybook.stu\_id

students = models.Student.objects.get(stu\_id = stu\_id)

price = mybook.price

students.money = students.money - price

penatly = models.Penalty(borrow\_id = borrow\_id,stu\_id = stu\_id, pen\_money = price,pen\_type = "丢书")

mybook.save()

penatly.save()

students.save()

books = models.BorrowOrder.objects.filter(stu\_id = request.session.get('stu\_id'),return\_status = 0)

return render(request,'student/stu\_book\_lost.html',{'books':books})

def teacher\_give\_score(request):

if request.method == 'GET':

students = models.StudentChoice.objects.filter(score\_\_isnull=True)

return render(request,'teacher/teacher\_give\_score.html',{'students':students})

if request.method =='POST':

student\_choice\_id = request.POST.get('student\_choice\_id')

print(student\_choice\_id)

score = request.POST.get('score')

print(score)

addscore = models.StudentChoice.objects.get(student\_choice\_id=student\_choice\_id)

addscore.score = score

addscore.save()

students = models.StudentChoice.objects.filter(score\_\_isnull=True)

return render(request,'teacher/teacher\_give\_score.html',{'students':students})

def stu\_skills\_order(request):

courses = models.StudentChoice.objects.filter(score\_\_gte=90, stu\_id = request.session.get('stu\_id'),status = 0)

return render(request,'student/stu\_skills\_order.html',{'courses':courses})

def instructor\_sp\_confirm(request):

rules = models.AddCreditOrder.objects.filter(status = 0)

return render(request,'instructor/instructor\_sp\_confirm.html',{"rules":rules})

def add\_review(request):

add\_id = request.GET.get('add\_id')

c\_order = models.AddCreditOrder.objects.get(add\_id=add\_id)

c\_order.status = 1

c\_order.save()

rules = models.AddCreditOrder.objects.filter(status = 0)

return render(request,'instructor/instructor\_sp\_confirm.html',{"rules":rules})

def teacher\_login(request):

if request.method == 'GET':

return render(request, 'teacher/teacher\_login.html')

if request.method == 'POST':

user = models.Teacher.objects.get(tea\_id = request.POST['teacher\_id'])

request.session['teacher\_name'] = user.teacher\_name

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

return render(request,'teacher/teacher\_index.html')

def teacher\_index(request):

return render(request,'teacher/teacher\_index.html')

def jwc\_index(request):

return render(request,'jwc/jwc\_index.html')

def index(request):

return render(request,'index.html')

def instructor\_login(request):

if request.method == 'GET':

return render(request, 'instructor/instructor\_login.html')

if request.method == 'POST':

user = models.Instructor.objects.get(instructor\_id = request.POST['instructor\_id'])

request.session['instructor\_name'] = user.instructor\_name

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

return render(request,'instructor/instructor\_pro\_review.html')

def teacher\_choose\_order(request):

cultivate = models.CultivatePlan.objects.get(course\_id = request.GET.get('course\_id',None))

cultivate.class\_field += 1

status = 0

time = cultivate.time

teacher\_id = request.session.get('teacher\_id',None)

certification = cultivate.certification

year = 2019

teacher\_choose = models.TeacherChoice(course\_year = year,status = status,time = time,certification = certification,teacher\_id = teacher\_id,course\_id = request.GET.get('course\_id'))

teacher\_choose.save()

cultivate.save()

courses = models.CultivatePlan.objects.filter(class\_field\_\_lt=F('limit'))

return render(request,'teacher/teacher\_choose\_course.html',{'courses':courses})

def stu\_choose\_order(request):

teacher\_course = models.TeacherChoice.objects.get(teacher\_choice\_id = request.GET.get('teacher\_choice\_id',None))

teacher\_choice\_id = request.GET.get('teacher\_choice\_id',None)

stu\_id = request.session.get("stu\_id")

course\_id = teacher\_course.course\_id

save\_course = models.StudentChoice(teacher\_choice\_id= teacher\_choice\_id,stu\_id = stu\_id, course\_id =course\_id, status=0)

save\_course.save()

students = models.StudentChoice.objects.filter(stu\_id=request.session.get("stu\_id"))

stu\_list = []

for student in students:

stu\_list.append(student.course\_id)

courses = models.TeacherChoice.objects.all()

return render(request,'student/stu\_choose\_course.html',{'courses':courses,'stu\_list':stu\_list})

def stu\_job\_certification(request):

if request.method == 'GET':

return render(request,'student/stu\_job\_certification.html')

if request.method == 'POST':

stu\_id = request.session.get('stu\_id')

job\_name = request.POST.get("job\_name")

days = request.POST.get("days")

jobs = models.JobCertification(stu\_id = stu\_id,job\_name = job\_name,days=days,status = 0)

jobs.save()

return render(request,'student/stu\_job\_certification.html')

def job\_review(request):

job\_id = request.GET.get('job\_id')

thisjob = models.JobCertification.objects.get(job\_id = job\_id)

thisjob.status = 1

thisjob.save()

jobs = models.JobCertification.objects.all()

return render(request,'instructor/instructor\_job\_review.html',{'jobs':jobs})

def instructor\_job\_review(request):

jobs = models.JobCertification.objects.all()

return render(request,'instructor/instructor\_job\_review.html',{'jobs':jobs})

def received\_products(request):

product\_id = request.GET.get('product\_id')

myproduct = models.Orders.objects.get(product\_id = product\_id)

myproduct.received = 1

price = myproduct.price

seller\_id = myproduct.seller\_id

myproduct.save()

stu\_id = request.session.get('stu\_id')

moneychange = models.Student.objects.get(stu\_id= seller\_id)

moneychange.money += price

moneychange.save()

products = models.Orders.objects.filter(buyer\_id =stu\_id)

return render(request,'student/stu\_cart.html',{'products':products})

def stu\_cart(request):

stu\_id = request.session.get('stu\_id')

products = models.Orders.objects.filter(buyer\_id =stu\_id)

return render(request,'student/stu\_cart.html',{'products':products})

def stu\_pay\_penalty(request):

stu\_id = request.session.get('stu\_id')

penaltys = models.Penalty.objects.filter(paid = 0, appeal = 0, stu\_id = stu\_id)

return render(request,'student/stu\_pay\_penalty.html',{'penaltys':penaltys})

def pay\_penalty(request):

stu\_id = request.session.get('stu\_id')

stu\_pen\_id = request.GET.get('stu\_pen\_id')

mypen = models.Penalty.objects.get(stu\_pen\_id=stu\_pen\_id)

price = mypen.pen\_money

mypen.paid = 1

mypen.save()

moneylose = models.Student.objects.get(stu\_id = stu\_id)

moneylose.money = moneylose.money-price

moneylose.save()

penaltys = models.Penalty.objects.filter(paid = 0, appeal = 0, stu\_id = stu\_id)

return render(request,'student/stu\_pay\_penalty.html',{'penaltys':penaltys})

def stu\_course(request):

stu\_id = request.session.get('stu\_id')

courses = models.StudentChoice.objects.filter(stu\_id = stu\_id)

return render(request,'student/stu\_course.html',{'courses':courses})

def stu\_course\_delete(request):

models.StudentChoice.objects.get(student\_choice\_id = request.GET.get('student\_choice\_id',None)).delete()

stu\_id = request.session.get('stu\_id')

courses = models.StudentChoice.objects.filter(stu\_id = stu\_id)

return render(request,'student/stu\_course.html',{'courses':courses})

def teacher\_course(request):

teacher\_id = request.session.get('teacher\_id')

courses = models.TeacherChoice.objects.filter(teacher\_id = teacher\_id)

return render(request,'teacher/teacher\_course.html',{'courses':courses})

def stu\_buyer\_comment(request):

stu\_id =request.session.get('stu\_id')

if request.method == 'GET':

buyers = models.Orders.objects.filter(buyer\_comm\_\_isnull=True,buyer\_id = stu\_id)

return render(request,'student/stu\_buyer\_comment.html',{'buyers':buyers})

if request.method == 'POST':

order\_id = request.POST.get('order\_id')

score = request.POST.get('score')

addscore = models.Orders.objects.get(order\_id = order\_id)

addscore.buyer\_comm = score

addscore.save()

buyers = models.Orders.objects.filter(buyer\_comm\_\_isnull=True,buyer\_id = stu\_id)

return render(request,'student/stu\_buyer\_comment.html',{'buyers':buyers})

def stu\_seller\_comment(request):

stu\_id =request.session.get('stu\_id')

if request.method == 'GET':

buyers = models.Orders.objects.filter(seller\_comm\_\_isnull=True,seller\_id = stu\_id)

return render(request,'student/stu\_seller\_comment.html',{'buyers':buyers})

if request.method == 'POST':

order\_id = request.POST.get('order\_id')

score = request.POST.get('score')

addscore = models.Orders.objects.get(order\_id = order\_id)

addscore.seller\_comm = score

addscore.save()

buyers = models.Orders.objects.filter(seller\_comm\_\_isnull=True,seller\_id = stu\_id)

return render(request,'student/stu\_seller\_comment.html',{'buyers':buyers})

def pro\_review(request):

pro\_id = request.GET.get('pro\_id')

changestatus = models.Products.objects.get(pro\_id = pro\_id)

changestatus.status =1

changestatus.save()

products = models.Products.objects.filter(status = 0)

return render(request,'instructor/instructor\_pro\_review.html',{'products':products})

def skills\_order(request):

student\_choice\_id = request.GET.get('student\_choice\_id')

changestatus = models.StudentChoice.objects.get(student\_choice\_id=student\_choice\_id)

changestatus.status = 1

changestatus.save()

score = 100

year = 2019

stu\_id = request.session.get('stu\_id')

status = 0

addcredit = models.Credit(stu\_id = stu\_id,year =year,score = score, status =status)

addcredit.save()

courses = models.StudentChoice.objects.filter(score\_\_gte=90, stu\_id = request.session.get('stu\_id'),status = 0)

return render(request,'student/stu\_skills\_order.html',{'courses':courses})

def teacher\_skills\_review(request):

credits = models.Credit.objects.filter(status = 0)

return render(request,'teacher/teacher\_skills\_review.html',{'credits':credits})

def skills\_review(request):

credit\_id = request.GET.get('credit\_id')

changestatus = models.Credit.objects.get(credit\_id = credit\_id)

changestatus.status = 1

changestatus.save()

credits = models.Credit.objects.filter(status = 0)

return render(request,'teacher/teacher\_skills\_review.html',{'credits':credits})

def instructor\_index(request):

losts = models.BorrowOrder.objects.filter(return\_status = 0)

counts = 0

for lost in losts:

if (datetime.datetime.now()-lost.supposedate).days >180:

lost.actdate = datetime.datetime.now()

lost.return\_status = 1

price = lost.price

borrow\_id =lost.borrow\_id

stu\_id = lost.stu\_id

addlost = model.Penalty(stu\_id = stu\_id,pen\_money = price,type = '丢书')

student = model.Student.objects.get(stu\_id = stu\_id)

student.money -= price

student.save()

lost.save()

addlost.save()

counts = count+1

message = "有{}个借阅被判定为自动丢书".format(counts)

return render(request,'instructor/instructor\_index.html',{'message':message})

def stu\_appeal(request):

if request.method == 'GET':

teachers =models.Teacher.objects.all()

appeals =models.Penalty.objects.filter(stu\_id = request.session.get('stu\_id'),paid =0)

return render(request,'student/stu\_appeal.html',{'teachers':teachers,'appeals':appeals})

if request.method == 'POST':

order\_id = request.POST.get('order\_id')

teacher\_id = request.POST.get('tea\_id')

addpen = models.PenAppeal(stu\_pen\_id = order\_id,status = 0, teacher\_id = teacher\_id)

changestatus = models.Penalty.objects.get(stu\_pen\_id = order\_id)

changestatus.appeal = 1

changestatus.save()

addpen.save()

teachers =models.Teacher.objects.all()

appeals =models.Penalty.objects.filter(stu\_id = request.session.get('stu\_id'),paid =0)

return render(request,'student/stu\_appeal.html',{'teachers':teachers,'appeals':appeals})

def teacher\_pen\_review(request):

pens = models.PenAppeal.objects.filter(teacher\_id = request.session.get('teacher\_id'),status =0)

return render(request,'teacher/teacher\_pen\_review.html',{'pens':pens})

def pen\_review(request):

pen\_id =request.GET.get('pen\_id')

changestatus = models.PenAppeal.objects.get(pen\_appeal\_id = pen\_id)

changestatus.status = 1

changestatus.save()

pens = models.PenAppeal.objects.filter(teacher\_id = request.session.get('teacher\_id'),status =0)

return render(request,'teacher/teacher\_pen\_review.html',{'pens':pens})

def teacher\_cheat(request):

if request.method == 'GET':

students = models.StudentChoice.objects.filter(score\_\_isnull=True)

return render(request,'teacher/teacher\_cheat.html',{'students':students})

if request.method == 'POST':

student\_choice\_id = request.POST.get('student\_choice\_id')

student =models.StudentChoice.objects.get(student\_choice\_id=student\_choice\_id)

student.score =0

stu\_id = student.stu\_id

student.save()

addcheat = models.Cheat(stu\_id = stu\_id,student\_choice\_id = student\_choice\_id)

addcheat.save()

students = models.StudentChoice.objects.filter(score\_\_isnull=True)

return render(request,'teacher/teacher\_cheat.html',{'students':students})

def scholarship(request):

if request.method == 'GET':

return render(request,'instructor/scholarship.html')

if request.method == 'POST':

cleartable = models.Scholarship.objects.all()

cleartable.delete()

students = models.Student.objects.all()

stu\_id\_list = []

scholarship\_list=[]

basic\_money = 1000

i = 0

for student in students:

stu\_id\_list.append(student.stu\_id)

print(stu\_id\_list)

for stu\_id in stu\_id\_list:

find\_student = models.StudentChoice.objects.filter(score\_\_isnull=False,stu\_id = stu\_id).aggregate(Avg("score"))

scholarship\_list.append([stu\_id,find\_student['score\_\_avg']])

print(scholarship\_list)

new\_scholarship\_list=sorted(scholarship\_list, key=lambda x:x[1], reverse = True )

print(new\_scholarship\_list)

for \_list in new\_scholarship\_list:

if i < 3:

addscholarship = models.Scholarship(stu\_id = \_list[0],status = 0, money = basic\_money)

addscholarship.save()

print(\_list)

print(basic\_money)

basic\_money -= 300

i +=1

else:

break

return render(request,'instructor/scholarship.html')

def stu\_scholarship(request):

scholarships = models.Scholarship.objects.filter(stu\_id = request.session.get('stu\_id'),status =0)

return render(request,'student/stu\_scholarship.html',{'scholarships':scholarships})

def sch\_order(request):

sch\_id = request.GET.get('sch\_id')

mysch = models.Scholarship.objects.get(sch\_id = sch\_id)

mysch.status = 1

stu\_id = mysch.stu\_id

money = mysch.money

addmoney = models.Student.objects.get(stu\_id = stu\_id)

addmoney.money += money

addmoney.save()

mysch.save()

scholarships = models.Scholarship.objects.filter(stu\_id = request.session.get('stu\_id'),status =0)

return render(request,'student/stu\_scholarship.html',{'scholarships':scholarships})

def test\_grade(year):

if datetime.datetime.now().month in [1,2,3,4,5,6,7,8]:

i = 1

else:

i = 0

return (datetime.datetime.now().year-year)-i+1

def credit\_score(request):

if request.method == 'GET':

return render(request,'instructor/credit\_score.html')

if request.method =='POST':

cleartable = models.SepScore.objects.all()

cleartable.delete()

stu\_scholarship = []

stu\_gpa = []

stu\_paper = []

stu\_job =[]

stu\_cheat = []

stu\_c =[]

stu\_credit=[]

stu\_penalty=[]

stu\_total\_labor = []

stu\_total\_debt =[]

students = models.Student.objects.all()

for student in students:

stu\_id = student.stu\_id

thisstu = models.Student.objects.get(stu\_id = stu\_id)

grade = test\_grade(int(thisstu.stu\_year))

scholarship = test\_none(models.Scholarship.objects.filter(status =1, stu\_id = stu\_id).aggregate(Sum("money"))['money\_\_sum'])

if scholarship == 0:

scholarship = 100

else:

scholarship = scholarship/10 +100

if scholarship>200:

scholarship =200

stu\_scholarship.append([stu\_id,scholarship])

gpa = test\_none(models.StudentChoice.objects.filter(score\_\_isnull=False,stu\_id = stu\_id).aggregate(Avg("score"))['score\_\_avg'])

stu\_gpa.append([stu\_id,gpa\*2])

paper = test\_none(models.AddCreditOrder.objects.filter(status =1, stu\_id = stu\_id).aggregate(Sum("score"))['score\_\_sum'])

if paper == 0:

paper = 100

else:

paper = 100+paper

if paper > 200:

paper =200

stu\_paper.append([stu\_id,paper])

cheat = models.Cheat.objects.filter(stu\_id = stu\_id).count()

if cheat > 0:

cheat = 0

else:

cheat =200

stu\_cheat.append([stu\_id,cheat])

credit= models.Credit.objects.filter(stu\_id = stu\_id,status = 1).count()

if credit ==0:

credit =100

else:

credit = 100+20\*credit

if credit > 200:

credit = 200

stu\_credit.append([stu\_id,credit])

job = test\_none(models.JobCertification.objects.filter(stu\_id = stu\_id,status = 1).aggregate(Sum('days'))['days\_\_sum'])

if job == 0:

job = 50

else:

job = 50+job

if job > 200:

job =200

stu\_job.append([stu\_id,job])

s\_c = models.Orders.objects.filter(buyer\_id = stu\_id,seller\_comm\_\_isnull=False).aggregate(Avg('seller\_comm'))['seller\_comm\_\_avg']

b\_c = models.Orders.objects.filter(seller\_id = stu\_id,buyer\_comm\_\_isnull=False).aggregate(Avg('buyer\_comm'))['buyer\_comm\_\_avg']

s\_c = test\_none(s\_c)

b\_c = test\_none(b\_c)

if (s\_c+b\_c) == 0:

s\_b = 200

else:

s\_b = s\_c+b\_c

stu\_c.append([stu\_id,s\_b])

penatly = test\_none(models.Penalty.objects.filter(stu\_id = stu\_id,paid =1).aggregate(Sum('pen\_money'))['pen\_money\_\_sum'])

yourpen = penatly

if penatly == 0:

newscore =200

else:

newscore = 200-yourpen\*5

if newscore <0:

newscore =0

stu\_penalty.append([stu\_id,newscore])

stu\_scholarship = sorted\_rank(stu\_scholarship)

score\_type = 'stu\_scholarship'

savedata(stu\_scholarship,score\_type)

stu\_gpa = sorted\_rank(stu\_gpa)

savedata(stu\_gpa,'stu\_gpa')

stu\_paper = sorted\_rank(stu\_paper)

savedata(stu\_paper,'stu\_paper')

stu\_job = sorted\_rank(stu\_job)

savedata(stu\_job,'stu\_job')

stu\_cheat = sorted\_rank(stu\_cheat)

savedata(stu\_cheat,'stu\_cheat')

stu\_c = sorted\_rank(stu\_c)

savedata(stu\_c,'stu\_c')

stu\_credit = sorted\_rank(stu\_credit)

savedata(stu\_credit,'stu\_credit')

stu\_penalty = sorted\_rank(stu\_penalty)

savedata(stu\_penalty,'stu\_penalty')

print(stu\_scholarship,stu\_gpa,stu\_paper,stu\_job,stu\_cheat,stu\_c,stu\_credit,stu\_penalty,sep ='\n')

return render(request,'instructor/credit\_score.html')

def savedata(yourlist,score\_type):

for ylist in yourlist:

savedata = models.SepScore(score\_type =score\_type ,stu\_id = ylist[0],score =ylist[1],rank = ylist[2])

savedata.save()

def test\_none(number):

if number == None:

number = 0

else:

number = number

return number

def stu\_index(request):

stu\_id = request.session.get('stu\_id')

student = models.Student.objects.get(stu\_id = stu\_id)

money = student.money

grade = test\_grade(int(student.stu\_year))

name = student.stu\_name

scholarship = models.SepScore.objects.get(score\_type='stu\_scholarship',stu\_id=stu\_id)

gpa = models.SepScore.objects.get(score\_type = 'stu\_gpa',stu\_id = stu\_id)

paper = models.SepScore.objects.get(score\_type = 'stu\_paper',stu\_id = stu\_id)

job = models.SepScore.objects.get(score\_type = 'stu\_job',stu\_id = stu\_id)

cheat = models.SepScore.objects.get(score\_type = 'stu\_cheat',stu\_id = stu\_id)

c = models.SepScore.objects.get(score\_type = 'stu\_c',stu\_id = stu\_id)

credit = models.SepScore.objects.get(score\_type = 'stu\_credit',stu\_id = stu\_id)

penalty = models.SepScore.objects.get(score\_type = 'stu\_penalty',stu\_id = stu\_id)

print(stu\_id,money,grade,name)

data = np.array([scholarship.score,gpa.score,paper.score,job.score,cheat.score,c.score,credit.score,penalty.score])

dataLenth = 8

labels = np.array(['奖学金','绩点','学生工作','校外工作','校内纪律','二手商品','技能','罚款情况'])

print(data)

angles = np.linspace(0, 2\*np.pi, dataLenth, endpoint=False)

data = np.concatenate((data, [data[0]]))

angles = np.concatenate((angles, [angles[0]]))

fig = plt.figure()

ax = fig.add\_subplot(111, polar=True)

ax.set\_rlim(0,200)

ax.plot(angles, data,'bo-', linewidth=2)

ax.fill(angles, data, facecolor='r', alpha=0.25)

ax.set\_thetagrids(angles \* 180/np.pi, labels, fontproperties="SimHei")

ax.set\_title("{}的雷达图".format(student.stu\_name), va='bottom', fontproperties="SimHei")

ax.grid(True)

plt.savefig('C:/Users/16437/Desktop/毕设/bishe/static/img/test1.jpg')

return render(request,'student/stu\_index.html',{'grade':grade,'money':money,'name':name,'scholarship':scholarship,'gpa':gpa,'paper':paper,'job':job,'cheat':cheat,'c':c,'credit':credit,'penalty':penalty})

def sorted\_rank(yourlist):

newlist = sorted(yourlist,key = lambda x:x[1],reverse =True)

for i in range(len(yourlist)):

newlist[i].append(i+1)

return newlist

def labor\_score(request):

if request.method == 'GET':

labors = models.LaborScore.objects.all()

return render(request,'instructor/labor\_score.html')

if request.method =='POST':

students = models.Student.objects.all()

score\_list = []

for student in students:

stu\_id = student.stu\_id

stu\_name = student.stu\_name

thisstu = models.Student.objects.get(stu\_id = stu\_id)

grade = test\_grade(int(thisstu.stu\_year))

stu\_scholarship = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_scholarship').score

stu\_gpa = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_gpa').score

stu\_paper = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_paper').score

stu\_job = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_job').score

stu\_cheat = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_cheat').score

stu\_c = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_c').score

stu\_credit = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_credit').score

stu\_penalty = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_penalty').score

stu\_score = 0.05\*grade\*50+0\*stu\_scholarship+0.1\*stu\_gpa+0.05\*stu\_paper+0.2\*stu\_job+0.2\*stu\_cheat+0\*stu\_c+0.2\*stu\_credit+0.2\*stu\_penalty

score\_list.append([stu\_id,stu\_name,stu\_score])

print(score\_list)

newlist = sorted(score\_list,key = lambda x:x[2],reverse = True)

for i in range(len(newlist)):

newlist[i].append(i+1)

add\_labor\_score(newlist)

return render(request,'instructor/labor\_score.html')

def labor\_index(request):

labors = models.LaborScore.objects.all()

return render(request,'labor\_index.html',{'labors':labors})

def add\_labor\_score(yourlists):

cleartable = models.LaborScore.objects.all()

cleartable.delete()

for yourlist in yourlists:

savedata = models.LaborScore(stu\_id = yourlist[0],stu\_name = yourlist[1],score = yourlist[2],rank = yourlist[3])

savedata.save()

def debt\_score(request):

if request.method == 'GET':

labors = models.DebtScore.objects.all()

return render(request,'instructor/debt\_score.html')

if request.method =='POST':

students = models.Student.objects.all()

score\_list = []

for student in students:

stu\_id = student.stu\_id

stu\_name = student.stu\_name

thisstu = models.Student.objects.get(stu\_id = stu\_id)

grade = test\_grade(int(thisstu.stu\_year))

stu\_scholarship = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_scholarship').score

stu\_gpa = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_gpa').score

stu\_paper = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_paper').score

stu\_job = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_job').score

stu\_cheat = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_cheat').score

stu\_c = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_c').score

stu\_credit = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_credit').score

stu\_penalty = models.SepScore.objects.get(stu\_id = stu\_id,score\_type = 'stu\_penalty').score

stu\_score = 0.1\*grade\*50+0.15\*stu\_scholarship+0\*stu\_gpa+0.1\*stu\_paper+0.1\*stu\_job+0.25\*stu\_cheat+0.05\*stu\_c+0\*stu\_credit+0.25\*stu\_penalty

score\_list.append([stu\_id,stu\_name,stu\_score])

print(score\_list)

newlist = sorted(score\_list,key = lambda x:x[2],reverse = True)

for i in range(len(newlist)):

newlist[i].append(i+1)

add\_debt\_score(newlist)

return render(request,'instructor/debt\_score.html')

def add\_debt\_score(yourlists):

cleartable = models.DebtScore.objects.all()

cleartable.delete()

for yourlist in yourlists:

savedata = models.DebtScore(stu\_id = yourlist[0],stu\_name = yourlist[1],score = yourlist[2],rank = yourlist[3])

savedata.save()

def debt\_index(request):

debts = models.DebtScore.objects.all()

return render(request,'debt\_index.html',{'debts':debts})