上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

本科毕业设计(论文)

FINAL PROJECT/THESIS OF UNDERGRADUATE

(2018 届)

学生选课管理信息系统设计与实现

Design and Implementation of Students Selection Course Management Information System

学　　院	光电信息与计算机工程学院
专　　业	计算机科学与技术
学生姓名	王启帆
学　　号	1412480521
指导教师	刘亚
完成日期	2018 年 05 月

承诺书

本人郑重承诺：所呈交的毕业论文“学生选课信息管理系统设计与实现”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

针对现在各高校使用的使用旧版本的 HTML 语言（当前最新 HTML5）开发的学生选课管理系统中学生选课和课程管理模块不够人性化和用户友好性不足，本次毕业设计结合当前流行的 Web 开发框架和更安全的 Web 开发机制与 ASP.NET 动态网页技术相结合，设计并实现一个新的选课管理信息系统，即对应教务管理系统中的选课和管理课程模块，主要用到了技术如 Ajax 网页异步刷新机制、Bootstrap 栅格系统布局等。该系统前端部分在实现过程中，使用到 ASP.NET 动态网页开发、Model-View-Controller 设计理念、Javascript 中的 Ajax 异步刷新；数据库的实现方面，使用了目前流行的小型关系型数据库 MySQL，由于它的开源特性，网络上提供丰富的嵌入式语言插件，使其能嵌套在后端语言 C#中实现数据库的访问。

用户分为三类，即管理员、教师和学生。其中教师和学生均需注册，系统会根据数据库中的账号进行判断，管理员无需审核每一次用户的加入。在课程时间不冲突的情况下，学生可以在课程列表页面直接选择课程到自己的课表；在个人课表界面，当天要上的课和本周还未完成的课程会高亮显示出来；对于教师而言，发布课程之后，如果选课人数少于 25 人，则教师可以考虑将该门课撤销；教师无论是发布课程还是撤销课程都需要管理员的审核，管理员会根据相关数据决定该课程是下架还是保持上线状态。

关键词：选课　ASP.NET　MVC　Ajax　Bootstrap　Javascript

ABSTRACT

In the circumstances that most universities nowadays still use the old-fashioned, slow-reacted course select system, it is very encouraging to develop a new kind of course select system. Therefore, I am very interested in doing this. The reason why I am going to do this is: firstly, the outdated system we are using right now are remarked badly, due to its reacting speed and its user-unfriendly user interface; secondly, the technology that the old system use is also not secure which uses cookie in every connection from client to server, and this won't happen if we use the new features in front-end HTML5, i.e. webstorage containing mainly sessionStorage and localStorage; thus not only releases the burden on the server, but also seems a lot more secure. Mainly connected to technologies like Ajax page asynchronous refresh and Bootstrap table system. The implementation of the frontend part is mainly based on technologies like: active server pages, model-view-controller design mode and asynchronous JavaScript and xml; as to the backend part, I preferred the open-sourced relational database, MySQL, its popularity made it possible to run under C# language to access the database tables.

The system is in development using the currently popular .NET framework combined with MySQL database, both of which are free shared in open source projects. The MVC, i.e. model-view-controller is also very welcome nowadays in web development, which is also integrated into the system. In addition, Bootstrap are new features that polishes the system, building robust and good-looking user interface, also high-performance data access. Moreover, the usage of Json to achieve data exchange between pages makes the data format is more simple and easier to read and write.

KEY WORDS: ASP.NET MVC Ajax Bootstrap Javascript

目　录

第 1 章　绪论

1.1 背景

由于现在高校多数采取选课制度，而经过了许多年，教务管理系统的发展并没有跟上互联网浪潮向前的脚步。过去的许多网站，因为历史遗留性，其渐渐变得只适合在 PC 端浏览，而对于现在非常流行的平板电脑和智能手机，支持还不够。而教务管理系统就是这样一种具有历史遗留性的网站，它包含了庞大的后台数据库，而前端页面在手机和平板电脑上难以适配，这使得大量数据在嵌入式操作系统的浏览器中显得非常冗余和杂乱，而在 Windows 操作系统上进行开发，使用当前最新的网站栅格系统布局，则在其他所有类型的设备上都可以进行访问，它会自适应屏幕的尺寸并对相关内容进行排版，然后再进行显示。此外，本学生选课管理信息系统还具有优化的前端，将最复杂的信息以最简单的形式展示出来，Glyphicon 和 Font-Awesome 前端插件使得展示大量信息达到新的水平，其界面美观自然，信息内容充分又不冗余，且一目了然。

与以往的教务管理系统相比，除了实现对教师发布的课程完成编排和学生选课的进行实现，它的功能更多地侧重了学生和教师的互动，学生和老师可以在访问个人主页之后互相评价[1]。

1.2 学生选课管理信息系统的目的和意义

学生选课管理信息系统是专为老师的课程编排和学生课表安排而设计的信息系统。其中包括了发布课程和管理课程功能，主要是为了解决教学中在某一时间段遇到的课程冲突或者教学地点冲突等问题。

网络技术作为教学辅助工具，可以充分有效的支持教学的各个环节，除了考虑系统的操作便捷性，界面友好性，还要注重学生间、师生间、学生和资源间的交互，强调教师教学活动设计功能的支持，提倡合作学习的理念。

本学生选课管理信息系统的意义，在于能够使用最新的前端技术完成一些旧技术无法带来的体验。好的体验可以提升用户使用系统的感受，也会提高用户对系统的使用效率[2]。

1.3 本论文的主要内容与结构

本论文研究并设计实现了基于 ASP.NET MVC 的学生选课管理信息系统。其前端使用了目前国内外最流行的开源框架——Bootstrap，界面美观，开发方便；后端应用了 ASP.NET MVC 架构，开发分层十分明确，同时便于维护；数据库使用了 Oracle 旗下最流行也是最稳定的小型关系型数据库 MySQL Database 5.1.7，既简单清晰，又高效安全；另外还结合了 jQuery，Ajax 等前端技术，实现了教师管理自己的课程（例如发布/修改课程）、学生选择课程、安排课表、管理员管理（如审核）课程、教师学生互相留言等功能。

第 1 章概要介绍了设计该选课管理信息系统的背景原因、设计目的以及现实意义。

第 2 章概要介绍了开发该系统所使用到的技术和开发工具，包括开发环境和运行环境等。

第 3 章对该系统的设计进行了简单的需求分析，由先整体后局部的分析方法，先对该系统整体进行需求分析，弄清楚一些最基本的需求（类别）；然后在对每个需求类别进行细致的需求分析。随后针对前面得到的结论进行数据库分析和可行性分析。将分析结果以其他形式展示出来例如 Excel 图标和流程图等等；使用这种图文结合的方式不仅可以使他人更便于理解该系统，同时也更利于开发者（自己）梳理思路，整理脉络，从而快速高效的进行系统的开发。

第 4 章是系统设计，包含了该系统的概要设计和详细设计，介绍了系统的基本结构、数据库设计、页面设计和类设计。通过图表介绍各个页面间的跳转以及相互关系和各个类的设计。

第 5 章是实现与测试，展示了系统实现的页面效果，介绍了后期单元测试的用例及结果。

第 6 章是总结，针对本次毕业设计做出总结以及反思毕业设计的不足以及后期可以改进的地方。

第 2 章　开发使用技术简介

2.1 ASP.NET 框架与 MVC 模式

1996 年，ASP 1.0（Active Server Pages）版本出现了，它引起了 Web 开发的新革命，降低了动态网页开发的难度。以前开发动态网页需要编写大量繁杂的 C 代码，编程效率非常低下，而且需要 Web 网页开发者掌握非常高的编程技巧。而 ASP 使用简单的脚本语言，能够将代码直接嵌入 HTML，使设计 Web 页面变得更简单。虽然 ASP 非常简单，但却能够实现非常强大的功能，这一切得益于其组件。特别是 ADO 组件，使得在网页中访问数据库易如反掌。这一切推动了动态网页的快速发展与建设，同时使 ASP 得到迅速流行。

2000 年 6 月，微软公司宣布了自己的.NET 框架。.NET 框架的基本思想是：把原有的重点从连接到互联网的单一网站或设备转移到计算机、设备和服务群组上，而将互联网本身作为新一代操作系统的基础。这样，用户将能够控制信息的传送方式、时间和内容，从而得到更多的服务。

2001 年，ASP.NET 浮出水面。它最初的名字为 ASP+，后来改为 ASP.NET。ASP.NET 是微软公司开发的一种建立在.NET 之上的 Web 运行环境，它不是 ASP 的简单升级，而是新一代的 Active Server Pages。ASP.NET 是微软公司新体系结构 Microsoft.NET 的一部分，其中全新的技术架构使编程变得更加简单。借助于 ASP.NET，可以创造出内容丰富的、动态的、个性化的 Web 站点。ASP.NET 简单易学、功能强大、应用灵活、扩展性好，可以使用任何.NET 兼容语言 [3]。

2.2 jQuery 与 Bootstrap

jQuery 是一个简洁快速的 JavaScript 框架。jQuery 设计的宗旨是“write Less，Do More”，使用写更少的代码完成更多的事情。它封装了 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、Ajax 交互等各种功能。

它的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的 CSS 选择器，并且可对 CSS 选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。

Bootstrap，来自 Twitter，是目前很受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JavaScript 的，它简洁灵活，使得 Web 开发更加快捷。它由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发，是一个 CSS/HTML 框架。Bootstrap 提供了优

雅的 HTML 和 CSS 规范，它即是由动态 CSS 语言 Less 写成。Bootstrap 一经推出后颇受欢迎，一直是 GitHub 上的热门开源项目，包括 NASA 的 MSNBC（微软全国广播公司）的 Breaking News 都使用了该项目。国内一些移动开发者较为熟悉的框架，如 WeX5 前端开源框架等，也是基于 Bootstrap 源码进行性能优化而来。

2.3 AJAX 与 JSON

AJAX 指的是一种异步 JavaScript 和 XML，它是一种新的网页开发技术，用于创建交互式网页应用。AJAX 是一种用于创建快速动态网页的技术。它通过在服务器与后台进行轻量级的数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

JSON 是一种轻量级的数据交换格式。它采用完全独立于编程语言的文本格式来存储和表示数据。因为其简洁和清晰的层次结构，它成为理想的数据交换语言。它有易于人阅读和编写、易于机器解析和生成以及有效地提升网络传输效率的优点。

2.4 开发工具

(1) MySQL Workbench 工具简介

MySQL Workbench 是一款专为 MySQL 设计的 ER/数据库建模工具。它是著名的数据库设计工具 DBDesigner4 的继任者。MySQL Workbench 可以用来设计和创建新的数据库图示，建立数据库文档，以及进行复杂的 MySQL 迁移。

(2) Visual Studio 2017 Community 工具简介

Visual Studio 2017 是微软于 2017 年 3 月 8 日正式推出的新版本，是迄今为止 最具生产力 的 Visual Studio 版本。其内建工具整合了 .NET Core、Azure 应用程序、微服务（microservices）、Docker 容器等所有内容。

Visual Studio（简称 VS）是美国微软公司的开发工具包系列产品。VS 是一个基本完整的开发工具集，它包括了整个软件生命周期中所需要的大部分工具，如 UML 工具、代码管控工具、集成开发环境(IDE)等等。所写的目标代码适用于微软支持的所有平台，包括 Microsoft Windows、Windows Mobile、Windows CE、.NET Framework、.NET Compact Framework 和 Microsoft Silverlight 及 Windows Phone。

第 3 章　需求分析

3.1 系统需求及设计思路

本系统需求如表 3.1 系统需求所示。

表 3.1 系统需求

系统名称	功能模块
学生选课管理信息系统	学生课程管理模块
	教师课程管理模块
	学生教师交互模块

本毕业设计目的为设计一个与教务管理系统中选课模块对应的学生选课管理信息系统，所涉及到的处理逻辑和处理方法皆来自于日常使用教务管理系统之后的心得和感想，以及进行总结之后得到的结论[4]。

3.2 系统需求分析

3.2.1 功能需求分析

本系统主要有以下用例（未全部列出，见用例说明）：

1. 用户登录/注册
2. 教师管理（发布/修改/查看）课程
3. 教师/学生互相查看信息
4. 学生管理（选择/收藏/查看）课程
5. 学生/教师相互留言
6. 教师/学生管理个人信息

用例说明：图 3.1 的用例中包含三个角色：教师、学生和管理员以及六个用例（如前所述）。这三者在学生选课管理信息系统中扮演的角色各不相同；学生用户作为该系统的主要使用者，主要能够通过系统完成选择课程、查看自动生成的课表、对已经上过的课程进行查看，以及查看教师列表、查看所有课程列表等；教师用户作为系统的第二使用者，可以通过该系统进行课程发布、课表查看、已教过学生查看；管理员

则可以通过该系统进行学生管理、教师管理和课程管理；其中管理员是该系统的最高权限者，可以对系统的一些参数进行修改以维护系统的正常运行[5]。

用例图总图如下，见图 3.1：

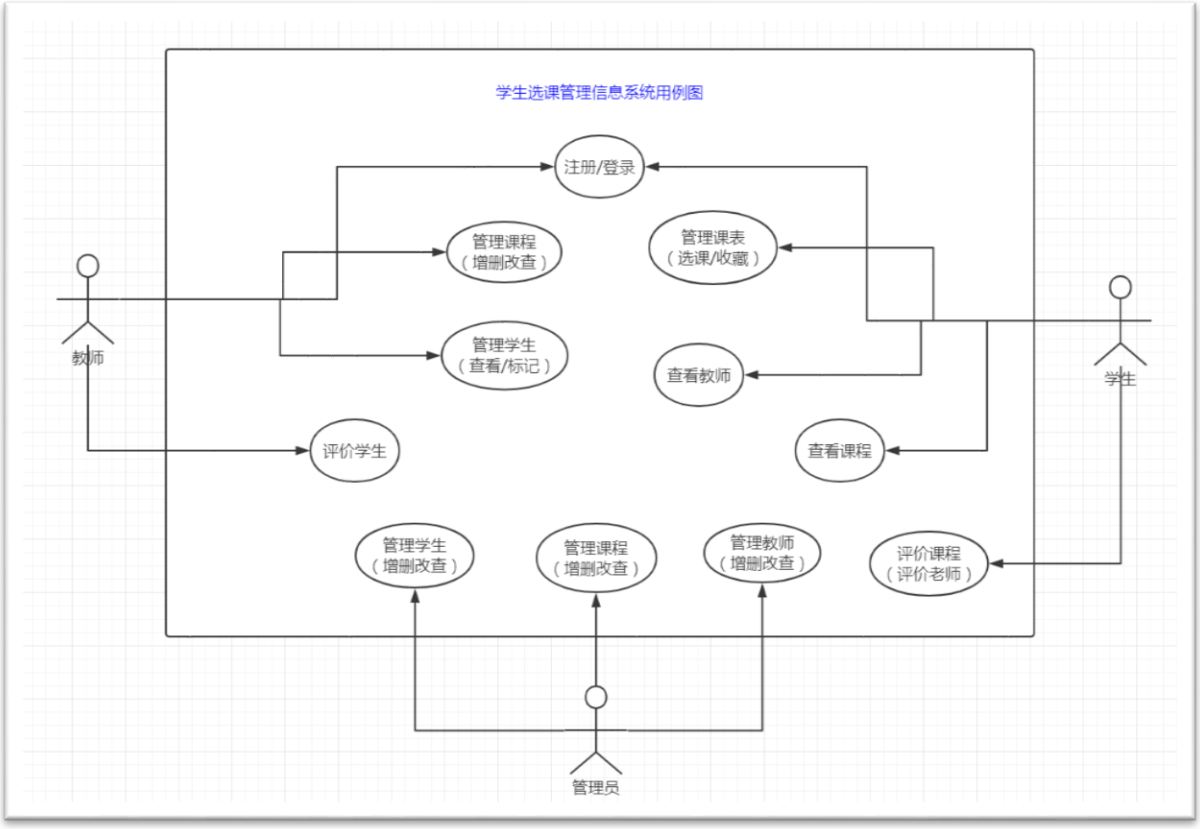

图 3.1 学生选课管理信息系统用例图-总图

详细用例说明如下：

(1) 用户登录/注册用例

用例说明：教师/学生分别根据学号/工号进行系统注册/登录；并根据其身份在系统内进行相关的操作。

用例名：用户登录/注册。

主要参与者：教师/学生。

概述：教师/学生分别根据学号/工号进行系统注册/登录；并根据其身份在系统内进行相关的操作。

前置条件：登录：该用户存在且合法；注册：无重复用户名和密码。

基本路径：

1) 当用户进入该系统并准备登录/注册时，用例开始；

2) 用户输入正确用户名及密码；

3) 用户成功登录并显示相应权限界面，用例结束。否则进行第（4）步。

4) 用户根据输入的用户名和密码进行该身份的注册。

可选路径：

1) 在选择提交前，用户都可以选择取消。若登录取消，用例结束；

2) 在基本路径第 II 步，如果用户名或密码有误，系统提示用户用户名或密码有误，请重新输入用户名和密码。

后置条件：如果登录成功，跳转到教师/学生对应的个人主页界面；不成功停留在登录界面并给出相应错误提示。

(2) 教师管理（发布/修改/查看）课程用例

用例说明：教师增/删/改/查课程（并交由管理员审核），此处以“增”为用例进行说明。

用例名：管理课程（增删改查）。

主要参与者：教师。

概述：教师增/删/改/查课程（并交由管理员审核），此处以“增”为用例进行说明。

前置条件：一个存在且合法的用户——教师已经正确登录系统。

基本路径：

1) 当用户进入该系统并点击新增课程时，用例开始；

2) 用户输入课程相关信息，具体见数据库设计中的课程类设计。课程 ID，教师 ID 将分别由系统生成和获取；系统对输入进行判断无误之后，点击提交；

3) 用户成功提交课程，用例结束。

可选路径：

1) 在选择提交前的任何时候，用户都可以选择取消。这次新增取消，用例结束。

2) 在基本路径第 II 步，如果课程相关信息填写不符合规范，系统提示用户检查并重新填写。如果一直存在不合法内容，则只能取消或停留在当前页面。

后置条件：如果提交成功则更新数据库，否则数据库将不做更新。

(3) 管理学生（查看/标记）用例

用例说明：教师对于自己曾教过的学生可以进行查看访问个人主页。

用例名：管理学生（查看/标记）。

主要参与者：教师。

概述：教师对于自己曾教过的学生可以进行查看访问个人主页。

前置条件：一个存在且合法的用户（教师）已经正确登录系统。

基本路径：

1) 当用户（教师）进入该系统并查看自己的学生时，用例开始；

2) 用户（教师）点击列表中某一个自己的学生，进入该学生的个人主页，查看

该学生的相关信息；

3) 教师在个人主页可以选择留言，或者返回学生列表；

4) 留言成功/返回，用例结束。

可选路径：

1) 在选择提交前的任何时候，用户都可以选择取消。取消，则用例结束。

后置条件：如果上传成功跳转主界面，不成功则停留在当前界面

(4) 管理学生（选择/收藏/查看）课程用例

用例说明：对系统内的非管理员用户进行管理（增删改查）。

用例名：管理学生（增删改查）。

主要参与者：管理员。

概述：对系统内的非管理员用户进行管理（增删改查）。

前置条件：管理员已经登陆系统（合法）。

基本路径：

1) 当管理员进入系统时，用例开始；

2) 管理员点击管理学生标签页；

3) 选择特定学生用户，进行特定信息的管理；

4) 变更相关信息成功，用例结束。

可选路径：

1) 在选择提交前的任何时候，用户都可以选择取消。这次修改取消，用例结束。

后置条件：如果修改成功则更新数据库，否则数据库不做更新。

(5) 管理课程（增删改查）用例

用例说明：对所有教师发布的课程进行管理（增删改查）。

用例名：管理课程（增删改查）。

主要参与者：管理员。

概述：对所有教师发布的课程进行管理（增删改查）。

前置条件：管理员已经登陆系统（合法）。

基本路径

1) 当管理员进入系统时，用例开始；

2) 管理员点击管理课程标签页；

3) 选择特定课程，进行特定信息的管理；

4) 变更相关信息成功，用例结束。

可选路径

1) 在选择提交前的任何时候，用户都可以选择取消。这次修改取消，用例结束。

后置条件：如果修改成功则更新数据库，否则数据库不做更新。

(6) 管理教师（增删改查）用例

用例说明：管理员管理教师（增删改查）。

用例名：管理教师（增删改查）。

主要参与者：管理员。

概述：管理员管理教师（增删改查）。

前置条件：管理员已经登陆系统（合法）。

基本路径

1) 当管理员进入系统时，用例开始；
2) 管理员点击管理教师标签页；
3) 选择特定教师，进行特定信息的管理；
4) 变更相关信息成功，用例结束。

可选路径

1) 在选择提交前的任何时候，用户都可以选择取消。这次修改取消，用例结束。

后置条件：如果修改成功则更新数据库，否则数据库不做更新。

(7) 评价课程（评价老师）用例

用例说明：学生对曾经教过自己的老师进行评价（针对课程的评价）。

用例名：评价课程（评价老师）。

主要参与者：学生用户。

概述：学生对曾经教过自己的老师进行评价（针对课程的评价）。

前置条件：一个存在且合法的学生用户已经登录系统。

基本路径

1) 当学生进入该系统并点击我的课程后，点击已完成课程时，用例开始；
2) 学生可以针对这门课提交一定数量文字的评价；
3) 用户点击提交，用例结束；

可选路径

1) 在选择提交前的任何时候，用户都可以选择取消。这次修改取消，用例结束。

后置条件：如果修改成功则更新数据库，否则数据库不做更新。

(8) 查看课程用例

用例说明：学生/教师针对特定课程，进行详细信息的查看。

用例名：查看课程。

主要参与者：学生/教师。

概述：学生/教师针对特定课程，进行详细信息的查看。

前置条件：一个存在且合法的用户已经登录系统。

基本路径

1) 当用户进入该系统，点击我的课程或者查看所有可选课程时，用例开始；
2) 用户查看该门课程的详细信息；
3) 点击返回，用例结束；

无可选路径

(9) 查看教师用例

用例说明：学生选课之前可以查看该门课的任课教师信息。

用例名：查看教师修改用户信息。

主要参与者：学生用户。

概述：学生选课之前可以查看该门课的任课教师信息。

前置条件：一个存在且合法的学生用户已经正确登录系统。

基本路径

1) 当用户进入该系统并点击下学期选课后，点击选择某门课程时，用例开始；
2) 用户查看教师的详细信息以及之前的评价；
3) 用户点击返回，用例结束；

无可选路径

(10) 管理课程（选课/收藏）用例

用例说明：学生选择下一学期将要进行的课程。

用例名：管理课程（选课/收藏）。

主要参与者：学生。

概述：学生选择下一学期将要进行的课程。

前置条件：一个存在且合法的学生用户已经正确登录系统。

基本路径

1) 当学生进入下学期选课标签页并点击选择/收藏相关课程时，用例开始；
2) 学生可以选课/收藏/退选课/取消收藏课程；
3) 用户点击提交，用例结束；

可选路径

1) 在选择提交前的任何时候，用户都可以选择取消。这次修改取消，用例结束。

后置条件：如果修改成功则更新数据库，否则数据库不做更新。

3.2.3 数据库需求分析

本系统采用的是 Oracle MySQL 5.1.7 Database 数据库。学生选课管理信息系统中主要的实体有学生、教师、管理员、课程、考试、教室、教学楼、学科门类、课表等。其中，学生可以登录系统，查看自己的课表、添加新的课程到自己的课表、收藏课程、评价已经修读过的课程等；教师可以登录系统，查看自己的课表、发布课程、查看历史开课记录等；课程具有如课程名、课程类别、课程归属教师、课程上课时间/地点、课程相关考试等；教室这个实体类的存在主要是为了方便系统的使用——将新的教室以及教学楼添加到数据库，这方便了实时获取最新信息；学科门类也是为了方便系统的使用，将课程按照学科门类标准进行分类；课表主要包含了选课人、授课人、选课时间以及选课类型（选课/收藏）等属性。

3.2.4 性能需求分析

本系统开发软件采用 Visual Studio 2017 作为开发工具；后台配合 MySQL 5.1.7 作为存储数据的数据库开发工具；并于 Windows 10 操作系统下进行开发。但是本系统支持 Windows7、8、10 等平台。其硬件需求为 512M 以上内存；至少 80G 以上的硬盘；32 倍速以上的 CD-ROM；10MB/100MB 自适应的网络适配器；并使用 TCP/IP 协议的局域网。

3.3 可行性分析

本毕业设设计使用 ASP.NET MVC 作为开发框架，HTML+Javascript+C#作为主要的开发语言实现了学生选课管理信息系统。该系统主要有三类用户——即学生、教师和管理员；这三者之间使用系统时具有不同的权限，管理员针对所有用户都具有查看其相关信息的权限，而管理员之外用户只能查看并管理自己的个人信息；对于教师而言，本系统主要提供了一个方便发布课程的功能，以及查看自己当前发布过的课程和上课时间地点；对于学生而言，使用本系统方便了进行课程选择和收藏，学生可以一目了然的查看所有开课的教师以及起开设的所有课程。除了这三种用户，本系统不涉及到与其他人有关的业务处理逻辑。

该系统采用 ASP.NET MVC 作为框架结合前端技术和 C#嵌入式语言实现开发，并配合 Oracle MySQL Database 存储数据，在软件开发平台这一条件上已经基本成熟可行。硬件方面根据现在普遍的计算机配置都可满足。由上述因素综合分析，本系统在技术上是可行的。

本系统采用的开发软件、使用的框架都是开源免费的，开发成本较低，没有特别的额外支出。另外本系统采用 MVC 架构，分层明确，便于维护，后期的测试维护更新不会占用和浪费过多资源，由上述因素综合分析，本系统在经济上是可行的。

本系统适用于学校组织，根据权限功能的不同划分了不同的角色，每种角色能够

完成自己能力范围内的功能。另外，本系统操作简单，界面人性化，对用户没有特别的技术要求，易于掌握，由上述因素综合分析，本系统在操作上是可行的。经可用，或预计将会推出[6]。

第 4 章　系统设计

4.1 系统结构图

图 4.1 结构图

本学生选课管理信息系统的逻辑结构如图 4.1 所示（详图可以查看链接：https://www.processon.com/view/link/5afa8fece4b07510694e0a36；链接地址见附件）。该系统的主要处理逻辑分为三个模块：学生选课模块、教师发布课程模块和管理员管理课程模块。这三个角色同时使用该系统，根据其权限的不同将在系统内完成各种不同的需求。首先教师和学生这两种角色都具有登录（已注册）和注册（新加入）的功能；教师可以在系统中完成查看自己当前周课表的功能、发布新课程、修改个人信息等功能；学生用户可以在系统中完成查看当前课表和所有课程的功能，并且可以选课、收藏课程；管理员则可以进行对学生、教师和未发布课程的管理。除以上功能之外，学生可以对所选课程进行管理，比如退选、收藏、取消收藏等功能；教师可以对发布的课程进行查看，如果教师发布的课程未审核通过，则无法出现在学生选课界面。管理员可以对学生信息进行管理、也可以对教师信息进行管理；除此之外还可以审核课程是否可以加入学生的培养计划和选课列表。其他一些细节的功能如查看当前教学周、查看课程评论等也是本系统的功能[7]。

4.2 数据库设计

下图展示了最简单的用户之间的数据联系：

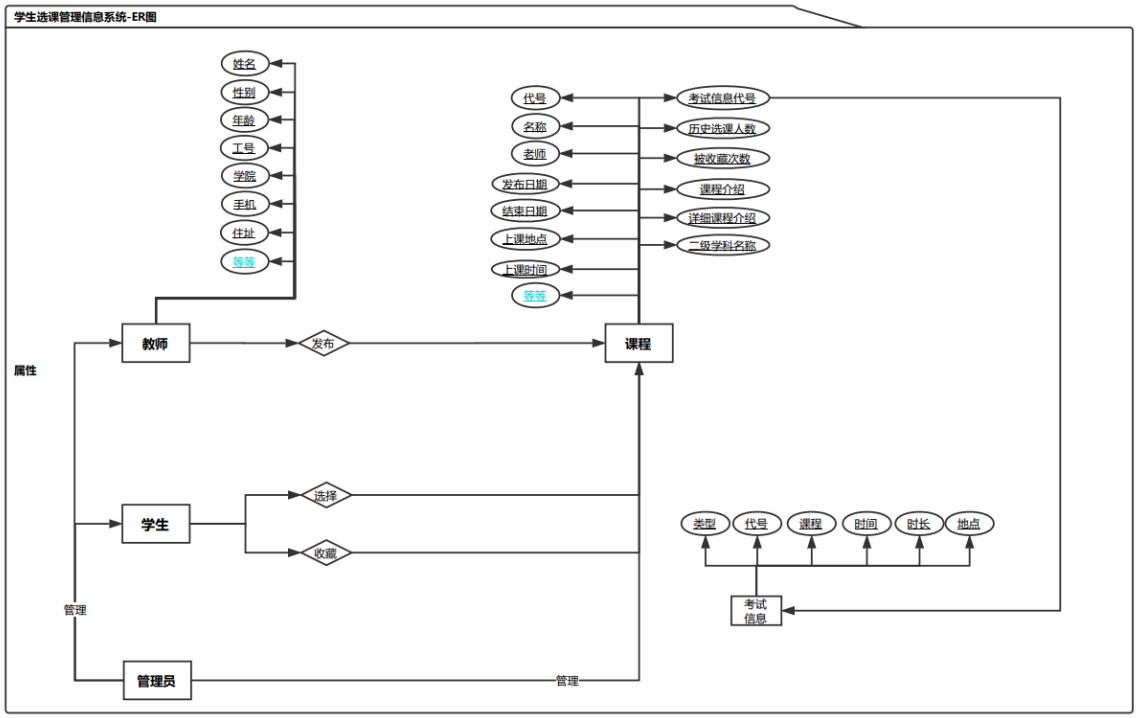

图 4.2 系统 ER 图

图 4.2 展示了最简单的教师和学生之间的联系——从数据库的角度。关于他们的数据表远不止这些，可以在第四章系统概要设计中的第二小节：数据库设计中找到更多信息。

表 4.1 登录信息表

字段名称	类型	说明
uniqueClientID	唯一编号	char(10)
username	用户名	varchar(50)
password	密码	varchar(50)
registryDate	注册日期	datetime
registryType	注册类型	char(1)

表 4.1 为登录信息表，其中第一个字段为该表的主键，为用户的唯一编号；该表中存储了系统中所有用户的登录信息（用户名、密码、注册日期、注册类型（学生或

教师))。特别地，管理员的登录信息也在该表中，但不是由注册而来，而是由 DBA 直接加入表中。

表 4.2 教师信息表

字段名称	类型	说明
uniqueClientID	唯一编号	char(10)
name	姓名	varchar(25)
gender	性别（‘M’/‘F’/‘U’）	char(1)
age	年龄	smallint
email	邮箱	varchar(50)
wechat	微信	varchar(25)
qqNumber	QQ 号码	varchar(25)
phone	手机号码	varchar(25)
university	学校	varchar(100)
college	学院	varchar(100)
major	专业	varchar(100)
courses	所授课程	varchar(125)
workNo	教工号	varchar(20)
workPass	教工密码	varchar(20)

表 4.2 为教师信息表，其中第一个字段为该表的主键，是教师用户的唯一编号，表 4.1 中的 uniqueClientID 字段是它的外键；剩下的字段则包括了教师用户的基本个人信息（姓名、性别、年龄、邮箱、微信、QQ 号码、手机号码、学校、学院、专业、所授课程、教工号、教工密码）。其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

表 4.3 学生信息表

字段名称	类型	说明
uniqueClientID	唯一编号	char(10)
name	姓名	varchar(30)

gender	性别（‘M’/‘F’/‘U’）	char(1)
age	年龄	smallint
email	邮箱	varchar(50)
wechat	微信	varchar(50)
qqNumber	QQ 号码	varchar(12)
phone	手机号码	varchar(30)
university	学校	varchar(100)
college	学院	varchar(100)
major	专业	varchar(100)
grade	年级	varchar(10)
cardNo	学生学号	varchar(20)
cardPass	学号密码	varchar(20)

表 4.3 为学生信息表，其中第一个字段为该表的主键，是学生用户的唯一编号，表 4.1 中的 uniqueClientID 字段是它的外键；剩下的字段则包括了学生用户的基本个人信息（姓名、性别、年龄、邮箱、微信、QQ 号码、手机号码、学校、学院、专业、年级、学生学号、学号密码）其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

表 4.4 课程信息表

字段名称	类型	说明
courseID	课程代号	char(10)
courseName	课程名称	varchar(100)
courseTeacher	代课老师	char(10)
publishDate	发布日期	date
startDate	开始日期	date
endDate	结束日期	date
venue	上课地点（特定西文字符/汉字）	varchar(60)
period	上课时间	varchar(50)
examDate	考试时间	datetime

examDuration	考试时长	digit(2)
examVenue	考试地点	varchar(35)
examID	考试信息代号	char(10)
chosen	历史选课人数	int
collected	被收藏次数	int
courseInfo	课程介绍	varchar(255)
detailInfo	详细课程介绍	varchar(2048)
subTag	二级学科名称	varchar(50)

表 4.4 为课程信息表，其中第一个字段为该表的主键，为课程的唯一编号；其他字段则包括了每门课程的基本信息（课程名称、代课老师、发布日期、开始日期、结束日期、上课地点、上课时间、考试时间、考试时长、考试地点、考试信息代号、历史选课人数、被收藏次数、课程介绍、详细课程介绍、二级学科名称）。其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

表 4.5 考试信息表

字段名称	类型	说明
examID	考试代号	char(10)
courseID	课程代号	char(10)
examDate	考试时间	datetime
examDuration	考试时长	digit(2)
examVenue	考试地点	varchar(60)
examType	考试类型	varchar(10)
examTaken	考试人数	int
examRate	通过率	varchar(10)
ava	是否结束	char

表 4.5 为考试信息表，其中第一个字段为该表的主键，为考试代号（唯一），它是课程信息表中 examID 字段的外键；其他字段包括了每门考试基本信息（课程代号、考试时间、考试时长、考试地点、考试类型、考试人数、通过率、是否结束）。其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

表 4.6 教室信息表

字段名称	类型	说明
roomID	教室代号	char(10)
roomName	教室名称	varchar(60)
roomTag	教室别称	varchar(35)
roomBuilding	所属建筑	varchar(35)
roomCapacity	可容人数	Int

表 4.6 为教室信息表，其中第一个字段为该表的主键，为教室的唯一代号，它是课程信息表中 venue 字段、考试信息表中 examVenue 字段和教室使用表 roomID 字段的外键；其他字段则包括了教室的基本信息（教室名称、教室别称、所属建筑、可容人数）。其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

表 4.7 教室使用表

字段名称	类型	说明
roomID	教室代号	char(10)
weekday	占用工作日	varchar(10)
period	占用时段	char(2)
type	占用类型	char(1)
occupier	占用人	char(10)
dayperiod	（结合字段）	char(2)

表 4.7 为教室使用表，其中第一个字段为该表的主键、它记录了每个教室在特定时间的使用对象。其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

表 4.8 学科门类表

字段名称	类型	说明
name	学科门类	varchar(20)
code	学科代号	char(2)

表 4.8 为学科门类表，这是一个由管理员管理直接进行管理的信息表。其中第二个字段为它的主键。

表 4.9 评论信息表

字段名称	类型	说明
remarkid	评论唯一标识 ID	varchar(10)
obj	评论课程 ID	varchar(10)
auth	评论者作者 ID	varchar(10)
time	发表评论时间	Datetime
remark	评论内容	varchar(2048)

表 4.9 为评论信息表，其中第一个字段为该表的主键，是每一条评论的唯一标识。其他字段包括了评论的基本信息（评论课程 ID、评论者作者 ID、发表评论时间、评论内容）。其中一些字段以特定的数据格式存储在数据库中，这些格式的说明请见附录 B。

4.3 实体类设计

（1）用户类设计

表 4.10 用户类

class User
public string UniqueClientID { get; set; } public string Username { get; set; } public string Password { get; set; } public string RegistryDate { get; set; } public char RegistryType { get; set; }

//构造函数：初始化字段
public User
{
}

表 4.10 是用户类，该类将会作为教师类和学生类的父类被它们继承。它具有用户最基本的信息，包括了唯一编号、用户名（登录名）、登录密码、注册日期和注册类型（学生还是教师）。这个类将会在该项目的后台中（即服务器端）扮演重要的角色，对于每一个用户，无论是教师学生，还是管理员，其个人信息在运行的时候都会被作为 Session[“S_user”]存在于服务器端。所以说这个类是非常重要的。

表 4.11 教师类（继承）用户类

class Teacher
public string UniqueClientID { get; set; }
public string Name { get; set; }
public char Gender { get; set; }
public int Age { get; set; }
public string Email { get; set; }
public string Wechat { get; set; }
public string QQ { get; set; }
public string Phone { get; set; }
public string University { get; set; }
public string College { get; set; }
public string Major { get; set; }
public char Position { get; set; }
public string Courses { get; set; }
public string WorkNo { get; set; }
public string WorkPass { get; set; }
//构造函数：初始化字段
public Teacher
{
}

表 4.11 是教师类，其继承自用户类这个基本类，该类只作为教师基本信息以及详细信息的实体类，系统在运行使所有有关教师的数据全部都被存放在 ViewData[“Teacher”]中。该实体类包含的教师信息有：教师姓名、教师性别、教师年

龄、电子邮箱、微信号、QQ 号、手机号、毕业高校、毕业学院、所学专业、学位、所授课程、工号、密码。

表 4.12 学生类（继承）用户类

class Student
public string UniqueClientID { get; set; } public string Name { get; set; } public char Gender { get; set; } public int Age { get; set; } public string Email { get; set; } public string Wechat { get; set; } public string QQ { get; set; } public string Phone { get; set; } public string University { get; set; } public string College { get; set; } public string Major { get; set; } public char Position { get; set; } public string Grade { get; set; } public string CardNo { get; set; } public string CardPass { get; set; }

//构造函数：初始化字段
public Student
{
}

表 4.12 是学生类，其继承自用户类这个基本类，该类只作为学生基本信息以及详细信息的实体类，系统在运行使所有有关教师的数据全部都被存放在 ViewData[“Student”]中。该实体类包含的教师信息有：学生姓名、学生性别、学生年龄、电子邮箱、微信号、QQ 号、手机号、本科高校（如果是研究生）、学院、专业、学位、当前年纪、学号、密码。

（2）课程类设计

表 4.13 课程类

class Course
public string CourseType { get; set; }
public string CourseID { get; set; }
public string CourseName { get; set; }
public string CourseTeacher { get; set; }
public string PublishDate { get; set; }
public string StartDate { get; set; }
public string EndDate { get; set; }
public string Venue { get; set; }
public string Period { get; set; }
public string ExamID { get; set; }
public int Chosen { get; set; }
public int Collected { get; set; }
public string CourseInfo { get; set; }
public string DetailInfo { get; set; }
public string SubTag { get; set; }
//构造函数：初始化字段
public Course()
{
}

表 4.13 是课程类，该类是本系统中最重要的类之一，存储了学生上课的课程信息和教师发布课程的课程信息，它包含了课程的基本和详细信息（课程门类、课程唯一编号、课程名称、授课教师、发布日期、开始日期、结束日期、上课地点、上课时间、考试信息编号、选课人数、收藏人数、课程基本介绍、课程详细介绍、课程二级学科标签）。该类中的 ExamID 字段外键于考试类的主键字段。

（3）课表类设计

表 4.14 课表类

class CourseTable
public string UniqueClientID{ get; set; } public string CourseID { get; set; } public string ManiType{ get; set; } public string ManipDate{ get; set; } public string IfDeleted{ get; set; }

//构造函数：初始化字段
public CourseTable()
{
}

表 4.14 是课表类，包含一个学生上课的基本信息（学生唯一编号、课程唯一编号、学生与课程关系（选课、收藏）、产生关系时间、是否有效）。而这些信息只是索引，具体的学生和课程的详细信息要到其对应的主表中查询。

（4）教室类设计

表 4.15 教室类

class Room
public string RoomID { get; set; } public string RoomName { get; set; } public string RoomTag { get; set; } public string RoomBuilding { get; set; } public int RoomCapacity { get; set; }

//构造函数：初始化字段
public Room()
{
}

表 4.15 是教室类，该类的设计是为了方便该系统的正常运行，它包含了教室唯一编号、教室名称、教室代号、教室所属建筑、教室容量这些信息。

（5）考试类设计

表 4.16 考试类

class Exam
public string ExamID { get; set; } public string CourseID { get; set; } public string ExamDate { get; set; } public string ExamDuration { get; set; } public string ExamVenue { get; set; } public string ExamType { get; set; }

//构造函数：初始化字段
public Exam()
{
}

表 4.16 是考试类，该类中的主键即第一个字段，是另一个实体类课程类的考试信息代号的外键。该类是课程类的衍生，因为课程对应着一门考试或者多门考试，所以这里设计了一个考试实体类。

（6）学科类设计

表 4.17 是学科类，该类的设计是为了方便该系统的正常运行，首先每门课都应该对应一个学科门类，而学科门类表是一个不经常改变的表，所以这个实体类是一个静态类，基本不会改变。该类中的所有数据均是直接从数据库中直接获取的。

表 4.17 学科类（枚举）

enum Subject	
//**Code**	**Name**
//"CA">	哲学
//"CB">	经济学
//"CC">	法学
//"CD">	教育学
//"CE">	文学
//"CF">	历史学
//"CG">	理学
//"CH">	工学
//"CI">	农学
//"CJ">	医学
//"CK">	军事学
//"CL">	管理学
//"CM">	艺术学

（7）教室占用类设计

表 4.18 教室占用类

class RoomUse
public string weekday { get; set; }
public string period { get; set; }
public string type { get; set; }
public string occupier { get; set; }
public string dayperiod { get; set; }
public string roomID { get; set; }

```
//构造函数：初始化字段
public RoomUse
{
}
```

表 4.18 是教室占用类，学生选课不仅仅要和课程及教师产生联系，也会和教室及教室所在建筑产生联系，该实体类存储的是教师和教室之间产生的关系的数据，包括了上课工作日 、上课具体时间、上课类型（授课还是考试）、占用教师以及占用教室唯一编号。

4.4 页面设计

（1）用户登录/注册页面

图 4.3 是登录/注册功能的时序图，完成顺序可以是：1）用户（包括教师/学生/管理员）可以选择登录，如果登录信息正确无误，则会返回正确的结果——即跳转到对应用户的个人主页。如果登录信息有误，则直接停留在该页面，返回一个错误消息；2）用户（包括教师/学生/管理员）可以选择注册，如果经过验证注册信息无误，则会返回正确值并跳转到用户主页继续完成一些个人信息的填写；如果注册信息有误，则会停留在该页面，直接返回一个错误消息。

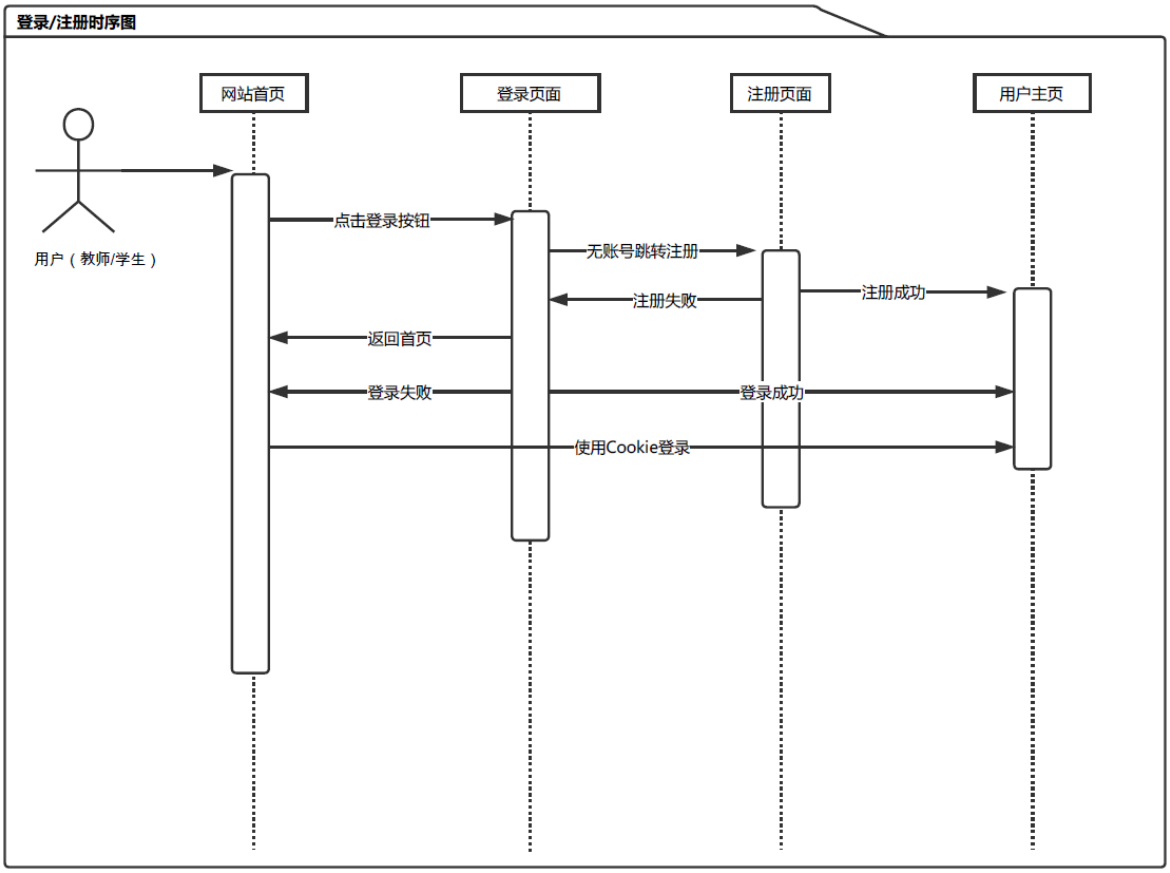

图 4.3 用户登录/注册顺序图

（2）管理员管理教师/学生

图 4.4 是管理员管理教师/学生的时序图，完成顺序可以是：管理员进入其个人主页，可以查看到四个标签页，分别对应着管理学生/管理教师/管理课程和管理其他；当管理员选择后面两个标签页的时候，请见本小节 5.1.3 管理员管理课程；如果管理员选择前面两个标签页，则可以对教师/学生的个人信息进行增删改查操作；当管理员做出变更时，客户端会自行验证管理员是否作出的是合法的变更，如果不合法点击提交则会直接从客户端发出相应提示，告知变更和修改不合法。如果合法，管理员点击提交之后数据会从客户端直接以 HTTP Post 方式提交到服务器，如果服务器端对数据库进行操作返回了正确的值，则从服务器传回客户端的布尔变量值为真值，否则为假；客户端根据接收到的值来告知管理员其变更操作是否完成。如果完成则管理员本次操作完成，否则需要重新向服务器发起请求，过程则与上面描述一致。

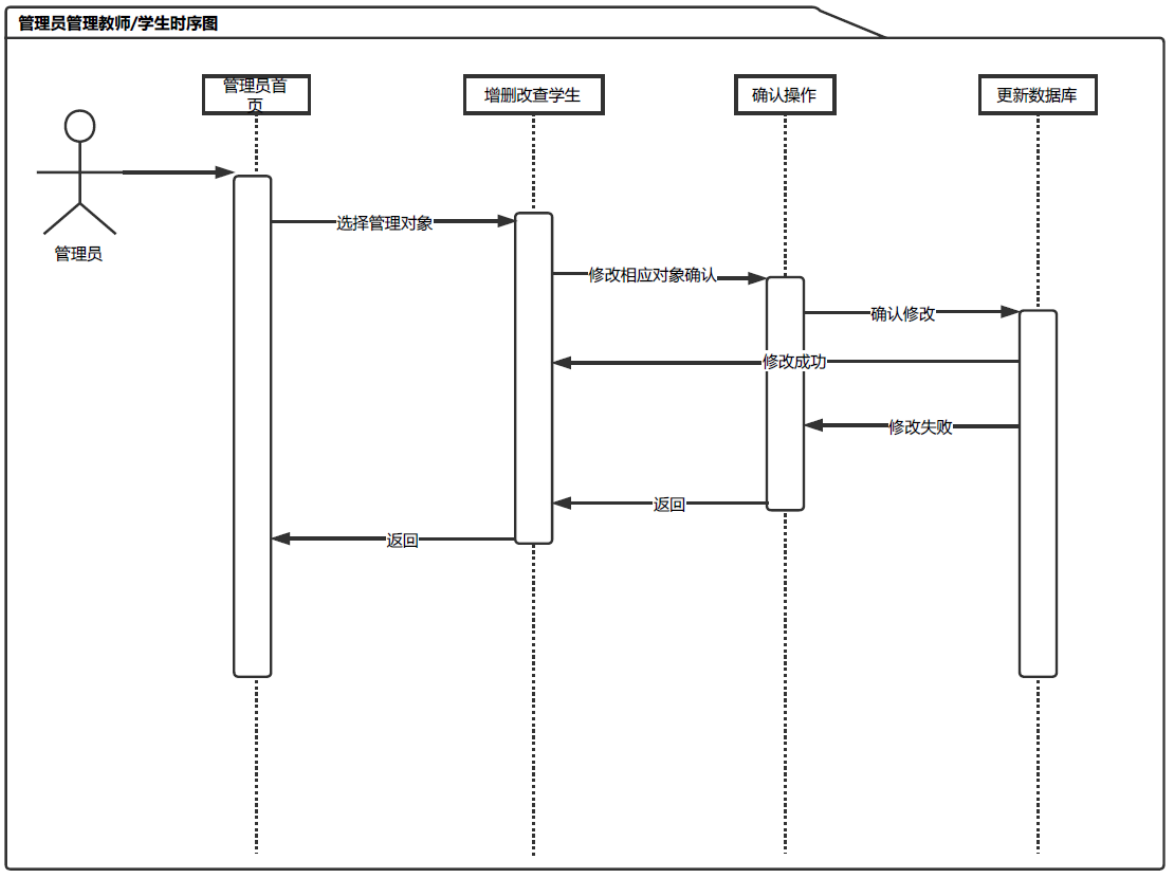

图 4.4 管理员管理教师/学生顺序图

（3）管理员管理课程

图 4.5 是管理员管理课程时序图，完成顺序可以是：管理员首先已经进行了正确的登录，否则系统不会给其该权限；管理员进入其管理主页之后，点击相应的标签页（第三个）——管理课程，则可以对课程进行增删改查操作，当管理员做出变更时，客户端会自行验证管理员是否作出的是合法的变更，如果不合法点击提交则会直接从客户端发出相应提示，告知变更和修改不合法。如果合法，管理员点击提交之后数据会从客户端直接以 HTTP Post 方式提交到服务器，如果服务器端对数据库进行操作返回了正确的值，则从服务器传回客户端的布尔变量值为真值，否则为假；客户端根据接收到的值来告知管理员其变更操作是否完成。如果完成则管理员本次操作完成，否则需要重新向服务器发起请求，过程则与上面描述一致。

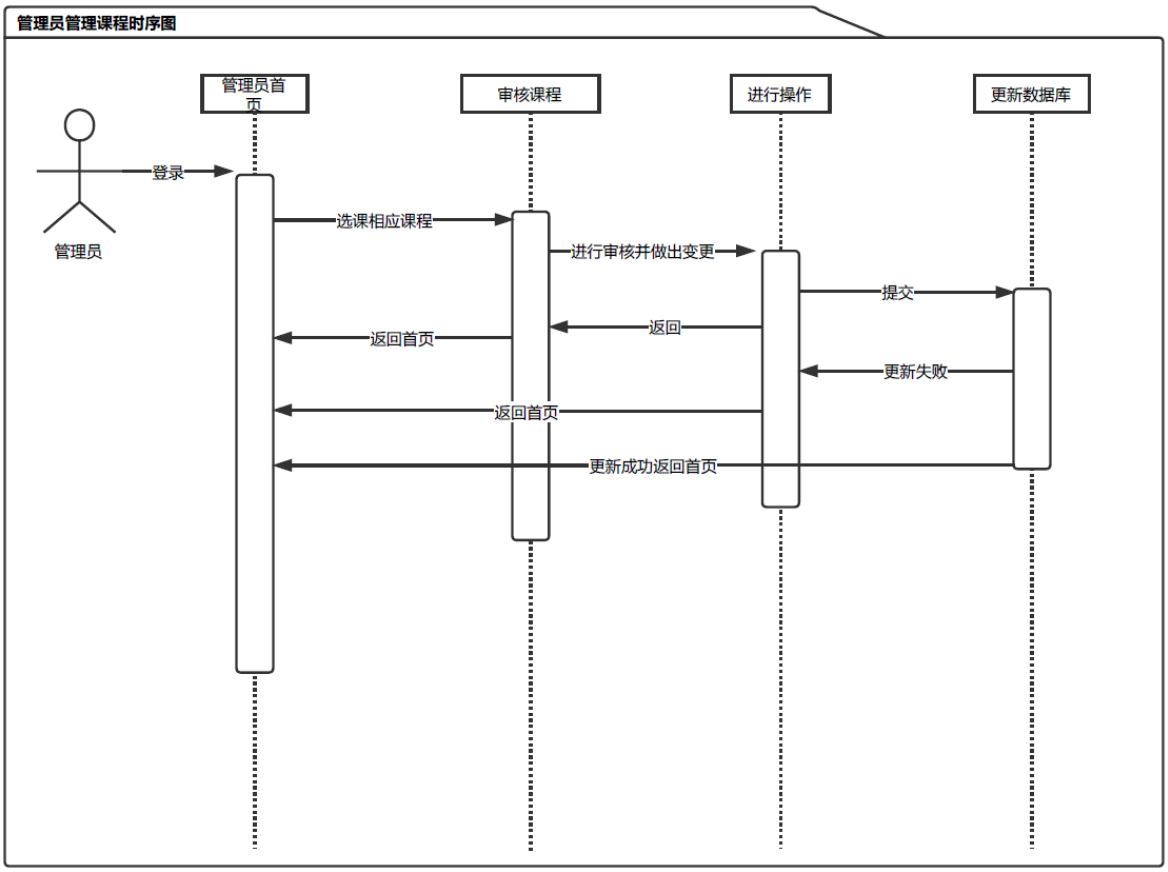

图 4.5 管理员管理课程顺序图

（4）教师管理课程

图 4.6 是教师管理课程时序图，完成顺序可以是：教师首先已经进行了正确的登录，否则系统不会给其该权限，当教师进入其个人主页的时候会看到自己的本周个人课表。除此之外，教师可以对自己的所有课程进行查看，并决定是否发布课程。如果教师决定发布课程，那就要点击发布课程的相关标签，当管理员做出变更时，客户端会自行验证管理员是否作出的是合法的变更，如果不合法点击提交则会直接从客户端发出相应提示，告知变更和修改不合法。如果合法，管理员点击提交之后数据会从客户端直接以 HTTP Post 方式提交到服务器，如果服务器端对数据库进行操作返回了正确的值，则从服务器传回客户端的布尔变量值为真值，否则为假；客户端根据接收到的值来告知管理员其变更操作是否完成。如果完成则管理员本次操作完成，否则需要重新向服务器发起请求，过程则与上面描述一致。

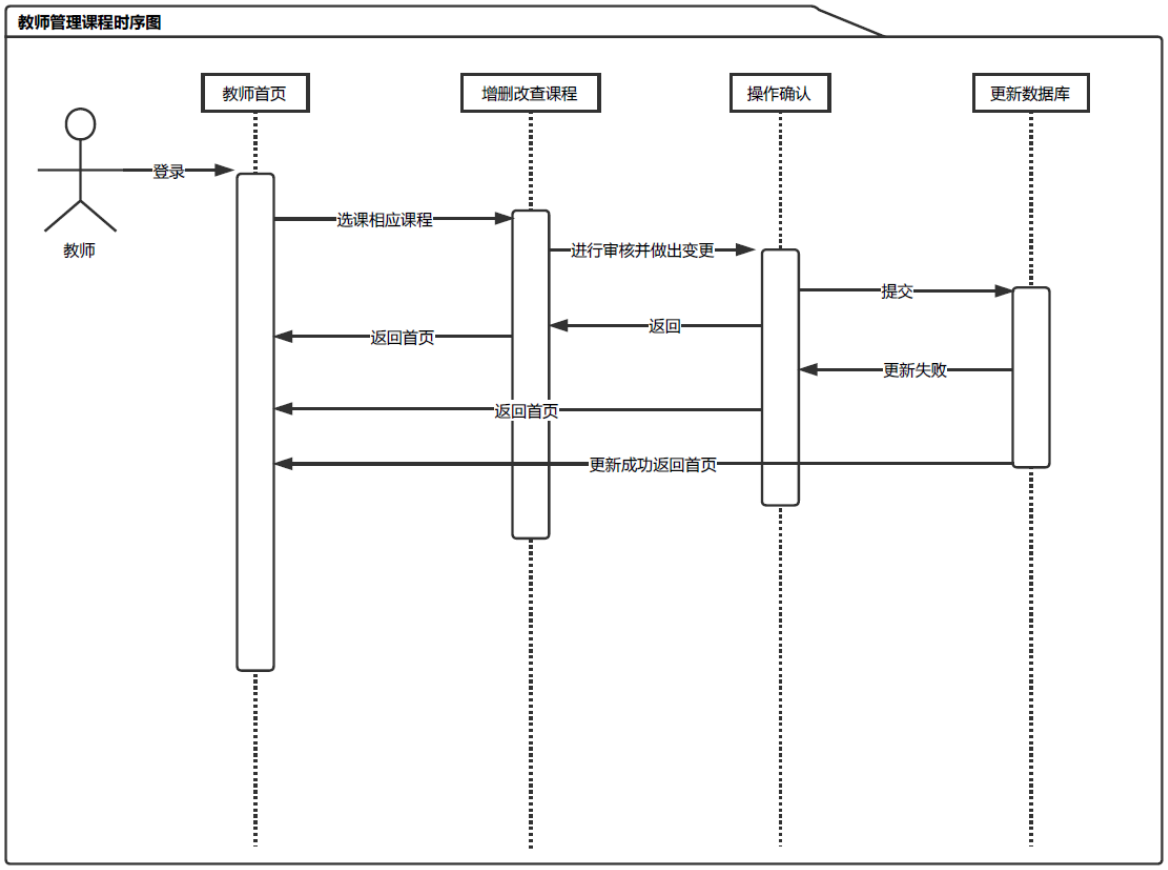

图 4.6 教师管理课程顺序图

（5）教师/学生相互评论

图 4.7 是教师/学生相互评论的时序图，完成顺序可以是：教师可以到学生的个人主页进行查看，如果该学生是该教师曾经教过的，则该教师可以对该学生进行评论；对学生来说亦是如此，如果学生访问教师的个人主页，发现该教师曾经教授过自己课程，则可以对该教师进行评价。当教师/学生做出评价时，客户端会自行验证管理员是否作出的是合法的提交，如果不合法则会直接从客户端发出相应提示，告知变更和修改不合法。如果合法，管理员提交之后数据会从客户端直接以 HTTP Post 方式提交到服务器，如果服务器端对数据库进行操作返回了正确的值，则从服务器传回客户端的布尔变量值为真值，否则为假；客户端根据接收到的值来告知管理员其变更操作是否完成。如果完成则管理员本次操作完成，否则需要重新向服务器发起请求，过程则与上面描述一致。

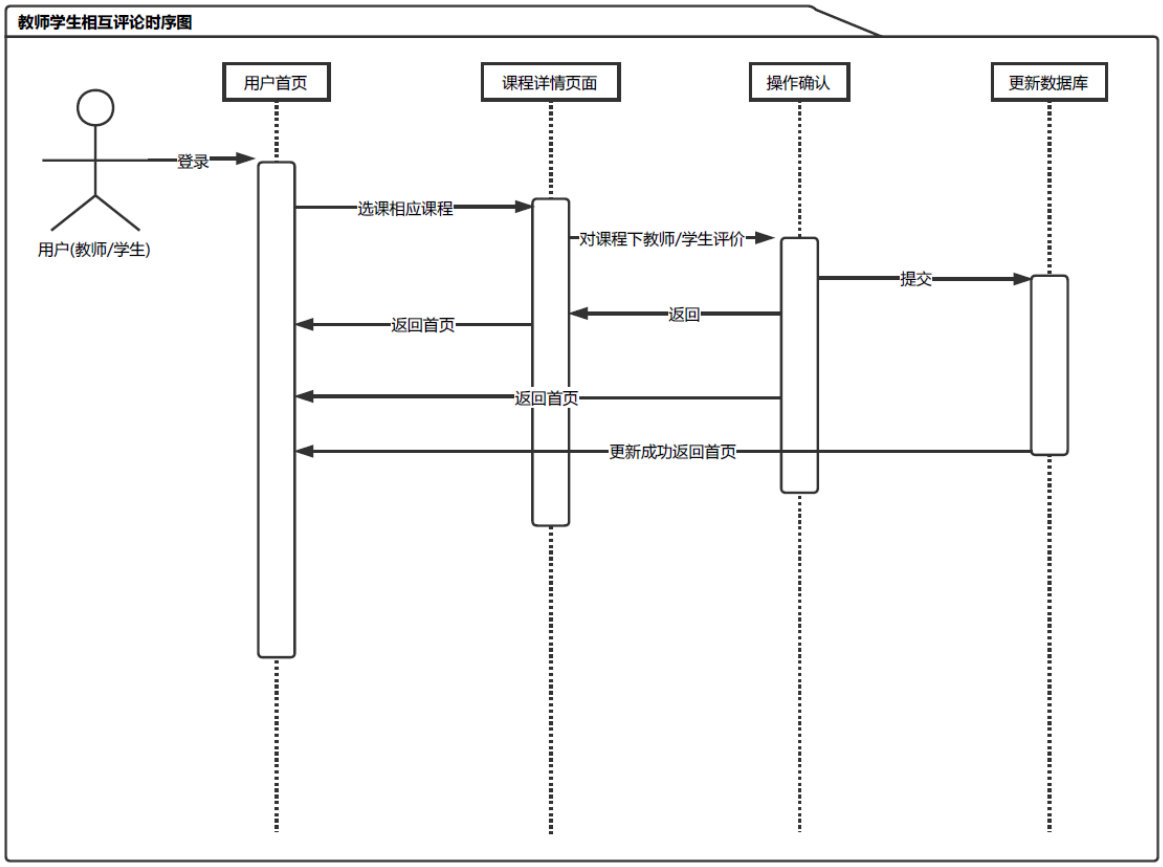

图 4.7 教师/学生相互评论顺序图

（6）教师管理学生

图 4.8 是教师管理学生时序图，完成顺序可以是：教师用户首先已经进行了正确的登录，否则系统不会给其该权限；教师进入其主页之后，可以查看自己当前有哪些学生，也可以查看自己教过的所有学生。教师能够查看到所有上过自己课程的学生。如果教师不查看，其实系统会自动显示出来。

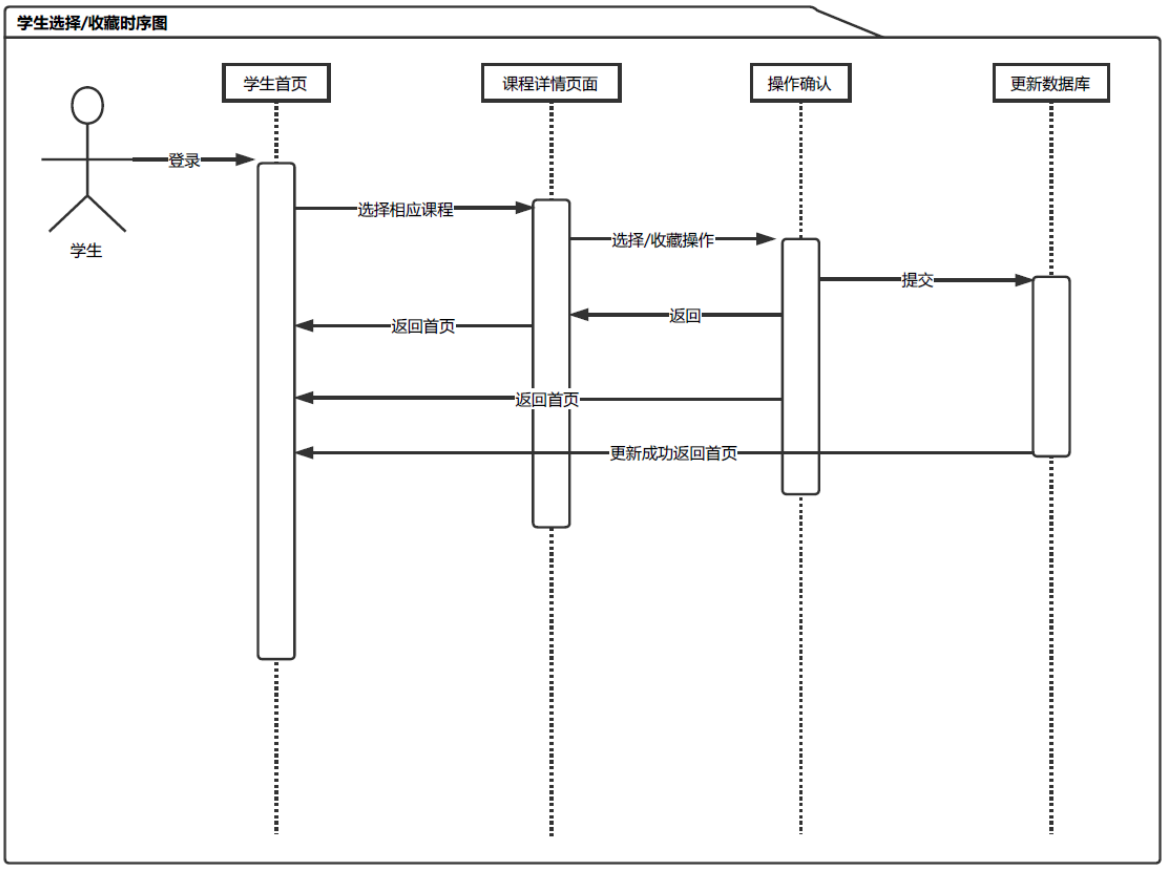

图 4.8 教师管理学生顺序图

（7）学生选择/收藏课程

图 4.9 是学生选择/收藏课程时序图，完成顺序可以是：学生用户首先已经进行了正确的登录，否则系统不会给其该权限；学生用户进入其管理主页之后，如果之前自己已经选择过课程，则会发现首页显示出了自己的课表，否则课表中的内容应该为空。当学生还没有选择过课程/已经选择了某些课程但还想再选时，可以选择查看当前所有可选课程列表，进入列表之后可以搜索自己想上的课程，如果找到了，则可以点击进入课程详情页面。进入该页面之后，学生用户可以对课程技能型选择和收藏操作。收藏课程不会把课程加入该学生的课表，而是放入起收藏列表。如果用户点击了选择该门课，则这门课会进入其个人课表。当学生不想选这门课或不再想收藏这门课程的时候，同样地，可以进入该课程详细信息界面，进行取消选择和取消收藏。当学生用户作出了变更的时候，系统会在后台跟服务器进行交互，实时地对数据库进行操作。

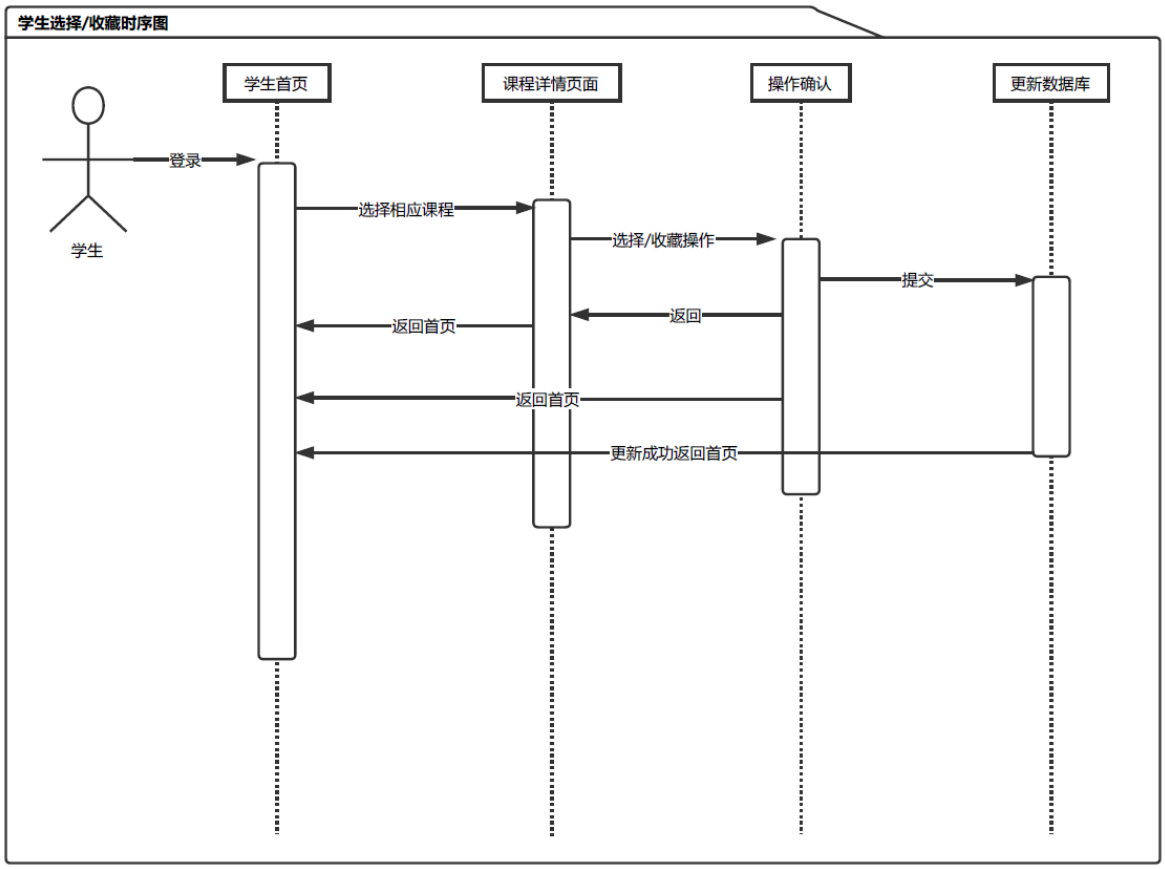

图 4.9 学生选择/收藏课程顺序图

（8）教师/学生修改个人信息

图 4.10 是教师/学生修改个人信息的时序图，完成顺序可以是：教师用户/学生用户首先已经正确地登录了该系统，否则不会有修改这些信息的权限。教师/学生点击进入个人中心，会发现可以变更自己的个人信息，当教师/学生变更其个人信息之后，点击提交的时候，客户端会自行验证教师/学生是否作出的是合法的提交，如果不合法则会直接从客户端发出相应提示，告知变更和修改不合法。如果合法，管理员提交之后数据会从客户端直接以 HTTP Post 方式提交到服务器，如果服务器端对数据库进行操作返回了正确的值，则从服务器传回客户端的布尔变量值为真值，否则为假；客户端根据接收到的值来告知管理员其变更操作是否完成。如果完成则管理员本次操作完成，否则需要重新向服务器发起请求，过程则与上面描述一致。

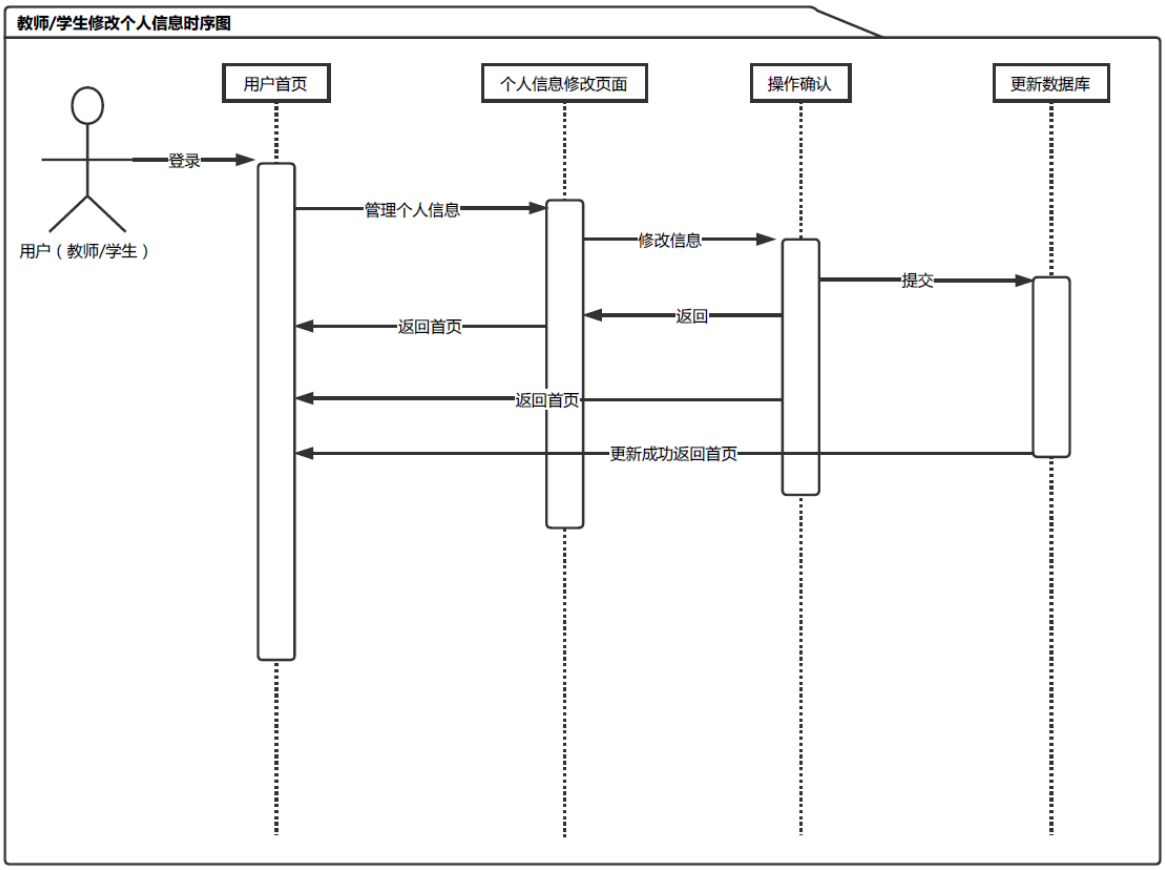

图 4.10 教师/学生修改个人信息顺序图

（9）教师/学生私信互动

图 4.11 是教师/学生私信互动时序图，完成顺序可以是：教师用户/学生用户首先已经正确地登录了该系统，否则不能完成向对方发送消息功能。，当学生或教师点击彼此的个人主页并选择留言（即发送消息），客户端会自行验证管理员是否发送的是合法的信息，如果不合法则会直接从客户端发出相应提示，告知信息不合法。如果合法，管理员提交之后数据会从客户端直接以 HTTP Post 方式提交到服务器，如果服务器端对数据库进行操作返回了正确的值，则从服务器传回客户端的布尔变量值为真值，否则为假；客户端根据接收到的值来告知管理员其变更操作是否完成。如果完成则管理员本次操作完成，否则需要重新向服务器发起请求，过程则与上面描述一致。

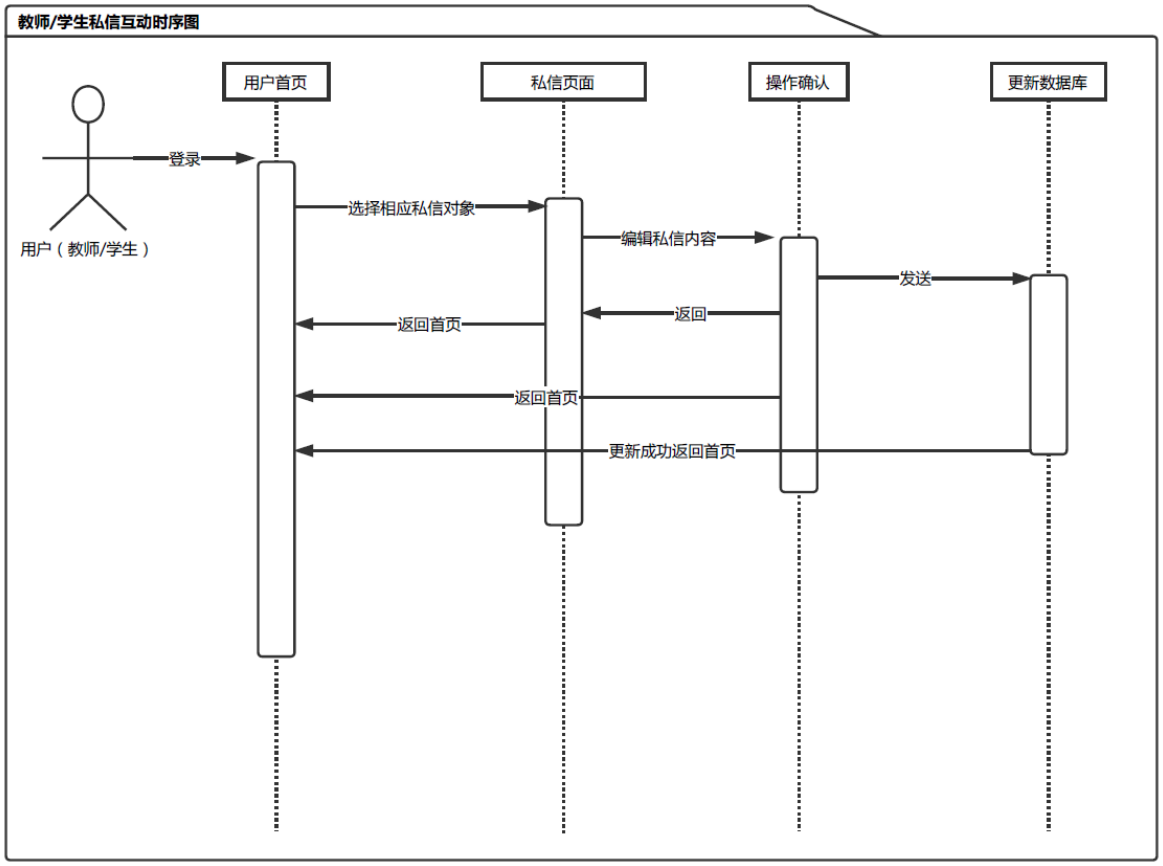

图 4.11 教师/学生私信互动顺序图

第 5 章　软件的实现与测试

5.1 软件界面实现

（1）用户登录/注册界面

图 5.1 为网站的首页，只有一个用户登录界面，用户在这里点击绿色按钮（如图 5.1 所示）将会跳转到用户登录界面（如图 5.2 所示）；如果没有用户名则可以进行注册点击登陆界面下方的注册按钮，即可跳转到注册界面，（如图 5.3 所示）；无论是学生还是老师或是管理员，都会通过这个界面，输入正确的用户名密码登录到学生选课管理信息系统。如果用户进入该网站只是停留在该页面，则不会进入任何其他界面[8]。

图 5.1 首页界面

图 5.2 是所有用户的登录界面，无论是学生、管理员或者教师都会通过这个界面进行登录。当然如果用户之前登录的时候记住了密码，则进入该页面之后会直接跳转到其对应的个人主页。

图 5.2 登录界面

图 5.3 是所有用户（管理员除外）的注册界面，还未该系统拥有账号的学生或者教师必须通过该页面进行注册之后才能进入本系统。该界面的输入会被客户端自行验证，如果哦没有非法字符则会停滞在该界面或者用户主动点击返回主页。

图 5.3 注册界面

注册界面进行注册之后，会跳转到输入个人信息页面，如图 5.4，验证无误之后就提交信息跳转个人主页。注意提交信息这里会有许多和身份相符的信息不能直接填入汉字而要根据系统的提示选择对应的标签之后才能正确的提交数据，否则插入新数据的时候会报错。

图 5.4 输入个人信息界面

（2）用户首页展示界面

图 5.5 为教师登陆成功之后立刻跳转的页面——即教师的个人主页界面。教师可以在该页面中的三个标签页中自由切换（无需跳转），根据每个标签页提供的功能完成相关操作。三个标签页分别为：首页-个人中心和我的消息。其中我的消息标签页包含与学生有关的消息和与课程有关的消息。该页面提供了获取当前教学周的小插件，位于欢迎文字正下方。首页展示的信息为教师当前周的课表，以及本学期开设的所有课程。如果教师需要查看所有自己开设过的课程，则需到要个人中心里我的课程中进行筛选。图 5.5 中绿色高亮标出的文字显示为当前工作日（或非工作日）以及当前需要（即未完成）上的课程。（图见下页）

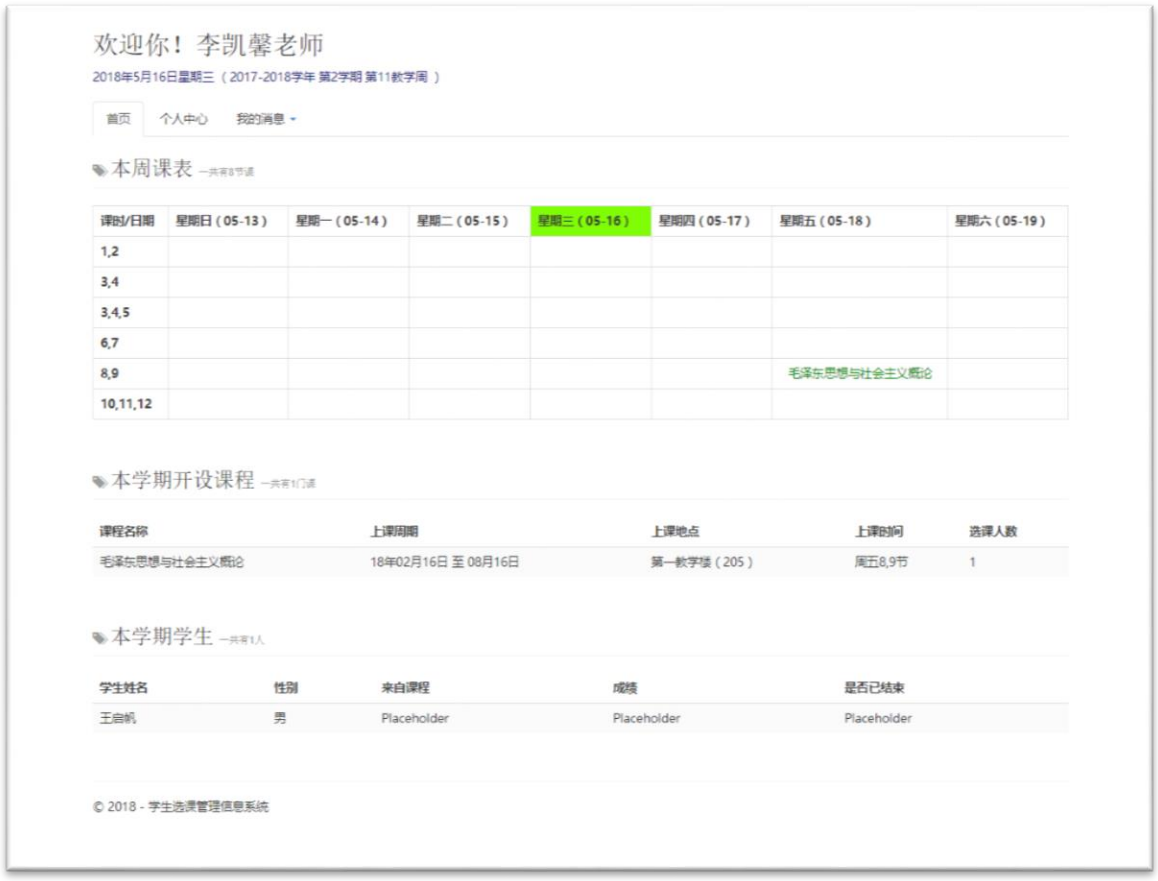

图 5.5 教师主页界面

图 5.5 中包含的信息相对较多，但是条理也很清晰，教师用户可以看到一目了然的信息。

教师点击第二个标签页，即个人中心，这里有教室这些年所教过学生的基本数据。教师在个人中心里选择发布新课程即可开始添加自己名下的课程。如图 5.6 所示。（图见下页）

（3）教师管理课程信息界面

图 5.7 是教师查看自己所开设课程的详细信息。在此处可以看到自己开设过的课程，以及教过的学生。这些都在我开设的课程和我教过的学生标签下。（图见下页）

图 5.6 新增课程信息界面

图 5.7 课程列表信息界面

（4）用户管理课表信息界面

图 5.8 为学生用户登录之后跳转的主界面，在此可以看到自己本周的课程表信息，也可以查看所有课程，然后进行选课或者收藏课程。当然选课要到个人中心标签页下的查看下学期选课按钮，点击之后可以进入下学期可选课程列表，然后就可以进行选课了，当然系统会进行时间冲突判断，如果时间冲突了，系统会高亮显示同一时段中后加入课表的课程时间，这两门课都会出现在课表中，由学生用户决定自己想上哪门课，如果觉得冲突，则学生可以退选掉其中一门课。（图见下页）

图 5.8 学生管理（查看）课表界面

（5）查看当前所有课程界面

图 5.9 为查看本学期所有课程的选课列表，可以看到上课时间这一列红色部分即与自己现有课程时间发生冲突的课程但这个时候学生用户依然可以决定自己想选哪一门，选任何一门或者都选——由自己决定。（图见下页）

图 5.9 查看当前所有课程信息界面

（6）收藏/选择/评论课程界面

图 5.10 为进入某一门课程的详细介绍界面。该界面中主要展示了该门课程的课程容量，并且和已经有多少人选择了这门课 。除此之外，学生用户可以决定将该门课加入课表或者移出课表（如果已经选择了的话），又或者是收藏该门课或者取消收藏该门课。（图见下页）

（7）查看所有教师/学生信息界面

图 5.11 为管理员用户查看所有用户（管理员自己除外）的界面。该界面中包含了所有用户的基本信息，数据来自 User 实体类（见详细设计中的实体类设计），若想查看某一个用户的详细信息，则点击该用户的用户名（高亮显示出），就可以跳转到这个用户详细信息的界面（如图 5.13 所示）。用户的个人信息详细界面的数据由 Student 实体类或 Teacher 实体类提供。（图见下页）

图 5.10 课程详细信息界面（选择/收藏）

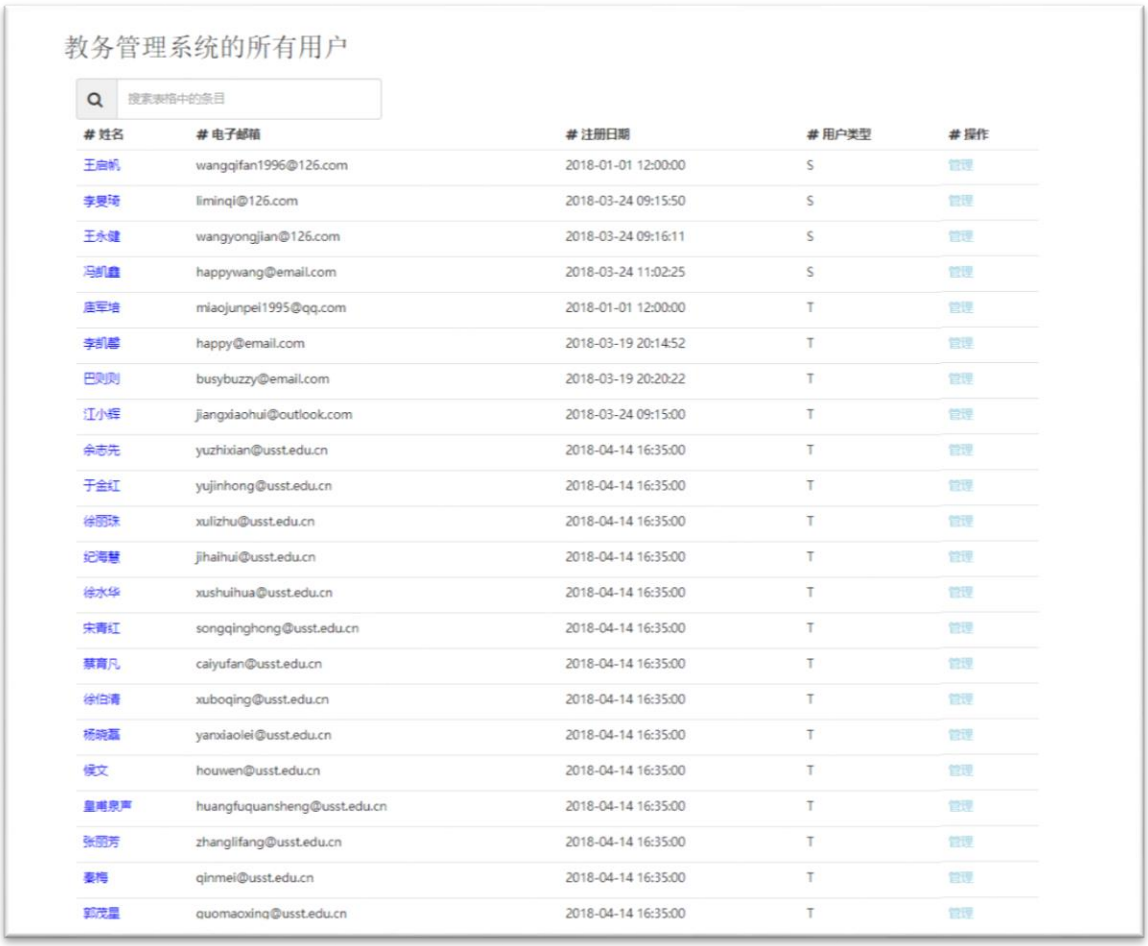

图 5.11 所有用户信息界面

（8）查看指定教师/学生信息界面

图 5.12 为学生或者教师详细信息界面。

图 5.12 指定用户详细信息界面

（9）网站每日一句界面

图 5.13 为该系统的额外的一个人性化的功能，可以让用户学到许多单词，或者看到漂亮的图片。

图 5.13 网站每日一句界面

（10）教师/学生我的消息界面

图 5.14 为教师/学生的我的消息界面，这里可以看到学生对教师或者教师对学生的评价（即消息）。

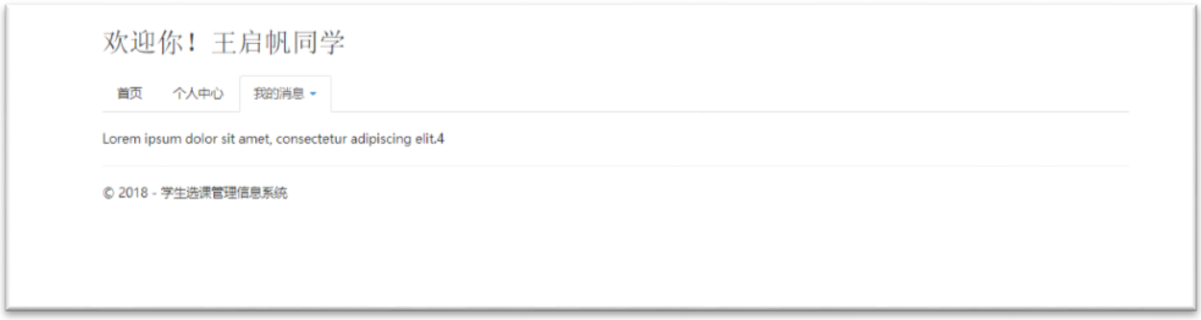

图 5.14 教师学生我的消息界面

（11）网站联系方式页面

图 5.15 为该系统的所属人和联系方式，其中还人性化的提供了图片和百度地图地址插件。（图见下页）

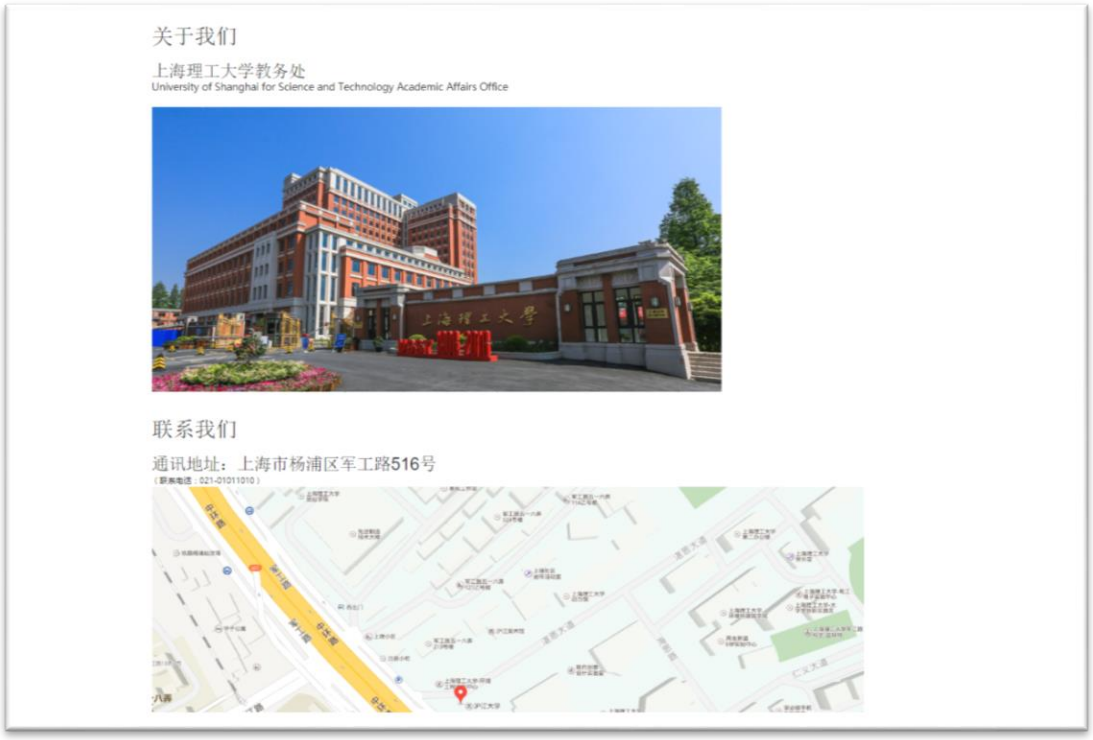

图 5.15 网站联系方式界面

（12）管理员管理课程界面

图 5.16 为 管理员管理课程界面，其中有四个标签页，可以对学生用户、教师用户、课程信息（是否发布课程）、其他系统信息（例如教学楼、教室、学科门类分类）进行管理。

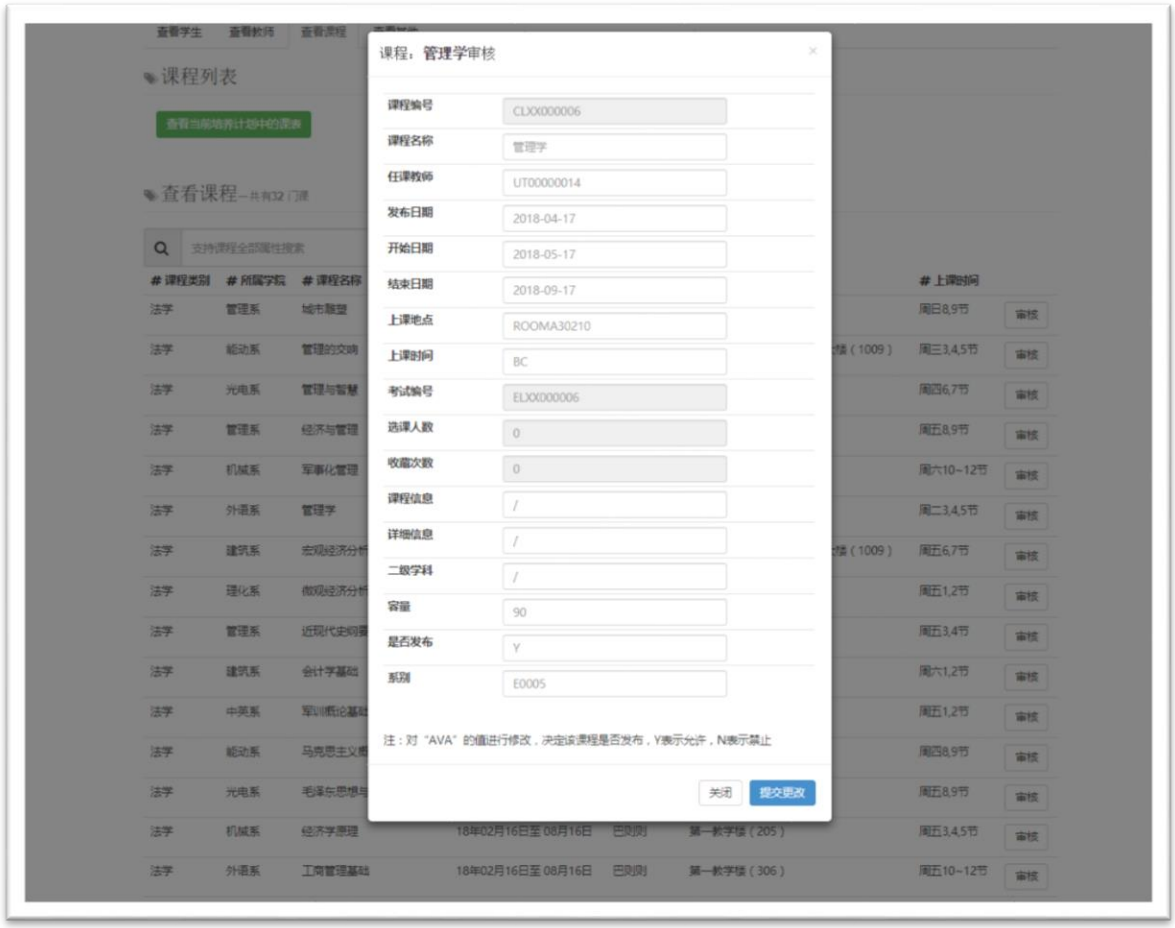

图 5.16 管理员管理课程界面

5.2 测试环境和测试工具

(1) 操作系统：windows2008 及以上（测试主机：window10）；

(2) CPU：1ghz；

(3) Ram：512；

(4) 网络畅通；

5.3 测试用例

本次系统测试主要采用单元测试。单元测试通常用于集中监测软件设计的模块，即其最小单元。一般来说，编码和单元测试属于软件过程中同一个阶段。程序员在编写出源程序并且通过基础语法检查之后，就可以参考系统详细设计的描述，对重要的执行通路进行单元测试，以便及时发现模块内部的错误并加以改正。单元测试可以应用计算机检测和人工检测两种不同类型的测试方法，两种不同的测试方法各有所长，互相补充。大多数单元测试主要使用白盒测试技术，可以并行的对多个模块的进行测试。

(1) 用户登录/注册

表 5.1 用户登录/注册测试用例

功能描述	用户登录/注册	
用例目的	测试合法用户是否可以正确登录/注册	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通。登录：已经正确注册过了账号；注册：还未注册账号。	
输入/动作	期望的输出/相应	实际情况
用户输入正确用户名及密码	成功跳转相应界面	成功跳转相应界面
用户输入错误的用户名或密码	提示用户名或密码错误	网页显示输入用户名或密码不存在或不正确

(2) 添加课程信息

表 5.2 教师添加课程信息测试用例

功能描述	教师添加课程信息	
用例目的	测试教师是否可以正常的添加课程信息	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，管理员用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况
教师用户点击添加课程，并且正确填写所有必要信息。	提示提交成功	网页提示提交成功，并且正确更新数据库。
用户点击添加课程，但是部分必要信息为空。	提示必填信息为空	网页提示必填信息为空，数据库没有更新。

用户点击添加课程，但是考试 ID 重复。	提示考试 ID 重复	网页考试 ID 重复,数据库没有更新。

(3) 教师查看个人课表

表 5.3 教师查看个人课表测试用例

功能描述	教师查看个人课表	
用例目的	测试教师是否可以在主页看到自己的课表的正确显示信息	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，教师用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况
登陆系统，点击主页	主页显示出了当前课表	如期望所预测，显示出了个人课表

(4) 学生查看个人课表

表 5.4 学生查看个人课表测试用例

功能描述	学生查看个人课表	
用例目的	测试学生是否可以正确查看自己的课表	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，教师或学生用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况
作为学生用户登录系统，成功登录之后，点击主页页面	主页已经正确地显示出了学生的个人课表	如预期所想，课表被正确显示

登录系统后，退选所有的课程，点击主页页面	主页没有显示课表，或者课表中没有课程	课表中没有课程
作为新注册用户，点击主页页面	主页没有显示课表，或者课表中没有课程	主页没有显示课表

(5) 选择（收藏）课程

表 5.5 选择（收藏）课程测试用例

功能描述	选择课程/收藏课程	
用例目的	测试学生是否可以将课程加入课表或者将课程加入收藏	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，学生用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况
学生用户点击选择/收藏一门未选择或者收藏过的课程	提示选课（收藏）成功	选课成功，选课按钮变成了退选按钮
学生用户选择（收藏）报名已选/收藏的课程	无法加入课表/无法加入收藏	选课失败/收藏失败，无法找到入口（因为已经选择/收藏）

(6) 管理员管理学生用户

表 5.6 管理员管理学生用户测试用例

功能描述	管理员管理学生用户	
用例目的	测试管理员是否可以对学生用户进行信息修改，添加或删除	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，教师或学生用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况

管理员点击管理学生标签页	显示出学生列表，每一条学生数据都展示出来	如预期所料，学生列表中的数据可以进行查看和修改
管理员点击修改学生用户信息。修改特定学生用户的某一条数据，并改为错误的数据	可以进行修改，但是无法提交	的确可以进行修改，但无法提交

(7) 管理员管理教师用户

表 5.7 管理员管理教师用户测试用例

功能描述	管理员管理教师用户
用例目的	测试管理员是否可以查看并修改教师用户的信息
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，管理员用户已正确的登录系统，并打开相应界面。

输入/动作	期望的输出/相应	实际情况
管理员点击管理教师标签页	显示出教师列表，每一条教师数据都展示出来	如预期所料，教师列表中的数据可以进行查看和修改
管理员点击修改教师用户信息。修改特定教师用户的某一条数据，并改为错误的数据	可以进行修改，但是无法提交	的确可以进行修改，但无法提交。同时修改多人的信息，如果有一条不合格，则全部无法提交

(8) 管理员管理课程

表 5.8 管理员管理课程测试用例

功能描述	管理员管理课程
用例目的	测试管理员是否可以正确修改课程的信息并设置其是否发布
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，教师或学生用户已正确的登录系统，并打开相应界面。

输入/动作	期望的输出/相应	实际情况

管理员点击管理课程标签页。选择特定课程信息，输入正确的课程信息（审核），点击提交	提示提交成功	提示提交成功，并正确更新数据库
管理员点击管理课程标签页。选择特定课程信息，输入不合规定的课程信息（失误审核），点击提交	提示必须按照规则填写所有信息	网页提示必须按照规则填写所有信息，数据库没有更新

(9) 所有课程信息查看

表 5.9 所有课程信息查看测试用例

功能描述	所有课程信息查看	
用例目的	测试教师或学生是否可以正确查看所有课程信息	
前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，学生用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况
用户点击课程列表。	显示所有可用课程。	网页显示所有可用课程。
用户选择搜索条件并输入搜索关键词，例：搜索条件：任意	显示所有符合筛选提交的课程	网页显示所有符合条件的所有合法已发布的课程。
用户选择搜索条件并不输入搜索关键词	显示所有课程	网页显示所有课程
用户点击查看详情可查看该课程详情	显示查看详情界面	网页显示查看详情界面

(10)管理员管理其他系统信息/参数

表 5.10 管理员管理其他系统信息/参数测试用例

功能描述	管理员管理其他系统信息/参数
用例目的	测试管理员是否可以修改对系统有用的参数/信息

前提条件	Windows 2008 及以上操作系统，cpu 1ghz ram 512mb,网络畅通，学生/教师用户已正确的登录系统，并打开相应界面	
输入/动作	期望的输出/相应	实际情况
管理员点击管理教学楼；然后点击添加教学楼	系统显示可以添加教学楼信息，正确输入的情况下可以提交	输入正确并无重复的教学楼信息，点击提交，显示提交成功
管理员点击管理学科门类以及院系信息	系统显示可以添加学科门类以及院系信息，正确输入的情况下可以提交	输入正确并合法的学科门类和院系信息，点击提交，显示提交成功

5.4 测试结论

本次测试主要进行了单元性的功能测试，基本覆盖了所有功能需求所提及的所有功能，保证了基本功能的实现和完善，测试目标基本完成，通过所有测试。但是界面过于简单，可在之后进行进一步的修改。

第 6 章　结束语

6.1 工作总结

最后一学期快进入尾声，我的毕业设计也迎来了新的篇章——结束语。作为计算机专业的学生，毕业设计是我接手的最大的项目，在前面三年的学习生活中，我更多的是让自己的心静下来，将课堂上学到的东西沉淀下来，变成自己的知识储备。所以过去没有经历过项目开发，不了解一个项目的构成和开发所需要做的准备。但是毕业设计给了我这个机会，着实让我体验了一把什么是项目的 Deadline。在完成论文的过程中，回忆起了各种不同的基础及专业知识，它们现在凝聚在我的毕业论文和设计中，成为我一生的财富。

在 2017 年十二月份，同老师一起讨论拟定毕业设计的题目；在二月份基本确定毕业设计系统的基本结构并做课题分析，查找资料以及数据库设计等前期准备工作；在三月份完善毕业设计思路并学习可能用到的技术，研究需求分析以及可行性方案；在四月份动手实现毕业设计各种功能并进行测试确认系统是否完善；五月份撰写毕业设计论文并准备毕业答辩等事宜。

论文的准备过程中，通过工具将所需要用到的技术整理成文档仔细思索各个功能实现的所有细节，上网搜索补充自身的知识薄弱点，研究遇到的困难。在这个过程中，我接触到许多优秀的技术和工具，技术如 ASP.NET 动态网页技术、JavaScript 技术、AJAX 技术等等；工具如 GitHub、Visual Studio、ProcessOn、MySQL Workbench 等等。期间会碰到很多意想不到的问题，可能会有计划之外的疏忽、技术上的难关，但是学会如何随机应变、及时解决问题也正是我们在其中学到的珍贵的技能之一。在遇到问题，解决问题的过程中，慢慢学会了发现，思考，总结，记录，反思，改正[9]。

6.2 工作展望

本次系统前台使用了来自 Twitter 的开源框架 Bootstrap，它使得前台页面的开发更加容易便捷，但也使得使用这个框架时，限于对框架本身的不了解，局限了部分功能的实现。在前期准备时应该更多参考框架本身的使用手册以及理解开源代码。

本次系统使用了 MVC 最根本的.NET 的 HTML 结合 Razor 引擎的方法实现。虽然较为基础通用，但是却不是现今主流使用框架，之后应该更多学习类似 spring mvc 等现今主流框架[10]。

参考文献

[1] Dan Pilone. UML 2.0 Pocket Reference: UML Syntax and Usage [M]. O'Reilly Media Publish House [M], 2013

[2] JON DUCKETT. JavaScript and jQuery: Interactive Front-End Web Development Hardcover [M]. Wiley Publish House, 2014

[3] 萨默维尔 (Ian Sommerville). 软件工程[M]. 北京:机械工业出版社, 2011

[4] 弗瑞曼 (Adam Freeman). 精通 ASP.NET MVC5[M]. 北京:人民邮电出版社, 2016

[5] 唐汉明, 翟振兴, 关宝军, 王洪权. 深入浅出 MySQL:数据库开发、优化与管理维护（第 2 版）[M]. 北京:人民邮电出版社, 2014

[6] 珍妮弗•凯瑞恩 (Jennifer Kyrnin) (作者), 姚军 (译者). Bootstrap 入门经典（第 1 版）[M]. 人民邮电出版社, 2016

[7] 陈华. Ajax 从入门到精通（第 1 版）[M]. 清华大学出版社, 2008

[8] 明日科技. C#从入门到精通(第 4 版) [M]. 清华大学出版社, 2017

[9] 龙马高新教育. 网站建设从入门到精通（第 1 版）[M]. 北京大学出版社, 2017

[10] 雷宁. 零基础学 HTML+CSS（第 1 版） [M]. 机械工业出版社, 2009

致　谢

在这次完成毕业设计的过程中，我要感谢我的指导老师刘亚，感谢她在这整个过程中，一直给予我耐心指导。从立题到分析需求，从系统设计到实施构建，从完成编程到后期测试，刘亚老师一直都悉心为我解答遇到的疑惑和困难，教导我如何克服遇到的困难。另外，我还要感谢每一位传授过我知识的老师，在大学的四年间我从不同的老师那里学习到了各种知识，而这些知识也必将成为我未来工作的基石和生活的基础。其次，我要感谢我的同事和朋友以及家人，感谢他们在完成毕设的期间给予我的帮助和建议。

附录 A: 主要源代码

详细源代码见附件一（以下给出后台处理逻辑代码）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;
using System.Web.Mvc;
using GradutionProject.Models;

namespace GradutionProject.Controllers
{
    public class CoursesController : Controller
    {
        // GET: Courses
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult AddCourses()
        {

            DataTable
                //building = DBManip.GetBuildingList(),
                subject = DBManip.GetSubjectList(),
                room = DBManip.GetRoomList();
            DataSet set = new DataSet();
            //building.TableName = "BuildingList";
            //set.Tables.Add(building.Copy());

            subject.TableName = "SubjectList";
            set.Tables.Add(subject.Copy());

            room.TableName = "RoomList";
            set.Tables.Add(room.Copy());


            return View(set);
        }
        public object AddCourse(FormCollection fc)
        {
```

```
            //未设置重复添加课程的检测，只有此处 Session 用作判断，但没什么
实质作用
            if (Session["S_if_added_this_course"] == null ||
Session["S_if_added_this_course"].ToString() == "no")
            {
                Session["S_if_added_this_course"] = "yes";
            }
            else
            {
                Session["S_if_added_this_course"] = "no";
                return "You have added this course";
            }
            Course course = new Course
            {
                CourseName = fc["courseName"],
                StartDate = fc["startDate"],
                EndDate = fc["endDate"],
                Venue = fc["venue"],
                Period = fc["period"],
                CourseInfo = fc["courseInfo"],
                CourseType = fc["subject"]
            };
            Exam exam = new Exam
            {
                ExamDate = fc["examPeriod"],
                ExamVenue = fc["examVenueClass"],
                ExamDuration = fc["examDuration"],
                ExamType = fc["examType"] == "open" ? ExamType.开
卷.ToString() : ExamType.闭卷.ToString()
            };

            //此处问题：不能强制将 User 转为 Teacher 类型
            //【未解决】
            User teacher = (User)Session["S_user"];
            course.CourseTeacher = teacher.UniqueClientID;
            DBManip.AddCourse(course, exam);

            return "Error";
        }

        [CookieFilter]
        public object DisplayCourses()
        {
            string studentID = ((User)Session["S_user"]).RegistryType == 'S' ?
((User)Session["S_user"]).UniqueClientID : "";
            //string studentID = ((User)Session["S_user"]).UniqueClientID;
            DataSet set = DBManip.GetCourseTable(studentID);
            ViewData["browseType"] = studentID;
```

```
            return View(set);
        }

        [CookieFilter]
        public ActionResult SpecificCourse(string id, string state)      // id = courseID;
state = selected/not selected
        {
            if (id == null)
            {
                return Redirect("MainPage");
            }
            else if (id.Length != 10)
            {
                return PartialView("FoF");
            }

            //tc = teacher_course_information
            DataSet tc = DBManip.GetCourseInfo(id);
            Session["S_courseid"] = id;
            ViewData["V_uid"] = ((User)Session["S_user"]).UniqueClientID;
//stduent id
            switch (state)
            {
                //Only three situations: select/collect/both
                case "A":
                case "B":
                case "C":
                case "N":
                    break;
                default: state = ""; break;
            }
            //Session["S_state"] = state;
            //本来使用的是 ViewData["V_state"] = state;  但现在似乎两者没什么区
别：因为这里 TempData 并不能跨 Action 传值。
            TempData["T_state"] = state; //Original: Session["S_state"];

            return View(tc);
        }

        public ActionResult MyCourse(string id)    // id = courseID
        {
            DataSet tc = DBManip.GetCourseInfo(id);
            ViewData["V_uid"] = ((User)Session["S_user"]).UniqueClientID;
//teacher id
            return View(tc);
        }

        public object ChooseCourse()
        {
```

```
        //Session["S_courseid"] selected course id : String()
        //Session["S_student"]    the info of the one who select : User()

        string selectedCourseId = Session["S_courseid"].ToString();
        string selectedBy = ((User)Session["S_user"]).UniqueClientID;
        char selectType = 'A';//which means Choose course

        DBManip.SelectCourse(selectedBy, selectedCourseId);

        return selectType;              // Chosen
    }

    public object CollectCourse()
    {
        string selectedCourseId = Session["S_courseid"].ToString();
        string selectedBy = ((User)Session["S_user"]).UniqueClientID;
        char selectType = 'B';//which means Choose course

        DBManip.SelectCourse(selectedBy, selectedCourseId, selectType);

        return selectType;              //Collected
    }

    public object UnchooseCourse()
    {
        //if (Session["S_state"].ToString() == "C")
        //{
        //      //TempData["T_state"] = "B";
        //      Session["S_state"] = "B";
        //}
        //else
        //{
        //      Session["S_state"] = "";
        //}

        string selectedCourseId = Session["S_courseid"].ToString();
        string selectedBy = ((User)Session["S_user"]).UniqueClientID;
        char selectType = 'a';//which means Choose course

        DBManip.DeselectCourse(selectedBy, selectedCourseId);

        return selectType;              //Unchosen
    }

    public object UncollectCourse()
    {
        //if (Session["S_state"].ToString() == "C")
        //{
        //      //TempData["T_state"] = "A";
```

```
        //      Session["S_state"] = "A";
        //}
        //else
        //{
        //      Session["S_state"] = "";
        //}
        string selectedCourseId = Session["S_courseid"].ToString();
        string selectedBy = ((User)Session["S_user"]).UniqueClientID;
        char selectType = 'b';//which means Choose course

        DBManip.DeselectCourse(selectedBy, selectedCourseId, selectType);


        return selectType;              //Uncollected
    }

    public object ValidateVenuePeriod(string json)
    {
        if (json == "" || json.Contains('_') == false)
        {
            return PartialView("FoF");
        }
        else
        {
            string[] para = json.Split('_');      //[0] venue    [1] period
            if (para.Length != 2)
            {
                return PartialView("FoF");
            }
            else
            {
                string result = DBManip.IfConflicted(para[0], para[1]);
                if (result == "false")
                {
                    return Content("false");
                }
                else
                {
                    return Content(result);
                }
            }
        }
    }

    public object ValidateVenuePeriodForStudent(string json)
    {
        if (json == "" || json.Contains('_') == false)
        {
            return PartialView("FoF");
```

```
                }
                else
                {
                    string[] para = json.Split('_');        //[0] period    [1] studentID
                    if (para.Length != 2)
                    {
                        return PartialView("FoF");
                    }
                    else
                    {
                        bool result = DBManip.IfConflictedStudent(para[0], para[1]);
                        if (result == false)
                        {
                            return Content("false");
                        }
                        else
                        {
                            return Content("true");
                        }
                    }
                }
            }
        }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace GradutionProject.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            bool iflogged = false;
            if (Session["S_user"] != null)
            {
                iflogged = true;
            }
            return View(iflogged);
        }

        public object IfLogged()
        {
            if (Session["S_user"] != null)
```

```
        {
            return null;
        }
        return Content("N");
    }
    public object Logout()
    {
        //HttpCookie CookieMessage;
        //string cookieName;
        //for (int i = 0; i < Request.Cookies.Count; i++)
        //{
        //    cookieName = Request.Cookies[i].Name;
        //    CookieMessage = new HttpCookie(cookieName)
        //    {
        //        Expires = DateTime.Now.AddDays(-1)
        //    };
        //    Response.Cookies.Add(CookieMessage);
        //}
        Session.Clear();
        return View();
    }

    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }

    public ActionResult Everyday()
    {
        ViewBag.Message = "Your contact page.";

        return View();
    }
    public ActionResult Test()
    {
        return View();
    }
    public ActionResult Test1()
    {
        return View();
    }
    public JsonResult AjaxTest()
    {
        bool result = true;
        return Json(result);
    }
    public JsonResult CheckUsernameTest(string name)
```

```
        {
            bool result = true;

            if (name == "admin")
            {
                result = false;
            }
            return Json(result);
        }
        // GET: Miscellaneous
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult AddRoom()
        {
            DataTable table = DBManip.GetBuildingList();

            return View(table);
        }
        public object AddRooms()
        {
            return "添加成功";
        }
        public ActionResult AddBuilding()
        {
            DataTable table = DBManip.GetBuildingList();
            //table.Columns.Remove(table.Columns["rooms"]);
            return View(table);
        }

        // GET: Users
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Login()
        {
            //【未解决】该页面存在一个问题：使用浏览器的回退（Back）则会清
除 Session。
            if (Session["S_user"] != null)
            {
                if (((User)Session["S_user"]).RegistryType == 'T')
                {
                    return RedirectToAction("TeacherMainPage");
                }
                else if (((User)Session["S_user"]).RegistryType == 'S')
```

```
            {
                return RedirectToAction("StudentMainPage");
            }
            else
            {
                return RedirectToAction("AdminMainPage");
            }
        }
        else
        {
            return View();
        }
    }

    [HttpGet]
    public object LoginTypeCheck()
    {
        return RedirectToAction("Login");
    }

    [HttpPost]
    public object LoginTypeCheck(FormCollection fc)
    {
        User user = null;
        if (Session["S_user"] != null)
        {
            user = (User)Session["S_user"];
        }
        else
        {
            user = new User
            {
                UniqueClientID = fc["Uid"],
                Username = fc["inputEmail"],
                Password = fc["inputPassword"]
            };
            user.RegistryType = user.UniqueClientID[1];
            Session["S_user"] = user;
        }

        if (user.RegistryType == 'S')
        {
            return Redirect("StudentMainPage");
        }
        else if (user.RegistryType == 'T')
        {
            return Redirect("TeacherMainPage");
        }
        else
```

```
            {
                return Redirect("AdminMainPage");
            }

        }

        public ActionResult Register()
        {
            return View();
        }

        [CookieFilter]
        public ActionResult AdminMainPage()
        {
            DataTable teacher = DBManip.GetTeacherTable(),
                                student = DBManip.GetStudentTable(),
                                course = DBManip.GetAllCourses();
            //此处注意：DataSet 的 Add()方法不能添加两个没有命名的
DataTable；见下网址：
            //https://www.cnblogs.com/chenhuzi/archive/2010/11/02/dataset-add-more-
table-example.html

            DataSet set = new DataSet();
            student.TableName = "student";
            set.Tables.Add(student.Copy());

            teacher.TableName = "teacher";
            set.Tables.Add(teacher.Copy());

            course.TableName = "AllCourses";
            set.Tables.Add(course.Copy());
            return View(set);
        }

        [HttpGet]
        [CookieFilter]
        public ActionResult TeacherMainPage()
        {
            ViewData["V_uname"] =
DBManip.GetUser(((User)Session["S_user"]).UniqueClientID);
            DataSet set =
DBManip.GetCourseOfTeacher(((User)Session["S_user"]).UniqueClientID);
            return View(set);
        }

        [HttpPost]
        public object TeacherMainPage(FormCollection fc)
        {
            User user = (User)Session["S_user"];
```

```
            Teacher teacherInfo = new Teacher
            {
                UniqueClientID = user.UniqueClientID,
                Name = fc["name"],
                Gender = Convert.ToChar(fc["gender"]),
                Age = Convert.ToInt32(fc["age"]),
                Wechat = fc["wechat"],
                QQ = fc["qq"],
                Phone = fc["phone"],
                University = fc["university"],
                Position = Convert.ToChar(fc["position"]),
                College = fc["college"],
                Major = fc["major"],
                WorkNo = fc["workNo"],
                WorkPass = fc["workPass"]
            };
            DBManip.AddUser(user, teacherInfo);
            ViewData["V_uname"] = teacherInfo.Name;
            return Redirect("TeacherMainPage");
        }

        [HttpGet]
        [CookieFilter]
        public ActionResult StudentMainPage()
        {
            ViewData["V_uname"] =
DBManip.GetUser(((User)Session["S_user"]).UniqueClientID);
            DataSet set =
DBManip.GetCourseOfStudent(((User)Session["S_user"]).UniqueClientID);
            return View(set);
        }

        [HttpPost]
        public object StudentMainPage(FormCollection fc)
        {
            User user = (User)Session["S_user"];
            Student studentInfo = new Student
            {
                UniqueClientID = user.UniqueClientID,
                Name = fc["name"],
                Gender = Convert.ToChar(fc["gender"]),
                Age = Convert.ToInt32(fc["age"]),
                Wechat = fc["wechat"],
                QQ = fc["qq"],
                Phone = fc["phone"],
                Grade = fc["grade"],
                Position = Convert.ToChar(fc["position"]),
                College = fc["college"],
                Major = fc["major"],
```

```
            CardNo = fc["cardNo"],
            CardPass = fc["cardPass"]
        };
        DBManip.AddUser(user, studentInfo);
        ViewData["V_uname"] = studentInfo.Name;
        return Redirect("StudentMainPage");
    }

    [CookieFilter]
    public ActionResult DisplayUsers()
    {
        if (((User)Session["S_user"]).RegistryType != 'A')
        {
            return Redirect("MainPage");
        }
        DataTable table = DBManip.GetUserTable();

        return View(table);
    }

    [CookieFilter]
    public ActionResult DisplayTeachers()
    {
        DataTable table = DBManip.GetTeacherTable();

        return View(table);
    }

    [CookieFilter]
    public ActionResult DisplayStudents()
    {
        DataTable table = DBManip.GetStudentTable();

        return View(table);
    }

    [HttpGet]
    public ActionResult AddStudent()
    {
        //404 not found
        return PartialView("FoF");
    }

    [HttpPost]
    public ActionResult AddStudent(FormCollection fc)
    {
        User student = new User(fc["inputEmail"], fc["inputPassword"], 'S');
        bool existence = DBManip.CheckUserExistence(ref student);
        if (existence == true)
```

```
        {
            //User has been registered, do the re-register
            return Redirect("Register");
        }
        //DBManip.AddUser(student);
        Session["S_user"] = student;
        return View();
    }

    [HttpGet]
    public ActionResult AddTeacher()
    {
        //404 not found
        return PartialView("FoF");
    }

    [HttpPost]
    public ActionResult AddTeacher(FormCollection fc)
    {
        User teacher = new User(fc["inputEmail"], fc["inputPassword"], 'T');
        bool existence = DBManip.CheckUserExistence(ref teacher);
        if (existence == true)
        {
            //User has been registered, do the re-register
            return Redirect("Register");
        }
        //DBManip.AddUser(teacher);
        Session["S_user"] = teacher;
        return View();
    }

    public ActionResult SpecificUser(string id)
    {
        string[] tableTest = new string[2];
        tableTest[0] = DBManip.GetSpecificUser(id);
        tableTest[1] = "A TestTable";
        return View(tableTest);
    }

    public object GetSpecificUser(string id)
    {
        string data = DBManip.GetSpecificUser(id);
        return data;
    }

    [HttpPost]
    public object ModifySpecificUser(FormCollection fc)
    {
        string[] arraydata = fc["ajaxData"].ToString().Split('&');
```

```
            string tableName = arraydata[0][1] == 'S' ? "student_information" :
"teacher_information";

            string cmdTxt = "update " + tableName + " set ", tmp = "";
            //From one
            for (int i = 1; i < arraydata.Length; i++)
            {
                tmp += arraydata[i] + ',';
            }

            tmp = tmp.Substring(0, tmp.Length - 1);       //除掉最后的‘,’号
            cmdTxt = cmdTxt + tmp + " where uniqueClientID='" + arraydata[0] + "'";
            int rowAffected = DBManip.ModifyDBwithSQL(cmdTxt);

            if (rowAffected == 0)
            {
                return PartialView("Error");
            }
            return Redirect("DisplayUsers");
        }

        //[CookieFilter]              //Not yet perceived how it worked.
        public object MainPage()
        {
            if (Session["S_user"] != null)
            {
                if (((User)Session["S_user"]).UniqueClientID[1] == 'T')
                {
                    return RedirectToAction("TeacherMainPage");
                }
                else if (((User)Session["S_user"]).UniqueClientID[1] == 'S')
                {
                    return RedirectToAction("StudentMainPage");
                }
                else
                {
                    return RedirectToAction("AdminMainPage");
                }
            }
            else
            {
                return RedirectToAction("Login");
            }
        }

        public object CheckExist(string email, string passwd)
        {
            User user = new User
            {
```

```
                //unknown which type
                Username = email,
                Password = passwd
            };
            bool Existence = DBManip.CheckUserExistence(ref user, true);
            Session["S_user"] = user;
            if (Existence == false)
            {
                return Content("N");
            }
            else
            {
                return Content(user.UniqueClientID);
            }
        }

        public object LoadDenpendencies()
        {
            return DBManip.LoadDenpendencies();
        }

        public object ModifyMultipleUsers(FormCollection fc)
        {
            if (fc["cmdTxt"] == null)
            {
                return null;
            }
            int rowAffected = DBManip.ModifyDBwithSQL(fc["cmdTxt"].ToString());
            if (rowAffected == 0)
            {
                return PartialView("Error");
            }
            return Redirect("AdminMainPage");
        }

        public object ModifySpecificCourse(FormCollection fc)
        {
            if (fc["cmdTxt"] == null)
            {
                return null;
            }
            int rowAffected = DBManip.ModifyDBwithSQL(fc["cmdTxt"].ToString());
            if (rowAffected == 0)
            {
                return PartialView("Error");
            }
            return Redirect("AdminMainPage");
        }
    }
```

```
}
```

附录 B: 软件使用说明书

1.软件描述

本系统是为了为学生选课和教师安排课程提供一个方便的平台，使用网站的形式利于教师和学生在任何设备上都能访问并完成操作。

1.1 系统主要功能

本系统主要功能如下：

(1) 登录：根据角色权限不同，登录后可以完成不同角色功能。

(2) 课程信息管理：教师用户可以发布课程及编辑课程；管理员用户可以管理（审核删除）课程；学生用户可以选择课程，安排课表。

(3) 用户信息管理： 管理员用户可以管理教师或学生的账户信息；教师或学生用户可以管理个人信息。

(4) 排课和选课管理：教师可以查看系统得到的当前周需要进行的课程；学生可以查看当前课表，也可以退选课。（当选课人数达到 25 人时，教师进行删除课程必须通过管理员审核；在 25 人之前则无需审核。）

2.软件的安装

2.1 系统环境需求

(1) 基本硬件平台

CPU：PII 或更高

内存：412MB 或更高

显示器：17’或更高

(2) 基本软件平台

操作系统：Windows Server 2003 或 Windows XP 以上

数据库：SQLSERVER 2008

服务器：IIS5.0 或更高

分辨率：1024*768 或更高

2.2 使用说明

在系统部署好以后，管理员或者用户可以直接打开浏览器输入网址即可访问，登录成功后进入对应的主页。对于管理员用户来说，有许多需要注意的系统参数都是按

照特定格式存储在数据库中，请查阅此链接文件 1（学生选课管理系统数据库设计），链接文件 2（学生选课管理系统数据库设计_新版）并阅读。