摘　要

随着高等院校的教育改革，现在国产大多数高校的课程安排采用的是学生自主选课的模式。这样做一方面有利于提高学生上课的积极性，另一方面，也有利于满足不同学生的个性化发展。然而学生在选课时，对备选课程与授课教师往往是完全陌生的，不知道这些课程所教授的内容，教授这门课程的教师有哪些，这些教师有什么特点，教的好不好等。同时学生完成一门课程后，对大多数学生来讲，这门课程相关的书籍就变成了废品了，而高校课程繁多，如果每门课程都购买新书，对部分经济困难的学生可能也是一笔不小的开支。此外，基于环保的理念，合理利用这些旧书，不仅可以减轻学生的经济压力，还能对环境保护做出一份贡献。

虽然目前国内有超级课程表，课程格子等一系列课程表应用， 但它们关注点都在帮助学生解决课程管理的问题。本系统的主要目标主要有两个：其一，让学生对将要学习的课程有一个全面清晰的认识，选择适合自己教师与课程，提升学习效率与热情；其二，充分利用师生的资源，包含二手书籍、复习资料、学习笔记等，减少资源浪费，保护环境。

本系统采用分布式架构，将系统拆分为多个独立的子系统并通过 WCF 技术进行进程间的通信。前端系统采用 B/S 架构进行设计，后端服务系统采用三层架构，是的系统具有良好的稳定性与可扩展性。

本系统严格按照软件设计流程进行，经过需求分析，概要设计，详细设计，实现与测试验证后，系统能够顺利运行。

关键字： 高校课程社区； 分布式技术； .NET 技术； WCF 技术； HTTP 协议；

Abstract

With the reform of education in colleges and universities, the curriculum arrangement of most universities in China now adopts the model of students' independent choice of courses. On the one hand, it is beneficial to improve the students' enthusiasm for class, on the other hand, it can also help to meet the individualized development of different students. Students in course selection, however, the optional courses and teaching teachers often is a complete stranger, don't know what these courses teach, what are the course of the teacher, a professor at what is the distinguishing feature of these teachers, teach good, etc. After students complete a course at the same time, for most students, the course books becomes waste, various courses in colleges and universities, if each course are buying new books, with some economic difficulties students may also have a lot of spending. In addition, based on the concept of environmental protection, reasonable use of these old books can not only reduce the economic pressure of students, but also make a contribution to environmental protection.

Although there are a number of curriculum programs, such as the super curriculum and curriculum grid, the focus is on helping students solve the problem of curriculum management. The main goal of this system basically has two: first, lets the student will learn the course has a comprehensive clear understanding, choose suitable for their teachers and courses, improve the learning efficiency and enthusiasm; Second, make full use of the resources of teachers and students, including second-hand books, review materials, study notes, etc., to reduce waste of resources and protect the environment.

This system uses a distributed architecture to break up the system into separate subsystems and communicate with each other through WCF technology. The front-end system is designed with B/S architecture, and the back-end service system adopts three-layer architecture, and the system has good stability and extensibility.

The system is carried out strictly according to the software design process, and the system can run smoothly after the requirement analysis, summary design, detailed design,

implementation and test verification..

Key words: College Curriculum Community; Distributed Technology; WCF Technology; HTTP;

目 录

绪论 1
1 需求分析 1
1.1 系统需求概述 1
1.2 用户交互系统 2
1.2.1 用户信息需求 2
1.2.2 课程评教需求 2
1.2.3 教师评教需求 2
1.2.4 二手交易市场需求 2
1.2.5 论坛与博客 3
1.2.6 运营管理需求 3
1.3 后端服务系统 3
1.3.1 业务接口层 3
1.3.2 数据处理层 3
1.3.3 数据访问层 4
1.4 资料管理需求 4
1.5 本章小结 4
2 系统设计 4
2.1 系统的概要设计 4
2.2 系统的详细设计 5
2.2.1 用户信息管理详细设计 5
2.2.2 课程信息模块详细设计 6
2.2.3 教师信息模块详细设计 8
2.2.4 二手市场市场详细设计 9
2.2.5 论坛博客区详细设计 11
2.2.6 管理员页面详细设计 13
2.3 数据库设计 13
2.3.1 关系数据库 E-R 图 13
2.3.2 数据库详细设计 14
2.4 本章小结 16
3 设计方案论证 16
3.1 系统设计中的关键问题与解决方案 16
3.1.1 HTTP 的安全性 16
3.1.2 服务器之间的通信 17
3.1.3 图像的上传与下载 17
3.1.4 邮箱可用性验证 17
3.1.5 不良信息过滤 18
3.2 系统开发中的关键技术 18
3.3 本章小结 18
4 系统的具体实现 18

4.1 交互系统 18
4.1.1 课程信息具体实现 18
4.1.2 教师信息具体实现 19
4.1.3 二手交易市场具体实现 19
4.1.4 博客论坛去具体实现 20
4.2 后端服务系统 21
4.3 本章小结 21
5 系统的测试与调试 22
5.1 测试的目的及意义 22
5.2 测试的内容与方法 22
5.2.1 黑盒测试 22
5.2.2 白盒测试 22
5.2.3 单元测试 22
5.2.4 集成测试 22
5.2.5 系统测试 22
5.3 测试的结果 22
5.4 本章小结 23
结论 23
参考文献 24
致谢 25
附录 数据库 SQL 语句 26

绪论

随着高等院校的教育改革，现在国产大多数高校的课程安排采用的是学生自主选课的模式。这样做一方面有利于提高学生上课的积极性，另一方面，也有利于满足不同学生的个性化发展。然而学生在选课时，对备选课程与授课教师往往是完全陌生的，不知道这些课程所教授的内容，教授这门课程的教师有哪些，这些教师有什么特点，教的好不好等。同时学生完成一门课程后，对大多数学生来讲，这门课程相关的书籍就变成了废品了，而高校课程繁多，如果每门课程都购买新书，对部分经济困难的学生可能也是一笔不小的开支。此外，基于环保的理念，合理利用这些旧书，不仅可以减轻学生的经济压力，还能对环境保护做出一份贡献。

虽然目前国内有超级课程表，课程格子等一系列课程表应用， 但它们关注点都在帮助学生解决课程管理的问题，例如让学生知道本周或当天的课程安排，上课的教师在哪，是哪个老师在上课等。且这类 APP 在每个新学期都拥有较高的日活数，这是因为刚开学，大多数学生对课程安排都不熟悉的缘故。而随着时间的推移，学生渐渐对比较固定的课程安排比较熟悉了，也就减少了对课程表的依赖，此时这类 APP 的日活数就开始有了一个阶段性的下降，一直到学期末，学生需要安排考试信息，这类 APP 的日活才会再迎来一个提升期。

本系统的主要目标主要有两个：其一，让学生对将要学习的课程有一个全面清晰的认识，选择适合自己教师与课程，提升学习效率与热情；其二，充分利用师生的资源，包含二手书籍、复习资料、学习笔记等，减少资源浪费，保护环境。

1 需求分析

1.1 系统需求概述

本系统是基于对课程与老师评教的基础上构建的一个具有二手物品交易以及学习心得等分项的综合性的社区。通过对整个系统的综合性的分析，暂且将整个系统分为三个子系统，分别为后台服务系统、资料管理系统与用户交互系统。

顾名思义，后台服务系统的功能主要是作为核心业务层与数据访问层，具有直接操作数据库的权限，它将核心业务进行封装，提供前端显示层一个通用的接口，它隔离了前端与数据库的直接交互，确保了数据安全性以及整个系统的可维护性与可扩展性。

资料管理系统的主要的功能是对整个系统在运行中产生的各类数据进行一个统一的管理，使得数据更集中，更易于管理。其中最核心的就是图像文件的上传与下载功能，例如用户注册的头像资料，二手商品的的图像信息等图像文件的管理。

前端交互系统是直接与用户进行交互的一套系统，它又可以分为多个子系统，包括 PC 端、Web 端、Android 端、IOS 端以及微信公众号等。它的核心就是为用户提供良好

的交互体验与正确的数据的数据服务。它的主要功能就是用户的注册于登录一个个人信息的管理、课程信息的浏览与评教、教师信息的浏览与评教、二手商品的交易以及论坛的浏览与发帖等。其中最核心的功能就是用户注册的邮箱与手机的认证，这是保证后面功能能正确执行的保障。

1.2 用户交互系统

1.2.1 用户信息需求

用户注册是系统最基础也是最核心的功能需求。用户需要用一个可用的邮箱进行注册，同时还需填写用户姓名，手机号码，注册密码以及学校信息。用户注册后会跳转到用户信息页面，这个页面包含所有用户已经填写的信息与还未填写的信息，同时还会给注册邮箱发一封验证邮件，用以验证这个邮箱的是否是用户本人的邮箱。如果用户邮箱未通过验证或还没验证，邮箱信息后面会有一个红色的×标记，反之则是一个绿色的√标记。然后用户可以选择继续完善用户信息，包括上传头像，设置昵称或修改注册基本信息等。

1.2.2 课程评教需求

课程评教需求的基本要求是用户能查看课程的信息以及对课程进行评教。用户可以以游客模式浏览课程信息，但是必须登录后才能评教。课程的详细信息与用户的评教内容显示在同一页，因此整个页面分为两个模块，上面是课程信息，下面是用户的评教内容并对评教内容进行分页。评教的内容包含用户的头像，昵称，评教时间与评教的内容。而且用户可以选择撤回已经发出去的评论。

1.2.3 教师评教需求

与课程评教需求一样，教师评教需求的基本要求是用于可以以游客模式浏览教师的信息，但必须登录后才能对教师进行评教。所有内容同样是在一个页面显示。当评教内容达到阈值时会将评教的内容进行分页。用户可以撤销已经发出去的评论。

1.2.4 二手交易市场需求

二手交易市场需求是本系统的一个重要功能点。设计它的初衷是让师生能让课程结束后无用的笔记与二手数等学习资料得到合理的利用，不但帮助了下一届学生学习的效率，也降低了二手物品变成无用垃圾的转换率。

它的基本功能需求是二手物品的展示，游客可以通过快捷方式搜索自己感兴趣的物品。用户进入个人中心，可以查看与管理自己已经购买或已经发布的商品。同时用户可以进入二手物品发布页面，申请上架售卖自己手的物品。其中物品类型是系统预定义的，因此用户只能售卖已经定义的各个类别的物品。提交申请后等待后台管理员审核通过后才可进行上架售卖。不仅如此，在个人中心，用户可以删除购买或销售记录或者下架自

己正在售卖的物品。

用户选中自己感兴趣的物品后可以选中“我想要”按钮进入最终确认页面或者点击“详细”按钮进入商品详细页，浏览与发布留言以确认商品的具体信息，如果用户最终确认要购买此物品，那么系统就会给卖家发送一封电子邮件通知卖家有人有意向购买其售卖的二手商品。后续的发货与付款全部交由卖家与买家在线下完成，本系统不参与这些环节。因此系统发送邮件给卖家后会自动判定本次交易完成。本系统拒绝卖家自卖自买，但卖家可以回复用户的留言，页面会在卖家留言后边标记“卖家”。

1.2.5 论坛与博客

博客与论坛可以统一为帖子，不同是他们的内容与类型有区别。帖子专区的首页是所有用户发布的帖子，用户可以根据类型来选择自己想看的内容。游客选择一个帖子后可以进入详细页面，然后游客就可以浏览帖子的详细内容与其他用户的回帖信息。用户需要登录后才能发帖与回帖。

用户在发帖的时候只能选择系统预定的帖子类型，同时可以选择是否允许回帖，如果禁止回帖则所有人（包括发帖者）都无法回复这个帖子。用户可以进入个人中心管理所有帖子信息，包括修改贴子的状态与内容或删除帖子。

1.2.6 运营管理需求

管理员模块是专门给管理员使用的后台管理系统，它主要包含课程信息管理，教师信息管理，二手交易信息管理与综合信息管理几大模块。

课程信息管理包括课程信息的发布与管理，用户申请添加的课程信息的审核以及对恶意评论的处理。

教师信息管理包括教师信息的发布与管理以及对恶意评论的处理。

二手交易信息管理包括用户申请的交易的审核，在线交易物品的上架与下架权限以及对恶意评论的处理。

综合管理模块包括除了上述的三大模块的其他所有运营或管理需求。例如，数据的批量导入与导出，用户找回密码的申请审核，敏感词汇设置以及运营权限的设置等。

1.3 后端服务系统

1.3.1 业务接口层

为交互系统调提供数据服务，包括课程信息服务接口，教师信息服务接口，交易信息服务接口与其他服务接口等几大类。

1.3.2 数据处理层

主要包含敏感信息与令人不适信息的过滤，邮件的发送，信息安全处理与信息验证等处理。

1.3.3 数据访问层

数据访问层的主要功能是数据的直接读写，事务处理，请求的数据的一致性与完整性。

1.4 资料管理需求

这个系统主要包含图像的上传与下载以及各类数据文件的管理。

1.5 本章小结

本章首先介绍了本系统的设计目的与意义，然后分别详细介绍了各个子系统所需实现的功能需求。

2 系统设计

2.1 系统的概要设计

整个系统拆分为 3 个子系统，各个子系统通过 WCF 与 HTTP 协议进行通信，用户直接访问交互系统。他们之间的关系如图 2.1 所示。

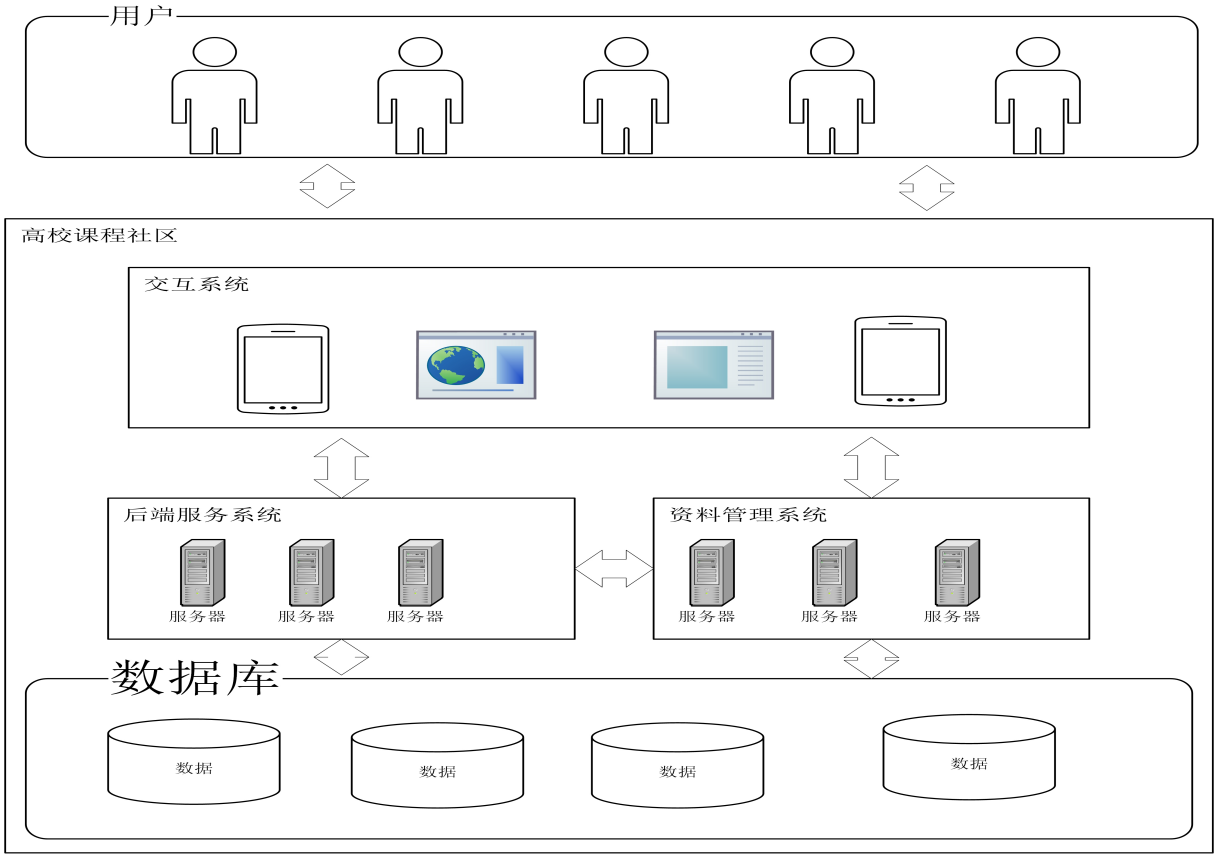

图 2.1 用户系统交互概要设计图

交互系统通过 WCF 获取服务系统提供的数据，通过 HTTP 协议访问资料管理系统

来上传与下载图像数据。服务系统与资料管理系统直接访问数据库。其中后端服务系统存的是业务数据，资料管理系统存的是文件的路径数据。

2.2 系统的详细设计

2.2.1 用户信息管理详细设计

游客通过手机 APP 或网页进入高校课程社区系统的首页，然后点击注册按钮进入新用户注册页。然后交互系统向业务系统（后端服务系统，下同）提交一个注册申请，由业务系统验证用户信息并向注册邮箱发送邮箱验证邮件，最后返回注册结果到交互系统并由交互系统告知用户注册结果。注册成功的用户会自动切换到用户个人信息页。用户登出后再次登录同样是由交互系统向业务系统提交登录申请，业务系统进行必要的检查与处理后返回交互系统登录结果。游客的注册于用户的登录流程如图 2.2 所示。

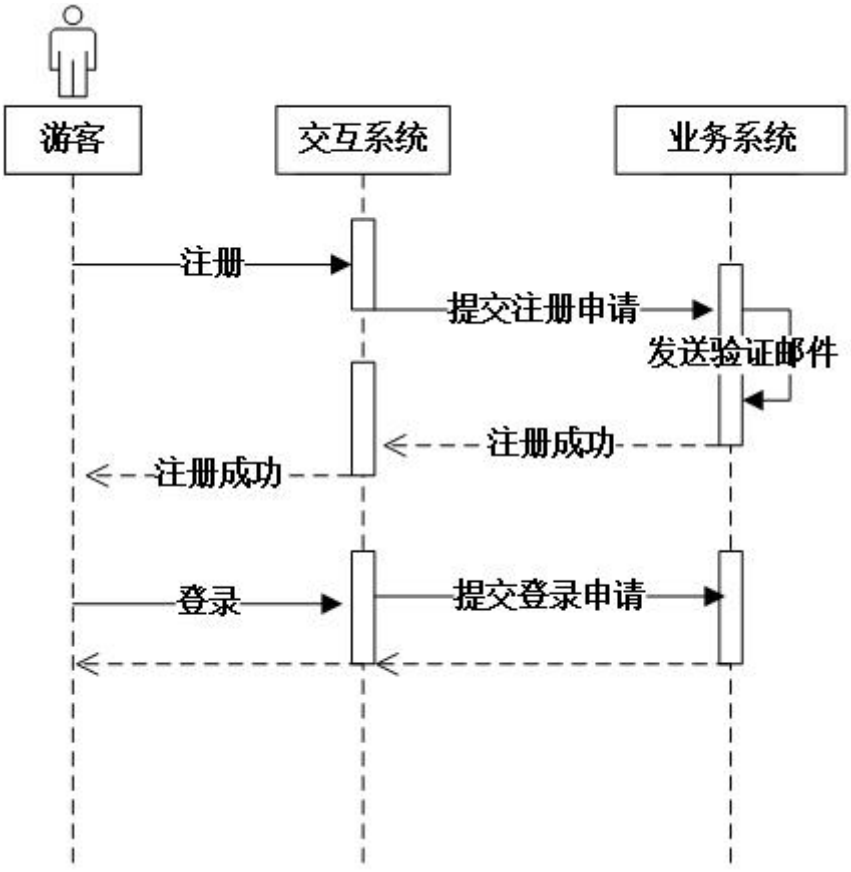

图 2.2 游客注册于登录流程图

用户进入个人信息的详细页面后可以继续完善个人信息，如图 2.3 所示，用户完善个人信息后，系统会先将用户的头像上传到资料管理系统，然后资料管理系统会返回一个图片 URL 给交互系统，最后由交互系统把用户的完整信息提交到业务系统进行处理。其中与资料管理系统的通信是通过 HTTP 协议，而与业务系统通信采用的是 WCF 远程过程调用框架。

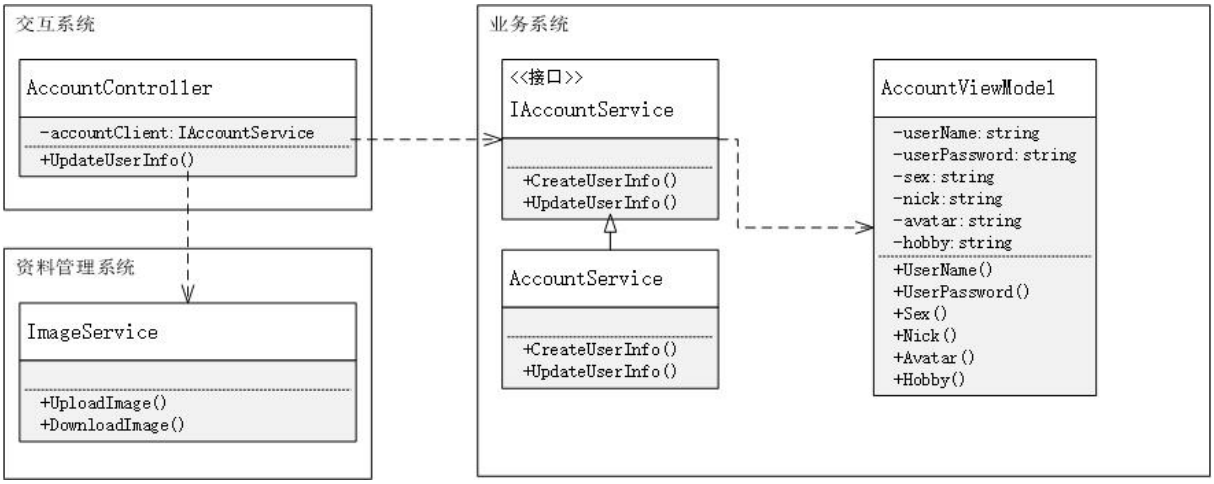

图 2.3 修改个人信息详细设计

用户如果忘记了登录密码，可以申请找回密码进行密码重置。首先，用户进入找回密码页填写账号信息与验证问题，然后提交等待管理员进行审核。系统会将审核结果发送用户的注册邮箱，如果审核通过，用户点击右键附带的链接跳转进行密码重置。详细操作流程如图 2.4 所示。

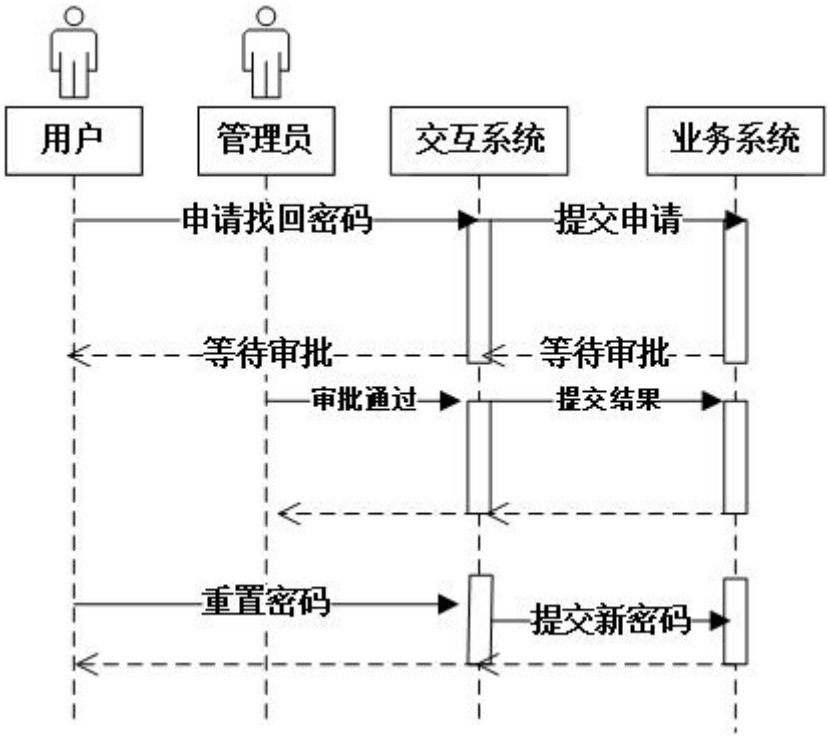

图 2.4 找回密码详细设计

2.2.2 课程信息模块详细设计

课程信息的操作的类结构设计如图 2.5 所示。业务系统定义了一个 ICourseService 接口给交互系统进行课程信息的读写操作，其各个方法的含义如下：

GetValidCode()，获取一个有效的课程编码，提高管理员导入课程数据的效率。

GetCourseInfoList()，获取课程信息列表。

AddCourseInfo()，添加新的课程信息。

ModCourseInfo()，修改课程信息。

GetCourseInfo()，根据条件查询课程信息。

DelCourseInfo()，删除课程信息。

AddCourseApply()，用户申请添加课程信息。

AddComment()，对课程添加评论，对内容进行敏感词汇过滤。

DelComment()，撤销或删除课程评论。

GetCourseApplyList()，获取用户申请添加的课程信息列表。

ReviewPass()，通过用户的课程信息申请。

ReviewFailled()，审核不通过。

GetCommentList()，获取课程的评论信息列表。

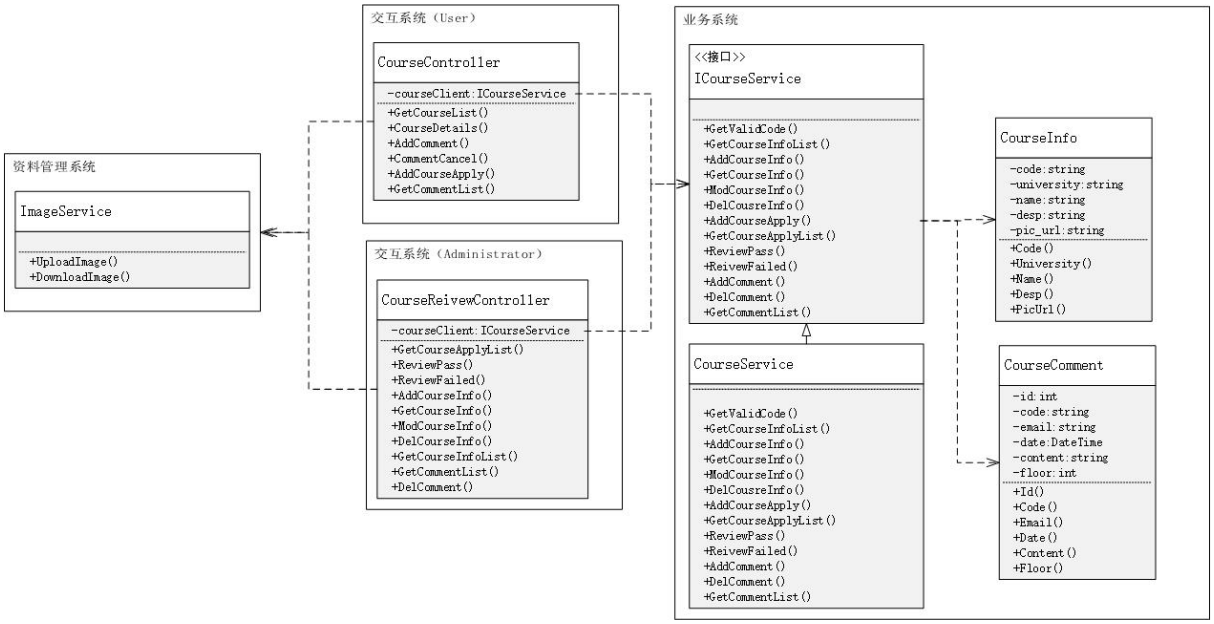

图 2.5 课程信息管理详细设计

普通用户可以通过 CourseController 查看课程信息，在课程信息下面发表评论，也可以申请添加系统暂时未导入的课程信息。其主要成员函数的定义如下：

GetCourseList()，获取课程信息列表。

CourseDetails()，查看课程详细信息。

AddComment()，添加课程评论。

CommentCancel()，撤销课程评论。

GetCommentList()，获取课程的评论信息。

AddCourseApply()，申请添加系统未导入的课程信息。

管理员通过 CourseReviewController 管理课程信息，其主要成员的定义如下：

GetCoruseApplyList()，获取用户申请添加的课程信息。

ReviewPass()，审核通过。

ReivewFailed()，审核不通过。

AddCourseInfo()，添加新的课程信息。

GetCourseInfo()，根据条件查询课程信息。

ModCourseInfo()，修改课程信息。

DelCourseInfo()，删除课程信息。

GetCourseInfoList()，获取课程信息列表。

GetCommentList()，获取课程的评论列表。

DelComment()，删除不当评论。

AddCourseInfo 和 ModCourseInfo 这两个方法会访问到资料管理系统的 ImageService 接口，通过 ImageService 的 UploadImage 获取图像的 URL，然后存储在业务服物系统的数据库中。

在交互系统于业务服务器通信的时候，WCF 定义的数据协议如上图所示。

2.2.3 教师信息模块详细设计

课程信息模块的类结构如图 2.6 所示。业务服务器提供 ITeacherService 接口供交互系统处理教师信息的相关数据。普通用户通过 TeacherController 浏览教师信息与发表评论，管理员通过 TeacherReviewController 管理教师信息。

图 2.6 教师信息管理详细设计

ITeacherService 接口定义了 WCF 通信的协议接口，其具体的操作由 TeacherService 类实现：

GetTeacherInfoList()，获取教师信息列表。

GetTeacherInfo()，根据条件查询教师信息。

ModTeacherInfo()，修改指定的教师的信息。

AddTeacherInfo，添加教师信息。

DelTeacherInfo()，删除教师信息。

GetCommentList()，获取教师的所有评论信息。

GetComment()，获取评论的详细信息。

DelComment()，删除不良评论。

前端页面的控制器类 TeacherController 通过对 ITeacherController 的方法进行一层封装，为普通用户提供教师信息的相关服务：

GetTeacherInfoList()，获取教师信息列表。

TeacherInfoDetails()，根据 Email 查询教师的详细信息与用户评论。

AddComment()，添加评论，评论内容有业务服务系统进行敏感词过滤。

CommentCancel()，撤销评论。

GetCommentList()，根据 Email 获取对教师的评论信息。

TeacherReviewController 封装了 ITeacherController 的部分接口，让管理员能够添加教、修改、查询与删除教师信息，同时对用户的评论进行管理，删除不合适的评论信息：

GetTeacherInfoList()，获取教师信息列表。

AddTeacherInfo()，添加新的教师信息。

ModTeacherInfo()，修改教师信息。

GetTeacherInfo()，根据 Email 查询教师信息。

DelTeacherInfo()，删除教师信息。

GetCommentList()，根据 Email 查询对教师的评论信息。

DelComment()，删除不良评论信息。

2.2.4 二手市场市场详细设计

新二手物品上架流程如图 2.7 所示，用户先进入添加商品页详细填写物品信息，然后提交等待管理员审核，通过后才能进入主页进行售卖，如果审核失败则需要重新提交新的申请。

用户可以进入个人中心查看购物记录以及管理上架物品的销售状态，详细设计如图 2.7 所示。

用户购买销售的商品的详细流程如图 2.8 所示，用户可以通过主页“我想要”进入最终确认页面，或者点击详细进入商品详细页面查看商品的详细信息与用户留言，然后确认时候购买。

IMarketService 是业务系统提供给交互系统处理交易信息的借口，它的具体操作由 MarketService 实现，它的核心操作定义如下：

UserAddGoods()，添加商品上架申请。

GetUserAddGoodsList()，获取用户的申请列表。

SetGoodsInfoStatus()，设置货物的销售状态。

GetGoodsInfo()，根据条件查询货物信息。

DelGoodsInfo()，删除物品消息。

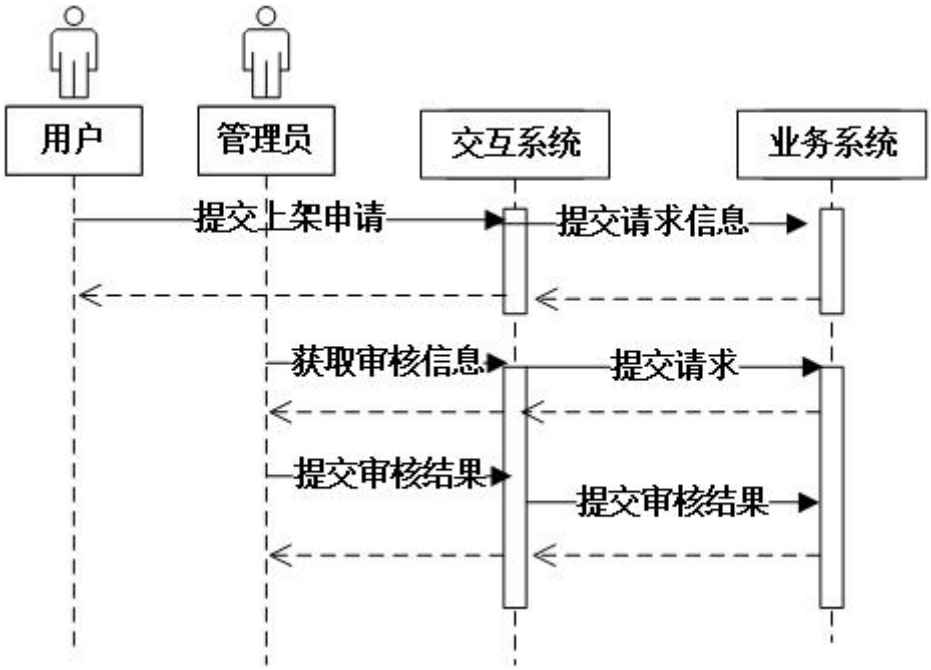

图 2.7 货物上架审核详细设计

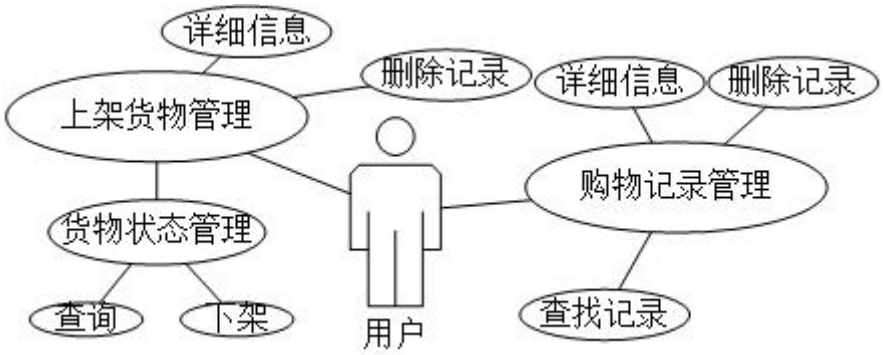

图 2.8 销售物品管理与购物记录管理详细设计

AddMsg()，添加留言。

DelMsg()，撤销留言。

用户通过 MarketController 进行商品信息的浏览，购物信息的管理，上架信息的管理以及商品交易，其核心操作定义如下：

UserAddGoods()，卖家提交商品上架申请。

GetUserAddGoodsList()，卖家获取申请列表，追踪申请进度。

GetGoodsInfoList()，获取正在销售的商品列表。

GetMsgOfGoods()，根据商品 Id 获取留言信息。

MakeComment()，添加留言。

CommentCancel()，撤销留言。

Buy()，进入订单页。

ConfirmBuy()，提交订单。

GetMyOnsaleGoodsInfoList()，获取卖家正在销售中的商品列表。

DelMyGoodsInfo()，删除卖家的销售记录。

DropOffMyGoods()，下架卖家正在销售的商品。

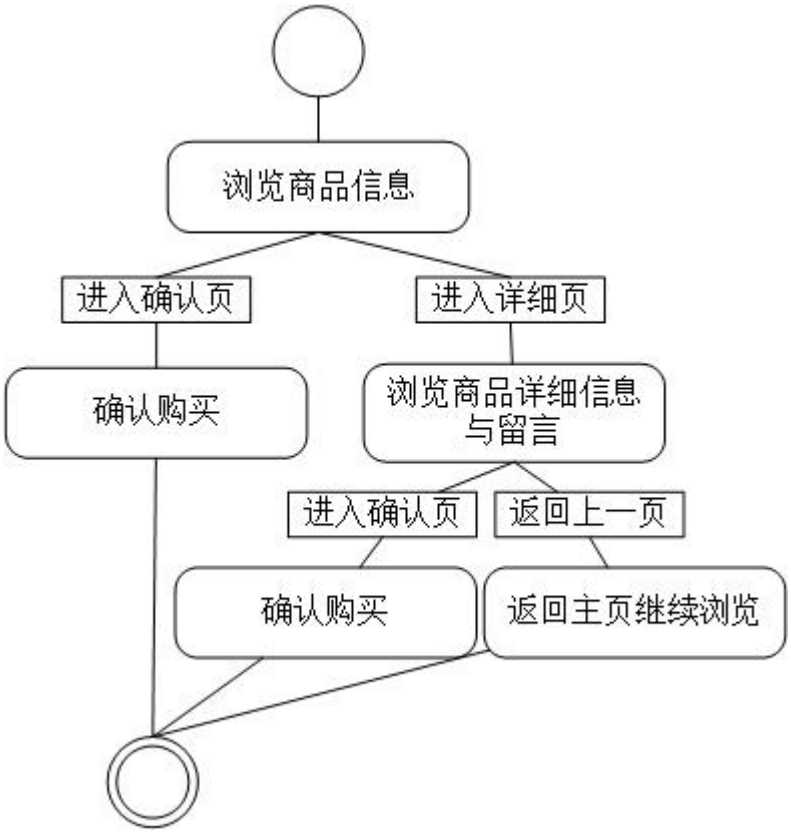

图 2.9 购物流程详细设计

管理员通过 MarketReviewController 对交易模块进行管理，其核心功能包括用户商品上架审核、货物的销售状态管理以及商品留言的敏感信息处理等：

ReviewPass()，审核通过。

ReviewFailed()，审核不通过。

GetUserAddGoodsList()，获取卖家商品上架申请列表。

SetGoodsStatus()，设置商品的销售状态。

DelMsg()，删除不良留言信息。

GetMsgList()，根据商品 Id 获取留言信息。

2.2.5 论坛博客区详细设计

用户发帖流程如图 2.10 所示，用户发帖不需要管理员进行审核，但帖子的内容会由系统自动进行敏感词过滤，管理员所需要做的事情就是在后台设置好敏感词汇。敏感词汇的设置在管理页的综合区，管理员可以从敏感词库中批量导入敏感词，也可以根据发

帖内容逐条添加。

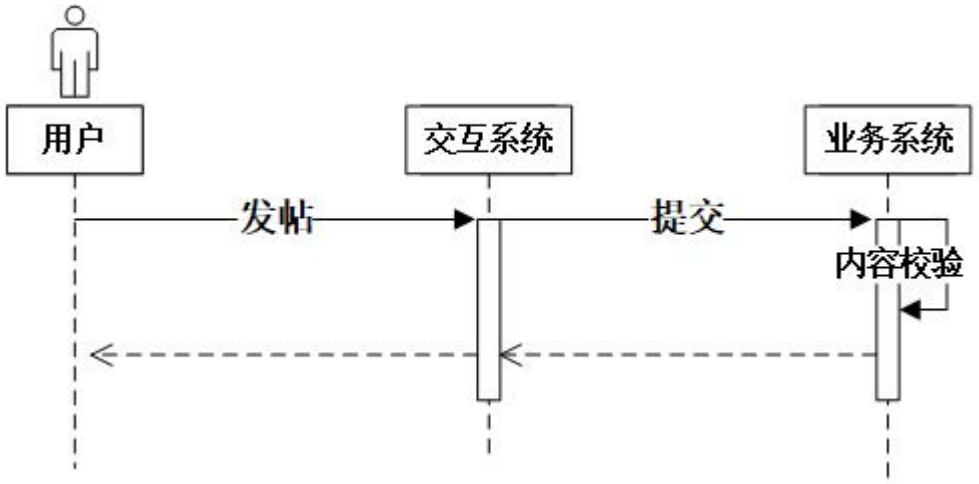

图 2.10 发帖流程详细设计

帖子管理流程如图 2.11 所示，用户可以设置帖子的状态，修改帖子信息等。

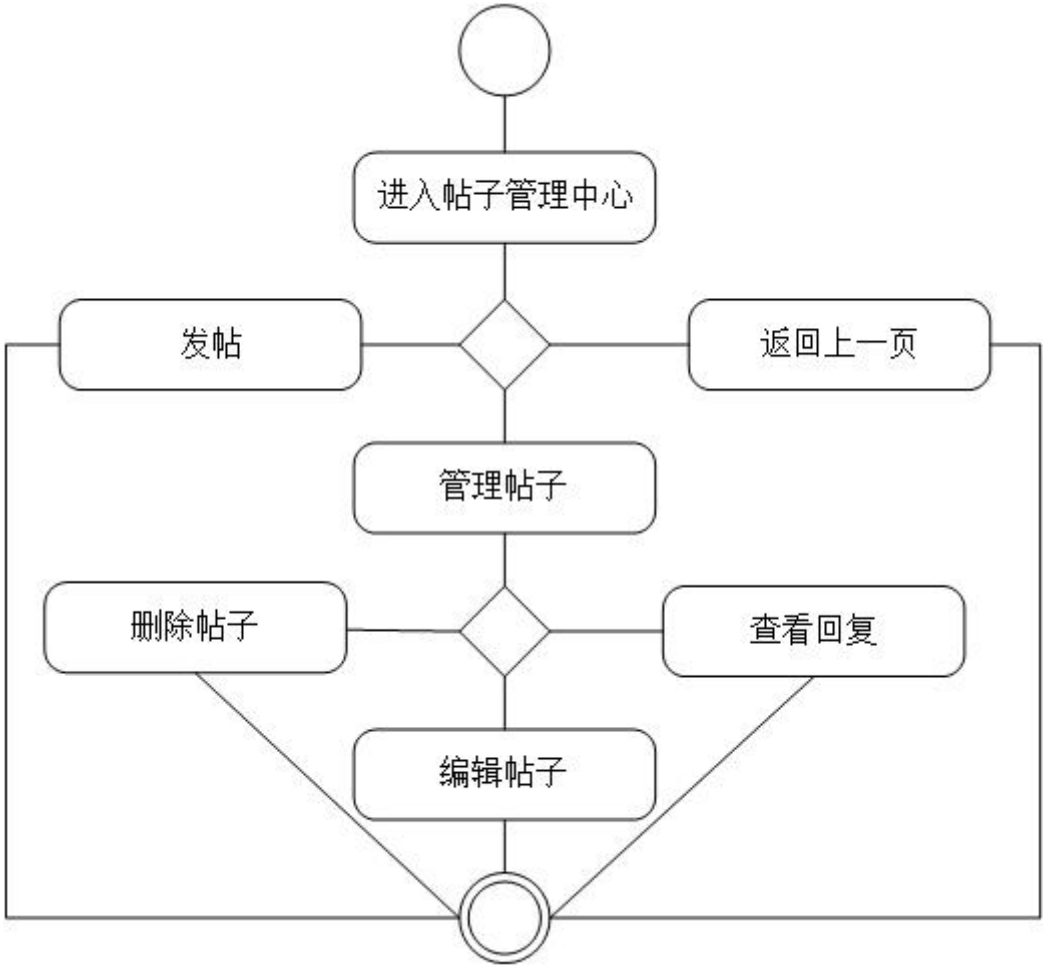

图 2.11 帖子管理详细设计

IForumService 是业务系统提供给交互系统处理帖子信息的借口，功能由 ForumService 实现，其核心操作如下所示：

AddPost()，发帖。

DelPost()，删除帖子。

GetPostList()，获取帖子信息列表。

ModPost()，修改帖子。

GetPost()，根据条件查询帖子。

ForumController 封装了业务系统提供的借口，为用户提供了发帖，回帖与管理等功能：

AddPost()，发帖。

DelPost()，删除帖子。

PostReply()，回帖。

DelPostReply()，撤销回复。

GetPostList()，获取帖子列表。

GetReplyList()，获取回帖列表。

2.2.6 管理员页面详细设计

如图 2.12 所示，管理员需要管理课程信息，教师信息，交易信息与其他综合信息。

图 2.12 管理员管理页详细设计

管理员需要进入专门的 Admin 系统，用管理员身份进行登录。系统默认的最高管理员是 Admin，具有系统的所有权限，可以由 Admin 用户创建新的管理员账号，并赋予其新的操作权限，但新的管理员账号的权限不会超过创建者的权限。

2.3 数据库设计

2.3.1 关系数据库 E-R 图

在设计数据库的时候先画图出 E-R 可以更直观的分析各个表之间的依赖关系，更容易定义表的主外键。图 2.13 是本系统的关键表的 E-R 图。

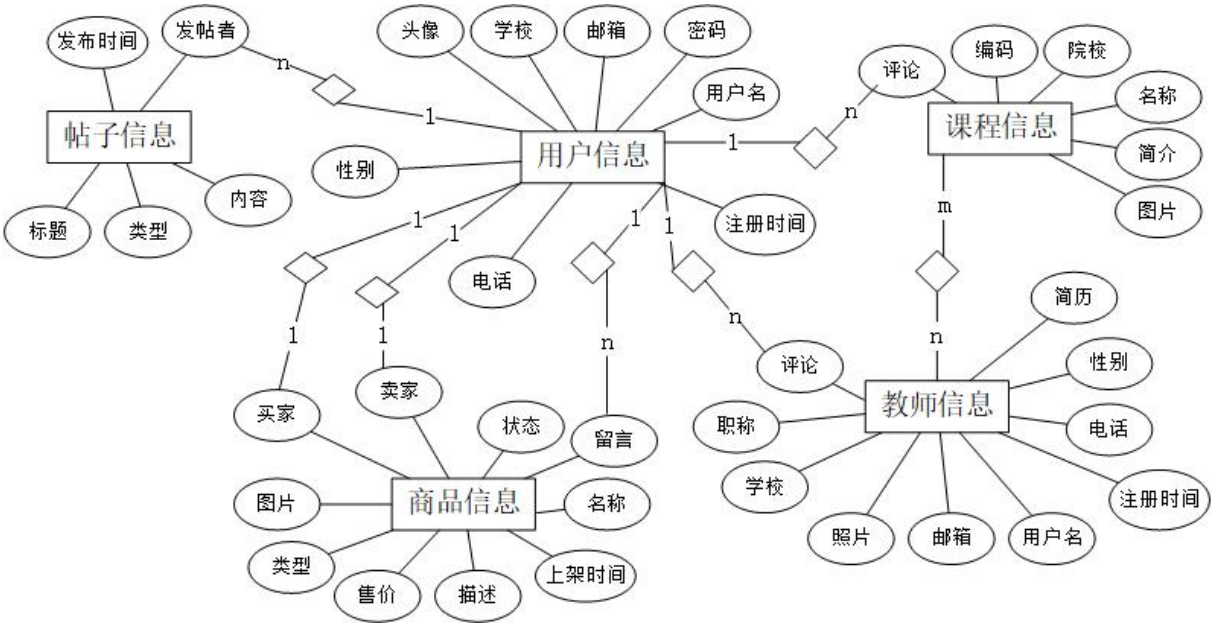

图 2.13 关系数据库的 E-R 图

通过 E-R 图，我们可一减少后期修改的次数，缩短开发周期。

2.3.2 数据库详细设计

用户信息表详细设计如表 2.1 所示，用户信息表 UserSets 用来存储注册用户的详细信息，该表的主键是 Email，外键 University 引用 cfg_University 表的 Id，Avatar 头像字段存储的是图想在资料服务器中的 URL，除了注册日期为 datetime 类型外，其他字段都是 nvarchar 类型。

表 2.1 UserSets

字段名	类型	备注
Email	nvarcahr(20)	邮箱，主键
Password	nvarchar(20)	密码
UserName	nvarchar(20)	用户名
RegisterDate	datetime	注册日期
Tel	nvarchar(20)	联系方式
Sex	nvarchar(10)	性别
Avatar	nvarchar(40)	头像
University	int	学校，外键

课程信息表详细设计如表 2.2 所示，课程信息表 CourseSets 用来存储有效的课程信息主键是 Code，外键 University 引用 cfg_University 的 Id，课程图片字段 PicUrl 存储的是存在资料服务中的图片的 URL。用户申请添加的课程信息存在 tmp_CourseSets 表中，评论信息存在 CourseCommentSets 表中。

教师信息表详细设计如表 2.3 所示，教师信息表 TeacherSets 用来存储教师信息，主

键是教师 Email，外键 University 引用 cfg_University 的 Id，外键 JobTitle 引用 cfg_JobTilte 的 Id，教师头像字段 Avatar 存储的是存储在资料服务器中的图像的 URL。

表 2.2 CourseSets

字段名	类型	备注
Code	nvarchar(20)	编码，主键
Name	nvarchar(20)	名称
Desp	nvarcha(2000)	简介
University	int	学校，外键
PicUrl	nvarchar(40)	课程图片

表 2.3 TeacherSets

字段名	类型	备注
Email	nvarchar(20)	教师邮箱，主键
Name	nvarchar(20)	教师姓名
Sex	nvarchar(20)	教师性别
Avatar	nvarchar(40)	教师头像
University	int	任职学校
RegisterDate	datetime	入职时间
JobTitle	int	职称
Tel	nvarchar(20)	联系方式
Resume	nvarchar(2000)	简历

帖子信息表的详细设计如表 2.4 所示，帖子信息表 ForumSets 存储所有用户的发帖信息。

表 2.4 ForumSets

字段名	类型	备注
Id	int	主键，自增长
Publisher	nvarchar(20)	发布者，外键
Title	nvarchar(20)	标题
PublishDate	datetime	发布时间
PostType	int	帖子类型，外键
Content	nvarchar(2000)	帖子内容
AllowResponse	int	允许发帖

Id 主键自增长，外键 Publisher 引用用户信息表 UserSets 的 Email，外键 PostType 引用 cfg_PostType 的 Id，AllowResponse 字段用于设定帖子是否允许回帖。

商品信息表的详细设计如表 2.5 所示，商品信息表 GoodsSets 存储所有商品的信息。

表 2.5 GoodsSets

字段名	类型	备注
Id	int	主键，自增长
Seller	nvarchar(20)	卖家，外键
Buyer	nvarchar(20)	买家，外键
Name	nvarchar(20)	商品名称
Desp	nvarchar(200)	商品描述
Price	double	商品售价
GoodsType	int	商品类型，外键
PicUrl	nvarchar(40)	商品图片
SaleStatus	int	销售状态，外键
OnsaleDate	datetime	上架时间
Comment	nvarchar(200)	卖家评论

主键 Id 自增长，外键 Seller 与 Buyer 都引用 UserSets 的 Email，外键 GoodsType 引用 cfg_GoodsType 的 Id，外键 SaleStatus 引用 cfg_SaleStatus 的 Id，PicUrl 字段存储的是存储在资料服务器上的图片的 URL。

2.4 本章小结

本章从系统结构到各个子系统的具体功能设计再到数据库的 E-R 设计与关键表的结构设计，详细介绍整个系统的设计方案。整个系统拆分为了 3 个子系统，其中具体介绍了资料管理系统，业务系统，管理员界面与用户界面的设计方案。

资料管理系统的主要功能是管理图像数据。

业务系统（后端服务系统）主要功能是访问数据库，将业务封装为一个统一的借口供交互系统调用。

交互系统又称为客户端系统，主要介绍了管理员客户端的设计方案与网页客户端的设计方案。

3 设计方案论证

3.1 系统设计中的关键问题与解决方案

3.1.1 HTTP 的安全性

交互系统采用的是 B/S 架构，通过 HTTP 协议进行数据的传输，为了防止 XSS（跨站脚本攻击）、伪造用户信息窃取数据、DDoS（分布式拒绝请求攻击）等威胁系统安全

性的操作发生，保证系统安全与稳性，本系统采取了诸多保证安心性的数据验证操作。

包含用户数据的请求操作都会使用 Post 方式进行提交。HTTP 的 Get 请求采用明文传输，会将参数直接附加在 URL 之后，例如使用百度搜索武汉科技大学的请求 URL 如下：https://www.baidu.com/s?wd=武汉科技大学，请求的内容是用户可见。而 HTTP 的 Post 请求会将请求数据进行一次编码加密，系统可以自定义加密方式来保证即使加密数据被不法者拦截了，也不会让真实数据泄露。

XSS（跨站脚本攻击）的原理是黑客将一段恶意代码注入到页面中，其他用户访问页面的时候就会执行这段恶意代码。常见的 XSS（跨站脚本攻击）危害有钓鱼网站，窃取用户账号，强制发送电子邮件，非法转移等。因为 XSS 攻击的方式的多样性，所以防御 XSS 攻击要根据具体的攻击策略采取对应的防御措施。本系统通过机器学习来分析异常的访问及其类型，然后交由对应的处理策略来进行处理。

DDoS（分布式拒绝服务攻击）的原理是利用病毒感染多台普通用户的客户端，然后在短时间内同时发起对网站的系统的大量请求，致使服务器无法及时响应而瘫痪，最后导致正常的请求也无法处理。本系统通过对增加一层 HTTP Agent 层来对 HTTP 进行转发，没当 Agent 系统收到一个 IP 的请求时，会根据其请求的内容，转发到对应的服务器进行处理。每个服务器配置一个 IP 请求队列，队列按照 IP 权重与等待时间进行从综合排序，如果同一个 IP 重复请求会导致权重自增。服务器会分局进程数从 IP 队列获取响应的请求进行处理。

3.1.2 服务器之间的通信

由于将整个系统拆分成了多个相互独立的子系统，每个系统运行在一台服务器上会产生一个独立的进程。为了保证各个进程的通信的稳定与安全，就不能采用 HTTP 协议，而直接使用 Socket 进行通讯又太多繁琐，因此本系统采用的解决方案是引入 RPC（Remote Procedure Call，远程过程调用）框架 WCF（Windows Communication Foundation， 微软通信基础类库）来减少重复的编码，从而专注于业务代码的编码。

3.1.3 图像的上传与下载

由于业务需要，本系统需要实现图像文件上传下载功能。本系统通过 JavaScript 读取图像数据，然后通过 HTTP 协议的 POST 请求传送到服务器，服务器将图像数据保存到硬盘中，然后返回客户顿一个 URL，客户端通过 FTP 协议来获取图像数据。

3.1.4 邮箱可用性验证

用户注册需要验证邮箱是否是用户本人的可用邮箱。本系统的解决方案是在用户注册的时候，通过邮件服务程序向注册邮箱发送一封验证邮件，如果用户通过验证，则可以进行二手商品的交易等操作。

验证邮件的内容中包含一个验证链接，只要用户点击这个链接就可以完成邮箱的可

用性验证操作。其实现原理是用户注册的时候会由系统随机生成一个 Token，然后系统会将这个邮箱地址与 Token 作为验证入口的参数生成一个 URL，只有用户点击这个 URL 就会跳转到验证页，其参数就是邮箱的地址与 Token。这个 Token 的过期时间是 5 分钟，如果用户超过 5 分钟未验证邮箱，则需要用户请求进行邮箱验证，然后系统重新发送一封新的验证邮件到用户邮箱中。

3.1.5 不良信息过滤

为了防止对教师与课程的恶意评论，以及用户发布不健康或不和谐的内容，本系统需要能够监控用户的评论内容以及发帖内容，并对恶意的评论，不健康与不和谐的内容进行过滤。

传统的解决方案是通过人工进行审核，然后又运营人员（管理员）删除不良信息。这样做不仅效率低下，而且陈本昂贵。为了减少人工的干预，本系统采用敏感词过滤与机器学习的方式来进行更加智能化的过滤。

敏感词过滤就是通过建立一个敏感词库，通过正则表达式匹配等手段对内容进行匹配扫描，如果内容中成功匹配到了敏感词就将其替换或删除。这种方式很暴力，但却只能删除词库中已经存在的敏感词，如果敏感词进行一下处理就可以绕过检测。例如有敏感词“测试”，如果用户使用“测，试”或者“CS”就无法将其过滤掉。

机器学习的方式就是通过机器学期手段分辨与处理用户发布的内容。这是一种更加高级的处理方式，通过足够的训练，它能做到语义分析，具有更加高效的过滤能力。

通过敏感词过滤作为第一次过滤手段，机器学习作为第二次手段，两者结合能实现更加高效的处理。

3.2 系统开发中的关键技术

整个系统的设计采用的是分布式架构，为了协同进程进行之间的数据处理，采用了 WCF 框架。交互系统采用 MVC 框架，保证了前后端的数据分离，减少了业务的耦合程度。业务系统采用的是三层架构，保证各个业务层之间的低耦合与每层内部的逻辑的高内聚，是的系统具有更好的扩展性。

3.3 本章小结

本章详细介绍了本系统设计中所遇到的各种问题以及对应的解决方案，以及整个系统所使用的框架技术等。

4 系统的具体实现

4.1 交互系统

4.1.1 课程信息具体实现

如图 4.1 所示为课程详细信息页，页面分为课程信息与用户评论两个部分。上面显示的是课程的详细信息，课程信息包含左侧的课程图片，与右侧的课程名称，授课教师以及课程简介等信息，右侧最后面是一个输入框与按钮，用来发表课程评论。下面是用户的评论列表，左侧是用户的头像，右侧是用户的昵称，发布时间与发布的内容，且当前用户的评论后面会出现一个红色的“撤回”按钮，用户可以点击按钮撤销当前评论。

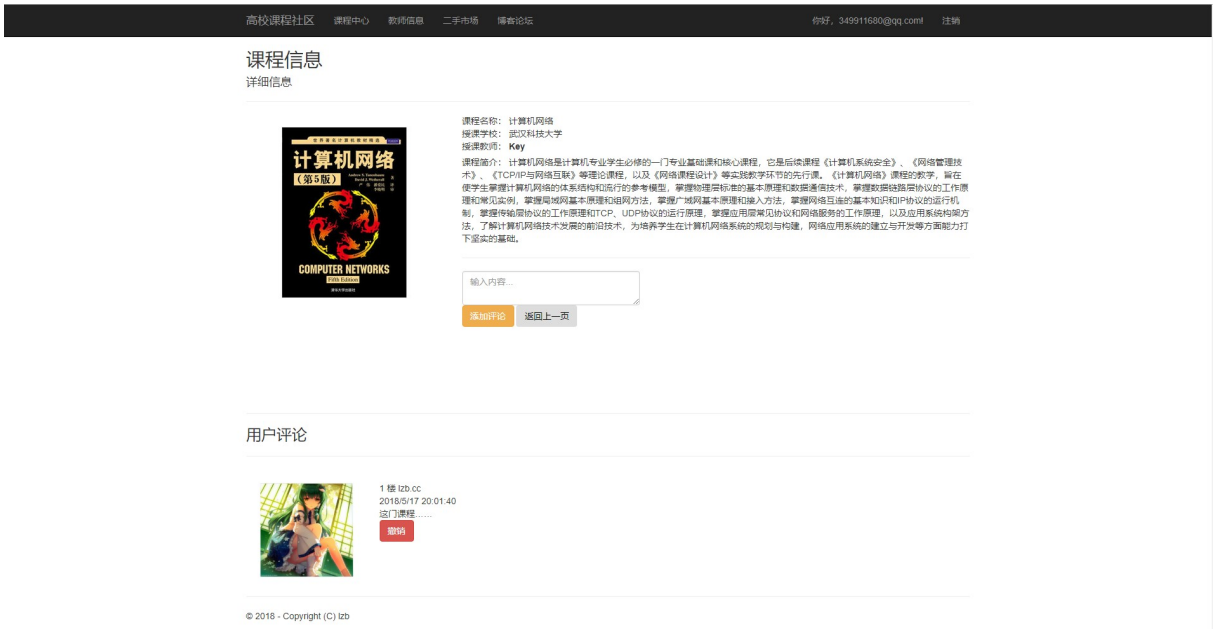

图 4.1 课程信息具体实现

4.1.2 教师信息具体实现

与课程信息页的布局相同，页面上面是课程教师的照片，所授课程，教师邮件以及教师的个人简历以及评论框和按钮。下面则是用户评论，包括用户头像，发布时间，发布内容，当前用户的评论后面会出现一个撤销按钮。

4.1.3 二手交易市场具体实现

交易中心主页如图 4.2 所示，页面中间是正在销售的商品信息，每件商品的信息包括商品的图片，价格，名称与描述，下面则是一个查看商品详细信息的按钮与进入订单页的按钮。上面有一个搜索模块，用户根据条件来搜索感兴趣的物品。

图 4.3 显示的是商品的额详细信息，用户可以添加留言，发表看法与询问卖家商品的具体信息，卖家可以回复用户的留言，或者添加补充说明等。用户可以撤销自己发布的留言信息，卖家留言会有一个粉红色“卖家”标记。

订单页（确认购买页）与商品的详细信息页面相同，不同的是没用用户留言。用户点击“我想要”后进入订单页，这里会显示商品与卖家的信息，如果确认购买，则系统发邮件通知卖家此商品有人购买，此商品状态修改为“销售完成”，且在主页中无法找到该商品信息。

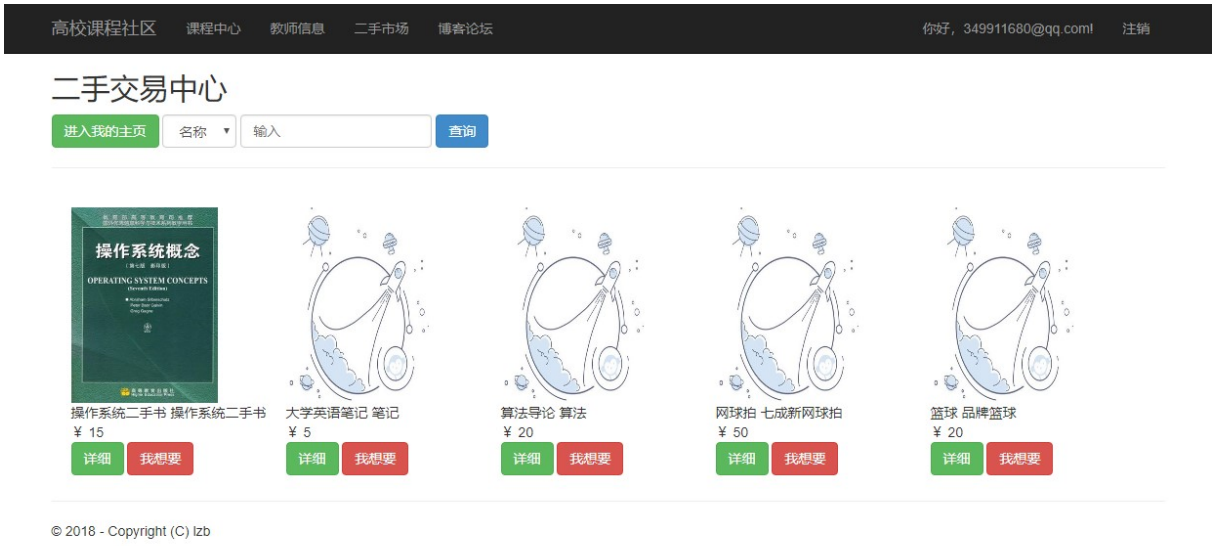

图 4.2 二手交易中心主页

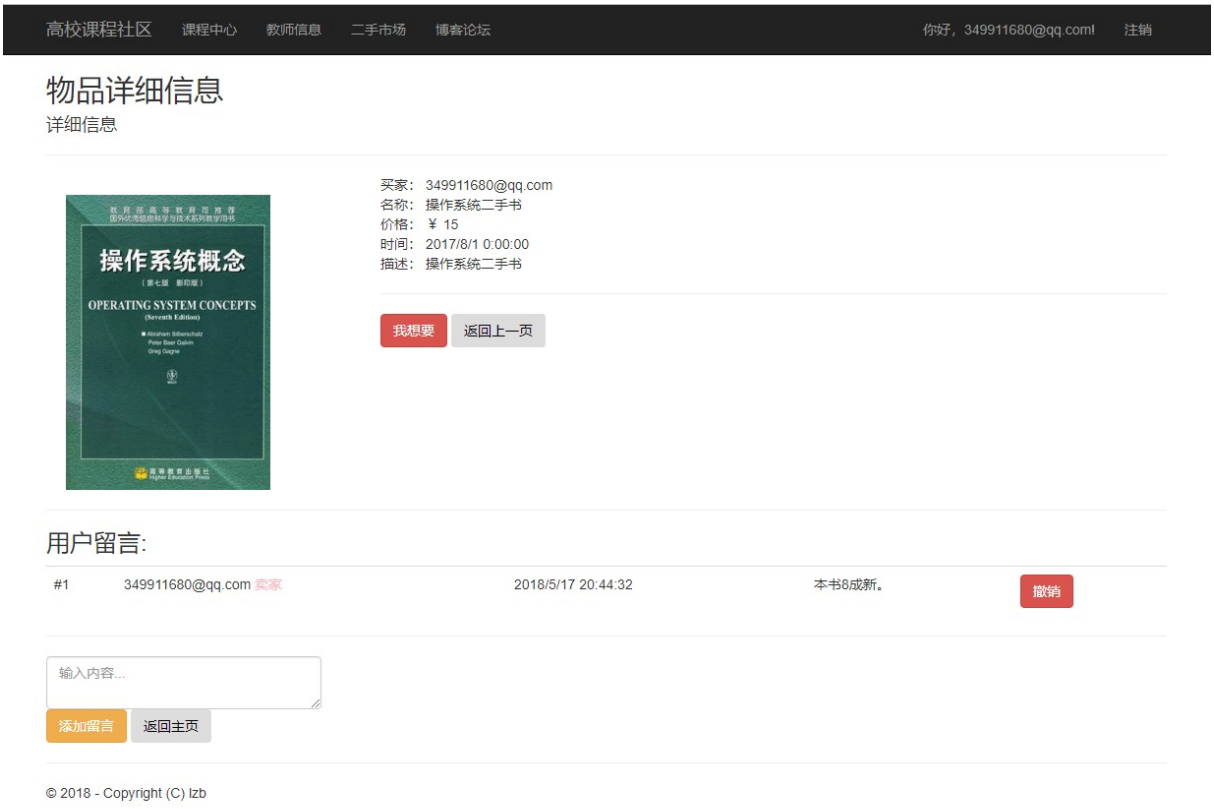

图 4.3 商品详细页

4.1.4 博客论坛去具体实现

帖子的详细信息如图 4.4 所示，帖子详细页包含帖子的发布者，发布日期，标题与内容。发帖者可以限制回复，当前回帖者可以撤销发出去的帖子信息。其他用户可以回复回帖。

图 4.4 帖子详情

4.2 后端服务系统

后台管理系统页面如图 4.5 所示，管理员登入后进入管理系统主页，管理员可以进入综合区创建新的管理员账号。

图 4.5 后台管理系统

4.3 本章小结

本章介绍了系统实现后得效果，展示了包括课程信息，教师信息，交易中心与博客论坛区的具体实现。。

5 系统的测试与调试

5.1 测试的目的及意义

测试是软件开发流程中必不可少的一个环节，通过测试可以查找出开发中产生的各种 Bug 与系统的潜在问题。通过测试与调试，能够有效的减少系统的 Bug 数量，提升系统的运行稳定率与用户体验。测试贯穿了整个系统的设计与开发，通过开发-测试-开发的流程，排除了系统的大部分 Bug。

5.2 测试的内容与方法

5.2.1 黑盒测试

黑盒测试也可称为功能测试，忽略函数或模块的具体实现，通过数据用例来测试函数是否达到了预期功能。具体操作方法是就是编写各种边界值用例，调用函数或模块来运行这些用例，根据用例的通过率来判定函数的功能是否达到了预期。

5.2.2 白盒测试

白盒测试又可以成为结构性测试，在已经知晓程序的内部结构的基础上，检查各个每个流程是否达到了预期。具体操作方式是根据各个流程编写测试用例来判定所有流程的结果是否达到预期，根据测试用例的通过率来判定结果是否达到预期。

5.2.3 单元测试

单元测试与黑盒测试类似，不同之处在于他的测试粒度更细，可以精确到函数内部的一条语句。

5.2.4 集成测试

集成测试主要是针对模块的接口进行测试，通过调用一个模块的借口来运行各个测试用例，根据测试用例的通过率来判定模块的功能是否达到预期。

5.2.5 系统测试

系统测试主要包括恢复测试，压力测试与安全测试。其主要目的是为了检测系统是否具有良好的容错处理，安全检查与抗压能力。

5.3 测试的结果

高校课程社区系统经过测试与调试后能够按照需求正常运行，基本没有错误，同时对用户操作时可能出现的错误都已进行了预判处理，提高了系统的健壮性，作为一个较为简易的系统模型，满足用户的各项需求，成功的完成了课题的要求。

5.4 本章小结

本章介绍了黑盒测试、白盒测试、单元测试、集成测试与系统测试，以及本系统通过这些测试方法测试的结果。

结论

通过 3 个月的设计开发与调试，高校课程社区系统成功运行并达到了设计预期。本系统的主要功能是为教师提供一个评教环境，学生可以对课程内容与教师的授课水平进行评论，教师可以跟进反馈。同时系统提供了一个二手物品交易平台，可以让师生能够将废物进行二次利用，不仅减少了资源的浪费，同时还能帮助其他师生以更小的代价获取所需资源，可谓是一举多得。

本系统根据需求拆分为了多个子系统，包括普通用户访问的网页系统与手机 APP，管理用于管理的后台管理系统，为其他系统提供服务的业务系统于资料管理系统，各个系统之间通过 HTTP 协议与 WCF 框架进行数据传输。系统设计与实现运用了大量的面向对象设计思想，不同模块之间通过接口进行通信，极大的满足了高内聚低耦合的软件设计思想，使得系统具有较高的稳健性、可维护性与可扩展性。

虽然高校课程社区系统能够满足的了设计预期，但因为开发周期有限以及个人能力不足的问题，系统依旧存在许多问题。比如：

完成度：为了满足用户在不同场景下可以浏览社区内容，交互系统应当包含 PC 客户端，网页系统，手机 APP 以及微信公总号等各个模块，但由于时间有限目前只实现了网页系统。

界面设计：现在的界面设计基本采用.NET Framework 的原始界面，虽然进行了一些修改，但依旧不够美观。

安全性：因为 HTTP 的诸多安全性问题，现在主流系统都开始采用 HTTPS 协议，本系统的也会朝着这方面改进。

数据库的设计：目前数据库的设计依旧不够完善，很多实体的属性没有考虑全面，存在诸多不足的地方。

根据现在系统还存在的各种问题，后期将针对性的完善，并不断优化设计，为用户提供更优质的服务。

参考文献

Cibraro P, Claeys K, Cozzolino F. Professional WCF 4: Windows Communication Foundation with .NET 4[M]. USA: Wrox, 2010.

致谢

时光匆匆而过，四年的大学生生活即将结束。在这短短的四年里，我学到了很多知识，收获颇多，不仅获取了技术上的进步，也获得了很多同伴的牢固友谊。在此我由衷地感谢每一位曾帮助和关心过我的人，是你们的支持和鼓励让我在学习和生活中有了动力和信念，并坚持不懈的奋斗。因此，在论文即将完成之际，谨向所有给予我指导和帮助的人献上我深深的感谢。

在此，我特别感谢我的毕设导师付晓薇教授，感谢她对我在毕业设计期间的诸多帮助。从论文选题到最后论文完成，都离不开付老师的悉心指导。付老师学识渊博，学术严谨，细心和耐心。感谢我的室友们，在毕业设计期间，我们共同勉励和相互支持，互相修改论文。感谢我的家人和亲朋好友，在我多年求学过程中始终给予的不断的鼓励与支持，让我有了向前的动力，以后我仍会继续努力，不断进取。

最后，感谢在百忙之中抽出宝贵时间审阅这篇论文并提出宝贵意见的评审老师们！

附录 数据库 SQL 语句

```
use master;
go
--如果数据库存在，删除重建
if exists(select * from dbo.sysdatabases where name = 'CmtyDB')
drop database CmtyDB;

create database CmtyDB
go

use CmtyDB;
go
--创建表
if OBJECT_ID('cfg_Universities') is not null drop table cfg_Universities;
create table cfg_Universities
(
    Id      int identity,
    name    nvarchar(20),
    primary key(Id)
);

if OBJECT_ID('cfg_AdminType') is not null drop table cfg_AdminType;
create table cfg_AdminType
(
    Id      int identity,
    desp    nvarchar(10),
    primary key (Id)
);

if OBJECT_ID('cfg_ReviewStatus') is not null drop table cfg_ReviewStatus;
create table cfg_ReviewStatus
(
    Id      int identity,
    Desp    nvarchar(20),
```

```
    primary key (Id)
);


if OBJECT_ID('cfg_JobTitle') is not null drop table cfg_JobTitle;
create table cfg_JobTitle
(
    Id     int identity,
    Desp nvarchar(20),
    primary key (Id)
);

if OBJECT_ID('UserSets') is not null drop table UserSets;
create table UserSets
(
    Email          nvarchar(20),
    Pwd               nvarchar(16),
    uName          nvarchar(20),
    rDate          datetime,
    Tel            nvarchar(11),
    university int,
    primary key (Email),
    constraint fk_university_us foreign key (university) references cfg_Universities (Id),
);

if OBJECT_ID('TeacherSets') is not null drop table TeacherSets;
create table TeacherSets
(
    Email          nvarchar(20),
    uName          nvarchar(20),
    rDate          datetime,
    Sex            nvarchar(10),
    Tel            nvarchar(11),
    university int,
    jTitle         int,
```

```
    Desp          nvarchar(2000),
    primary key (Email),
    constraint fk_university_ts foreign key (university) references cfg_Universities (Id),
    constraint fk_jTitle_ts foreign key (jTitle) references cfg_JobTitle (Id)
);

if OBJECT_ID('CourseSets') is not null drop table CourseSets;
create table CourseSets
(
     Id               nvarchar(20),
     university    int,
     name             nvarchar(20),
    desp            nvarchar(2000),
    pic_url         nvarchar(200),
     primary key (Id),
     constraint fk_university_cs foreign key (university) references cfg_Universities (Id)
);


if OBJECT_ID('TeacherCourseSets') is not null drop table TeacherCourseSets;
create table TeacherCourseSets
(
     Email           nvarchar(20),
     CourseId        nvarchar(20),
     primary key (Email, CourseId),
     constraint fk_Email_tcs foreign key (Email) references TeacherSets (Email),
     constraint fk_CourseId_tcs foreign key (CourseId) references CourseSets (Id)
);

if OBJECT_ID('BookSets') is not null drop table BookSets;
create table BookSets
(
     Id               int,
     name             nvarchar(50),
     author           nvarchar(50),
```

```
    pic             nvarchar(50),
    desp            nvarchar(2000),
    primary key (Id)
);

if OBJECT_ID('ExtraUserInfo') is not null drop table ExtraUserInfo;
create table ExtraUserInfo
(
    Email       nvarchar(20),
    Sex         nvarchar(10),
    Nick        nvarchar(20),
    Hobby       nvarchar(30),
    Avatar      nvarchar(120),
    primary key (Email),
    constraint fk_email_eui foreign key (Email) references UserSets (Email)
);

if OBJECT_ID('EmailCheckSets') is not null drop table EmailCheckSets
create table EmailCheckSets
(
    Email           nvarchar(20),
    Token           int,
    CheckStatus int,
    primary key (Email)
);

if OBJECT_ID('AdminUsers') is not null drop table AdminUsers;
create table AdminUsers
(
    Email       nvarchar(20),
    Pwd             nvarchar(20),
    authority   int,
    primary key (Email),
    constraint fk_authority_au foreign key (authority) references cfg_AdminType(Id)
);
```

```
if OBJECT_ID('tmp_CourseSets') is not null drop table tmp_CourseSets;
create table tmp_CourseSets
(
    Code            nvarchar(20),
    CommitUser      nvarchar(20),
    CommitDate      datetime,
    ReviewStatus int,
     university     int,
     name               nvarchar(20),
    desp            nvarchar(2000),
    pic_url         nvarchar(200),
    primary key (Code),
    constraint fk_CommitUser_tcs foreign key (CommitUser) references UserSets (Email),
    constraint fk_ReviewStatus_tcs foreign key (ReviewStatus) references cfg_ReviewStatus (Id)
);

if OBJECT_ID('CourseCommentSets') is not null drop table CourseCommentSets;
create table CourseCommentSets
(
    Id              int   identity,
    Code            nvarchar(20),
    Email           nvarchar(20),
    cDate           datetime,
    Content         nvarchar(200),
    CmtFloor        int,
    primary key (Id),
    constraint fk_Code_ccs foreign key (Code) references CourseSets (Id),
    constraint fk_Email_ccs foreign key (Email) references UserSets (Email)
);

if OBJECT_ID('TeacherCommentSets') is not null drop table TeacherCommentSets;
create table TeacherCommentSets
```

```
(
    Id              int    identity,
    T_Id            nvarchar(20),
    Email           nvarchar(20),
    cDate           datetime,
    Content         nvarchar(200),
    CmtFloor        int,
    primary key (Id),
    constraint fk_Code_ttcs foreign key (T_Id) references TeacherSets (Email),
    constraint fk_Email_ttcs foreign key (Email) references UserSets (Email)
);


if OBJECT_ID('cfg_SaleStatus') is not null drop table cfg_SaleStatus
create table cfg_SaleStatus
(
    Id    int identity,
    Desp    nvarchar(50),
    primary key (Id)
);

if OBJECT_ID('cfg_GoodsType') is not null drop table cfg_GoodsType
create table cfg_GoodsType
(
    Id          int identity,
    Desp        nvarchar(50),
    primary key(Id)
);

if OBJECT_ID('GoodsSets') is not null drop table GoodsSets
create table GoodsSets
(
    Id              int identity,
    Seller          nvarchar(20),
    Name            nvarchar(30),
```

```
    Mny         int,
    Pic_Url     nvarchar(200),
    Desp        nvarchar(200),
    PubDate     datetime,
    SStatus      int,
    Buyer       nvarchar(20),
    Comment         nvarchar(200),
    GType       int,
    primary key (Id),
    constraint fk_Seller_gs     foreign key (Seller)     references UserSets (Email),
    constraint fk_SStatus_gs foreign key (SStatus) references cfg_SaleStatus (Id),
    constraint fk_GType_gs      foreign key (GType) references cfg_GoodsType (Id)
);

if OBJECT_ID('LeaveMsg') is not null drop table LeaveMsg
create table LeaveMsg
(
    Id      int identity,
    Gid     int,
    Email   nvarchar(20),
    PubDatedatetime,
    Content nvarchar(200),
    LmFloor int,
    primary key (Id),
    constraint fk_Gid_lm foreign key (Gid) references GoodsSets (Id),
    constraint fk_Email   foreign key (Email) references UserSets (Email)
);

if OBJECT_ID('cfg_PostType') is not null drop table cfg_PostType
create table cfg_PostType
(
    Id   int identity,
    Desp nvarchar(20),
    primary key (Id)
);
```

```
if OBJECT_ID('PostMsg') is not null drop table PostMsg
create table PostMsg
(
    Id              int identity,
    Email           nvarchar(20),
    Title           nvarchar(20),
    PType           int,
    Content         nvarchar(2000),
    PDate           datetime,
    NoComments int,
    primary key (Id),
    constraint fk_Email_pm foreign key(Email) references UserSets(Email),
    constraint fk_PType_pm foreign key(PType) references cfg_PostType(Id)
);

if OBJECT_ID('PostReply') is not null drop table PostReply
create table PostReply
(
    Id          int identity,
    Email       nvarchar(20),
    Reply       int,
    Content nvarchar(2000),
    RDate       datetime,
    primary key (Id),
    constraint fk_Email_pr foreign key (Email) references UserSets(Email),
    constraint fk_Reply_pr foreign key (Reply) references PostMsg(Id)
);

if OBJECT_ID('PostReplyMsg') is not null drop table PostReplyMsg
create table PostReplyMsg
(
    Id          int identity,
    Email       nvarchar(20),
    Reply       int,
```

```
    Content nvarchar(2000),
    RDate    datetime,
    primary key(Id),
    constraint fk_Reply_prm foreign key (Reply) references PostReply(Id),
    constraint fk_Email_prm foreign key (Email) references UserSets(Email)
);

create table FilterStrings
(
    Name      nvarchar(20),
    primary key (Name)
);
go


use CmtyDB;
go
--cfg_Universities--
insert into cfg_Universities values (N'北京大学')
insert into cfg_Universities values (N'清华大学')
insert into cfg_Universities values (N'武汉大学')
insert into cfg_Universities values (N'华中科技大学')
insert into cfg_Universities values (N'武汉科技大学')

--cfg_UserType--
insert into cfg_AdminType values (N'超级管理员')
insert into cfg_AdminType values (N'普通管理员')
insert into cfg_AdminType values (N'操作员')

--AdminUsers--
insert into AdminUsers values (N'349911680@qq.com', N'123456', 1)


--cfg_ReviewStatus--
insert into cfg_ReviewStatus values (N'待审核')
```

```
insert into cfg_ReviewStatus values (N'审核中')
insert into cfg_ReviewStatus values (N'审核通过')
insert into cfg_ReviewStatus values (N'审核不通过')

--cfg_JobTitle--
insert into cfg_JobTitle values (N'助教')
insert into cfg_JobTitle values (N'讲师')
insert into cfg_JobTitle values (N'副教授')
insert into cfg_JobTitle values (N'教授')
insert into cfg_JobTitle values (N'院士')

--cfg_SaleStatus--
insert into cfg_SaleStatus values(N'待审核')
insert into cfg_SaleStatus values(N'审核中')
insert into cfg_SaleStatus values(N'审核不通过')
insert into cfg_SaleStatus values(N'销售中')
insert into cfg_SaleStatus values(N'等待收货')
insert into cfg_SaleStatus values(N'销售完成')
insert into cfg_SaleStatus values(N'售后/退货')
insert into cfg_SaleStatus values(N'退货完成')
insert into cfg_SaleStatus values(N'下架')
insert into cfg_SaleStatus values(N'删除记录')

--cfg_GooldsType--
insert into cfg_GoodsType values(N'其他')
insert into cfg_GoodsType values(N'课程资料')
insert into cfg_GoodsType values(N'二手书')
insert into cfg_GoodsType values(N'笔记')
insert into cfg_GoodsType values(N'二手电脑')
insert into cfg_GoodsType values(N'运动器材')

--cfg_PostType
insert into cfg_PostType values(N'技术问答')
insert into cfg_PostType values(N'技术分享')
insert into cfg_PostType values(N'IT 大杂烩')
```

```
insert into cfg_PostType values(N'职业生涯')
insert into cfg_PostType values(N'学习感悟')
insert into cfg_PostType values(N'求助专区')
insert into cfg_PostType values(N'生活琐事')
insert into cfg_PostType values(N'兴趣爱好')
insert into cfg_PostType values(N'升学指导')
insert into cfg_PostType values(N'博客专区')

--FilterStrings
insert into FilterStrings values(N'测试 1')
insert into FilterStrings values(N'测试 2')
insert into FilterStrings values(N'测试 3')
insert into FilterStrings values(N'测试 4')
go
use master;
```