

基于 Vue 框架的图书商城管理系统

开发与设计

**Vue Framework-based Bookstore
Management System Development and
Design**

专 业： 计算机科学与技术（信息处理）

姓 名： 黎泽成

指 导 教 师： 杨光磊

申请学位级别： 学 士

论文提交日期： 2021 年 5 月 20 日

学位授予单位： 天津科技大学

摘 要

随着互联网的迅速发展,电子商务已被广大的互联网用户所接受,网上书店系统作为其中的一部分也有了迅速的发展。随着全球经济一体化的逐步深入,网上书店已是现代传统书店必不可少的经营策略。目前,网上书店在国际互连网上可以实现的商务功能已经多样化,几乎以往传统书店功能都可以在互联网上进行电子化的高效运转。图书网上交易是图书销售领域未来发展之必然,也是出版发行行业摆脱困境,建立新的流通渠道,繁荣出版,繁荣市场的必由之路。由此可见,开发一套在线图书交易系统就有了重要的学术和商业价值。

本系统是一个线上购书系统。经过可行性分析及需求分析,最终决定,以运用 Vs Code 作为本系统前端页面开发工具,以 PyCharm 作为后端开发工具,开发语言选用 Vue.js 以及 Python,以功能强大的 MySQL 作为后台数据库。本系统包含用户端和管理员端两部分。在用户端,用户拥有查看附近店铺,查看商家页面详情,添加购物车,提交订单等功能。在管理员端的操作,主要包括店铺管理,书籍管理等。

本系统的推出,方便了读者购买书籍同时也让读者可以更快找到自己感兴趣的书籍,对自己想要的书籍在网上轻轻松松就能购得。避免了去线下书店寻找书籍花费过长时间的情况。

关键词: 线上购书系统; MySQL; Python; Vue.js

ABSTRACT

With the rapid development of the Internet, e-commerce has been accepted by the majority of Internet users, and the online bookstore system as part of it has also seen rapid development. With the gradual integration of the global economy, online bookstores have become an essential business strategy for modern traditional bookstores. At present, the business functions that can be realized by online bookstores on the international interconnection network have diversified, and almost all of the traditional bookstore functions of the past can be run electronically and efficiently on the Internet. Online book trading is the inevitable future development of the book sales field, and it is also the inevitable way for the publishing and development industry to get rid of the difficulties, establish new distribution channels, prosperous publishing and prosperous market. Thus, the development of an online book trading system has an important academic and commercial value.

This system is an online book purchasing system. After feasibility and requirement analysis, we finally decided to use Vs Code as the front-end page development tool, PyCharm as the back-end development tool, Vue.js and Python as the development language, and powerful MySQL as the back-end database. The system contains two parts: user side and administrator side. In the user side, users can view nearby stores, view merchant page details, add shopping cart, submit orders and other functions. In the administrator side, the operation mainly includes store management, book management, etc.

The launch of this system makes it convenient for readers to buy books and also allows them to find the books they are interested in more quickly and to purchase the books they want online easily. It avoids the need to go to offline bookstores and spend too much time looking for books.

Keywords: Online book buying system; MySQL; Python; Vue.js

目 录

第一章 绪论.....	1
第一节 研究背景.....	1
第二节 研究目的与意义.....	2
第三节 研究现状与未来分析.....	3
第二章 图书商城系统相关理论与技术.....	4
第一节 Vue 框架介绍.....	4
第二节 MVVM 架构模式介绍.....	5
第三节 前后端分离开发模式介绍.....	6
第四节 开发工具介绍.....	8
第五节 本章小结.....	8
第三章 图书商城系统的需求分析.....	9
第一节 业务需求分析.....	9
第二节 可行性分析.....	9
第三节 功能需求分析.....	10
第四节 系统开发步骤.....	11
第五节 本章小结.....	11
第四章 图书商城系统设计.....	12
第一节 系统的设计概要.....	12
第二节 系统详细功能设计.....	13
第三节 数据库表的设计.....	15

第四节 连接数据库的设计与实现.....	17
第五节 本章小结.....	18
第五章 系统功能实现.....	19
第一节 用户端实现.....	19
第二节 管理员端实现.....	32
第三节 本章小结.....	38
第六章 系统的测试.....	39
第一节 系统测试的目的.....	39
第二节 系统测试的方法.....	39
第三节 系统测试的过程.....	39
第四节 系统测试的总结.....	44
第五节 本章小结.....	44
第七章 总结与展望.....	45
第一节 总结.....	45
第二节 展望.....	46
参考文献.....	47
致 谢.....	48

第一章 绪论

第一节 研究背景

互联网的出现最开始是在美国,但是到了如今,随着互联网技术的不断发展与完善,全世界的各个地区的国家都具备了互联网条件,互联网已经在全世界范围内覆盖了。随着用户的不断增加,其规模迅速扩大,它的领域也走向多元化,除了原先的科学技术和教育外,互联网已进入了文化、经济、政治、新闻、体育、娱乐、商业和服务业。^[1]因此我们能够预见,在未来互联网技术一定可以为我们提供一种全新的生活方式和工作方式。

随着计算机科学技术的不断发展,到了 20 世纪末期,互联网中已经在广泛地应用网络技术和数据库技术了,这样一来,更加方便了广大的网络用户,提升了用户体验。电子商务就是通过电信网络进行的生产、经营、销售和流通等活动,简单地讲,电子商务是指利用电信网络进行的商务活动。^[2]正是因为这种形势的发展,在网上兴起了很多的利用电子商务开展的网络贸易,从而让传统的经济形势开始发生了相应的转变。也因此,网上书店的研究也是在顺应这种趋势的发展,来进一步方便人们的生活。

网上书店的兴起,实际上是 Internet 电子商务在图书业发展的必然结果,它使传统的图书销售业发生了根本性的变革,同时也使传统的购书方式发生了根本性的变化。^[3]由于在国外有着更为优越的技术和完备的基础设施,同时还有着良好的消费群体,美国的网上书店便得到了充分的发展,有着呈现上升的趋势。这也使得世界上销售量最大的书店诞生了-亚马逊书店。亚马逊经历了几个发展阶段:先是作为一个网络书店,然后是一个网络市场,现在是一个应用服务提供商(ASP)。^[4]亚马逊书店不需要很多员工,但是该书店员工的人均销售额确非常的高,比拥有庞大员工人数的 Barnes & Nobel 公司的人均销售额高了 3 倍以上。

电子商务在这一切事情的实现中扮演着十分重要的角色。最早叩开中国大陆网上书店大门的商家,是美国的“亚马逊”,一经杀入中国市场,便一发不可收拾。^[5]因此我国很多企业也开始建立起自己的网站,不断开拓新的网上商机。这样,随着传统形势的转型加之应用电子商务,我国的电子商务领域会得到快速的发展。

但即使是这样,目前我国在这方面还是有很多的缺陷,不管是实力还是规模都与外国有着不小的差距,与预期的效果还是有着不小的差距,成交量有限,访问量也不多,很多的网上广告也还并没有得到广大商家的认可。

第二节 研究目的与意义

在图书销售这块领域所应用的一种动态网站称为网上书店。由于近些年来图书销售方面广泛应用电子商务技术，网上书店的规模一下子就变得壮大起来了，积累的用户量也在日益增长。

与传统的书店相比，网上书店既可以避免书籍订货的局限和盲目，又可以克服看样订货投入大，费用高，管理难的不足，而且网上选择范围广又能直观看样，可浏览内容，可随时添订，结算及时，快捷方便，周转高速。^[6]如果选择其他的交易方式是达不到这样子的效果的。

目前来看，有很大一部分的网上用户都习惯于在线购物的方式，而且数量有着不断上涨的趋势。相比于其他的用品，书籍类物品价格标准化，购买过程中风险极低，因此在线购买书籍这种交易方式是更被广大用户喜欢的。根据 CNNIC 于 2003 年 1 月的调查，高达 67.7% 的我国网民在过去一年内在网上购买过书刊^[7]。

网上书店通过不断的发展和完善，如今已成为同行中的中建力量了。网上书店迅速崛起和普及，成为一只新秀，开始慢慢鲸吞着图书市场这块大的商业蛋糕，目前，网上书店行业的发展，已经进入了信息化同步时代，竞争也呈现白热化状态。谁能够看准机遇、把握时机、抢占先机，谁就有可能成为最后的赢家。^[5]

我国的网上书店起源于 20 世纪 90 年代中期，但发展到现在还处于“热而不火”的局面。^[8]在当前环境下，我国依然存在着众多的习惯于传统经营模式的小型书店，它们所面临的生存形势非常严峻，一方面是网上的零售市场，另一方面则是新华渠道的大型书店。所以，中小型书店想要继续生存下去，必须要开始着手网上经营模式的应用，进一步开拓自己的市场范围。

随着信息化社会的发展以及互联网技术的不断进步，网络售书成为图书销售的发展趋势和方向，如何改善我国网上书店行业的经营现状、找出适合我国网上书店发展的对策，对我国的网上书店摆脱困境、获得新发展有极其重要的意义。^[8]因此本设计旨在推动我国网上书店的发展，让中小型书店获得更大范围的发展空间。

在对网上书店系统开发之前需要思考的问题是怎么让该经营模式的投入能够得到最大效益的回报。因此就必须对与之相关的一系列问题进行科学分析，这包括了需求分析、概要分析、系统功能模块设计等等。在完成这些分析之后，就可以开始布局具体的实施方案了。

第三节 研究现状与未来分析

在美国首先出现了网上书店,当前网上书店在经济发达的地区如欧美地区的分布比较密集,这些地区都具有起步较早、发展迅速、规模庞大等特点。而我国是在二十世纪末期才开始出现了网上书店,起步晚是最明显的特点,但是我国网上书店的起点相对比较高,发展起来也是相对迅速的。根据相关部门的不完全统计,一直到 1998 年在我国已经存在了 70 余家网上书店。

研究我国网上书店的发展,可以把其分为大致三个阶段:第一阶段:1995 年~1997 年,在这一阶段主要是注重网络这一新奇工具的宣传。第二阶段:1998 年~1999 年,在这一阶段已经存在着很激烈网络商业的竞争了,这时候就有一些敏感的业者感觉到仅仅依靠价格已经不再足以吸引消费者了。第三阶段:2000 年~至今,在这一阶段,尽管价格依然在考虑范围之内,但是全方位的需求考虑包括品质保证、售后服务、产品特点等等已经是大趋势了。

对于传统的线下书店,不仅书店受店面面积限制,而且也不可能罗列所有的图书,读者找起来也比较麻烦。现在有很多情况是,许多营业员也不知道有些书籍放在什么地方,只是知道一个大概区域。相比这种情况,网上书店确可以通过寻找相应书籍的标签和关键词来进行检索,同时可以提供可对比项,可以最快地帮助读者找到最合适的书籍,检索能力极强。

而且由于网上书店不需要店面,是依靠网络形成的,只要是在网络覆盖的地区,消费者就可以不受时间和地点的限制进行找书。还有就是多数的网上书店都会提供多种多样的服务,如新书推荐、畅销书籍排行、读者评论等栏目,使读者既可以全方位地把握图书信息,又可以进行在线讨论,交流心得。^[9]

随着网上书店的发展,目前图书市场的竞争可以说是越来越激烈了,近年来以新华书店为代表的实体书店也正在向线上市场进军,而且现在已经开设了网上书城,未来也将会有更多的实体书店纷纷建立网上销售渠道,从而加剧行业市场竞争。

在这种情况下,网上书店未来的营销策略也需要有所调整。首先在提高和保证图书质量上要下足功夫,同时也要增强自己的技术含量,建立功能强大的数据库管理系统。其次要选择正确的物流方式同时对物流进行改革,书店要根据自己的资金状况以及竞争力水平来考虑,选择适合的物流方式。最后,出版社对于网上书店的发展来说是非常重要的,这代表着书店对用户的保障。因此,必须要加强与出版社之间的合作,与此同时也需要社会各界的支持。

第二章 图书商城系统相关理论与技术

相关理论的学习是在每个项目开发中必不可少的事情，该章节主要是介绍在本次项目中所用到的理论知识，并分析其使用到的关键性技术，这些知识为后续的项目开发打下基础。

第一节 Vue 框架介绍

一、Vue 框架概述

Vue 框架是一套用于构建用户界面的渐进式框架，所谓渐进式就是指你可以控制一个页面的标签，也可以控制一系列的标签，甚至你也可以控制整个页面或者是整个前台项目。使用 Vue 框架你可以独立完成前后端分离式的 Web 项目的 Javascript 框架。Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关心视图层，不仅容易上手，而且还便于与第三方库或者既有项目整合；另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页面应用提供驱动。^[10]

在 Vue 项目开发过程中，一般是使用包管理工具来安装 Vue，但是直接在页面上加上 script 标签引用也可以。通过 new Vue() 语法来建立一个 Vue 的实例对象，该实例对象包含很多的属性，包括 el、data、methods、computed、watch 等等。el 代表节点元素，data 里面存放数据，methods 里面写各种方法，computed 内存储的也是方法，但是为计算数据。复杂逻辑的应该存储在 computed 中，计算属性是基于它们的依赖进行缓存的，由于 computed 带有一层缓存，所以只有在它的相关依赖发生改变时才会重新运行，而 methods 则是调用一次生成一次。^[10] watch 为监听属性，可以监听 data 中的属性值也可以监控对象。

在 Vue 中，html 要用 v-bind 来进行属性绑定，v-bind 可以简写为 :，监听事件的时候要用 v-on 来进行事件绑定，v-on 可以简写为 @，v-once 代表只能渲染一次，v-html 用来渲染 html，在 input、select 标签中我们可以用 v-model 来实现双向数据绑定，v-for 可以实现数据循环（循环的时候需要设置 key 值来唯一标识），v-if、v-else 用来实现判断，v-show 可以决定 dom 节点的显示或者隐藏，在 Vue 中还存在很多修饰符如取消事件 (.stop)，取消默认事件 (.prevent) 等等。^[11]

在 Vue 实例的对象下面还存在很多的生命周期钩子函数，总的来说分为三大块：创建阶段包含 beforeCreate、created、beforeMount、mounted 等钩子函数。更新阶段包含 beforeUpdate、updated 等钩子函数。销毁阶段包含 beforeDestory、destoryed 等钩子函数。

二、Vue 框架特点

前台框架比如：Angular 太过庞大，React 精通移动端。而 Vue 则吸收了前两者的优势，是一个轻量级的 JavaScript 框架，并且还能实现前后端分离开发，节约开发成本。从根本上来说，这要得益于 Vue 框架的几个特点：

1. 双向数据绑定

保留了 Angular 的特点，但是在数据操作方面更简单了。Vue.js 会自动对页面中的某些数据的变化做出同步的响应。也就是说，Vue.js 会自动响应数据的变化情况，并且根据用户在代码中预先写好的绑定关系，对所有绑定在一起的数据和视图内容都进行修改，而这种绑定的关系，就是以 input 标签的 v-model 属性来声明的。^[11]

2. 组件化

保留了 React 的优点，实现了 html 的封装和重用，在构建单页面应用方面有着独特的优势。^[12]Vue 通过组件，把一个单页面应用中的各种模块拆分到一个一个单独的组件中，我们只要在父级应用中写好各种组件的标签，并且在组件标签中写好要传入组件的参数，然后分别写好各种组件的实现，然后整个应用就算做完了。^[13]

3. 虚拟 DOM

虚拟 DOM，他就是一种可以预先通过 JavaScript 进行各种计算，把最终的 DOM 操作计算出来并优化，由于这个 DOM 操作属于预处理操作，并没有真实的操作 DOM，所以叫做虚拟 DOM，最后在计算完毕才真正将 DOM 操作提交，将 DOM 操作变化反映到 DOM 树上。^[14]

第二节 MVVM 架构模式介绍

一、MVVM 模式概述

MVVM 是由 Model、View、ViewModel 三部分构成，Model 代表的是数据模型，也可以在 Model 中定义数据修改和操作的相关逻辑；View 代表的是 UI 组件，负责将数据模型转为相应的 UI 展现出来；ViewModel 则是一个同步 View 和 Model 的对象。^[15]

在这种架构模式下，View 和 Model 并不会直接的有联系，是通过 ViewModel 进行联系的，因为 Model 和 ViewModel 之间的交互是双向的，所以在 View 中的数据变化会同步到 Model 中，反过来也一样。

在 ViewModel 中通过双向的数据绑定来把 View 和 Model 层连接了起来，而 View 和 Model 之间的同步工作是完全自动的，不需要外来的人为干涉。所以开发者只需要关注相应逻辑的开发，不需要再手动的操作 DOM 了，也不需要关注数据状态的同步问题，把复杂的数据状态维护全交给 MVVM 来管理。^[15]

架构模式如图 2-1 所示。

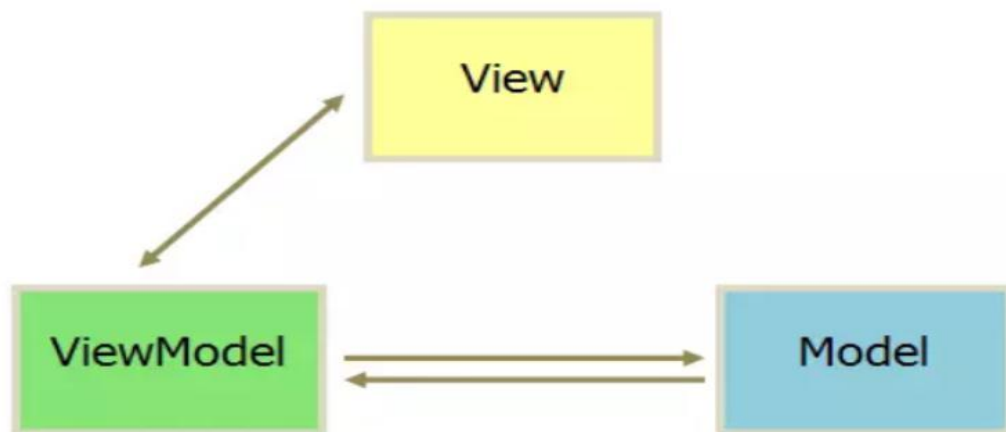


图 2-1 MVVM 架构模式图

二、MVVM 模式的优点

在 MVVM 模式下，比之前的 MVP 模式可以至少省下 30% 的编码量，甚至可以省下 70% DOM 操作，这得益于它的以下几个优点：

1. 低耦合

View 能够独立于 Model 的变化和修改，一个 ViewModel 能绑定到不同的 View 上，在 View 变化的时候 Model 可以不变，同时当 Model 变化的时候 View 也可以不变。^[16]

2. 可重复性

可以把功能完整、相对独立的视图逻辑封装起来，让其他的 View 页面利用这段视图逻辑。

3. 独立开发

在这里开发人员只需要考虑相应的逻辑开发和数据的设计，设计人员就只关注整个页面的开发与设计。

4. 可测试

页面测试起来是十分麻烦的，但是现在可以根据 ViewModel 来进行页面测试。

第三节 前后端分离开发模式介绍

在一般的传统的开发模式中，开发者将的前端代码写在 JSP 里面，有时候还会在 JSP 中加入后端代码。当用户访问网站时，页面数据也就是 Html 文档，由 Servlet 容器将 JSP 编译成 Servlet，然后将 JSP 中的 Html、Css、JavaScript 代码输

出到浏览器，这个过程需要经过很多步骤，才能响应用户的请求。^[17]用这种方式来编写代码非常的麻烦，而且效率上也非常低，整个代码加载起来就会很缓慢。从后期代码维护的角度上看，这种前后端代码混合在一起的方式，导致项目的维护性很差，同时也增加了维护成本。传统开发模式如图 2-2 所示。

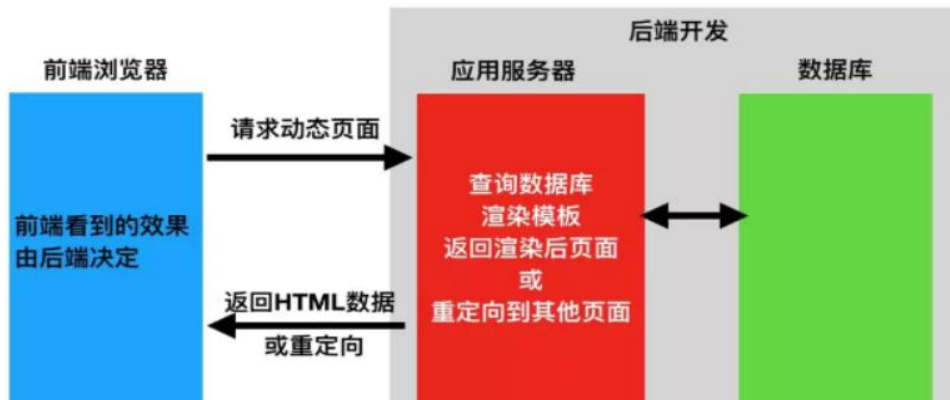


图 2-2 传统开发模式图

由于传统开发模式的弊端，之后就出现一种新的开发模式-前后端分离开发模式，在这种开发模式中，在后端只需要将前端需要的数据返回即可，在前端就只需要关注页面的渲染，给用户看到什么样的页面，如何把后端数据加载到前端页面，都是由前端开发者决定，后端只需要提供相应的接口，并且这种开发模式中前端与后端的耦合度很低。在这种模式中，我们通常将后端开发的每个视图都成为一个接口，或者 API，前端通过访问接口来对数据进行增删改查，总结一句话，后台负责提供数据，前端负责数据展示，职责分离，分工明确。^[17]前后端分离开发模式如图 2-3 所示。

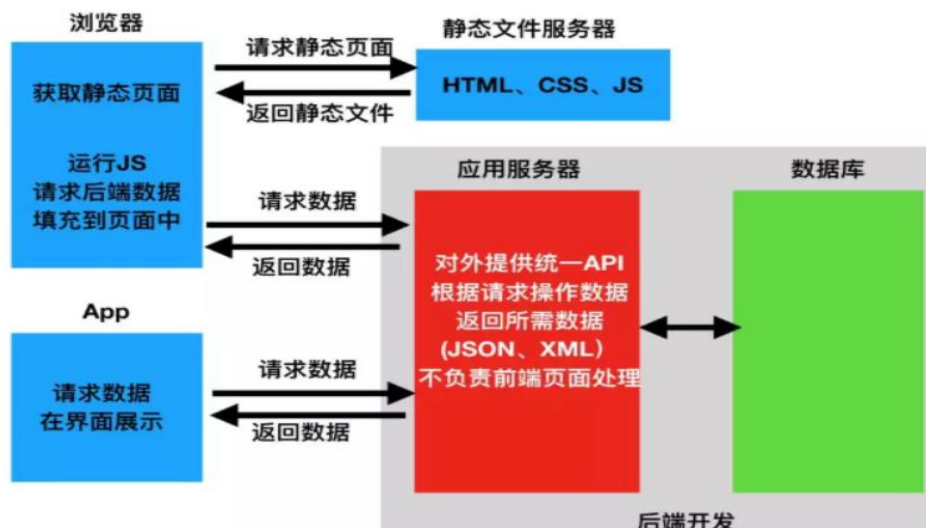


图 2-3 前后端分离开发模式图

第四节 开发工具介绍

一、Visual Studio Code

Visual Studio Code 是一款非常轻量级并且还跨平台的代码编译器。Vs Code 的使命是让开发者在编译器里拥有 IDE 那样的开发体验，比如代码的智能提示、语法检查、图形化的调试工具、插件拓展、版本管理等。^[18]它具有对 JavaScript、TypeScript、Node.js 的内置支持，同时它也支持其他的主流语言如 Java、Python 等。Vs Code 有非常多的快捷按键，按键设计也是非常合理且人性化。

二、PyCharm

PyCharm 是由 JetBrains 公司打造的一款 Python IDE。它带有一套可以帮助用户在使用 Python 语言开发的时候提高其工作效率的工具，如调试、语法高亮显示、Project 管理、代码跳转、智能提示、版本控制等。^[19]除此之外，这个 IDE 还提供了一些高级功能，用来支持 Django 框架下的 Web 开发，同时还支持 Google App Engine，更加的，它还支持 IronPython。

三、Navicat

Navicat 是一套数据库管理工具，不仅速度快而且价格也相当便宜，专门为了简化数据库管理管理和降低系统管理成本而设计的，它是以直觉化的图形用户界面设计的，操作起来非常的简单，可以安全的创建、组织、访问并共用信息。^[20]

四、Postman

API 是应用程序编程接口，而 API 测试就是为了检查它们的功能、性能、安全系，以及是否返回正确的响应，根据输入参数检查响应，并检查 API 检索和授权数据所花费的时间。^[21]Postman 正是一个通过向 Web 服务器发送请求并且获取响应来测试 API 的应用程序。

第五节 本章小结

在本章节中，是在项目开发之前对其中涉及的关键性技术进行相关概述。主要介绍了 Vue 框架的相关内容，进而还描述了 MVVM 架构模式和前后端分离开发模式。最后，对于本次项目开发所用到的开发工具也进行了简单的介绍。

第三章 图书商城系统的需求分析

图书商城系统的设计之前要进行相关的需求分析,进行需求分析的目的就是为了弄清楚用户需要什么、想要什么。在了解了这些之后,也就知道设计的系统需要什么要求和性能了。

第一节 业务需求分析

随着网络的普及以及各种技术飞速发展,近几年来,越来越流行网络支付了,同时网络支付手段也越来越完善了。相比于线下购物,人们更加倾向也更加习惯于网上购物了,只需要在手机上点击购买,东西就会很快的到自己的手中。同时加上网上购物会有很多的优惠,这也是人们选择网上购物的一个重要原因。

网络已经慢慢改变了传统的商业运转模式并提供了一种新的可行性方案:利用互联网技术,通过廉价的通讯手段,将卖家、买家、厂家和合作人紧密的结合的在一起,消除了时间上和空间上的隔阂,这也很大程度上降低了交易成本同时扩大了交易范围。

当然传统的线下书店也正在遭受互联网的冲击,与此同时,传统书店的购书观念也慢慢被网上购物观念所影响。人们期待着一种新的更加便捷的方式,所以线上书店就诞生了。

先如今,网上书店可以实现的业务功能也已经非常的多样化了,从最初只是发布一些消息到现在各种类别图书的展示以及在线交易功能、在线客户交流功能、在线管理功能等等,几乎传统书店能提供的东西在线上书店都可以很好的提供并进行电子化的高效运作,并且不需要考虑地域和店面问题,突破了传统书店的这些局限性,从而吸引了更多的用户数量。

第二节 可行性分析

进行可行性分析的目的是为了搞明白本项目的开发能不能够实现相应的价值、值不值得进行研究,实际上就是一次很大程度简化的系统分析和系统设计过程,所以,必须进行可行性分析。分析最开始的设计目标和进行了相关调查可以得出以下几点可行性分析:

一、操作可行性分析

从操作角度来分析,本项目小巧,页面简单易懂,只需要花个十多分钟就能很好的使用,对于用户来说,只需要简单的注册一个账号就可以在不同的书店找到不同的书籍,进行对比后再决定购买,同时,书店里面种类众多,相信所有用

户都能够找到自己心仪的书籍。而且，对于管理人员，管理操作也是非常的简单，同时与后台也具备很好的交互性。总的来说，本项目对任何人来说都是很容易上手的。

二、技术可行性分析

从技术角度分析可知，网站的搭建较为简单，而且目前基本上都是使用框架进行网页的搭建，本项目难度适中，环境的搭建比较简单，项目需要考虑不同平台的页面比例，做到不同用户都能有个很好的使用体验。本次设计中数据库方面使用功能强大的 MySQL 数据库系统。

三、经济可行性分析

从经济角度分析，线上书店主要以书籍、杂志以及其他商品为主，因此在网站上可以让客户提前浏览书籍、杂志，消费者初步了解了里面的内容之后在进行购物决定，这样也可以相应减少很多的退货问题。相比于传统的书店，开设网上书店并不会有很多的市场规模和经营模式的限制，突破了很多以往传统书店的限制，这是一种全新的销售渠道。在这些方面都可以大大降低资金的投入，用户也可以不受时间地点的限制，随时随地都能登录网站进行购物，增加书店的销售额度。因此不管是从商家还是用户的角度来看，网上书店具有很好的发展前景。

四、法律可行性分析

从法律角度分析，本项目是作为毕业设计与商业无关，同时是经过自己独立的开发设计，不需要考虑侵权的麻烦，在法律上是没有任何问题的。

第三节 功能需求分析

本章中的系统功能分析通过用例图方式分析。用例将需求和真实的设计很好的连接起来。在前文的分析中，可以明确知道本系统是提供给用户和管理员使用的，本系统所表现的用例图，如图 3-1 所示。

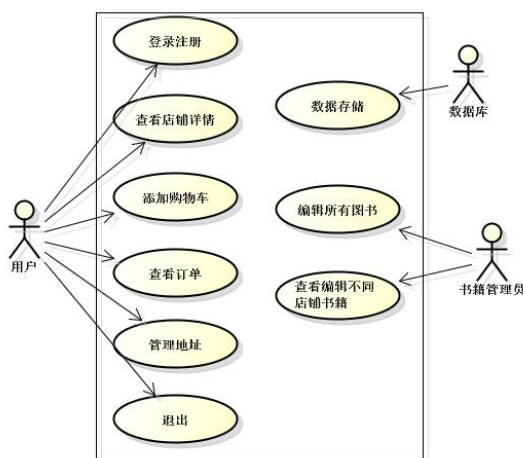


图 3-1 系统用例图

从本系统的用例图可以看出，本系统分为两类人员，对应使用的功能也是不相同的。比如对于用户来说，他们支持使用的功能包括了登录注册、查看店铺详情、添加购物车、查看订单、管理地址和退出等功能。在数据库中存放着关于书籍、店铺和用户的各种信息。而书籍管理员可以编辑所有图书，同时对于不同的店铺中的书籍，也可以进行相应的编辑。那么从该用例图入手，就可以构建出本系统的一些基本功能模块了。

第四节 系统开发步骤

在系统开发之前，要经历相应的几个步骤。首先就是进行需求的调研，在这一阶段要对本系统的主要大功能模块有大概的明确同时初步定义好少量的界面，然后进一步深入分析每个大功能下面的小功能应该有些什么。其次就是进行概要设计了，概要设计在整个系统开发过程中起着很关键的重要，包括设计系统的基本结构、主要模块、功能、接口、数据结构等等，这些为系统之后的详细设计提供了基础。之后就是详细设计，开发者需要具体实现相应功能，设计相应的数据结构和算法完成编码，同时也要完成数据库表结构的设计。在完成本系统设计之后就需要进行相应的系统测试了，以此来确保本系统的功能能够正常使用。最后一步就是把本系统真正交给用户使用。

第五节 本章小结

本章是对于本图书商城系统进行了相应的需求分析，首先从业务需求入手，对目前传统书店的弊端进行分析，接着阐述慢慢崛起的线上书店模式，其后介绍的是对应的可行性分析，分别分析了操作可行性、技术可行性、经济可行性和法律可行性并得出了相应的结论。最后的功能需求，通过了用例图的方式对本系统所需要的基本功能进行了相应的分析，不同的人员对应不同的功能。最后对系统开发的步骤进行了比较完整的介绍。

第四章 图书商城系统设计

第一节 系统的设计概要

一、整体思路

本系统设计的目的是为了迎合互联网发展的趋势，进一步方便人们的生活，让用户随时随地只需要上网进行需求的满足就可以了。系统的功能方面，用户在本系统找到附近店铺然后去店铺里面选择相应的自己想要的图书就可以了，后台的管理人员可以在本系统的管理端上看到数据库存在的所有书籍，同时还可以管理所有店铺，对于各个店铺里面的书籍可以进行相应的操作；在系统模块方面，每个功能模块进行单独设计，这样系统的设计和开发进度都会得到提升，系统的设计需要对所掌握的需求有一个更精准的了解。

二、功能结构设计与分析

本图书商城系统分为三个部分，分别为用户端、管理员端和数据库端。其中用户端设计为五大主要功能模块，分别为登陆注册、查看店铺详情、添加购物车、提交订单、个人信息管理。管理员端设计了两个主要功能模块，分别为管理所有图书信息和管理所有店铺以及该店铺下的所有书籍。数据库存储着本次系统的各种信息表。本系统的结构设计如图 4-1 所示。

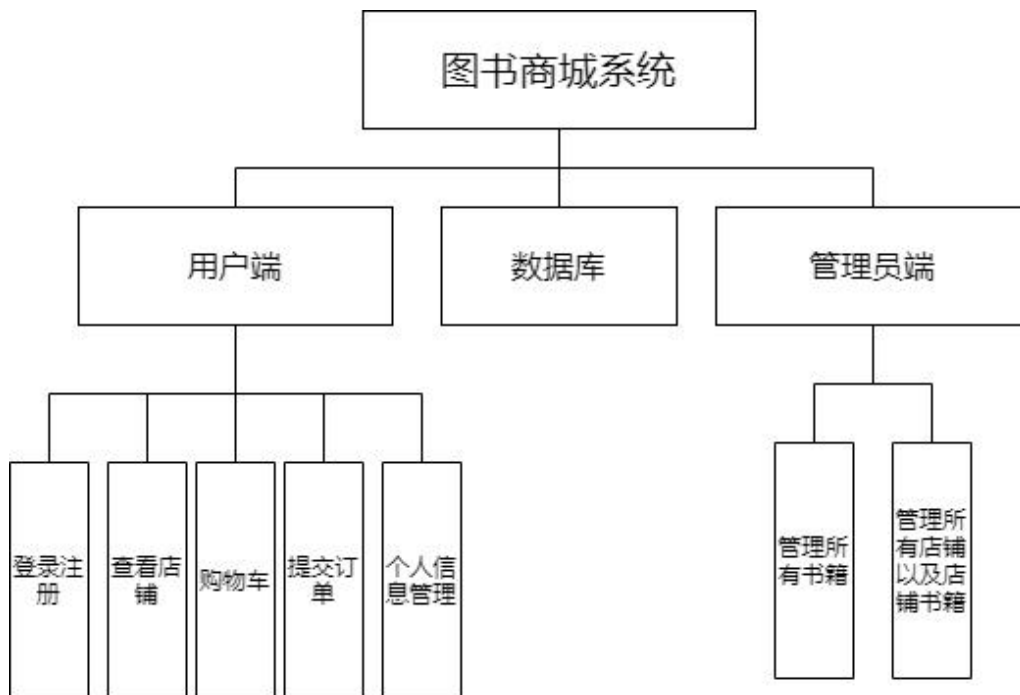


图 4-1 系统结构图

三、层次分析

在本系统开发和设计中，着重从 UI 层、数据库层和逻辑层入手来构建层次结构。UI 层的目的是为了更方便用户与系统的交互，数据库层完成数据的调用和其他使用数据的操作，逻辑层是整个系统的核心层，负责各个功能板块的逻辑设计，一个系统数据结构逻辑设计的好的话，相应开发起来的难度也会降低很多。采用前后端分离模式实现接口的对接，对接方式使用 JSON 文件格式。

第二节 系统详细功能设计

本系统的设计在系统结构图的基础上完成不同功能的开发与设计：

一、注册与登录功能

对于用户，首先进入登录页面，如果有账号的话可以直接进行登录，只要输入的账号和密码没有问题的话就会进入到主页面，但是对于没有账号的用户，首先需要点击注册按钮进入注册页面，在登录之前是进不去主页面的。进行账号和密码注册之后该信息会加入到数据库的相应表中，之后再返回登录页面，输入刚注册账号的信息，然后后台对输入的信息进行相应的校对，如果账号和密码都符合的话就会在几秒之后进入系统的主页面。在进入主页面之后同样也是不允许在返回登录页面了，只有在点击退出按钮之后系统会自动回到登录界面。

对于书籍管理员来说，操作也非常简单，只需要进入相应的管理员登录页面，输入相应的 **admin** 的密码，后台会找到相应的管理员表对该密码进行校验，没有问题的话就会进入相应的书籍管理页面。

不管是用户还是管理员，在登录之后，基本的信息都会暂时保存在本地的本地存储中，整个登录和注册页面的设计非常的简洁大方。

二、查看店铺内容功能

用户在进行登录之后，会进入到本系统的主页面，在该页面的顶部会有目前的时间信息，然后在下面会看到从数据库里获得的附近店铺的选项，上面有一些店铺的基本信息，比如配送费、起送费等等，点击进入之后就可以看到该店铺的不同类别的书籍信息了，如价格、折扣、库存等等。每个店铺的书籍信息也都是不相同的。

三、购物车功能

在店铺页面的最下方，会有一个购物车的图标，点击是可以展开的，如果用户当前没有添加任何图书的话，那么默认是隐藏的，在点击一本图书加入购物车之后，就会自动展开了，当然也可以收起隐藏继续添加其他的图书，在购物车中包含全选和一件清空购物车功能。用户还可以退出该店铺，进入其他的店铺添加其他店铺的书籍，在主页面的底部有个购物车选项，在里面汇总了用户在不同店铺的所有的购物车内容。购物车里面的内容是可以长时间存储，保存在本地的本

地存储里面。在店铺里面还有结算按钮，当然需要满足该店铺的起送规则，否则会提示还差多少钱同时点击也是无效的，在价格满足要求之后就可以点击该按钮了。

四、订单相关功能

在满足了店铺起送要求之后，点击结算按钮，加上配送费就可以进入到订单页面了，在该页面，顶部是用户的地址信息，当然如果设置好了地址的话在里面选择就可以了，如果当前没有地址的话是不可以提交订单的，需要到个人信息页面去创建新的地址。中间部分是在该店铺的购物车信息，包含价格和数量两部分，最后还有一个加上配送费的总价汇总。最下方就是订单提交按钮了，点击该按钮，会显示一个确认和取消框，点击空白区域可以返回订单页面，选择确认或取消之后该订单信息就会被加入到后台相应的表里面，然后会进入到一个专门的用户订单页面，里面包括了该用户在不同书店确认或者取消的订单信息。

五、个人信息功能

在主页面下方的导航选项，有个个人信息按钮，点击后可以进入个人信息页面，在这里的顶部展现该用户的基本信息，如 id 和姓名，下方就是对信息处理的一些功能了，有改名按钮，改名是不允许和其他用户名字一样的。同时也允许更改当前的密码，当然新旧密码也是不允许相同的。最重要的是用户的地址管理，点击该按钮会进入用户地址页面，在这里会展示目前用户已经创建的地址，如果没有地址可以点击新建进行地址的创建。每条地址信息可以点击，进入地址编辑页面，对当前的地址信息进行编辑或者删除。最后，还提供一个账号注销按钮，如果注销的话，与该账号有关的信息都会在数据库中被删除。

六、编辑所有图书功能

管理员登录之后，就会进入管理员首页，对于管理员来说，提供对所有书籍的管理和对店铺的管理功能，首先对所有书籍，可以新增图书，对当前已有书籍可以进行相应信息的更改或者图书的删除。进行相关操作就会在后台数据库相应的表当中得到体现。

七、店铺管理功能

在点击店铺信息之后进入店铺管理页面，在这里可以新开店铺，同时对已有的店铺可以进行基本信息的编辑，点击进入不同的店铺信息会出现相应店铺的书籍信息，对于每个店铺的图书信息也都是可以修改的。同样相关的操作都会在后台的数据库表中得到体现。

八、退出功能

不管是用户还是管理员，都提供一个退出系统的按钮，点击该按钮几秒钟之后就会回到最初的登陆页面，同时本地存储的本地信息会进行删除。

第三节 数据库表的设计

数据库设计在整个项目开发中非常重要，在数据库中包含有多张表结构，通过概念设计可以将其转化为 R-R 图。设计完了数据库之后，就可以开始项目的开发了，后续的功能都需要围绕表来进行，所以在一定程度上，设计好的表结构能够降低程序开发的难度。

一、管理员表

管理员可以对书籍进行操作，同时也可以编辑每个店铺和店铺下的书籍，本项目中的管理员表设计了 id、name、password 三个字段，其中 id 设置为主键。结构如表 4-1（管理员表）所示。

表 4-1 管理员表

名	类型	长度	是否可以 NULL	主键
id	int	10	否	是
name	varchar	10	否	
password	varchar	12	否	

二、店铺表

店铺表存储了有关店铺的基本信息，本项目中的店铺表设计了 id、title、sales、expressLimit、expressPrice、slogan、imgUrl 七个字段，其中 id 设置为主键，expressLimit 为起送价，expressPrice 为配送费，imgUrl 为店铺图，slogan 为宣传语，sales 为月售。结构如表 4-2（店铺表）所示。

表 4-2 店铺表

名	类型	长度	是否可以 NULL	主键
id	int	10	否	是
title	varchar	10	否	
sales	int	10	否	
expressLimit	int	10	否	
expressPrice	int	10	否	
slogan	varchar	10	否	
imgUrl	varchar	30	否	

三、书籍表

书籍表存储了书的基本信息，本项目中的书籍表设计了 id、tab、name、imgUrl 四个字段，其中 id 设置为主键，tab 为书籍类别、imgUrl 为书籍封面。结构如表 4-3（书籍表）所示。

表 4-3 书籍表

名	类型	长度	是否可以为 NULL	主键
id	int	10	否	是
tab	varchar	10	否	
name	varchar	10	否	
imgUrl	varchar	10	否	

四、书籍店铺关联表

书籍店铺关联表代表着书籍和店铺的多对多关系，本项目中的书籍店铺关联表设计了 shop_id、book_id、stock、sales、price、oldPrice 六个字段，其中 shop_id 和 book_id 设置为外键，分别为店铺 id 和书籍 id。结构如表 4-4（书籍店铺关联表）所示。

表 4-4 书籍店铺关联表

名	类型	长度	是否可以为 NULL	外键
shop_id	int	10	否	是
book_id	int	10	否	是
stock	int	10	否	
sales	int	10	否	
price	decimal	10	否	
oldPrice	decimal	10	否	

五、订单表

订单表是用户购买图书之后的保存下来的信息，本项目中的订单表设计了 shop_id、id、address_id、user_id、shopName、isCanceled 六个字段，其中 id 设置为主键，shop_id、address_id、user_id 设置为外键，分别为店铺 id、地址 id、用户 id，shopName 为店铺名、isCanceled 决定是否取消状态。结构如表 4-5（订单表）所示。

表 4-5 订单表

名	类型	长度	是否可以为 NULL	主键	外键
id	int	10	否	是	
shop_id	int	10	否		是
address_id	int	10	否		是
shopName	varchar	10	否		
isCanceled	varchar	10	否		
user_id	int	10	否		是

六、地址表

地址表是用来存储用户地址信息，本项目中的地址表设计了 id、user_id、name、phone、department、city、houseNumber 七个字段，其中 id 设置为主键，user_id 设置为外键，为用户 id，name 为收件人姓名，houseNumber 为房间号。结构如表 4-6（地址表）所示。

表 4-6 地址表

名	类型	长度	是否可以为 NULL	主键	外键
id	int	10	否	是	
user_id	int	10	否		是
name	varchar	10	否		
phone	int	10	否		
department	varchar	30	否		
city	varchar	10	否		
houseNumber	varchar	10	否		

七、用户表

用户表是用来存储已注册用户的基本信息，本项目中的用户表设计了 id、username、password 三个字段，其中 id 设置为主键。结构如表 4-7（用户表）所示。

表 4-7 用户表

名	类型	长度	是否可以为 NULL	主键
id	int	10	否	是
username	varchar	10	否	
password	varchar	12	否	

八、订单书籍关联表

订单书籍关联表代表着订单和书籍的多对多关系，本项目中的订单书籍关联表设计了 order_id、book_id、book_sales 三个字段，其中 order_id 和 book_id 设置为外键，分别为订单 id 和书籍 id，book_sales 为该书数量。结构如表 4-8（订单书籍关联表）所示。

表 4-8 订单书籍关联表

名	类型	长度	是否可以为 NULL	外键
order_id	int	10	否	是
book_id	int	10	否	是
book_sales	int	10	否	

第四节 连接数据库的设计与实现

数据库是计算机应用系统中的一种专门管理数据资源的系统。数据有多种形式，如文字、数码、符号、图形、图像及声音等，数据是所有计算机系统所要处理的对象，我们所熟知的一种处理办法是制作文件，即将处理过程编成程序文件，将所涉及的数据按程序要求组成数据文件，再用程序来调用，数据文件与程序文件保持着一定的关系。^[22]

简单来说，数据库就是一个存放数据的仓库，它里面的存储空间极大，可以

放置上百万条、千万条甚至亿万条数据。然而并不是说随意把数据丢进去就可以了，必须要符合一定的规则，不然到时候查询的效率会非常低。关系型数据库和非关系型数据库是如今比较常见的两种数据库。

随着计算机技术的不断发展，文件式的管理方法不仅使得数据通用性变差，而且还不便于移植，会造成存储空间的浪费，已经不太能满足当前的需求了。但是数据库系统的出现很好地解决了这个问题，数据库系统从对自身数据的管理出发，把数据都存放在数据库当中，这样以来，通过数据库管理系统就可以很方便的使用数据了。

MySQL 是一款安全、跨平台、高效的，可以和主流编程语言紧密联系的数据库系统。当前 MySQL 在中小型网站中被广泛使用，因为其体积小、速度快、成本低、开放源码等原因，让很多的公司都使用 MySQL 数据库来降低公司的成本。^[23]

本项目后台连接数据库使用的是 Python 语言的 Flask 框架，Flask 是 Python 的一个 Web 微框架，只要导入相应的连接数据库的包，就可以很轻松的连接上数据库^[24]。当然在导入之前还需要进行数据库的配置，同时前后端分离开发模式还需要注意解决跨域的问题。

在连接上数据库之后，首先一步就是进行关系映射，简单来说，就是将数据库中的表结构与面向对象语言里面的类建立好一种对应关系。这样做的目的就是之后我们想要操作数据库、数据库中的表或者某条记录就可以直接通过操作类实例来完成。而 SQLAlchemy 就是 Python 里面最知名的 ORM 工具之一，可以高效率的操作数据库。本项目就是使用了 SQLAlchemy 进行表的关系映射，之后写接口的时候对数据库进行的操作就都是通过操作类实例来完成的了。

第五节 本章小结

本章对图书商城应用系统设计进行了相关的阐述，首先是概要设计，分析了本次设计的整体思路以及相关功能结构和层次结构。之后对本系统设计的主体功能模块进行了相应的介绍，包括登录注册、购物车、订单、个人信息等等。紧接着介绍了本次系统设计中的表结构，包括用户表、管理员表、书籍表、店铺表、订单表等等。最后对数据库以及数据库的连接进行了相关介绍。

第五章 系统功能实现

本项目使用 HBuilder X 工具将代码打包为 APK，该工具十分的轻巧、急速，主要是针对 Vue 生态打造的，使用起来非常的方便和简单，打包操作很容易就可以上手。

第一节 用户端实现

一、注册与登录

本项目的首页为登录页面，同时也设置了登录之前不允许访问其他页面，当然注册页面和管理员登录页面是可以访问的，如果已经有了账号的话只需要输入相应的用户名和密码，后台会获得相应的数据来进行对比，当用户名和密码都正确的话就能够成功登录了。如果还未有账号的话可以通过点击注册链接进去到注册页面，注册方式很简单，将想好的用户名和密码填入即可，如果该用户名没有被注册的话就能够成功注册，在成功注册之后会跳转到登录界面，然后把注册的账号的信息输入就可以登录了。

成功登录之后会进入到主页面，在主页面的最上放会实时显示当前的时间，中间部分是一些图标，由于目前的技术水平不够，没法做到真正意义上的分类跳转，页面可以使用鼠标滚轮就行翻滚，下方会有附近店铺的一个内容，点击相应的店铺就会进入其中，最下方是导航栏，具体内容在之后会介绍。

本系统的用户注册界面如图 5-1 所示：

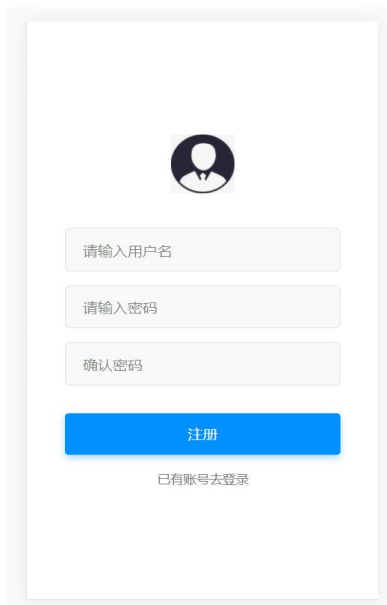
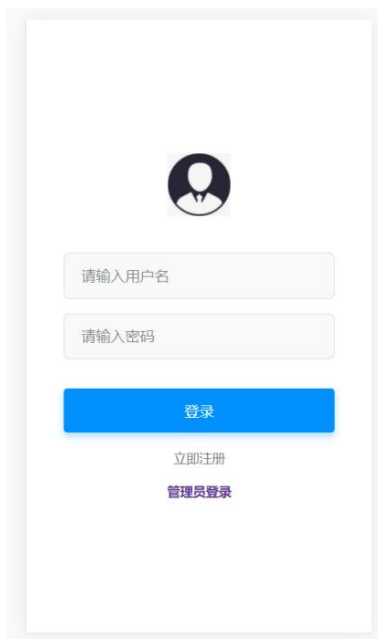
The diagram shows a user registration form. At the top is a circular icon of a person. Below it are three input fields: '请输入用户名' (Please enter username), '请输入密码' (Please enter password), and '确认密码' (Confirm password). Below these fields is a blue button labeled '注册' (Register). At the bottom of the form is a link labeled '已有账号去登录' (Already have an account, go to login).

图 5-1 用户注册界面图

用户注册功能的关键代码如下：该函数就是通过对前端发来的数据进行相应的操作来返回相应的结果给前端，首先一步就是判断该用户是否以及被注册，未注册的话就能够正常加入到数据库表当中。

```
# 注册逻辑
@app.route('/register', methods=['POST', 'GET'])
def register():
    session = DBSession()
    register_user = request.json # 获取post数据
    find_user = session.query(User).filter(User.username == register_user['username']).first()
    # 如果有记录代表该用户名存在
    if find_user is not None:
        session.close()
        return 'exist'
    # 否则添加进数据库
    else:
        new_user = User(username=register_user['username'], password=register_user['password'])
        session.add(new_user)
        session.commit()
        session.close()
        return 'ok'
```

本系统的用户登录界面如图 5-2 所示：



The image shows a user login interface. At the top, there is a circular icon of a person. Below the icon are two input fields: '请输入用户名' (Please enter username) and '请输入密码' (Please enter password). Below these fields is a blue button labeled '登录' (Login). Under the login button are two links: '立即注册' (Register now) and '管理员登录' (Admin login).

图 5-2 用户登录界面图

用户登录功能的关键代码如下：该函数是对前端发来的数据进行判断来返回相应的结果给前端，同样也要先判断用户是否存在，在数据库表中查找对应关键字的字段，如果存在才能进行用户名和密码的验证。

```

# 登录逻辑
@app.route('/login', methods=['POST', 'GET'])
def login():
    session = DBSession()
    login_user = request.json # 获得post数据
    # 判断是否存在该用户
    find_user = session.query(User).filter(User.username == login_user['username']).first()
    if find_user is None:
        session.close()
        return 'none'
    # 比较用户名和密码是否正确
    elif login_user['username'] != find_user.username or login_user['password'] != find_user.password:
        session.close()
        return 'error'
    else:
        session.close()
        return find_user.to_json()

```

本系统的用户成功登录界面如图 5-3 所示：



图 5-3 用户成功登录界面图

附近店铺展示的关键代码如下：该函数就是从数据库表中取出附近店铺的所有字段返回给前端。

```

# 附近店铺展示逻辑
@app.route('/showNearby', methods=['POST', 'GET'])
def showNearby():
    session = DBSession()
    shops = session.query(Shop).all()
    li = []
    for item in shops:
        shop = item.to_json()
        shop['imgUrl'] = "http://localhost:5000/" + shop['imgUrl']
        li.append(shop) # model转为json存放list
    dic = {'key': li} # list转为dict
    string = json.dumps(dic) # dict转为json
    session.close()
    return string

```

二、查看店铺内容功能

上面介绍了用户登录之后会进入到主页面，在主页面有附近店铺选项，每个店铺上面会有其基本的一些信息，进入店铺之后，最上部分会有一句标语以及一个回退的按钮，在页面的上部分也有该店铺的一些相关信息，如配送费，起送费等等，中间部分就是该店铺的主要内容了，包括了各种类别的书籍，点击左边的类别栏就可以看到该类别下的所有书籍了。每本书上会有该书的一些基本信息，比如现价、原价、库存等等，每本书的边上有加减按钮可以进行添加购物车的操作，购物车的具体内容会在之后介绍。页面最下方就是购物车功能，在没有添加书籍的时候是会默认不展开的。

本系统的店铺内容界面如图 5-4、图 5-5 所示：



图 5-4 店铺内容界面图

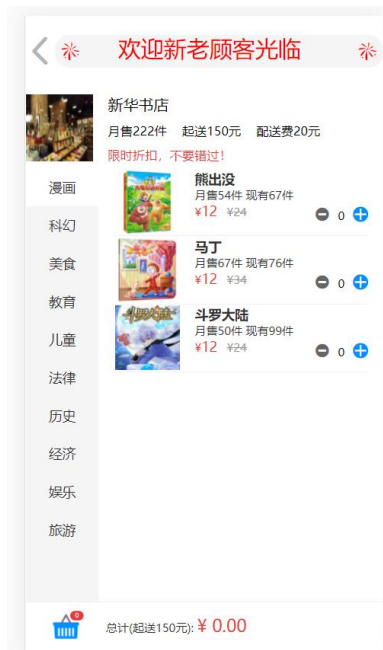


图 5-5 店铺内容界面图

当前店铺所有书籍展示的关键代码如下：该函数就是通过前端发来的店铺 id 来查找当前店铺下的相关的书籍全部字段并取出来返回给前端。

```
# 当前商铺书籍展示逻辑
@app.route('/showBook', methods=['POST', 'GET'])
def showBook():
    session = DBSession()
    gets = request.args.to_dict()
    config_book = session.query(BSconfig). \
        filter(BSconfig.shop_id == gets['id']).all() # 查询当前书店下所有书信息
    li = []
    for item in config_book: # 将关系表数据转为字典
        config_info = item.to_json()
        book_info = session.query(Book).filter(Book.id == config_info['book_id']). \
            filter(Book.tab == gets['tab']).first() # 查找满足条件的书
        if book_info is not None:
            book_info = book_info.to_json()
            book_info['imgUrl'] = "http://localhost:5000/static/img/books/" \
                + book_info['tab'] + '/' + book_info['imgUrl']
            book = dict(config_info, **book_info) # 字典合并
            li.append(book)
    dic = {'key': li}
    string = json.dumps(dic, cls=DecimalEncoder)
    session.close()
    return string
```

三、购物车功能

上文提到的在店铺内容页面最下面有一个购物车图标，未选择任何书籍的时候是没办法展开的，但在加入一本书籍之后再去点击的话就有用了，打开可以看到购买该书籍的数量和价格信息，接着可以点击阴影区域继续返回添加书籍，在购物车里面也可以选择一键清空。购物车图标的边上的信息指的是该店铺下的起送费，用户必须满足了起送要求才能去进行结算，否则的话显示的将会是目前还差多少钱起送。当然还可以退出该店铺，在别的店铺同样购入书籍，之后可以点击主页面下方的购物车导航栏，之后就能够看到目前用户所有店铺的购物车汇总了。购物车可以实现长时间的储存，保存在本地存储里面，当用户从店铺退出后再次进入依然会显示之前的购物车信息。

本系统的购物车加减操作界面如图 5-6 所示：

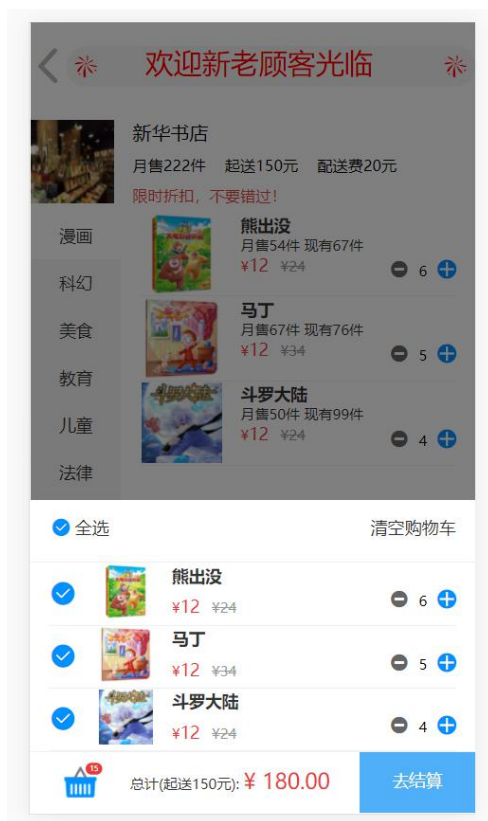


图 5-6 购物车加减操作界面图

加减购物车的关键代码如下：该函数就是通过设计的购物车结构，对点击的加减操作转化为正数与负数的判断从而进行不同的操作来让数量进行相应的增减。

```
changeCartItemInfo (state, payload) {
  const {shopId, productId, productInfo} = payload
  let shopInfo = state.cartList[shopId] || { // 购物车结构
    shopName: "",
    productList: {}
  }
  let product = shopInfo.productList[productId]
  if (!product) {
    productInfo.count = 0
    product = productInfo
  }
  if (payload.num > 0) {
    product.check = true
  }
  if (product.count < product.stock && payload.num > 0 || product.count > 0 && payload.num < 0) {
    product.count = product.count + payload.num
  } // 将加减操作转化为正负判断来进行相应增减
  shopInfo.productList[productId] = product
  state.cartList[shopId] = shopInfo
  setLocalCartList(state)
},
```

本系统的购物车汇总界面如图 5-7 所示：



图 5-7 购物车汇总界面图

购物车信息汇总的关键代码如下：该函数就是将本地存储的所有店铺里面的购物车信息全部取出来。

```
// 购物车展示逻辑
const useCartEffect = () => {
  const store = useStore()
  const cartList = store.state.cartList // 取出本地存储的购物车信息
  const cartListWithProducts = computed(() => {
    const newCartList = {}
    for (let shopId in cartList) {
      let total = 0
      const products = cartList[shopId].productList
      for (let productId in products) { // 计算不同购物车的不同书籍的数量和总价
        const product = products[productId]
        total += (product.count || 0)
      }
      if (total > 0) {
        newCartList[shopId] = cartList[shopId]
      }
    }
    return newCartList
  })
  return { cartListWithProducts }
}
```

四、订单相关功能

在满足店铺的起送要求之后，点击去结算按钮就可以进到相应订单页面了，在订单页面顶部有一个回退按钮，这下面就是该用户的地址信息了。如果当前用户没有任何的地址信息的话，是没办法提交订单的，需要先去个人信息页面进行

地址的创建,如果用户有地址的话,只需要点击在所有地址选择就可以了。中间部分就是在该店铺购物的订单信息了,包括购买书籍的数量、价格、和总价汇总。在页面的最下方有个订单提交按钮,边上显示的加上该店铺配送费之后的需要支付的总金额,点击提交订单会出现一个确认或者取消的框,点击阴影区域可以返回之前的操作,选择提交或者取消之后就会进入到用户的订单汇总页面了,这里显示的是用户在不同店铺的不同订单操作。

本系统的订单相关界面如图 5-8、图 5-9 所示:



图 5-8 订单相关界面图



图 5-9 订单相关界面图

提交订单的关键代码如下:该函数将前端传来的订单所有信息进行拆分分别加入不同表里面来完成本次订单提交操作。

```

# 订单逻辑
@app.route('/order', methods=['POST', 'GET'])
def order():
    session = DBSession()
    if request.method == 'POST': # 传入订单逻辑
        gets = request.json # 获取post数据
        # 加入订单
        new_order = Order(address_id=gets['addressId'], shop_id=gets['shopId'], user_id=gets['userId'],
                           shopName=gets['shopName'], isCanceled=gets['isCanceled'])
        session.add(new_order)
        session.commit()
        new_order_id = new_order.id
        products = gets['products']
        for item in products:
            product = P0config(order_id=new_order_id, book_id=item['id'], book_sales=item['num'])
            session.add(product)
            session.commit()
        session.close()
    return 'ok'

```

本系统的用户订单汇总界面如图 5-10 所示：



图 5-10 用户订单汇总界面图

用户订单展示的关键代码如下：该函数通过查询订单相关的表来取出当前用户的所有订单信息返回给前端。


```

elif request.method == 'GET': # 展示订单逻辑
    gets = request.args.to_dict() # 获取get数据
    orderInfo = session.query(Order).filter(Order.user_id == gets['id']).all() # 查找所有订单
    li1 = []
    for _order in orderInfo:
        _order = _order.to_json()
        del _order['address_id']
        del _order['user_id']
        li2 = []
        books_sales = session.query(POconfig).filter(POconfig.order_id == _order['id']).all()
        for book_sales in books_sales:
            book_sales = book_sales.to_json()
            del book_sales['order_id']
            book_price = session.query(BSconfig).filter(BSconfig.shop_id == _order['shop_id']). \
                filter(BSconfig.book_id == book_sales['book_id']).first()
            book_price = book_price.to_json()
            del book_price['shop_id']
            del book_price['book_id']
            book = session.query(Book).filter(Book.id == book_sales['book_id']).first()
            book = book.to_json()
            book['imgUrl'] = "http://localhost:5000/static/img/books/" \
                + book['tab'] + '/' + book['imgUrl']
            del book['id']
            bookInfo = dict(book_price, **book)
            book = {'bookInfo': bookInfo}
            bookInfo = dict(book_sales, **book)
            li2.append(bookInfo)
        books = {'books': li2}
        books = dict(_order, **books)
        del books['id']
        del books['shop_id']
        li1.append(books)
    dic = {"orders": li1}
    session.close()
    return json.dumps(dic, cls=DecimalEncoder)

```

五、个人信息相关功能

在主页面下方的导航栏有个个人信息选项，点击就会进入到个人页面。在该页面的上方，会显示当前用户的基本信息如 id、name 等。之后在下方会有很多的用户功能。

地址管理一栏点进去可以看到目前用户的所有地址汇总，当然如果该用户没有地址的话，可以点击新增地址，对于每条地址信息都支持编辑或者删除。

用户还可以选择变更用户名，当然要更改的用户名是不允许和其他用户的用户名一样的。用户也可以进行密码的变更，当然新旧密码也是不允许相同的。本系统还支持用户注销，当用户确认注销之后，会在后台将与该用户相关的信息都进行清除。最下面是点击退出按钮，点击后几秒钟之后就会回到最开始的登录界面。

本系统的用户个人信息界面如图 5-11 所示：

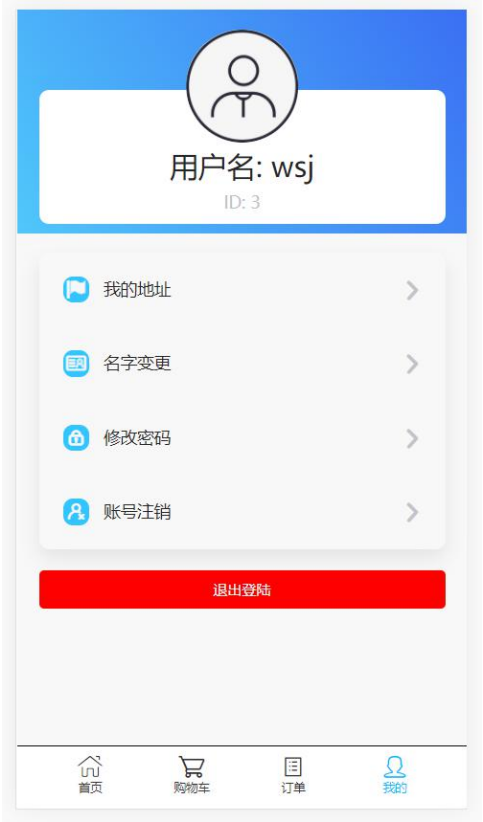


图 5-11 用户个人信息界面图

本系统的用户新增地址界面如图 5-12 所示：



图 5-12 用户新增地址界面图

增加地址的关键代码如下：该函数将获得到的前端地址信息加入地址表中来完成地址添加操作。

```
# 增加地址逻辑
@app.route('/addAddress', methods=['POST', 'GET'])
def addAddress():
    session = DBSession()
    addressInfo = request.json
    new_address = Address(city=addressInfo['city'], name=addressInfo['name'],
                          phone=addressInfo['phone'], houseNumber=addressInfo['houseNumber'],
                          user_id=addressInfo['userId'], department=addressInfo['department'])
    session.add(new_address)
    session.commit()
    session.close()
    return 'ok'
```

地址展示的关键代码如下：该函数通过前端传入的用户 id 来查询该用户下的所有地址信息返回给前端。

```
# 地址展示逻辑
@app.route('/showAddress', methods=['POST', 'GET'])
def showAddress():
    session = DBSession()
    _id = request.args.to_dict()
    addresses = session.query(Address).filter(Address.user_id == _id['id']).all()
    li = []
    for item in addresses:
        li.append(item.to_json())
    dic = {'key': li}
    string = json.dumps(dic)
    session.close()
    return string
```

本系统的用户编辑地址界面如图 5-13 所示：



5-13 用户编辑地址界面图

修改地址的关键代码如下：该函数通过获得前端传来的地址 id 在表中查找该地址信息，再将从前端获得的新地址信息填入。

```
# 修改地址逻辑
@app.route('/updateAddress', methods=['POST', 'GET'])
def updateAddress():
    session = DBSession()
    addressInfo = request.json
    for item in addressInfo:
        if item == 'id' or item == 'user_id':
            continue
        if addressInfo[item] != '':
            res = session.query(Address).filter(Address.id == addressInfo['id']). \
                update({item: addressInfo[item]})
            session.commit()
            new_address = session.query(Address).filter(Address.id == addressInfo['id']).first()
            new_address = new_address.to_json()
            if new_address[item] != addressInfo[item]:
                session.rollback()
    session.close()
    return 'ok'
```

删除地址的关键代码如下：该函数通过前端传来的地址 id 来删除地址表中该 id 的字段。

```
# 删除地址逻辑
@app.route('/deleteAddress', methods=['POST', 'GET'])
def deleteAddress():
    session = DBSession()
    addressInfo = request.json
    address = session.query(Address).filter(Address.id == addressInfo['id']).first()
    session.delete(address)
    session.commit()
    session.close()
    return 'ok'
```

第二节 管理员端实现

一、管理员登录功能

在系统首页下方有一个管理员登录按钮，点击就会进到专门的管理员登录页面，同样的在登录之前也是不允许访问其他页面的。只需要在该页面输入相应的管理员账号的密码，后台对该密码进行校验，没有问题的话就能进入到管理员端首页了，在管理员端提供查看所有书籍和对所有店铺的管理。

本系统的管理员登录界面如图 5-14 所示，管理员端首页如图 5-15 所示：

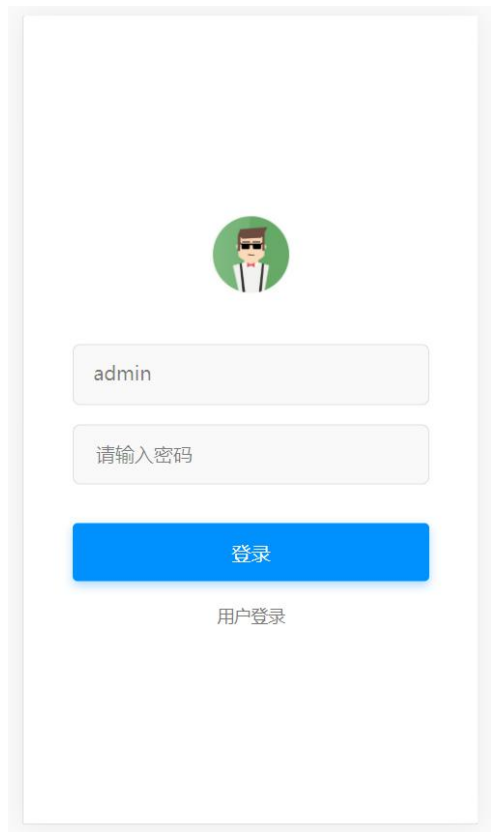


图 5-14 管理员登录界面图

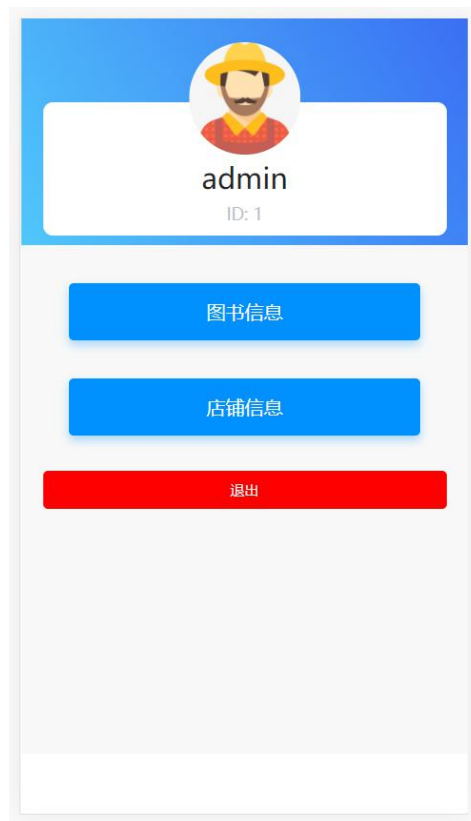


图 5-15 管理员首页图

管理员登录功能的关键代码如下：该函数获得前端输入的密码，与管理员表中的密码字段进行比较，正确的话就可以成功登录了。

```
# 管理员登录逻辑
@app.route('/admin', methods=['POST', 'GET'])
def admin():
    session = DBSession()
    adminInfo = request.json
    admin = session.query(Admin).filter(Admin.name == adminInfo['name']).first()
    if adminInfo['password'] != admin.password:
        session.close()
        return 'error'
    else:
        session.close()
        return admin.to_json()
```

二、管理所有书籍功能

在管理员首页有所有图书信息的选项, 点击的进入就可以看到当前数据库里面的所有书籍了, 同时对于每本书, 管理员都可以进行编辑和删除选项, 新增书籍也是可以的。

本系统的管理员管理书籍界面如图 5-16 所示:



图 5-16 所有书籍界面图

展示书籍的实现代码如下: 该函数将书籍表中的所有字段取出来一起打包返回给前端。

```
# 展示所有书籍逻辑
@app.route('/showAllBook', methods=['POST', 'GET'])
def showAllBook():
    session = DBSession()
    gets = request.args.to_dict()
    books = session.query(Book).filter(Book.tab == gets['tab']).all()
    li = []
    for item in books:
        item = item.to_json()
        item['imgUrl'] = "http://localhost:5000/static/img/books/" \
            + item['tab'] + '/' + item['imgUrl']
        li.append(item)
    dic = {'key': li}
    session.close()
    return dic
```

本系统的管理员增加书籍界面如图 5-17 所示：



图 5-17 增加书籍界面图

增加书籍的实现代码如下：该函数将前端发来的书籍信息加入到书籍表中完成本次添加操作。


```
# 增加书籍逻辑
@app.route('/addBook', methods=['POST', 'GET'])
def addBook():
    session = DBSession()
    bookInfo = request.json
    book = session.query(Book).filter(Book.name == bookInfo['name']).first()
    if book is not None:
        session.close()
        return 'exist'
    else:
        new_book = Book(tab=bookInfo['tab'], name=bookInfo['name'],
                        imgUrl=bookInfo['imgUrl'])
        session.add(new_book)
        session.commit()
        session.close()
        return 'ok'
```

本系统的管理员删除书籍界面如图 5-18 所示：

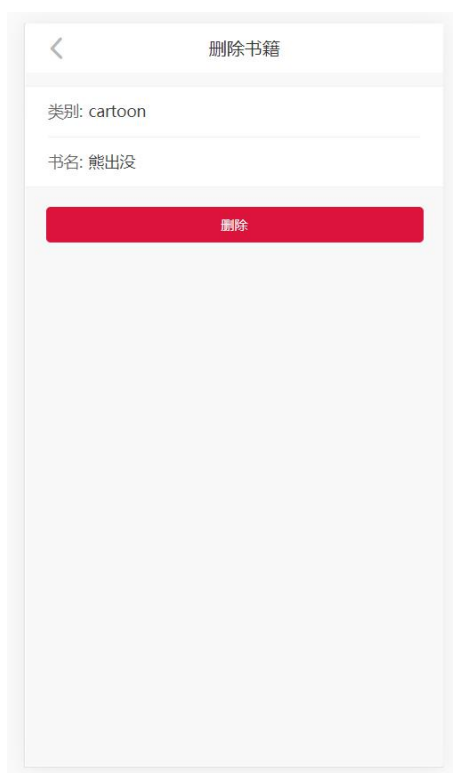


图 5-18 删除书籍界面图

删除书籍的实现代码如下：该函数通过前端传来的书籍 id 在表中找到该书籍进行删除来完成删除操作。


```
# 删除书籍逻辑
@app.route('/deleteBook', methods=['POST', 'GET'])
def deleteBook():
    session = DBSession()
    bookInfo = request.json
    book = session.query(Book).filter(Book.id == bookInfo['bookId']).first()
    session.delete(book)
    session.commit()
    session.close()
    return 'ok'
```

本系统的管理修改书籍界面如图 5-19 所示：



图 5-19 修改书籍界面图

修改书籍的实现代码如下：该函数获得前端传来的书籍 id 在表中查找该书籍，再将前端的需要更新的信息进行相应的替换。

```

# 修改书籍逻辑
@app.route('/updateBook', methods=['POST', 'GET'])
def updateBook():
    session = DBSession()
    bookInfo = request.json
    book = session.query(Book).filter(Book.name == bookInfo['name']).first()
    if book is not None:
        session.close()
        return 'exist'
    else:
        for item in bookInfo:
            if item == 'bookId':
                continue
            if bookInfo[item] != '':
                res = session.query(Book).filter(Book.id == bookInfo['bookId']). \
                    update({item: bookInfo[item]})
                session.commit()
                new_book = session.query(Book).filter(Book.id == bookInfo['bookId']).first()
                new_book = new_book.to_json()
                if new_book[item] != bookInfo[item]:
                    session.rollback()
        session.close()
        return 'ok'

```

三、管理店铺功能

在管理员首页里面有个店铺信息按钮，点击之后会进入包含所有店铺信息的页面。管理员不仅可以对所有店铺的基本信息进行增删改，同时也可以进入相应的店铺里面对该店铺里面的所有书籍进行增删改。

本系统管理员店铺信息相关操作界面如图 5-20 所示，店铺书籍相关信息操作如图 5-21 所示：



图 5-20 店铺信息界面图

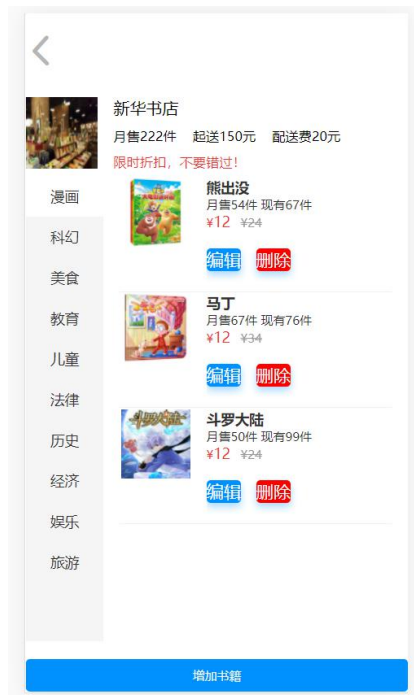


图 5-21 对应店铺书籍界面图

以上涉及的对店铺和店铺下的书籍进行操作的相关函数如下：由于相关增删改操作与之前对书籍表的操作类似，在此不再进行详细介绍。

```
# 增加店铺书籍逻辑
@app.route('/addShopBook', methods=['POST', 'GET'])
def addShopBook():...

# 获得当前店铺待操作书籍逻辑
@app.route('/theShopBook', methods=['POST', 'GET'])
def theShopBook():...

# 编辑店铺书籍逻辑
@app.route('/updateShopBook', methods=['POST', 'GET'])
def updateShopBook():...

# 删除店铺书籍逻辑
@app.route('/deleteShopBook', methods=['POST', 'GET'])
def deleteShopBook():...

# 修改店铺信息逻辑
@app.route('/updateShop', methods=['POST', 'GET'])
def updateShop():...

# 新开店铺逻辑
@app.route('/addShop', methods=['POST', 'GET'])
def addShop():...

# 关闭店铺逻辑
@app.route('/deleteShop', methods=['POST', 'GET'])
def deleteShop():...
```

第三节 本章小结

在本章中对该系统功能进行了详细的介绍。包括了两个大方面：用户端和管理员端。首先用户端，介绍了五个大功能模块分别是登录注册、查看店铺详情、购物车、提交订单、个人信息管理，其中个人信息管理模块中带有很多的小功能如更改用户名、更改密码等等。然后是管理员端，介绍了三个模块分别是管理员登录、管理所有书籍、管理店铺，管理员的操作相对简单重复，只是对数据库的记录进行简单的操作，实现起来的代码也非常的简单。

第六章 系统的测试

第一节 系统测试的目的

在完成了系统功能的开发之后，下一阶段就是进行系统的相应测试了。系统测试的目的就是为了检测本系统在运行的过程中是否能够符合设计之前的预期状况。对于在测试的过程中，如果发生了不符合预期的情况或者出现了相应的错误，要及时找准错误位置并且进行正确的修改。

系统测试的初衷就是为了使本系统的实现能够达到最初设计的那样，同时在真正实际使用之前要尽可能的不出现任何的漏洞和缺陷在，不然的话会带来很多不必要的麻烦，这也是对使用者负责任。最后，该测试也是对本系统的进一步优化操作，这主要体现在相关页面的优化和简洁上，从而让系统的整体性大大提高，这才可以被广大用户所接受和认可。

第二节 系统测试的方法

等价类划分法广泛应用于系统测试的过程中，其中最广泛的是在黑盒测试中使用，这种方法在使用的时候，可以很好的减少测试时候的工作量，同时还能提高测试的有效性和准确性。^[25]上面所提的等价类划分法，其实主要就是把系统中所输入的数据分为有效数据和无效数据两种类别。其中，有效等价类的数据属于输入合理的情况，无效等价类属于数据输入不合理的情况。在测试中是不涉及系统的代码部分的，只是根据本系统的运行情况，通过输入不同的数据来观察得到的反馈情况是否和设计时候的预期符合一致，从而来判断该功能的运行是否存在相应的一些问题。接下来是针对本系统的主要功能所展开的测试。

第三节 系统测试的过程

一、用户端功能测试

在用户端的测试中，主要针对的是登录和注册功能的测试。之前在功能介绍一章提到过，对于已有账号的用户只需要进行输入用户名和密码的登录，如果后台判断的反馈情况正确就能够成功跳转到主页面，然而对于没有账号的用户来说，首先一步就是进行账号的注册，成功注册之后便会跳转到登录页面，输入之前的注册信息就可以进行登录了。

在用户注册功能测试中，主要是有输入信息为空、该用户已经被注册、两次密码输入情况不一致、注册成功这四种用户最常遇到的情况，对这四种状态分别

进行测试，使得结果更加的真实可靠。该功能的测试结果如表 6-1 所示。注册功能的测试过程如图 6-1、图 6-2、图 6-3、图 6-4 所示。

表 6-1 用户端注册功能测试结果表

输入等价类	有效等价类	编号	无效等价类	编号
用户名、密码、确认密码	用户名、密码都可以	1	用户名为空	2
			密码为空	3
			用户名存在	4
			两次密码不一致	5

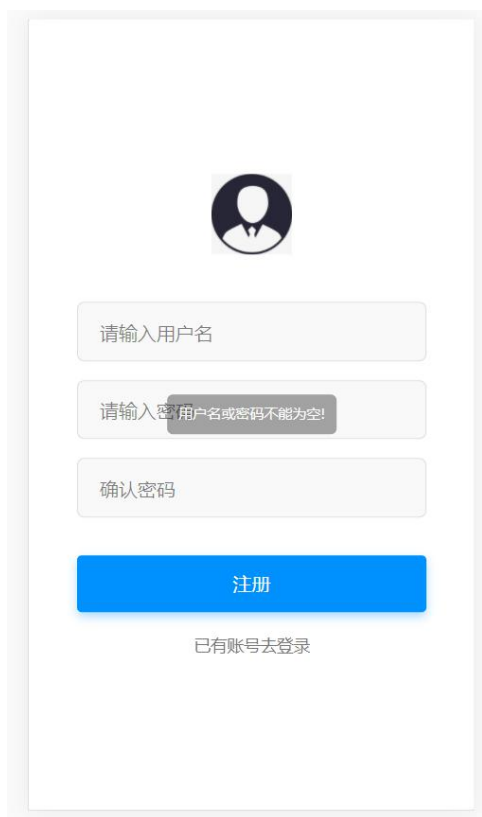


图 6-1 用户注册功能测试 01



图 6-2 用户注册功能测试 02



图 6-3 用户注册功能测试 03

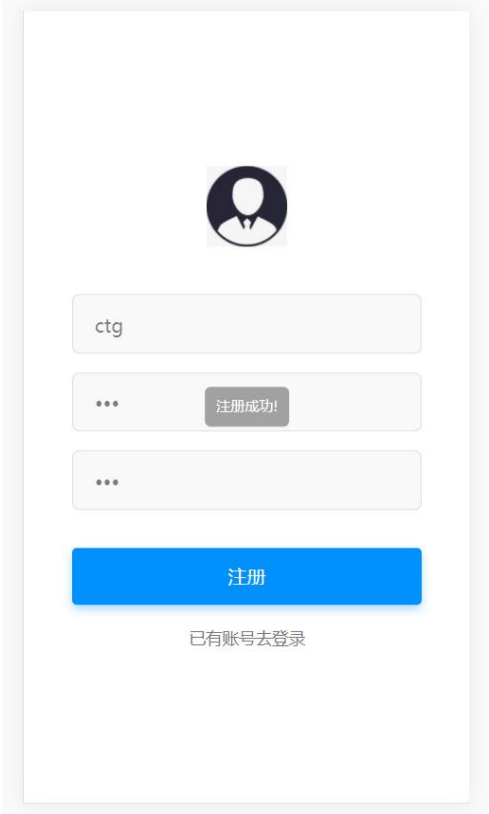


图 6-4 用户注册功能测试 04

在用户登录功能测试中, 主要是有输入信息为空、用户名或者密码输入有误、不存在该用户、登录成功这四种用户最常见的情况, 这四种情况分别进行测试, 使结果更加的真实可靠。该功能的测试结果如表 6-2 所示。登录功能的测试过程如图 6-5、图 6-6、图 6-7、图 6-8 所示。

表 6-2 用户端登录功能测试结果表

输入等价类	有效等价类	编号	无效等价类	编号
用户名、密码	用户名、密码都正确	1	用户名为空	2
			密码为空	3
			用户名不存在	4
			密码错误	5



图 6-5 用户登录功能测试 01



图 6-6 用户登录功能测试 02



图 6-7 用户登录功能测试 03

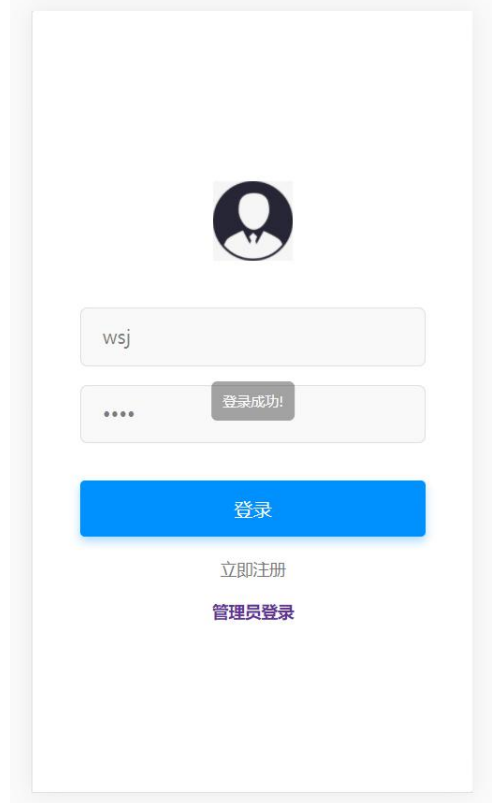


图 6-8 用户登录功能测试 04

二、管理员端功能测试

同样的，在管理员端的测试，主要是针对管理员登录的情况的一个测试。在之前也已经介绍过，管理员在进入相应的管理员页面之前也要进行登录，但与用户端不同的是，管理员的用户名是默认为 **admin** 的，管理员只需要进行相应密码的填充即可，后台会对该密码进行判断，如果没有问题的话就会跳转到管理员页面。

在管理员登录功能测试中，主要的情况就是管理员密码输入有误和登录成功两种情况，对这两种情况进行测试，使结合更加真实可靠。该功能的测试结果如表 6-3 所示。管理员登录功能测试如图 6-9、图 6-10 所示。

表 6-3 管理员端登录功能测试结果表

输入等价类	有效等价类	编号	无效等价类	编号
密码	密码正确	1	密码错误	2
			密码为空	3

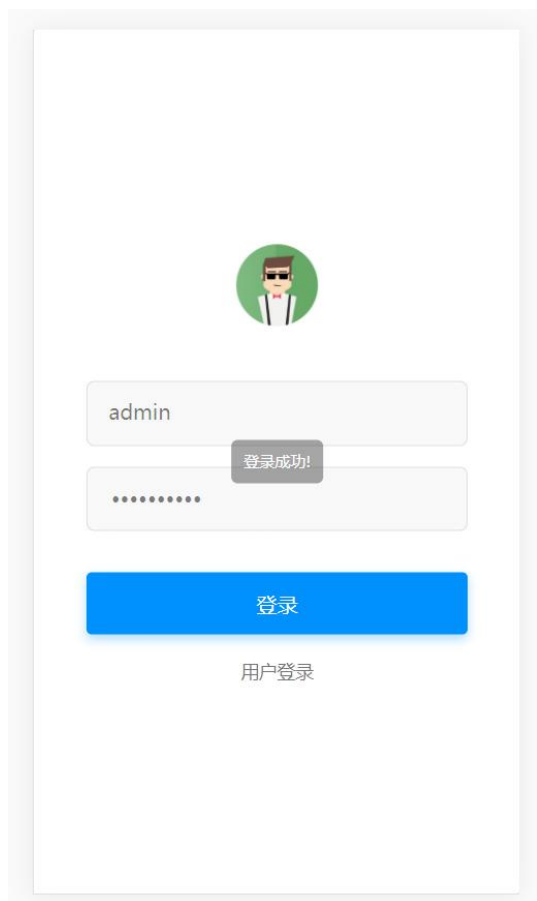


图 6-9 管理员登录功能测试 01

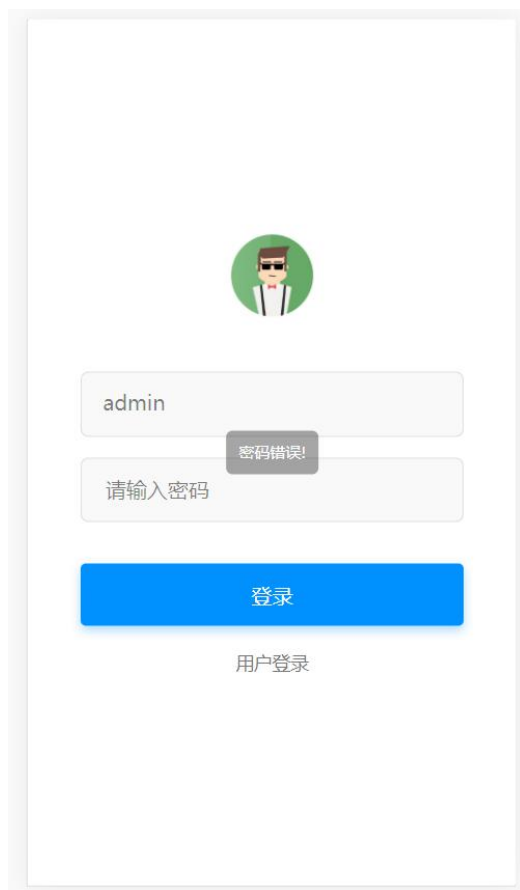


图 6-10 管理员登录功能测试 02

第四节 系统测试的总结

对于本次的系统测试，总的来说还是非常不错的，在测试的过程中发现了一些细小的问题，同时也认识到了一个系统必须要有很好的健壮性，对于各种考虑到的情况，要得到相应的一些反馈，因此系统在功能上还是有很多可以改进的地方。对于这些测试过程中发现的问题，其实都是在系统开发设计的初期遗忘的甚至是没有考虑到的东西，也许刚开始整个系统反应的问题并不会那么明显，但是到了后期，还是明显感觉到了系统健壮性的不足以及逻辑上存在的许多小问题，虽然只是一些细节上的小问题，但是你很难保证在真正的实际应用中这些问题会带来什么样的后果。对于这些问题的解决其实也就是对用户负责任，通过反反复复的系统测试，把系统中存在的细小问题找出来，并且要及时的进行相应代码的修改和精简。

通过本次系统测试，让我很深刻的体会到了，要开发设计一个真实的给用户使用的系统是多么的复杂和困难，不管是需求的构思还是真实的功能实现过程都不是轻轻松松就可以办到的，在边设计的时候更多也还需要进行深入的思考，在设计的时候会存在很多的相同的功能的反复进行，这就加大了工作量，正是不断的系统测试才可以让整个系统越来越完善，才能够更加的贴合用户使用，才能更好的创造社会价值。

第五节 本章小结

本章主要是介绍了关于系统设计完成之后的系统测试环节，包括了测试的目的以及测试方法等等。测试的过程包括用户端测试和管理员端测试。用户端测试了登录以及注册功能中的常见情况，管理员端测试了登录功能中常见的情况。最后，对本次系统测试进行了总结。

第七章 总结与展望

第一节 总结

随着网络的不断发展，电子商务也在不断的发展中，电子商务目前作为一种十分先进的手段在现代信息网络中起着十分重要的作用，这种十分创新的经济运作方式，其影响力也在慢慢超过商业邻域了。^[26]因此为了跟上这种发展潮流，每个人都应该多角度了解这种新的运营模式，积极参与进来，适应目前飞速发展的信息社会。

线上书店在其影响下也在不断发展，当然还有其他很多的邻域，或多或少的都会慢慢被影响，这种新型的运营模式已经是不可逆转的趋势了，给人们的生活带来很多方便。本系统在开发和设计的过程中，主要取得了以下研究成果。

在通过对目前我国的书店行业背景的分析中，了解了目前传统书店在发展过程中面临的许多问题和挑战，同时对于线上书店模式也进行了客观的分析以及相应的研究，在大量阅读相关文献之后了解了线上书店模式目前的发展情况以及未来的发展趋势。

在对本图书商城系统的开发设计过程中用到的相关技术知识的认真学习的基础上，详尽地分析了其中需要使用地关键性技术，同时也更加明确了还需要进一步进行什么知识的学习，以此来对这次设计提供更加强大的技术支撑。

再进一步也介绍了本系统的设计细节，使用用例图来对系统的功能进行了初步的分析。结合该分析的基础上，对于本项目会涉及到的功能模块进行更加详细分析。在找准了本项目的主要模块和结构之后，完成了本项目的设计和实现。

同时为了保证本项目具有可靠性，还进行了相应的系统测试环节，解决了本项目在开发和设计过程所遇到的一些问题，同时也积累了很多宝贵的经验。对于与本系统的相关的开发也提供了相应的参考价值。

本图书商城系统具有以下的一些特点：

1. 使用框架进行开发，不仅提高了开发效率，同时也有利于代码的维护，代码的可读性更佳。
2. 使用前后端分离开发模式，开发速度提升了很多，并且可以前后端同时开发。
3. 本系统的界面十分简洁、易懂，并且操作十分简单。

第二节 展望

项目的开发是需要不断提升的，同时也根据需求不断变化的动态过程。在经过四个多月的努力，我终于通过自己的努力完成了图书商城应用的设计，通过这次项目设计，我深刻的体会到了理论的学习和实际操作之间的差距，对于自己所学的各种知识，很多细小的地方，如果理解的不够清楚的话，在实际使用起来的时候会出遇到很多的问题。当然，在处理困难的过程中也是学习的过程，正是因此才能更好的掌握知识点，接着再学习更多的知识，拿来解决实际问题。

本次的项目设计界面比较简介美观，功能也是相对丰富。但是由于自己知识储备的不是很丰富，很多比较复杂的功能自己还没有办法实现，同时也由于自己技术水平的限制，一些功能做的比较简单，如果真的需要实际运用的话，还有很多的方面需要更进一步的完善。比如在线支付功能、定位功能以及购买完物品之后的一个物流功能等等，同时还需要一个更加完备的后台管理，最后，还有很多地方的代码设计也可以更加的简洁。

经过这几个多月的设计，我深刻明白了毕业设计并不是一项简单的任务作业，而是一次总结，不仅是对自己四年来书本知识掌握的一次考验，也是对自己课外知识积累的一次检测，更重要的是，这是一次理论转化为实际操作的过渡。这次的毕业设计不是终点，而是一次成长，教会我们勤于思考，勤于动手，让我明白成功的背后一定需要更多辛勤的汗水！

参考文献

- [1]秦喆.一种基于 XP 的测试模型的研究[D].西南大学,2010.
- [2]赵丽娟.浅谈电子商务及其物流的协同关系[J].商业时代,2014.
- [3]惠开敏. 网上书店的设计与实现[D]. 山东大学, 2009.
- [4]John Wiley ,Sons, Ltd. Amazon's Evolving Ecosystem:A Cyber-bookstore and Application Service Provider[J].Canadian Journal of Administrative Sciences Revue canadienne des sciences de l'administration 26: 332–343 (2009).
- [5]张雨露. 我国网上书店业态与发展研究[D]. 河南大学, 2013.
- [6]李春霞.黑盒测试方法探析[J].甘肃高师学报,2009.
- [7]谢新洲, 吴淑燕. 网上书店经营模式与理念分析[J]. 电子出版, 2003.
- [8]唐黎丽. 中外网上书店比较研究[D]. 华中师范大学, 2008.
- [9]张歌燕. 试析我国网上书店的优势、劣势及其发展对策[J]. 图书情报知识, 2003.
- [10]杨丽波. 浅析探索性软件测试方法[J]. 南方企业家, 2018.
- [11]麦冬, 陈涛, 梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017.
- [12]楚书来,李卫丽.基于等价类划分的黑河测试用例设计与实现[J].电脑知识与技术,2012.
- [13]William B.McGregor. Four counter-presumption constructions in shua (khob-Kwadi, Bostswana)[J].Lingua,2015.
- [14]唐斌斌, 叶奕. Vue.js 在前端开发应用中的性能影响研究[J]. 电子制作, 2020(10).
- [15]温梅标. MVVM 模式及其应用研究[J].电脑知识与技术,2020.
- [16]Paul Johson.Using MVVM Light with your Xamarin Apps[M].Apress,Berkelry,CA,2018.
- [17]杜艳美,黄晓芳.面向企业级 web 应用的前后端分离开发模式及实践[J].西南科技大学学报,2018.
- [18]石景恺.ITEM 发布系统设计与测试[D].电子科技大学,2015.
- [19]Paul Krill. Pycharm Python IED backs Apple Silicon[J].InfoWorld.com,2021.
- [20]Anonymous. EIVA Navicat Software for Catenary Simulation[J].Ocean News & Technology,2010.
- [21]Nur Atiqah Zaini,Siti Fadzilah Mat Noor,Tengku Siti Meriam Tengku Wook. Evaluation of APi Interface Design by Applying Cognitive Walkthrough[J].International journal of Advanced Computer Science and Applications,2019.
- [22]Leesley M. Praise for databases introduction[J].Elsevier, 1982.
- [23]Fontaine Rafamantanantsoa,Maherindefo Laha.Analysis and Neural Networks Modeling of Web Server Performances Using MySQL and PostgreSQL[J].Communications and Network,2018.
- [24]李辉. Flask Web 开发实战[M]. 机械工业出版社, 2018.
- [25]Sutiah S,Supriyono S. Software testing on e-learning Madrasahs using Blackbox testing[J].IOP Conference Series;Materials Science and Engineering,2021.
- [26]Meihang Han,Dandan Tan,Yuanyuan Zhang,Yiming Zhang. The Design and Implementation of Online Bookstore[J].Advanced Management Science,2017.

致 谢

时间过的真快，经过了三个多月的努力，我最终完成了最后的毕业设计和论文的写作。从最开始确定毕业设计题目到系统的实现，再到论文文章的完成，这里的每一步对我而言都是一次挑战，同时这也是我在大学期间独立完成的重大项目。在这段时间里，我学到了很多新的知识，同时对于自己四年来所学知识也进行了很好的巩固。从最开始的茫然，到之后的独立学习，查看相关的资料和文献，使得自己头脑中模糊的概念逐渐清晰，让自己非常稚嫩的项目一步步变得更加完善，在这里面的每一次改进都是我学习的收获，每一次试验的成功都会让我倍感惊喜，这也对自己学习成功的一次肯定。

我的论文作品还不是非常成熟，还是存在很多的不足之处，但是这一次写论文的经历让我终身受益。我深刻的体会到写论文是要真真正正用心去做的一件事情，是真正的自己学习的过程和研究的过程，没有学习就不会有研究的潜力。没有自己的研究，就不会有所突破。我期望这次的经历可以让我在之后学习生涯中激励我继续前进。

我想在此感谢我的女朋友张玲妍女士，感谢四年以来的陪伴，特别是在过去一年的考研时光里，有你的鼓励和支持，让我倍感温暖，我也很高兴最终实现了自己成为研究生的目标。虽然短短几句话语不足以表达我的感情，但我想说，未来还长，我们还长。在此祝你毕业快乐，希望你能很快适应接下来的工作阶段，在之后的生活中万事顺心。

在最后，我要感谢我的指导老师，杨光磊老师，他在这个过程中给了我很多宝贵的意见和指导，让我受益匪浅。同时我也要感谢在毕业设计和论文写作过程中给予我支持和鼓励的家人、朋友以及 435 的各位室友们。在此也谢谢各位论文评委老师们的辛苦工作！