廣西農業職業技術大學

GUANGXI VOCATIONAL UNIVERSITY OF AGRICULTURE

职业本科毕业论文(设计)

题目：基于 MVC 手工工艺品专卖店订购系统的设计与实现

学　　院：信息工程学院

专　　业：计算机应用工程

班　　级：

学　　号：

学生姓名：

指导教师姓名：

指导教师职称：

二〇二四年十二月

诚信声明书

本人郑重声明：所呈交的毕业论文（设计）是本人在导师的指导下独立进行研究、写作的成果，除文中已经注明引用的内容和致谢的地方外，本论文（设计）不包含任何其他个人或集体已经发表或撰写的研究成果。对本论文（设计）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

如本人的毕业论文（设计）涉及抄袭或剽窃等行为，本人愿承担由此造成的一切责任及后果。

论文作者签名： **日期：**

摘要

随着互联网技术的发展，线上购物已深入人们的日常生活。本项目设计并实现了一套基于 B/S 架构的手工工艺品专卖店订购系统，旨在提供便捷、高效的购物体验。系统包括商品管理、购物车管理、预订管理、订单支付、出入库记录等核心功能，并通过前后端分离提升了可扩展性和响应速度。

首先，进行了详细的用户需求分析，确定了商品浏览、购物车管理、订单支付、商品预订、退款申请及商品评论等功能。根据这些需求，前端界面设计简洁易用，用户可浏览商品、加入购物车、预订无库存商品、支付并生成订单，选择收货方式。系统还支持用户查看个人信息、历史订单，并进行商品评价。

后台管理功能包括商品的增删改查、上下架操作、用户门店购买及出库管理，便于统计和控制库存。管理员可以实时查看库存，自动记录商品的入库与出库，并管理订单状态，确保数据一致性和系统高效运行。同时，管理员可导出报表，便于库存和订单管理。

系统采用 Spring Boot 3.3.4 作为后端框架，MyBatis-Plus 简化数据库操作，并通过 Spring Security 进行权限管理。前端使用 Vue 3 与 Vite，Pinia 进行状态管理，Axios 处理数据交互，Element Plus 提供 UI 组件，确保了系统的高效联动。

本系统成功实现了手工工艺品订购、订单管理、库存控制及退款处理等功能，大大提高了专卖店的管理效率。用户能够流畅浏览商品，预订商品并完成支付，管理员则可实时管理订单和库存，确保商品信息的准确性和及时性。系统为专卖店的运营提供了技术支持和高效管理。

关键词：MySQL 技术；SpringBoot 框架；预订管理；出入库记录

Design And Implementation Of Order System For Handicraft Store Based On MVC

ABSTRACT

With the development of Internet technology, online shopping has penetrated into People's Daily life. This project designed and implemented a set of manual crafts store ordering system based on B/S architecture, aiming to provide convenient and efficient shopping experience. The system includes commodity management, shopping cart management, reservation management, order payment, warehouse records and other core functions, and improves scalability and response speed through front and back end separation.

First of all, the detailed analysis of user needs is carried out, and the functions of product browsing, shopping cart management, order payment, product reservation, refund application and product review are determined. Based on these requirements, the front-end interface is designed to be simple and easy to use, allowing users to browse items, add to a shopping cart, order out-of-stock items, pay and generate orders, and select delivery methods. The system also allows users to view personal information, order history, and evaluate products.

The background management function includes the addition, deletion, modification and inspection of the goods, the operation of the shelves, the purchase of the user store and the management of the warehouse, which is easy to count and control the inventory. Administrators can view inventory in real time, automatically record incoming and outgoing goods, and manage order status to ensure data consistency and efficient system operation. At the same time, administrators can export reports to facilitate inventory and order management.

The system uses Spring Boot 3.3.4 as the back-end framework, MyBatis Plus simplifies database operation, and implements permission management through Spring Security. The front-end uses Vue 3 for state management with Vite and Pinia, Axios handles data interaction, and Element Plus provides UI components to ensure efficient system linkage.

The system has successfully realized the functions of manual crafts ordering, order management, inventory control and refund processing, which greatly improves the management efficiency of the store. Users can smoothly browse products, order goods and complete payment, and administrators can manage orders and inventory in real time to ensure the accuracy and timeliness of product information. The system provides technical

support and efficient management for the operation of the store.

KEY WORDS: MySQL technology; SpringBoot framework; Booking management; Entry and exit records

目录

第 1 章 绪论

1.1 研究背景与意义

随着互联网技术的迅猛发展和电子商务的普及，线上购物已经成为现代消费者生活中不可或缺的一部分。传统零售模式逐渐向线上转型，特别是在手工艺品领域，越来越多的消费者选择在网上购买这些具有独特艺术价值和文化背景的商品[1]。通过互联网，消费者可以轻松地获取商品信息，并完成从浏览商品、选购、支付到提货或第三方物流配送的整个购物流程，这种方式不仅提升了购物效率，也拓展了商家的销售渠道[2]。

手工艺品作为文化传承和个性化商品的代表，近年来在电子商务平台上的需求不断增长。随着消费者购买习惯的转变，越来越多的人开始偏好在线购买手工艺品，据统计，绢花目前销量以每年 40%的速度增长，且随着生活水平及 装饰美化意识的日益提高，对绢花的需求也会越来越大[3][4]。设计一个专门的手工艺品在线订购系统，能够为商家开辟更广阔的市场，同时满足消费者对个性化、独特商品的需求。这一发展趋势不仅有助于促进文化创意产业的繁荣，也为消费者提供了更加便捷和多样化的购物体验。

此外，线上购物平台能够极大提升购物体验，消费者可以随时随地浏览商品，节省大量时间成本，并方便地进行价格和质量的对比，确保做出最佳选择。然而，随着市场需求的不断扩大，手工艺品在线销售平台的竞争愈发激烈。商家不仅需要提升商品质量和独特性，还需要优化用户体验和服务质量，才能在激烈的竞争中脱颖而出。

在这种背景下，库存管理成为商家面临的重要挑战。手工艺品具有独特性和少量生产的特点，因此如何有效管理库存，避免产品短缺或滞销带来的损失，是商家急需解决的问题。一个高效的在线订购系统能够帮助商家更好地管理库存，优化资源配置，提升整体运营效率[5]。

1.2 研究现状

近年来，手工工艺品订购系统在全球范围内得到了快速发展，尤其是在国外市场。平台如 Etsy 和 ArtFire 等专注于手工艺品的电商平台逐步成熟，为独立的手工艺者提供了全球化的展示和销售渠道。这些平台通过社区运营模式有效提升了用户参与度和品牌忠诚度，增强了消费者与创作者之间的互动，推动了平台的良性发展。然而，这些平台主要集中于大规模的电商市场，缺乏对特定手工艺品专卖店需求的针对性设计，尤其是在门店的库存管理、个性化订购和线下购买等方面存在不足。

国外的研究多集中在系统的运营模式、用户体验提升及供应链优化等方面，尤其

在多平台集成和跨境交易的便捷性上，平台通过不断优化物流体系、支付方式和跨文化沟通，提升了用户满意度和平台的竞争力。技术方面，国外手工艺品订购系统普遍采用现代化的 Web 框架，如 Spring Boot 和 Vue.js，结合前后端分离架构，确保了系统的高扩展性和维护性。通过采用 MVC 架构，系统能够动态响应用户需求，提供定制化服务，从而提高了系统的性能和灵活性[6][7][8]。但尽管技术不断进步，国外市场仍然缺少以手工艺品专卖店为核心的小规模、定制化的订购系统，这在个性化需求和线下购买场景中仍存在较大挑战。

国内手工工艺品订购系统的建设也在近年来得到了快速发展，得益于“互联网+”和新零售模式的兴起。根据数据显示，2019 年中国的电子商务交易额达到 35 万亿元，其中网上零售额为 10.63 万亿元，同比增长 16.5%[9]。手工工艺品作为具有文化传承和艺术价值的商品，吸引了大量消费者的关注。与传统线下销售模式相比，线上订购系统能够突破地域限制，满足消费者对个性化和独特商品的需求，因此，专注于手工工艺品的线上订购系统逐渐成为国内市场的研究热点。

目前国内手工工艺品订购系统的研究多集中在系统设计、用户体验优化、供应链管理和文化传承等方面。国内研究者普遍采用 B/S（浏览器/服务器）架构，结合 Spring Boot、Vue.js 等技术框架进行开发，以确保系统的稳定性、可维护性和扩展性[10][11][12]。然而，尽管许多系统在商品管理、库存控制和用户互动方面取得了一定进展，但国内市场仍缺乏专门针对手工工艺品专卖店的定制化系统。现有的系统更多侧重于商品的简单管理和购买流程，较少关注线下购买场景和手工艺品的个性化定制需求。

技术方面，国内手工工艺品订购系统也普遍采用 Spring Boot 和 Vue.js，结合前后端分离架构提升系统的响应速度和灵活性[13]。这些技术架构能够有效支撑用户对手工艺品的个性化需求，但依然存在诸如库存管理不精细、订单管理复杂等问题，特别是在处理线下购买、退货和退款等特殊场景时，系统往往缺乏高效的支持[14]。

总的来说，尽管国内外手工工艺品订购系统的技术不断发展，并取得了显著进展，但仍面临许多挑战，尤其是在针对专卖店需求、线下购买和个性化服务的优化方面。未来的发展方向应注重系统的定制化和场景化，提升用户体验，特别是在库存管理、订单处理和退货退款等环节的高效性和精确性，满足手工工艺品专卖店运营和消费者的需求。

1.3 本文主要工作

本文主要研究的是基于 Spring Boot 与 Vue 的手工工艺品订购设计与实现。具体研究内 容如下：

(1) 对目前订购普遍存在的问题进行调查研究。通过对订购系统目前的国内外 现状、发展背景及其意义进行分析，了解消费者对手工工艺品订购的体验感和功能需求。

(2) 通过消费者的需求对搭建手工工艺品订购所采用的技术框架、开发语言以及开

发 工具进行研究，选出最适用的研发架构。

(3) 以消费者对订购系统的需求为基础，对订购系统的架构设计、功能需求进行分析研究。

(4) 对订购系统的具体功能模块划分以及数据库设计进行分析研究。

(5) 基于用户商品购买率，对销售排行靠前的商品进行推荐。

(6) 对订购系统的具体业务逻辑进行分析

(7) 对手工工艺品订购系统进行测试分析研究

1.4 本文组织结构

本文主要分为七个章节来分析基于 Spring Boot 与 Vue 的订购系统设计与实现。其中具体的内容如下：

第一章：绪论。通过对目前订购系统背景的调查研究，指出其研究意义。并针对订购系统目前的国内外现状以及存在的部分问题，提出解决方案。还对本论文的研究内容 以及论文结构做简要介绍。

第二章：关键技术。主要介绍了研发订购系统过程中所涉及的相关技术框架、研发语言、数据库等。其中包括：Java 语言、Spring Boot 框架、Vue.Js 框架、Mybatis-Plus 框架、Mysql 数据库、IntelliJ IDEA 开发工具。构成了订购系统实现的技术基础。

第三章：系统需求分析。通过目前订购系统所存在的问题提出新的需求，对订购系统进行更加全面的需求分析，具体做法是对订购系统进行需求上的拆分，具体分析订购系统的功能需求以及非功能需求。明确订购系统的功能需求和非功能需求指标。其次是对订购系统的概要设计，包括订购系统的架构设计、模块划分以及对数据库的设计。

第四章：系统设计。主要是系统的开发环境以及技术架构、数据库表设计、部分模块的详细设计以及业务流程图等。

第五章：系统实现。主要介绍订购系统的实现过程。包括登录注册模块、后台管理模块、线下售卖模块、物品上下架模块、导出详情模块、前台首页模块、商品浏览模块、商品详情模块、购物车模块等功能模块的具体实现。

第六章：系统测试。主要是对测试环境的介绍以及对系统功能和非功能需求的测试， 通过测试结果确定订购系统是否具有实用性和可行性。

第七章：总结与展望。该部分主要分析了全文并进行总结，发现其中存在的不足之处，并提出改正。最后对未来的工作进行展望。

第 2 章 关键技术

2.1 开发工具和环境

IntelliJ IDEA：用于后端开发，配置 JDK 17 和 MySQL 8，结合 Spring Boot 和 MyBatis-Plus，提高开发效率，快速构建数据持久层。

Visual Studio Code：用于前端开发，安装 Vue.js 插件并结合 Vite 构建工具，快速启动开发环境，实现高效热模块替换。

2.2 SpringBoot 框架

Spring Boot 通过自动配置简化开发，支持快速构建可独立运行的 Java 应用。内置 Tomcat 服务器，无需额外配置，提升了开发与部署效率。同时，它与 MyBatis-Plus、MySQL 等技术无缝集成[15]。

2.3 Vue3 框架

Vue 3 提供高效的响应式数据绑定和组件化开发，支持 Composition API，提升代码灵活性和可维护性。与 Vite 配合，提供极速启动与热模块替换，极大提升前端开发效率。同时，Vue 3 集成了 Vue Router，支持客户端路由管理，可以方便地进行页面跳转和状态管理。通过 Vue Router，开发者可以在单页应用（SPA）中定义不同的路由规则，实现不同组件的渲染和页面导航，增强用户体验[16]。

2.4 其他相关技术介绍

MySQL：开源关系型数据库，提供高效的存储、管理和查询功能，适用于存储大量结构化数据，保证数据一致性。

MyBatis-Plus：MyBatis 的增强工具，简化数据库操作，支持自动生成代码，提供高效的数据库交互方案，兼容 MySQL 和其他数据库[17]。

2.5 本章小结

本章介绍了在开发订购系统中所使用的相关技术框架。首先，介绍了 Spring Boot 框架，它简化了 Java 应用程序的开发过程。然后，介绍了 Vue.js 框架， 它用于将 Java 的数据进行在前端渲染呈现给用户可视化界面。此外，还介绍了 MySQL 作为关系型数据库管理系统和 MyBatis-Plus 作为 Spring Boot 与数据库的交互桥梁，使得操作数据库更加方便快捷。

第 3 章 系统需求分析

3.1 系统需求概述

在订购系统开发过程中，对系统的功能、性能以及数据进行分析极其重要。因为其主要面对用户，所以需要了解业务需求和用户需要，理解系统在整个环境中具体做什么，才能确定系统具体功能。 随着订购系统以及计算机技术的发展，喜欢在进行线上购物的消费者更加注重用户体验， 对于购买的商品质量、售后、个人信息安全要求日益剧增。因此在当下时代，消费者对于订购平台具有如下需求：

(1)订购系统应该随时随地的为消费者提供进行登陆和注册，且用户界面应该是一个对消费者友好的界面，消费者打开就能在订购系统中熟练的进行购物下单操作。

(2) 订购系统的界面应是美观的，使消费者在浏览购物系统商品时具有良好的购物体验。

(3) 订购系统应该提供全面的商品搜索功能，能够快速的定位到消费者搜索的商品类目，并且系统对商品的介绍以及消费者对商品的评价需醒目，使消费者能够快速的了解商品有关信息。

（4）订购系统的用户信息以及商品质量安全必须得到有效保障。通过上述分析，订购系统不仅需要满足消费者的多样化需求，还要具备多项功能，同时确保消费者的个人信息安全，并保障系统的性能。因此，针对订购系统所面对的不同用户角色，需要制定相应的功能需求，从而明确系统的具体功能。

3.2 功能需求分析

本论文的目标是设计一个基于 MVC 架构的手工工艺品专卖店订购系统，致力于为消费者和商家提供一个便捷的在线交易平台。该系统不仅支持手工艺品的预订与销售，还积极推动农村传统手工艺的保护与传承。通过展示和销售具有地方特色的传统工艺品，激发消费者对这些文化遗产的兴趣和购买欲望，进一步促进农村手工艺品的市场化与现代化发展。在该平台中，消费者可以通过浏览器随时随地快速访问系统，查看商品信息并进行购买。系统为不同用户提供了多样化的功能，主要分为普通用户、管理员、游客角色，每个角色有不同的权限和功能。通过这些功能，系统不仅满足了消费者的购物需求，也为商家提供了销售和管理的便利，为管理员提供了高效的后台管理支持。具体的角色特点以及其权限如表 3-1 所示。

表 3-1 角色特点及权限表

序号	角色	工作内容
1	游客	浏览预订系统信息 浏览商品信息
2	普通用户	浏览预订系统信息 浏览商品信息 对个人信息的查询、修改 购物车商品的添加、修改、删除 购买订单查看、取消、退款 对商品进行预订、修改、取消 对收货地址添加、删除、修改 商品评论发布 接收系统通知
3	管理员	初始化订购系统基本数据 订单的查看、取消、删除 商品信息的添加、修改、删除 查看订单、取消、删除 查看销售信息图表 查看出入库记录、导出盘点信息 通知用户预订到货 处理用户退款申请 商品评论的删除、禁言用户评论

根据上表分析，下面对各个角色的具体功能需求进行详细介绍：

(1) 游客

游客用户是未进行注册的一般用户，该类角色的系统用例图如所示图 3-1 系统角色用例图。该类角色由于未进行注册专卖店订购系统，因此只能对专卖店订购系统信息以及商品信息进行查看。

在进行商品信息查看时，游客用户可通过点击商品查看商品详情。游客用户只有注册成为专卖店订购系统普通用户，才可进行商品的购买，添加购物车等操作。

(2) 普通用户

普通用户是已经进行注册的游客用户，该类角色的系统用例图如图 3-1 系统角色用例图所示。成为普通用户后，即可进行商品购买，除此之外，普通用户还具备以下权限： 1 普通用户可使用账号、密码登陆专卖店订购系统。2 普通用户进入个人中心，可对个人基本信息进行管理，例如修改用户头像、用户名称、修改密码等。 3 普通用户在浏览商品时，可以将喜欢的商品添加至购物车，在购物车中，可对商品的数量进

行修改，也可对不需要的商品进行删除。普通用户在选择好购买商品后可进行商品结算，提交订单后，可前往支付，如果用户取消支付，在取消订单管理修改地址。 4 普通用户购买商品生成订单后，便可在订单管理中心中进行查看和维护，对于未付款的订单，可在付款时间范围内点击订单去完成付款，用户完成付款后系统会通知用户收货信息用户可根据选择的收货方式进行取货。如果用户需要购物的商品为缺货状态，5 用户可预订商品，用户预订商品将支付 80%的预订付款，商家通知到货后，用户可前往支付尾款，支付尾款后用户可根据所需的方式取货。6 普通用户可在七日内对订单发起申请退款操作，由管理员审核是否通过。

(3) 管理员

管理员是专卖店订购系统的管理人员，拥有最高的系统权限。该类角色的系统用例图如图 3-1 系统角色用例图所示。管理员的具体权限如下： 1 管理员可对专卖店订购系统进行管理，轮播图内容等， 管理员可对其进行查看、修改和删除。 2 管理员可对商品进行管理，管理员可查看专卖店订购系统的所有商品信息，并且对商品的分类以及商品的基本信息可进行修改以及删除。同时还可添加新的商品。 3 管理员可对订单进行管理，管理员可以查看专卖店订购系统内的所有订单信息及其订单详情，并对不合理的订单进行取消，对所有预订的订单进行通知到货处理。 4 管理员可对用户进行管理，管理员可以新增新的管理员用户。 5 管理员可对系统进行管理，管理员可以初始化专卖店订购系统的基本数据。6 管理员可对用户申请的退款进行处理，7 管理员可查看出入库记录导出盘点信息等。8 管理员可查看销售额图表，销售排行前十图表等。

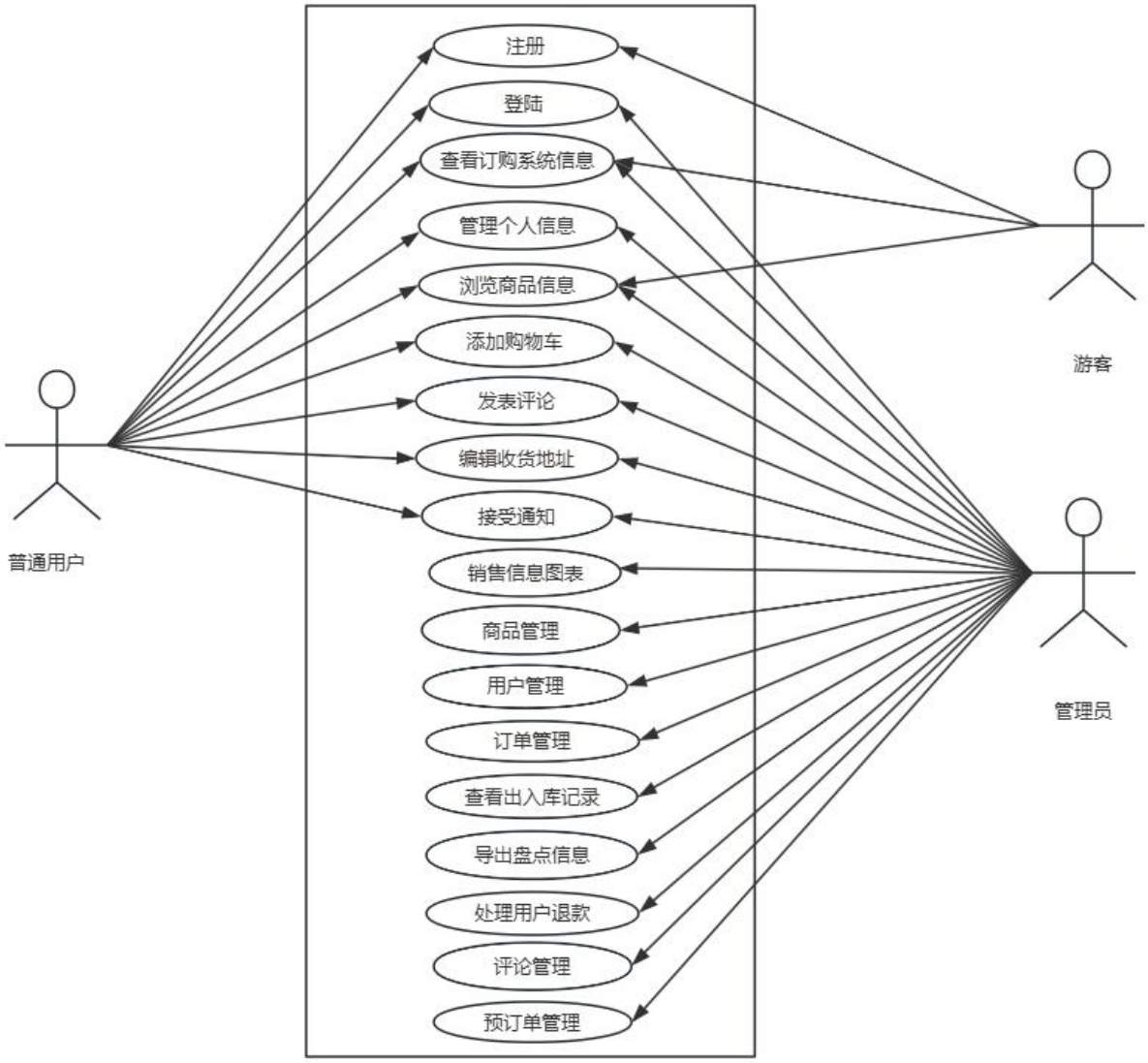
注册
登陆
查看订购系统信息
管理个人信息
浏览商品信息
添加购物车
发表评论
编辑收货地址
接受通知
销售信息图表
商品管理
用户管理
订单管理
查看出入库记录
导出盘点信息
处理用户退款
评论管理
预订单管理
普通用户
游客
管理员

图 3-1 系统角色用例图

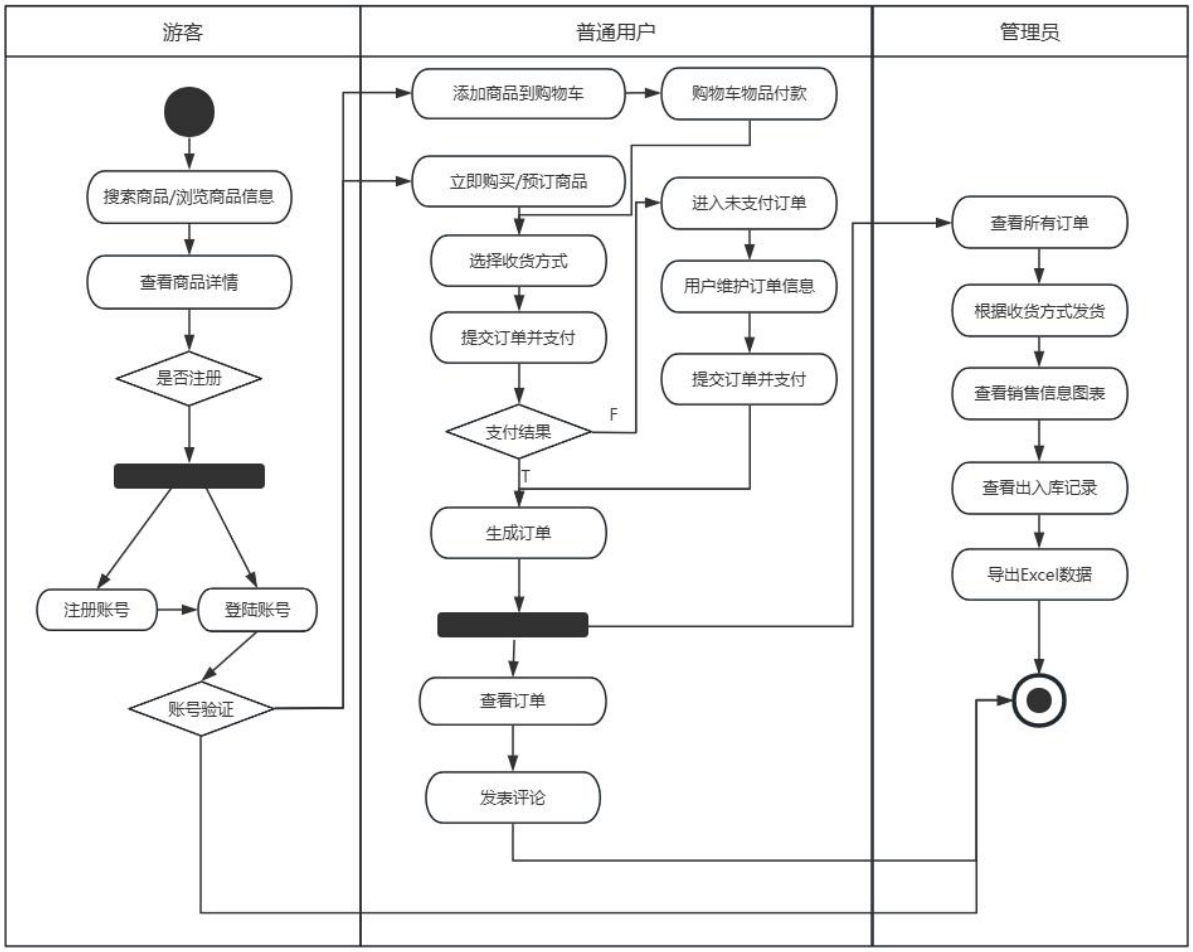
游客
普通用户
管理员
搜索商品/浏览商品信息
查看商品详情
是否注册
注册账号
登陆账号
账号验证
添加商品到购物车
购物车物品付款
立即购买/预订商品
选择收货方式
提交订单并支付
支付结果
F
T
进入未支付订单
用户维护订单信息
提交订单并支付
生成订单
查看订单
发表评论
查看所有订单
根据收货方式发货
查看销售信息图表
查看出入库记录
导出Excel数据

图 3-2　角色购物活动图

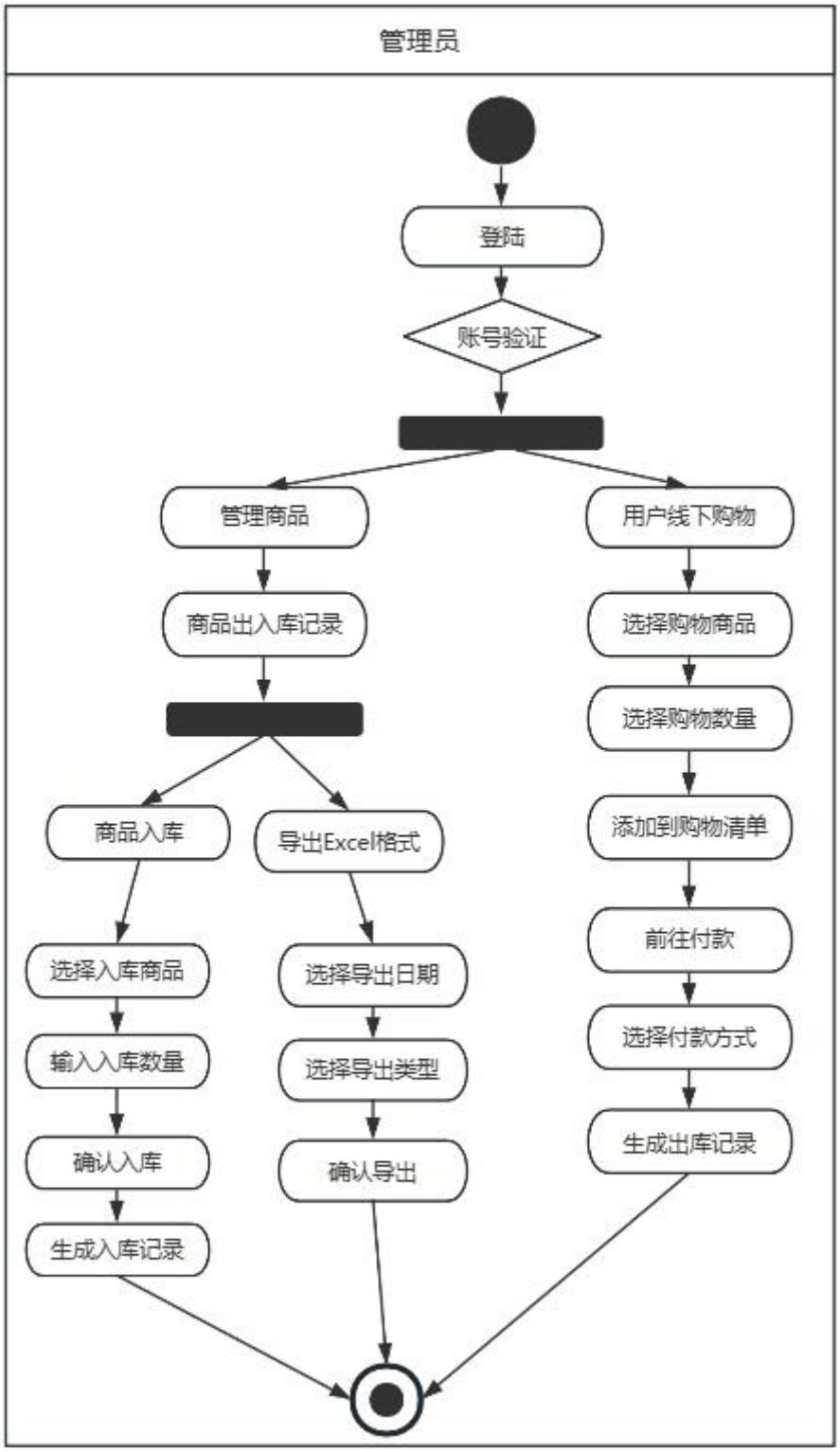

图 3-3　管理员出入库记录及用户购物活动图

3.3 本章小结

本章对专卖店订购系统的功能需求进行了详细分析，明确了系统中不同用户角色的权限和操作范围。根据用户的身份，系统划分为游客、普通用户和管理员三类角色，

并分别列出了每类用户的具体功能需求。

游客用户仅能浏览专卖店订购系统信息和商品信息，无法进行购买和其他操作。普通用户在完成注册后，能够进行商品购买、订单管理、购物车操作、商品预订和退款申请等功能。此外，普通用户还可管理个人信息并接收系统通知。管理员则拥有最高权限，负责管理系统数据、商品信息、订单、用户权限等，能对商品进行新增、修改和删除，并能够管理用户订单、处理退款申请和查看销售数据等。

通过对这些角色的功能分析，系统能够为不同的用户提供个性化的服务，确保每个用户只能访问其权限范围内的功能，增强了系统的安全性和灵活性。整体而言，本章为后续系统设计与实现提供了清晰的功能框架，确保了系统的高效运行和良好的用户体验。

第 4 章 系统设计

4.1 概要设计

前台包括购物车管理、首页设计、个人中心等模块。用户可进行加入购物车、商品搜索、订单操作、预订管理等操作。如图 4-1

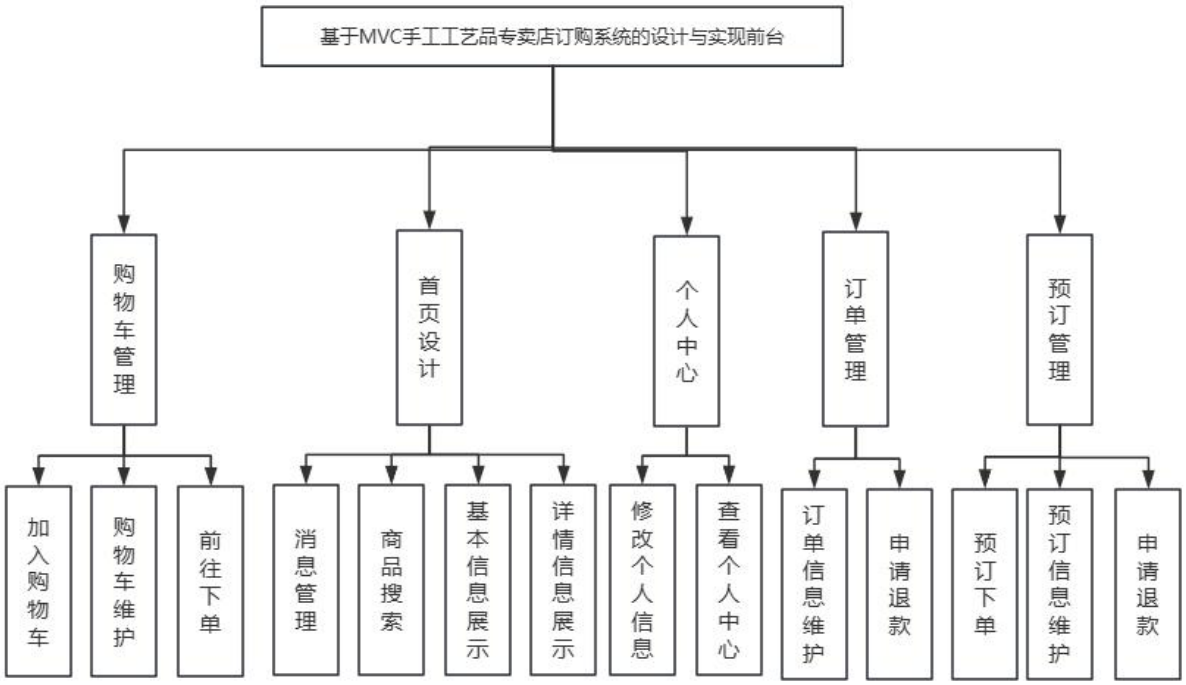

图 4-1 专卖店订购系统前台功能结构图

后台有商品管理、用户管理、订单管理等。商品管理涉及类型和物品维护等；用户管理包括用户评论、地址维护等；订单管理涵盖用户预订、审核退款等。此外，还有主页信息、营业额信息、盘点记录相关功能，包括可视化图表维护、营业额信息维护、导出报表等操作。

系统采用 MVC 架构，通过各模块协作，既方便用户选购手工艺品，又便于商家高效管理商品、用户、订单等，提升整体运营效率。如图 4-2

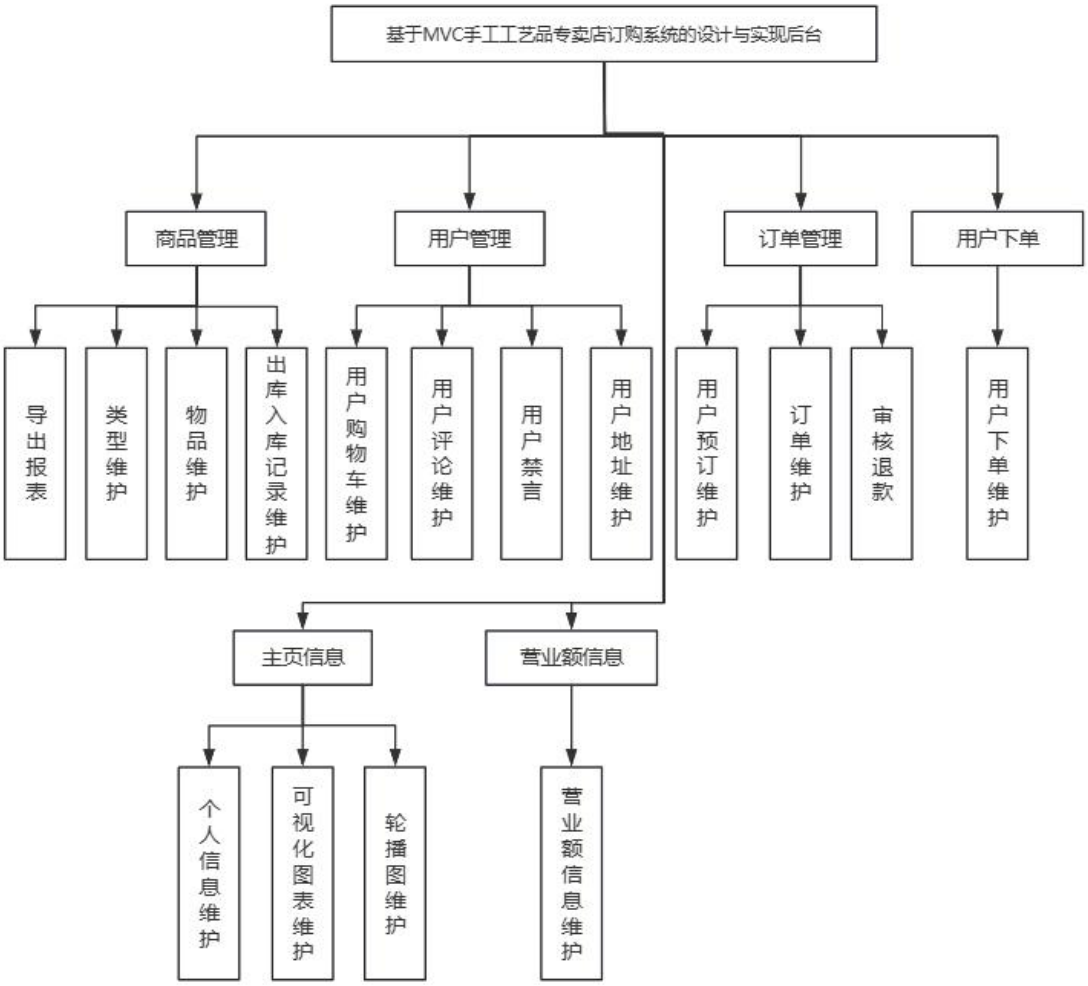

图 4-2 专卖店订购系统后台功能结构图

4.2 详细设计

4.2.1 前台功能设计

购物车管理：用户在前台页面浏览数据，在登陆成功后后可将心仪的商品加入购物车，用户在加入购物车时系统判断用户加入的数量是否大于库存数量，如果大于库存则无法加入，用户加入成功后可到购物车内管理购物车物品可对购物车物品删除、修改、查询，用户如需购买购物车数据则勾选对应数据前往付款，用户可选择收货方式，自提或者第三方邮寄，用户付款时系统会再次判定用户购买的数量是否小于库存数量。如果大于库存数量则付款失败，如果小于库存数量则付款成功并生成一个订单，并由系统通知用户。如图 4-3 所示。

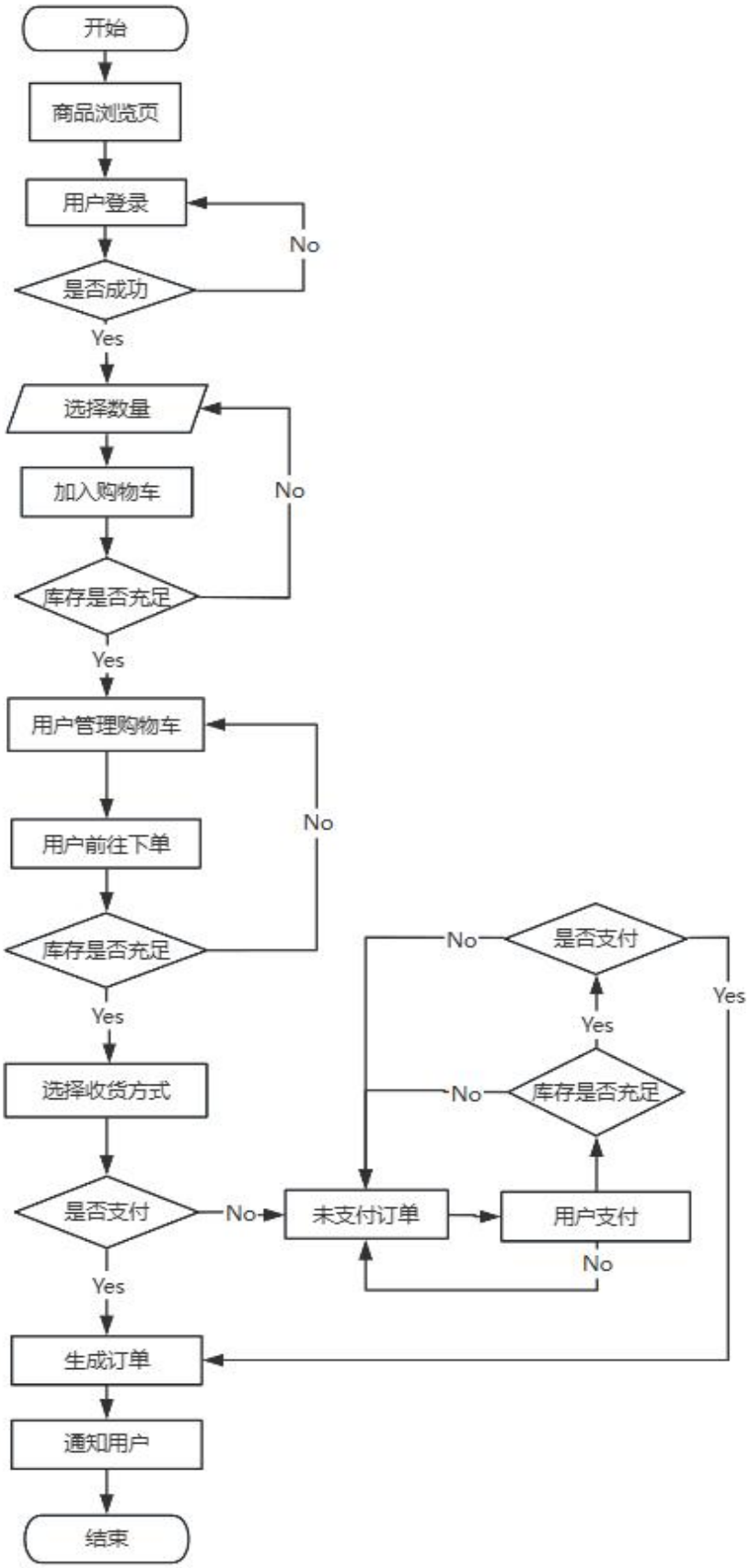
开始
商品浏览页
用户登录
No
是否成功
Yes
选择数量
加入购物车
No
库存是否充足
Yes
用户管理购物车
No
用户前往下单
库存是否充足
Yes
选择收货方式
是否支付
No
未支付订单
用户支付
No
Yes
No
是否支付
Yes
No
库存是否充足
Yes
生成订单
通知用户
结束

图 4-3 购物车流程图

预订管理：用户在前台点击商品详情时用户登陆成功后可预订无货商品，在用户选择收货方式后，用户预订的商品需要支付预订款，提交成功后将生成一条预订信息在用户预订管理中，用户可对预订的信息进行维护，如修改收货信息、申请退款等。如图 4-4。

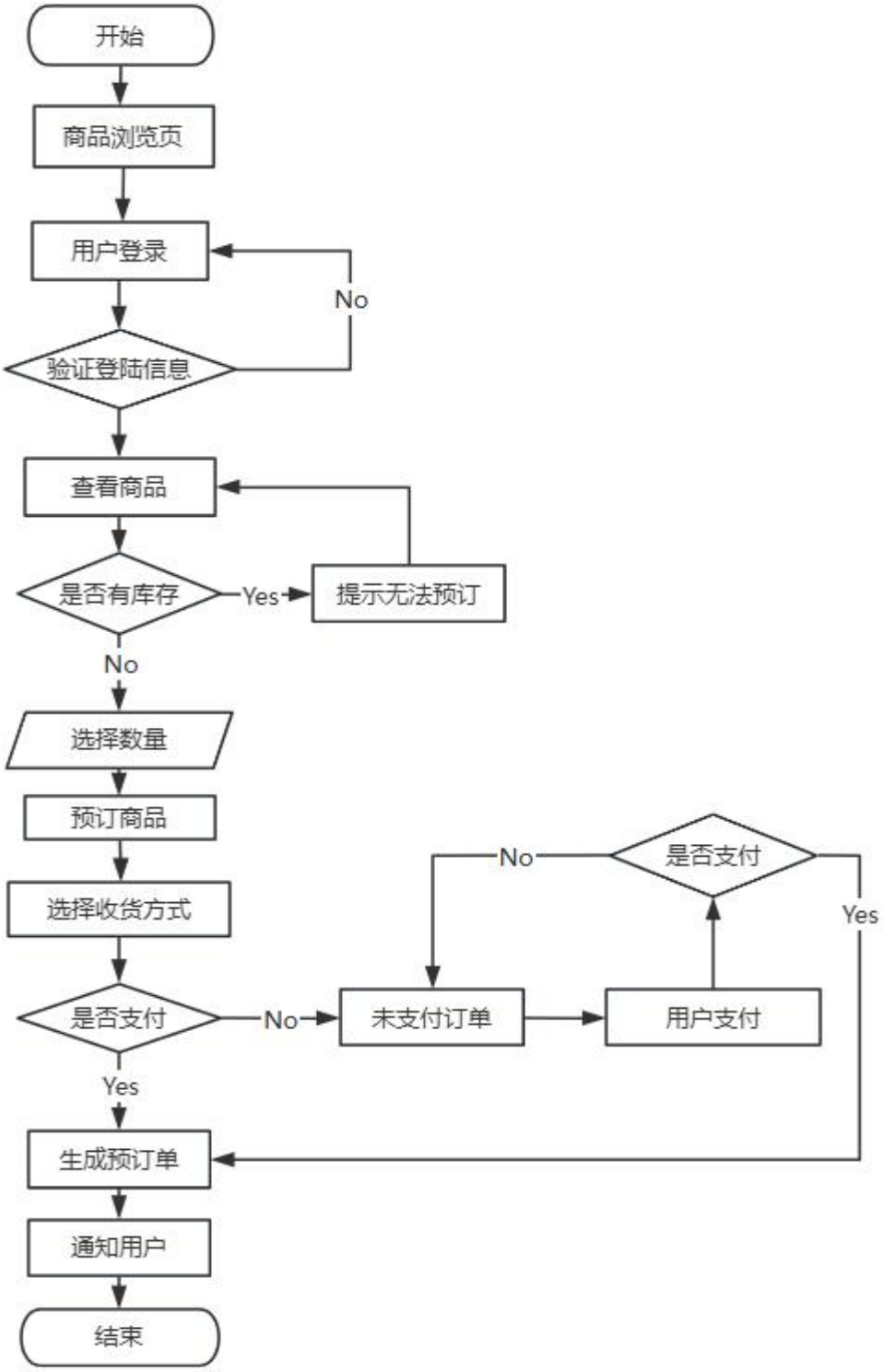

图 4-4 预订管理流程图

4.2.2 后台功能设计

用户线下购买：用户在线下购物时，商家可根据用户购买的商品选择用户需要购买的物品进行购物操作，将购买的商品选择数量后添加到清单中用户如需购买多个商品，可重复以上操作进行购物，用户选择完成后商家可选择用户需要的付款方式进行结账。如图 4-5

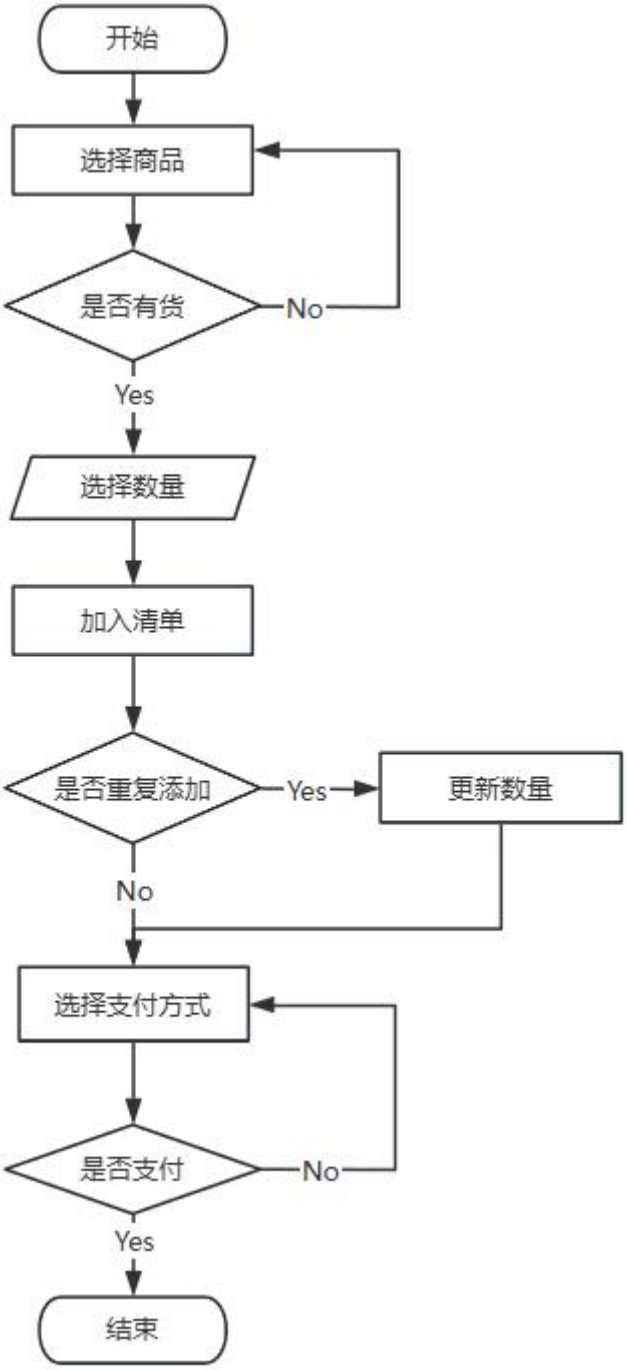

图 4-5 用户线下购买流程图

出入库记录：用户在后台有添加商品时可以在出入库记录菜单中选中添加物品入

库操作，用户可选择对应的物品进行入库操作，系统会监测商品变动情况，如果商品有变动则会自动添加到记录表中记录出库或入库数量，记录表无法删除修改等。如需盘点数据可点击导出 Excel 表格，用户在选择需要导出的日期、变动类型后可导出 Excel 在浏览器中下载。如图 4-6

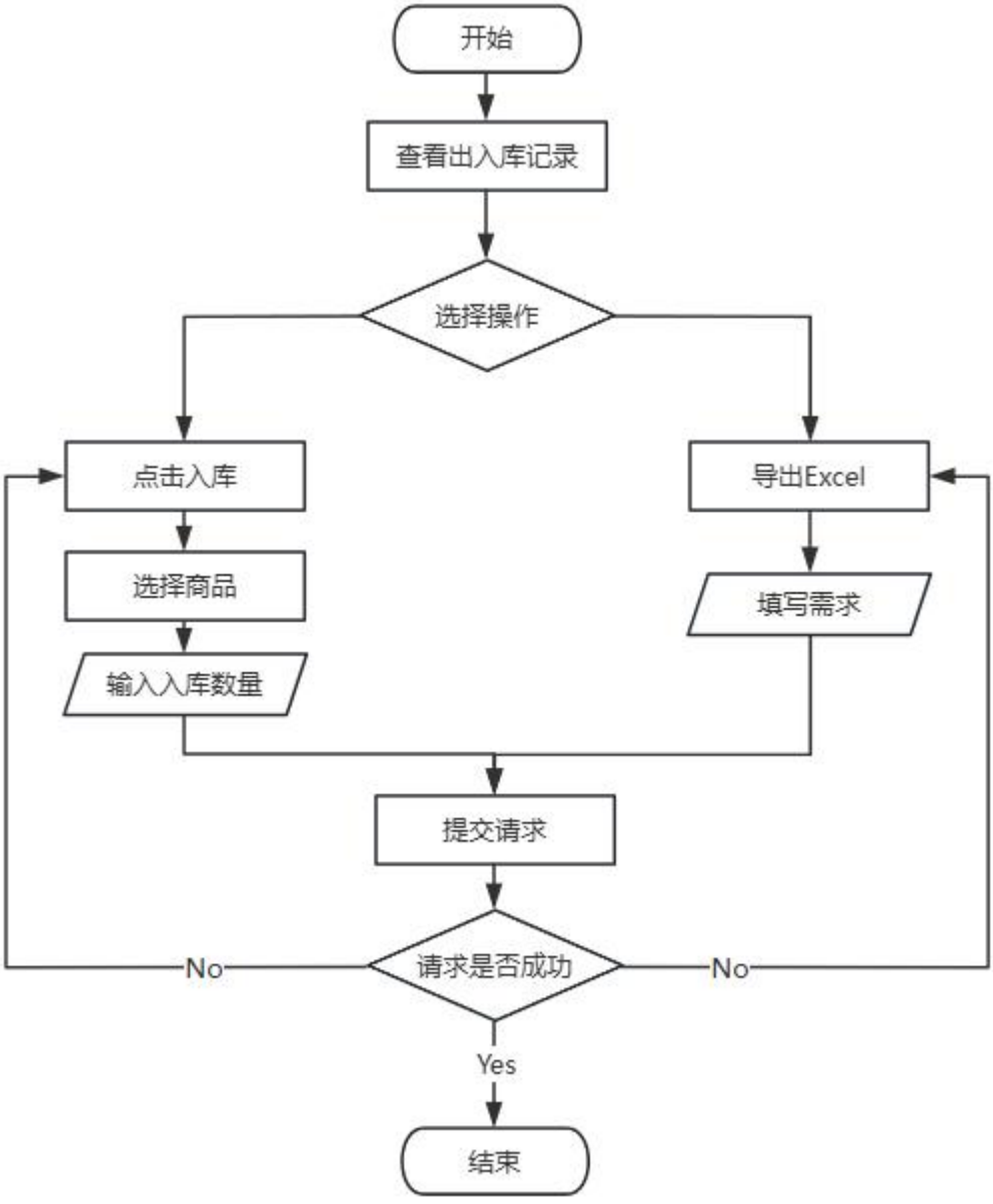

图 4-6 出入库记录流程图

4.3 数据库设计

该数据库系统设计支持手工艺品商店的用户管理、订单、支付、库存、购物车、评价和预订等功能，包含多对一、一对多和一对一、多对多关系，确保商品、用户、订单和支付数据的高效管理与操作流畅性。设计如图 4-7。

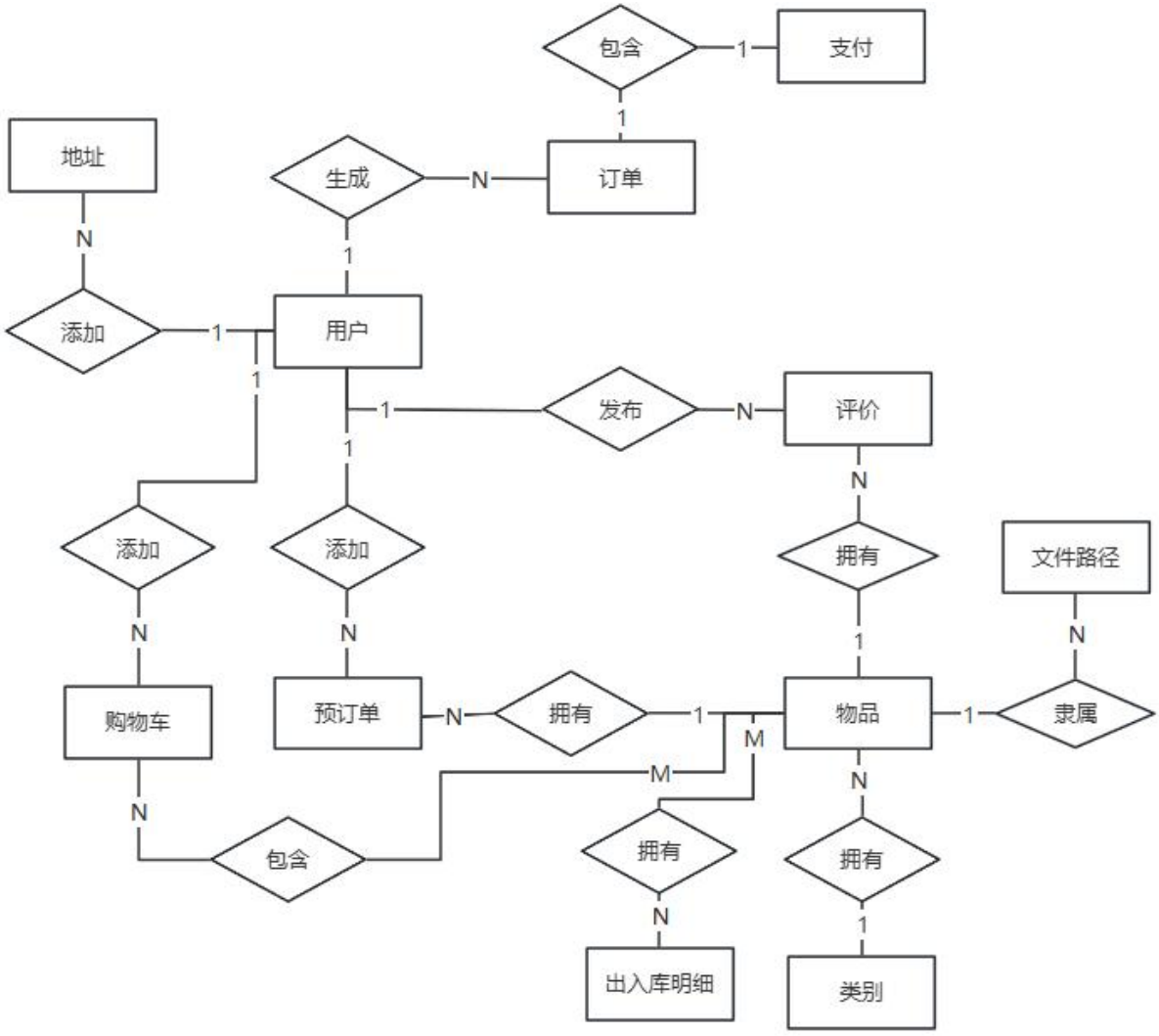

图 4-7 数据库 ER 图

一、用户表 users

用户表在系统中主要用于存储用户数据，以及登陆验证；设计如图 4-8

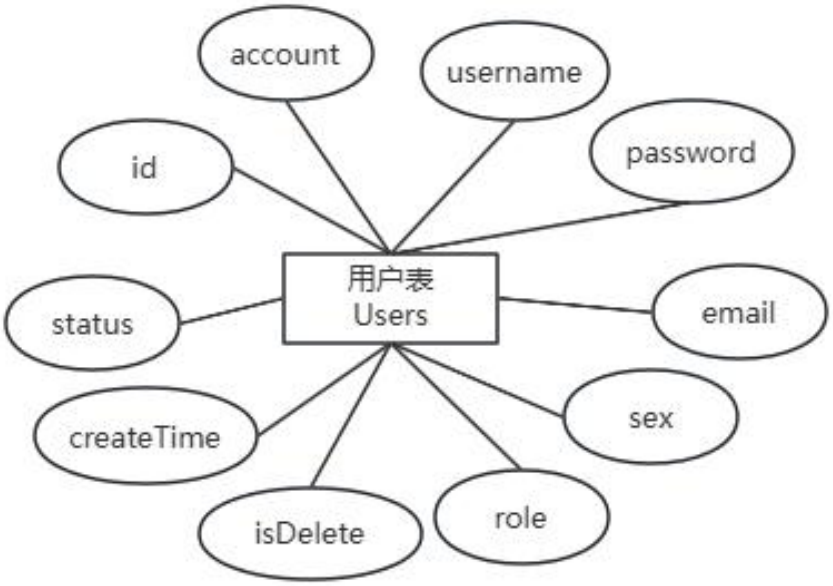

图 4-8　用户表实体类

表 4-1　用户表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
account	varchar	16	否	账号
username	varchar	50	否	用户名
password	varchar	255	否	密码
email	varchar	32	否	邮箱
sex	boolean	1	否	性别
role	boolean	1	否	用户角色
status	boolean	1	否	用户状态
createTime	datetime	19	否	创建时间
isDelete	boolean	1	否	是否删除

二、未支付订单表 unpaid_orders

未支付订单表在系统中主要用于存储用户取消支付的订单；设计如图 4-9

图 4-9　未支付订单表实体类

表 4-2　未支付订单表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
createtime	datetime	19	否	创建时间
expiretime	datetime	19	否	过期时间
payment_status	int	8	否	付款状态
reminder	boolean	1	否	提醒用户
isdelete	boolean	1	否	是否删除
delivery	varchar	25	否	购买方式
addresses_id	varchar	255	否	地址 ID

三、未支付订单物品表 unpaid_order_items

未支付订单物品表在系统中主要用于存储用户未支付的订单物品详情；设计如图 4-10。

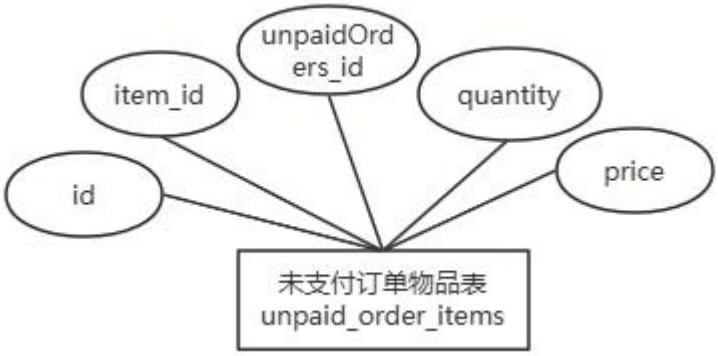

图 4-10　未支付订单物品表实体类

表 4-3　未支付订单物品表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
item_id	varchar	255	否	物品 ID
unpaidOrders_id	varchar	255	否	订单 ID
quantity	int	8	否	购买数量
price	decimal	10,2	否	购买单价

四、评价表 reviews

评价表在系统中主要用于存储用户对商品的评价；设计如图 4-11。

图 4-11　评价表实体类

表 4-4 评价表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
item_id	varchar	255	否	物品 ID
rating	int	1	否	评分
comment	varchar	255	否	评价内容
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

五、预订表 reservations

预订表在系统中主要用于存储用户预订商品信息；设计如图 4-12

图 4-12　预订表实体类

表 4-5 预订表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
item_id	varchar	255	否	物品 ID
quantity	int	8	否	预订数量
price	decimal	10,2	否	购买价格
balancePayment	decimal	10,2	否	尾款
isBalance	boolean	1	否	支付尾款
orderdate	datetime	19	否	取货日期
status	boolean	1	否	预订状态
notification	boolean	1	否	发送通知
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

六、支付表 payments

支付表在系统中用于存储用户支付信息；设计如图 4-13

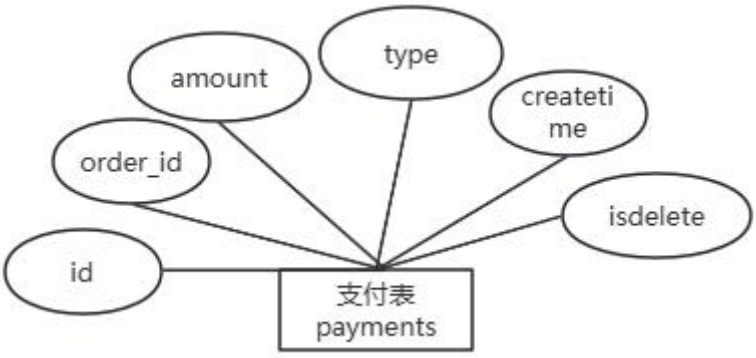

图 4-13　支付表实体类

表 4-6　支付表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
order_id	varchar	255	否	订单 ID
amount	decimal	10,2	否	支付金额
type	boolean	1	否	状态
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

七、订单表 orders

订单表在系统中用于存储用户订单信息；设计如图 4-14。

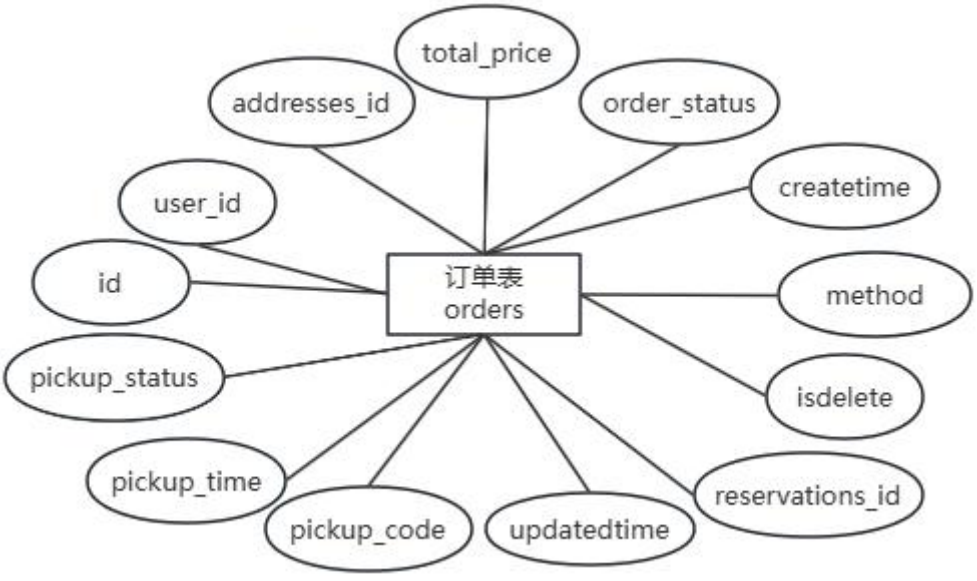

图 4-14　订单表实体类

表 4-7 订单表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
addresses_id	varchar	255	否	地址 id
total_price	decimal	10,2	否	订单总价
order_status	int	8	否	订单状态
createtime	datetime	19	否	创建时间
method	boolean	1	否	购物方式
pickup_status	int	8	否	取货状态
pickup_time	datetime	19	否	取货时间
pickup_code	varchar	10	否	取货码
updatedtime	datetime	19	否	更新时间
isdelete	boolean	1	否	是否删除
reservations_id	varchar	255	否	预订 id

八、订单详情表 order_items

订单详情表在系统中用于存储用户订单详情数据；设计如图 4-15。

图 4-15 订单详情表实体类

表 4-8 订单详情表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
order_id	varchar	255	否	订单 ID
item_id	varchar	255	否	物品 id

续表 4-9　订单详情表结构

字段名	数据类型	长度	是否主键	描述
quantity	int	8	否	购买数量
price	decimal	10,2	否	购买价格
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

九、通知表 notifications

通知表在系统中用于存储通知用户信息；设计如图 4-16。

图 4-16　通知表实体类

表 4-10　通知表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
message	varchar	255	否	通知内容
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除
readNot	boolean	1	否	是否已读

十、物品表 item

物品表在系统中用于存储商品信息，并用于管理商品展示，上架下架等；设计如图 4-17。

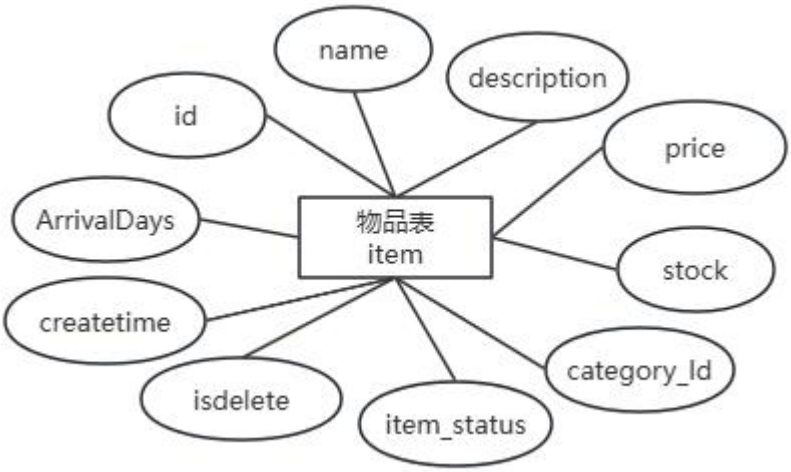

图 4-17　物品表实体类

表 4-11　物品表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
name	varchar	16	否	工艺名称
description	varchar	255	否	描述
price	decimal	10,2	否	物品价格
stock	int	8	否	库存数量
category_Id	varchar	255	否	类型 id
item_status	int	8	否	物品状态
ArrivalDays	int	8	否	到货天数
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

十一、库存明细表 inventory

库存明细表在系统中用于存储出库入库信息；设计如图 4-18。

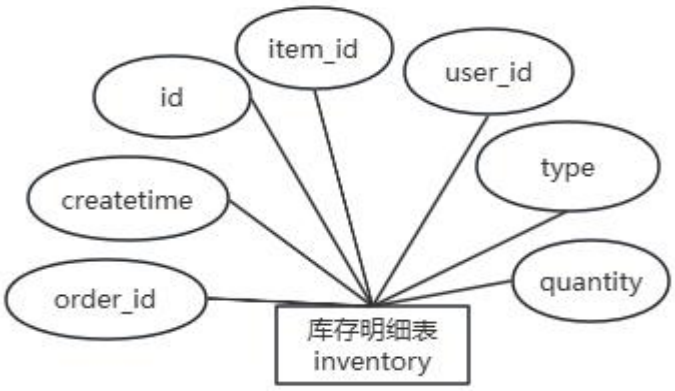

图 4-18　库存明细表实体类

表 4-12　库存明细表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
item_id	varchar	255	否	物品 ID
type	int	8	否	变动类型
quantity	int	8	否	变动数量
createtime	datetime	19	否	创建时间
order_id	varchar	255	否	订单 ID

十二、文件路径表 file_paths

文件路径表在系统中用于存储系统图片地址，提供给前台读取图片信息；设计如图 4-19。

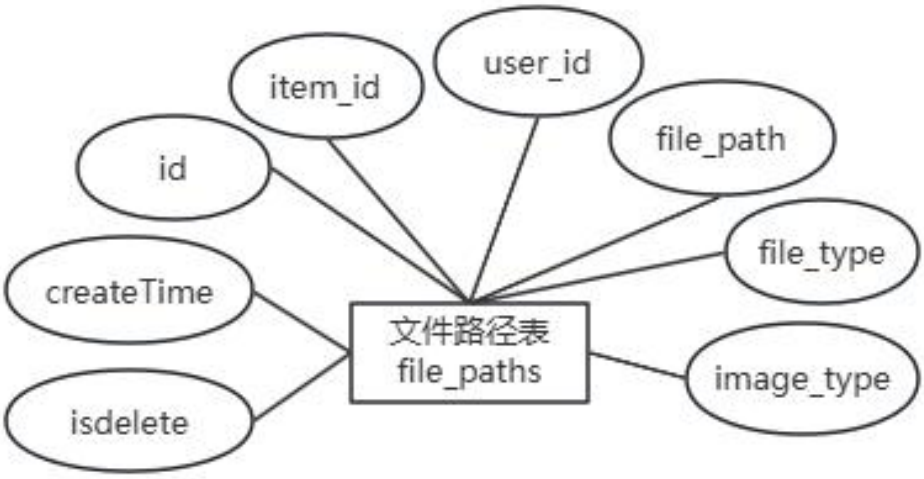

图 4-19　文件路径表实体类

表 4-13　文件路径表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
item_id	varchar	255	否	物品 ID
user_id	varchar	255	否	用户 ID
file_path	varchar	255	否	文件路径
file_type	varchar	25	否	文件类型
image_type	enum	255	否	图像类型
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

十三、类型表 category

类型表在系统中用于存储物品类型，提供物品表选择类型数据；设计如图 4-20。

图 4-20　类型表实体类

表 4-14　类型表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
name	varchar	255	否	类型名称
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

十四、购物车表 cart_items

购物车表在系统中用于存储用户加入购物车的数据；设计如图 4-21。

图 4-21　购物车表实体类

表 4-15　购物车表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
item_id	varchar	255	否	物品 ID
quantity	int	8	否	购买数量

续表 4-16　购物车表结构

字段名	数据类型	长度	是否主键	描述
price	decimal	10,2	否	价格
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除

十五、用户禁言表 banned_users

用户禁言表在系统中用于存储用户违反社区要求禁言记录；设计如图 4-22。

图 4-22　用户禁言表实体类

表 4-17　用户禁言表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
type	boolean	1	否	状态
createtime	datetime	19	否	创建时间
isdelete	boolean	1	否	是否删除
reason	varchar	255	否	禁言原因

十六、用户地址表 addresses

用户地址表在系统中用于存储用户地址信息，提供用户购买时选择地址数据。如图 4-23。

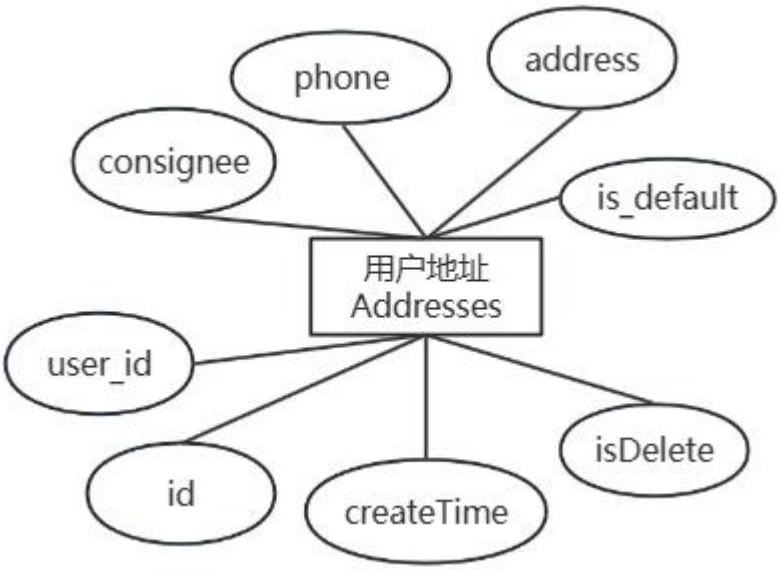

图 4-23　用户地址表实体类

表 4-18　用户地址表结构

字段名	数据类型	长度	是否主键	描述
id	varchar	255	是	ID
user_id	varchar	255	否	用户 ID
consignee	varchar	25	否	收货人
phone	char	11	否	联系电话
address	varchar	255	否	地址
is_default	boolean	1	否	默认地址
isDelete	boolean	1	否	是否删除
createTime	datetime	19	否	创建时间

4.4 本章小结

基于 MVC 手工艺品专卖店订购系统的设计分为前台和后台。前台方便用户操作，涵盖购物车、商品浏览、个人中心、订单和预订管理等功能。后台助力商家管理，包括商品、用户、订单、主页信息等管理模块。

系统采用 MVC 架构，模型层管理数据，视图层实现友好交互界面，控制器层处理业务逻辑。通过合理的数据库设计、功能模块划分与接口设计，确保系统的高效性和安全性。同时，考虑了性能优化措施，如数据库优化、缓存机制和前端优化，以保障系统流畅运行，为手工艺品专卖店的线上运营提供有力支持。

第 5 章 系统实现

5.1 开发运行环境

前端环境：前端使用了 Vue 3 和 Vite 创建的项目，提供快速的开发和构建体验。为了简化 UI 开发，使用了 Element Plus 组件库，结合了 Pinia 进行状态管理，且通过 Pinia 持久化插件 实现了数据的持久化存储。路由管理使用了 Vue Router，并使用 axios 进行 API 请求和数据交互。项目的构建和热更新依赖于 Vite，使得开发过程更加高效和便捷。

后端环境：后端使用了 Spring Boot 框架，搭配了 MyBatis-Plus 作为数据库交互层，简化了 Mysql 数据库操作。邮件功能通过 spring-boot-starter-mail 实现，支持发送邮件功能。代码中使用了 Lombok 注解库，简化了 getter、setter 等常见方法的编写，提升开发效率。

此环境配置能够确保前后端高效开发、实时调试、并通过插件和框架增强了代码的可维护性和功能的可扩展性。

前端开发环境搭配

前端开发在 cmd 中使用 vite 创建项目(本系统使用的是 vue3 版本)关键代码如下:

```
npm create vite //创建 vue.js 项目
```

创建项目成功后，通过 npm 命令对项目进行添加第三方工具依赖，关键代码如下：

```
npm install element-plus –save //安装 element-plus
npm install axios    //安装 axios
npm install pinia    //安装 Pinia
npm install vue-router –save    //安装 router
```

后端开发环境搭配

后端使用的是 SpringBoot 框架，通过 Maven 工具包添加环境依赖到项目中，关键配置如下：

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>3.3.4</version>
</parent>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.26</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-mail</artifactId>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.34</version>
</dependency>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

在以上基础搭建好项目后引入 MyBatis-Plus 依赖作为持久层与数据库交互。代码如下：

```
<dependency>
<roupId>com.baomidou</groupId>
<artifactId>mybatis-plus-spring-boot3-starter</artifactId>
<version>3.5.7</version>
</dependency>
```

以上依赖都引入完成后在项目的配置文件 application.yml 中配置项目的依赖连接，配置所需的连接数据，代码如下：

```
spring:
#   配置数据库链接
datasource:
  url:
jdbc:mysql://localhost:3306/ManualCraft?characterEncoding=utf-8&serverTimezone
=GMT%2B8
username: root
  password: 123456
#   配置邮箱信息
mail:
  host: smtp.qq.com
port: 465
username: 2308670776@qq.com    # QQ 邮箱
password: qwrppttewbceebje    # QQ 邮箱的授权码
properties:
  mail:
  smtp:
  auth: true
ssl:
  enable: true
servlet:
  multipart:
max-file-size: 10MB
max-request-size: 10MB
mybatis-plus:
  type-aliases-package: com.example.demo.Entity.pojo
type-enums-package: com.nzd.manualcraft.enums
global-config:
  db-config:
  id-type: assign_uuid
server:
```

后端开发环境配置至此后基本完成了。

5.2 用户购物车模块

用户购物车模块是专卖店订购系统的核心功能该模块在用户登陆成功后即可提供用户使用用户可将心仪的商品加入购物车中，如商品下架、缺货用户将无法加入购物车，只有商品有货时该功能才能够正常使用，用户在加入购物车后可前往购物车管理购物车数据，对添加多数量的商品数量进行修改，如用户错误的将商品加入购物车后，用户亦可将该商品清除出购物车当中，如用户需要购买商品时，则可勾选商品进行支付，在支付前系统将会验证用户选中的商品是否库存充足，如库存充足用户选择收货方式后前往付款订单即可完成。

后端添加购物车功能接口关键代码如下：

```
    /**
    *  添加到购物车
    *
    * @return
    */
    @PostMapping("/save")
    public ResponseEntity<String> SaveCartItems(@RequestBody CartItems cartItems, HttpSession session) {
    //              判断是否为空
      if (cartItems.getItemId() == null || cartItems.getItemId().trim().isEmpty() || cartItems.getQuantity() == 0 || cartItems.getQuantity() <= 0) {
          return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(" 未选择物品");
        }
      return cartItemsService.saveCartItems(cartItems,session);
    }
```

前端添加购物车功能关键代码如下：

```
const AddToCart = () => {
    if (token == null || token == "") {
        ElMessage.error("请先登录哦!");
        return;
    }
    const cartItemFrom = ref({//添加数据
        userId: token,
        itemId: props.itemId,
        quantity: num.value,
        price: props.item.price,
    });
    if (num.value < 1) {
        ElMessage({ message: "请选择添加数量", type: "error" });
        return;
    }
    axios.post("/api/cartitems/save", cartItemFrom.value)
        .then((res) => {
            if (res.status === 200) {
                ElMessage({ message: "添加成功", type:
"success" });
            } else {
                ElMessage.error("添加失败");
            }
        }).catch((error) => {
            ElMessage.error("添加失败");
        }).finally(() => {
            num.value = 0;
        });
```

后端查询购物车功能接口关键代码如下：

```
    /**
     * 分页查询数据
     */
    @GetMapping("/page")
    public Page<CartItemsDTO> queryCartItemsPage(@RequestParam(defaultValue = "1") int pageNo, @RequestParam(defaultValue = "30") int pageSize,  HttpSession session) {
        return cartItemsService.queryCartItemsPage(pageNo, pageSize,session);
    }
```

前端查询购物车功能关键代码如下：

```
    // 获取购物车数据
      const fetchCartItems = () => {
        axios.get("/api/cartitems/page", { params: { currentPage: 1, pageSize: 10 } })
          .then((res) => {
            cartItems.value = res.data.records;
          })
          .catch((error) => {
            ElMessage.error("Error fetching cart items:", error);
          });
      };
```

后端修改购物车功能接口关键代码如下：

```
    /**
     * 修改数据
     */
    @PutMapping("/update")
    public ResponseEntity<String> UpdateCartItems(@RequestBody CartItems
cartItems) {
        //          判断是否为空
        if          (cartItems.getItemId()          ==          null          ||
cartItems.getItemId().trim().isEmpty()) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("输入不能为空");
        }
        return cartItemsService.UpdateCartItems(cartItems);
    }
```

前端修改购物车功能关键代码如下：

```
    const SubmitCart = () => {
      axios.put("/api/cartitems/update", EditForm.value).then((res) => {
        if (res.status === 200) {
          ElMessage({ message: "提交成功", type: "success" });
          EditForm.value = [];
          openEditDialog.value = false;
          fetchCartItems(); // 重新加载数据
        }
      }).catch((error) => {
        ElMessage.error("提交失败");
      })
    }
```

后端删除购物车功能接口关键代码如下：

```
    /**
     * 软删除
     * @param id
     * @return
     */
    @PutMapping("/delete/{id}")
    public ResponseEntity<String> deleteCategory(@PathVariable("id")
String id) {
        //          判断是否为空
        if (id == null) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("id 不能为空");
        }
        return cartItemsService.deleteCartItems(id);
    }
```

前端删除购物车功能关键代码如下：

```
    // 点击删除
    const removeItem = (id) => {
      axios.put(`/api/cartitems/delete/${id}`)).then((res) => {
        if (res.status === 200) {
          ElMessage({ message: "删除成功", type: "success" });
          cartItems.value = cartItems.value.filter((item) => item.id !== id);
        }
      }).catch((error) => {
        ElMessage.error("删除失败: " + error.message);
      })
    };
```

加入购物车功能界面如下图 5-1 所示：

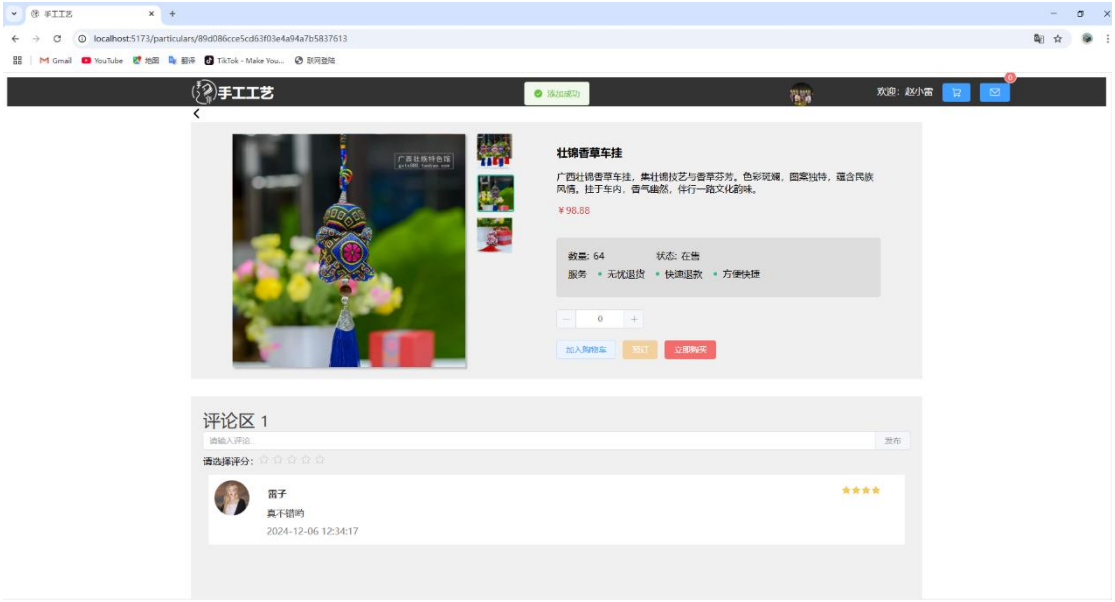

图 5-1 添加购物车界面

加载购物车功能界面如下图 5-2 所示：

图 5-2 购物车界面

修改购物车功能界面如下图 5-3 所示：

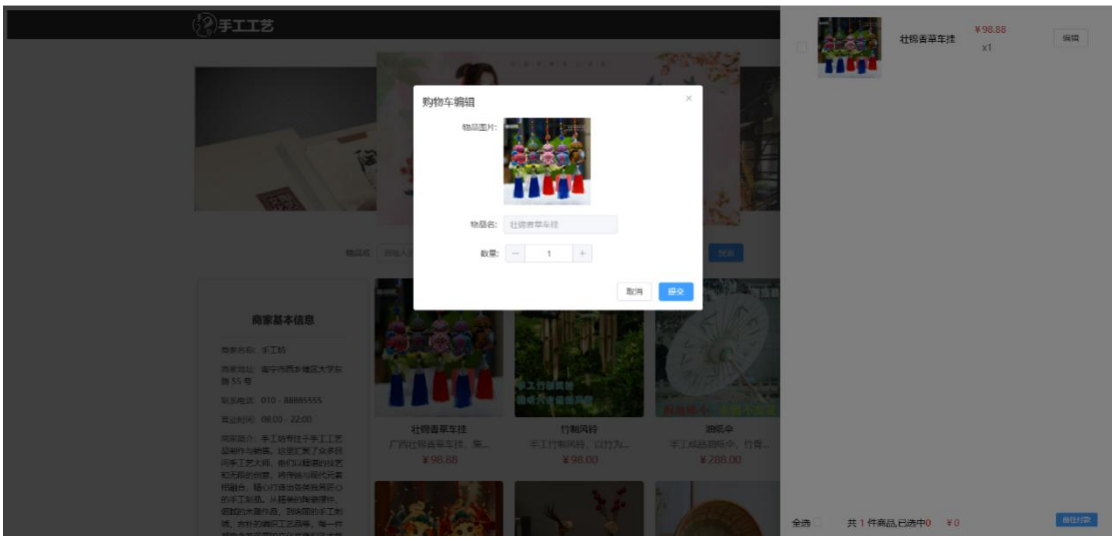

图 5-3 修改购物车界面

删除购物车功能界面如下图 5-4 所示：

图 5-4 删除购物车界面

5.3 预订管理模块

预订管理模块是专卖店订购系统的核心功能模块，在用户登录成功后即可提供用户使用订购功能，该功能只支持缺货商品预订，如商品在售或者下架不支持预订功能，当商品缺货时用户需要购买该商品可点击预订功能，用户点击预订功能后在选择收货方式前系统将验证用户是否预订过该商品，如预订过则提示用户商品已预订提示用户是否继续预订，如用户继续预订可选择收货方式后进行付款，预订款为商品总价 80%，付款成功后将会有系统通知用户预订成功，用户只需等待商家提示到货后付尾款即可按照取货方式进行取货。

后端添加预订功能接口关键代码如下：

```
    /**
    * 添加预订
      */
    @PostMapping("/save")
    public ResponseEntity<String> SaveCartItems(@RequestBody
ReservationsDTO reservationsDTO, HttpSession session) {
    //          判断是否为空
    if (reservationsDTO.getItemId() == null ||
reservationsDTO.getItemId().trim().isEmpty() || reservationsDTO
            .getQuantity() == 0 || reservationsDTO.getQuantity() <= 0) {
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("未选择物品");
            }
        return
reservationsService.saveReservations(reservationsDTO,session);
    }
```

前端添加预订功能关键代码如下：

```
      const reserve = (itemId) => {
          // 如果没有登录或者 quantity 不是大于 1 的则无法往下执行
          if (!isReserveEnabled.value) {
              return;
          }
    axios.get(`/api/reservations/selectSave/${itemId}`).then((res) => {
     if (res.status === 200) {
         popconfirmTitle.value = "已存在相同订单记录，是否继续预订？";
      } else if (res.status === 406) {
              popconfirmTitle.value = "确定预订吗？";
          }
      }).catch((error) => {
          console.error("预订出错了");
        });
     };
```

后端加载预订功能接口关键代码如下：

```
    @GetMapping("/page")
    public                                                    Page<ReservationsDTO>
queryCartItemsPage(@RequestParam(defaultValue = "1") int pageNo,

@RequestParam(defaultValue = "30") int pageSize,
                                                              @RequestParam
String name,
                                                              @RequestParam
String token, HttpSession session) {
        return    reservationsService.queryReservationsPage(pageNo,    pageSize,
name, token, session);
    }
```

前端加载预订功能关键代码如下：

```
    const loadData = () => {
        loading.value = true; //  设置加载状态
        axios
            .get("/api/reservations/page", {
                params: {
                    pageNo: currentPage.value,
                    pageSize: pageSize.value,
                    name: search.value, //  添加搜索参数
                    token: token, //  传递 token 作为筛选条件
                },
            })
            .then((res) => {
                //  根据状态字段，确保每一行都有自己的 PickUpStatus
                filterTableData.value = res.data.records.map((item) => {
                    return {
                        ...item, //  展开现有的项目属性
                          PickUpStatus: item.status === !true, //  根据实际状态设置“PickUpStatus”
                        };
                });
                TotalNumber.value = res.data.total; //  假设响应数据包含总数
                loading.value = false;
            })
            .catch((error) => {
                ElMessage.error("加载数据失败: " + error.message);
            })
            .finally(() => {
                loading.value = false; //  复位加载状态
            });
      };
```

后端修改预订功能接口关键代码如下：

```
    /**
     * 修改数据
     */
    @PutMapping("/update")
    public ResponseEntity<String> UpdateCartItems(@RequestBody
ReservationsDTO reservationsDTO) {
        //          判断是否为空
        if          (reservationsDTO.getId()          ==          null          ||
reservationsDTO.getId().trim().isEmpty()) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("输
入不能为空");
        }
        return reservationsService.UpdateReservations(reservationsDTO);
    }
```

前端修改预订功能关键代码如下：

```
        const SubmitCart = () => {
            if (deliveryMethod.value === "thirdParty") {
                if (EditForm.value.address === "" || EditForm.value.consignee ===
"" || EditForm.value.phone === "") {
                    ElMessage.error("收货信息不能为空");
                    return;
                }
            }
            const EditForms = ref({
                id: EditForm.value.id,
                ordersId: EditForm.value.ordersId,
                addressId: addressId.value,
                quantity: EditForm.value.quantity,
                delivery: deliveryMethod.value,
            })
            axios.put("/api/reservations/update", EditForms.value).then((res) => {
                if (res.status === 200) {
                    ElMessage({ message: "提交成功", type: "success" });
                    EditForm.value = [];
                    openEditDialog.value = false;
                    loadData();
                }
            }).catch((error) => {
                ElMessage.error("提交失败");
            })
        }
```

后端取消预订功能接口关键代码如下：

```
    /**
     * 取消预订
     */
    @PutMapping("/delete/{id}")
    public ResponseEntity<String> deleteCategory(@PathVariable("id") String id)
{
        //          判断是否为空
        if (id == null) {
            return  ResponseEntity.status(HttpStatus.BAD_REQUEST).body("id
不能为空");
        }
        return reservationsService.deleteReservations(id);
    }
```

前端取消预订功能关键代码如下：

```
    // 取消预订
        const CancelReservation = (id) => {
            axios.put((`/api/reservations/delete/${id}`)).then((res) => {
                if (res.status === 200) {
                    ElMessage({ message: "取消成功,货款将在 1-3 个工作日内原
路退回！ ", type: "success" });
                    loadData();
                }
            }).catch((error) => {
                ElMessage.error("取消失败: " + error.message);
            })
        }
```

添加预订功能界面如下图 5-5 所示：

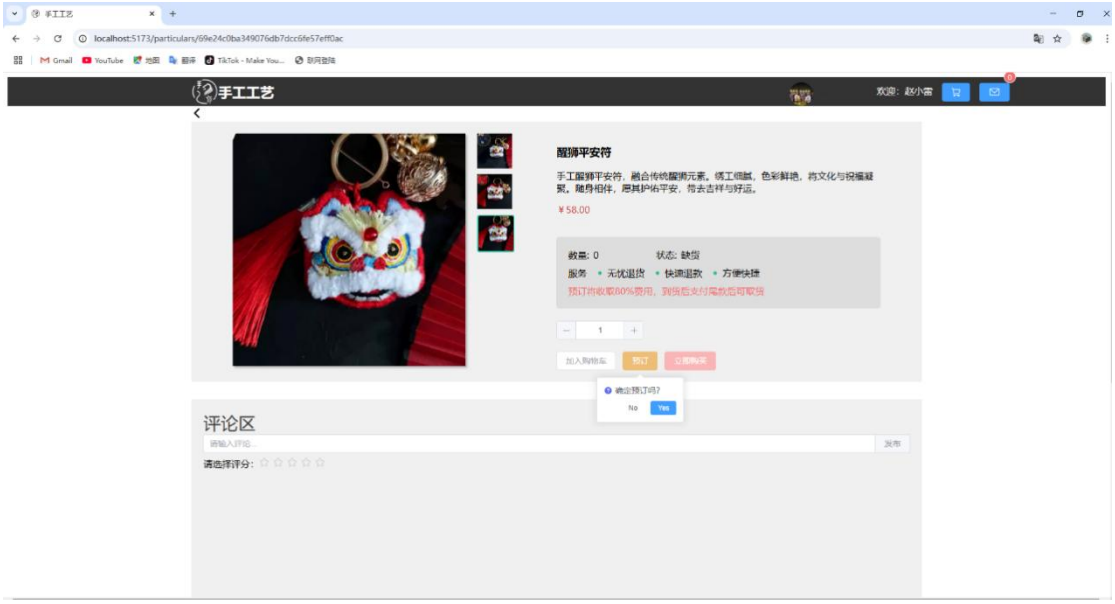

图 5-5　添加预订界面

加载预订成功功能界面及系统通知如下图 5-6 所示：

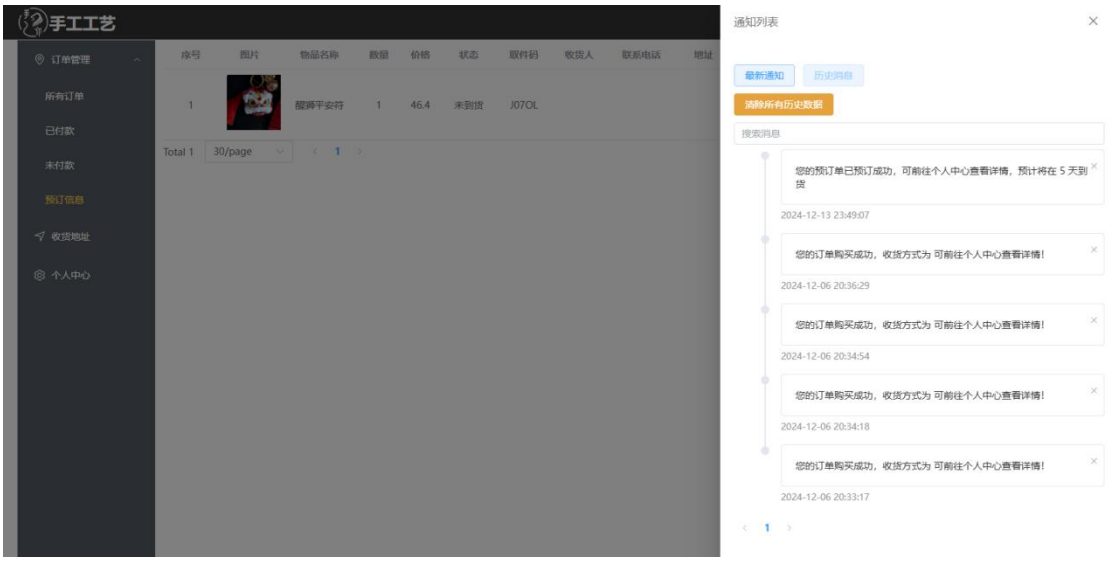

图 5-6　预订通知界面

修改预订功能界面如下图 5-7 所示：

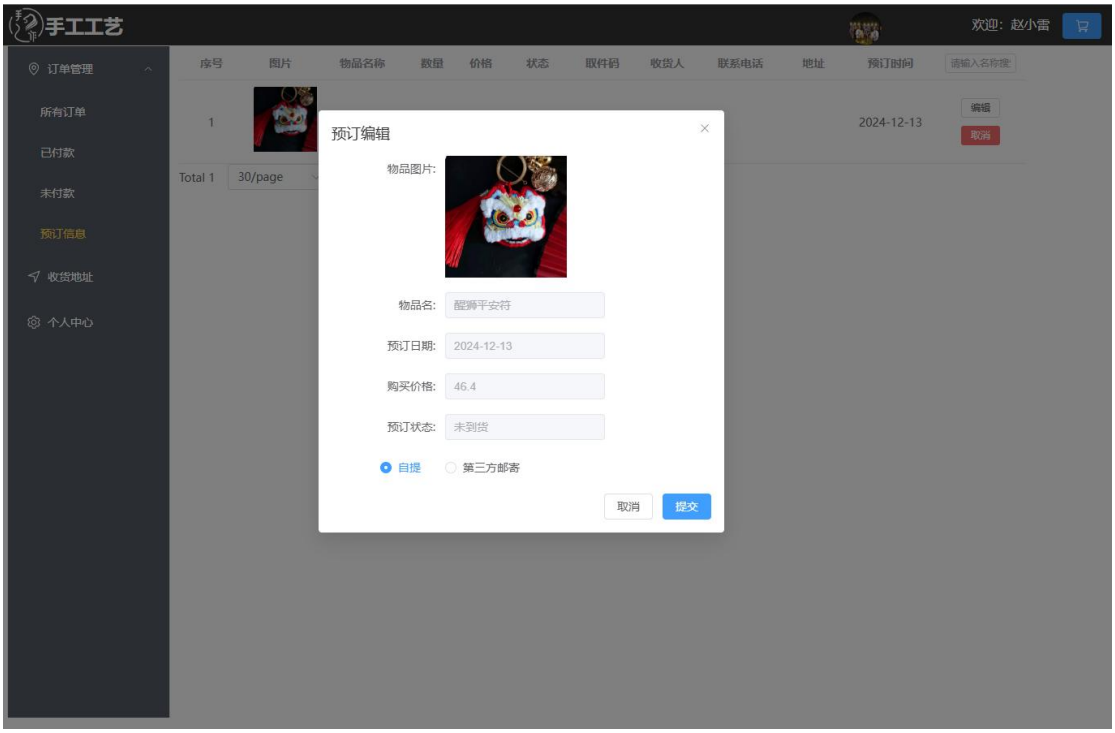

图 5-7　修改预订界面

删除预订功能界面如下图 5-8 所示：

图 5-8　删除预订界面

5.4 用户线下购买模块

用户线下购买模块为后端核心功能模块，主要用于用户线下购物商品出入库记录，用户在线下购物时，商家可在平台线下购物售出，商家只需要选中用户需要购物的商品选择购物数量后添加到购物清单，用户可选择线上支付还是线下支付，支付成功后用户则将购买完成，并且在系统中留下出入库记录。

后端用户线下购买功能接口关键代码如下：

```
        @PostMapping("/save")
        public ResponseEntity<String> SaveCartItems(@RequestBody CartItems
cartItems, HttpSession session) {
            if (cartItems.getItemId() == null ||
cartItems.getItemId().trim().isEmpty() || cartItems.getQuantity() == 0 ||
cartItems.getQuantity() <= 0) {
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("未选择物品");
            }
            return cartItemsService.saveCartItems(cartItems,session);
        }
```

前端用户线下购买功能关键代码如下：

```
    const handleSubmit = () => {
        if (formData.value.length === 0) {
            ElMessage.error("您还没有添加商品");
            return;
        }
        const transformedData = {
            userId: token,
                totalPrice: formData.value.reduce((total, item) => total +
(item.price * item.stock), 0),
            itemData: formData.value.map(item => ({
                id: item.id,
                quantity: item.stock,
                price: item.price,
                itemId: item.id,
                addressesId: "false",
            }))
        };
        sessionStorage.setItem('formData', JSON.stringify(transformedData));;
        // 跳转到 OrderInfo 页面，并通过路由传输 sessionStorage 数据
        showDialog.value = true;
    };
```

用户线下购买功能界面如下图 5-9 所示：

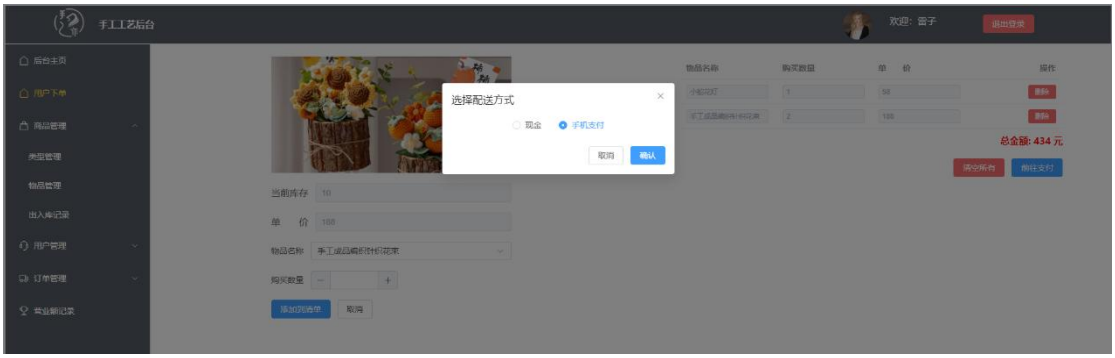

图 5-9 用户线下购买功能界面

5.5 出入库记录模块

出入库记录模块为后端核心功能模块，主要用于记录商品出入库变动情况，例如商品售出后则会在系统中自行添加出库记录，商家入库时则自行添加入库记录，商家亦可在页面中点击导出 Excel 表格，根据提示数据选择时间区间，以及变动类型获取 Excel 表格数据。点击导出后则提交请求，如成功后将会在浏览器下载 Excel 表格。

后端导出 Excel 功能接口关键代码如下：

```
    /**
     * 导出盘点
     */
    @PostMapping("/SDerivedData")
    public ResponseEntity<byte[]> SDerivedData(@RequestParam String startDate, @RequestParam String endDate,
                                               @RequestParam Integer type) {
        //          判断是否为空
        if (startDate == null || startDate.trim().isEmpty() || endDate == null || endDate.trim().isEmpty() || type == null) {
            return null;
        }
        return itemService.SDerivedData(startDate, endDate,type);
    }
```

前端导出 Excel 功能关键代码如下

```
const SubmitExport = () => {
    const formattedDates = formatDates();
    axios
        .post("/api/item/SDerivedData", null, {
            params: {
                startDate: formattedDates[0],
                endDate: formattedDates[1],
                type: Derived.value,
            },
            responseType: 'blob', // 设置响应类型为 blob
        })
        .then((res) => {
            // 处理返回的文件（Excel）
                        const blob = new Blob([res.data], { type: 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet' });
            const url = window.URL.createObjectURL(blob);
            const link = document.createElement('a');
            link.href = url;
            link.setAttribute('download', 'export_data.xlsx'); // 设置下载的文件名称
            document.body.appendChild(link);
            link.click();
            document.body.removeChild(link);
            window.URL.revokeObjectURL(url); // 清理
            handleDialogClose();
        })
        .catch((error) => {
            ElMessage({ message: "下载失败" });
        })
        .finally(() => {
            // 如果需要清理工作，可以在这里执行
        });
};
```

导出 Excel 功能界面如下图 5-10 所示：

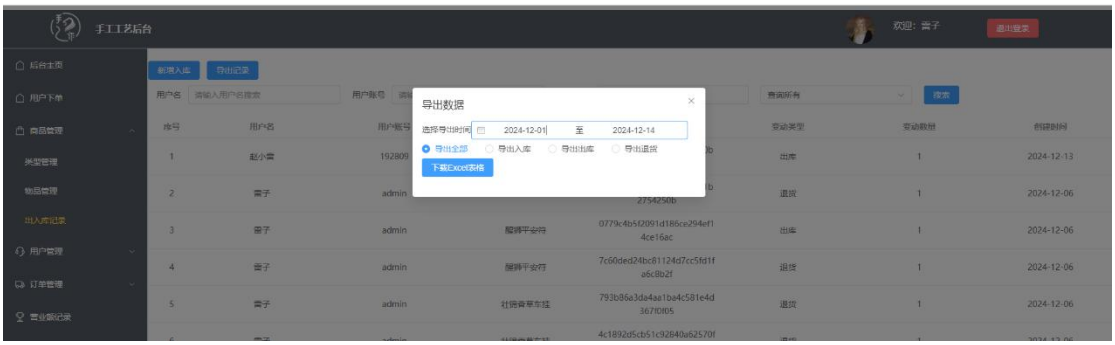

图 5-10　导出 Excel 功能界面

下载 Excel 功能界面如下图 5-11 所示：

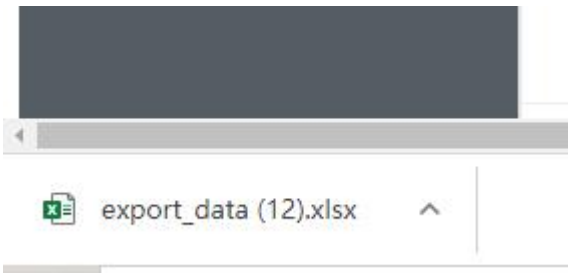

图 5-11　下载 Excel 界面

5.6 本章小结

本章介绍了专卖店订购系统的核心功能模块，包括购物车管理、预订管理、用户线下购买和出入库记录。各模块紧密协作，确保用户和商家在商品交易和管理中的顺畅体验。

购物车模块使用户能够在登录后将商品添加至购物车，并且仅对有货商品进行操作。用户可随时修改或删除购物车中的商品，在支付前验证商品库存，确保顺利完成支付。预订管理模块则专注于缺货商品的预订，用户可锁定缺货商品并预付部分款项，系统在预订成功后通知用户，并待商家到货。

用户线下购买模块记录商品的出入库情况，支持线上和线下支付方式，交易完成后自动生成出入库记录。出入库记录模块帮助商家管理库存变动，并提供导出 Excel 报表功能，便于商家分析库存数据。

这些模块结合使用，提升了用户体验和商家运营管理能力。

第 6 章 系统测试

6.1 测试环境

6.1.1 后端测试环境（Spring Boot）

创建项目：使用 Spring Boot 3.3.4 创建项目，添加 Spring Web, Spring Security, MyBatis-Plus 等依赖。

配置数据库：在 application.properties 配置 MySQL 连接信息（数据库 URL、用户名、密码等）。邮件服务：通过 spring-boot-starter-mail 配置邮件服务，用于发送通知邮件等。

集成 MyBatis-Plus：配置 MyBatis-Plus 进行数据库操作，简化 CRUD 操作。

测试：使用 JUnit 编写单元测试和集成测试，验证 API 功能是否正常。

6.1.2 前端测试环境（Vue 3 + Vite）

创建项目：使用 Vite 创建 Vue 3 项目，安装 Pinia, Axios, Element Plus 等依赖。

配置环境变量：在 .env 文件中设置前后端 API 路径。

集成测试：使用 Jest 或 Vue Test Utils 编写组件单元测试，模拟用户交互，验证功能是否正常。

前后端集成：通过前端发送请求与后端 API 交互，确保数据正确传输。

6.2 功能测试

表 6-1 前台购物车测试模块

序号	测试功能	测试步骤	预期结果	实际结果
1	用户将商品加入购物车	1、用户登录（用户名：192809，密码：123123123），进入商品详情页面。 2、选择一件有库存的商品，选择数 2，点击“加入购物车”按钮。 3、确认商品功加入购物车。	商品成功加入购物车，并在购物车页面中显示。	符合预期结果

续表 6-1 前台购物车测试模块

序号	测试功能	测试步骤	预期结果	实际结果
2	用户尝试将缺货商品加入购物车	1、选择一件缺货的商品，选择数量 1，点击“加入购物车”按钮。	无法使用加入购物车。	符合预期结果
3	用户修改购物车中商品的数量	1、用户登录（用户名：192809，密码：123123123），进入商品详情页面。 2、修改购物车中某商品的数量选择修改数量 1。 3、确认数量修改后，点击“提交”按钮。	商品数量成功更新。	符合预期结果
4	用户选择商品进行支付时，系统验证库存是否不足	1、用户登录（用户名：192809，密码：123123123），进入商品详情页面。 2、选择一件库存不足的商品，点击“支付”按钮。 3、确认系统提示库存不足，无法支付。	系统提示库存不足。	符合预期结果

表 6-2 前台预订测试模块

序号	测试功能	测试步骤	预期结果	实际结果
1	用户在商品缺货时可以预订商品	1、用户登录（用户名：192809，密码：123123123），进入商品详情页面。 2、选择一件缺货的商品选择预订数量 2，点击“预订”按钮。	系统允许用户进行预订，并跳转至收货方式选择页面。	符合预期结果
2	用户尝试预订在售商品	1、选择一件在售的商品选择预订数量 1，点击“预订”按钮。	无法使用预订。	符合预期结果

续表 6-2 前台预订测试模块

序号	测试功能	测试步骤	预期结果	实际结果
3	用户尝试重新	1、选择一件缺货的商品选	系统提示用	符合预期

	预订已预订的商品	择预订数量 1，点击“预订”按钮，并成功预订。 2、刷新系统，再次选择该商品进行预订。	户该商品已被预订，询问是否继续预订。	结果
4	用户支付预订款成功后，系统通知预订成功	1、用户支付预订款（80%商品总价）。 2、系统验证支付成功后，发送预订成功的通知给用户。	用户收到预订成功的通知，显示预订成功信息。	符合预期结果

表 6-3　用户线下购买测试模块

序号	测试功能	测试步骤	预期结果	实际结果
1	商家成功将用户需要购买的商品添加到购物清单中	1、商家登录（用户名：admin，密码：123123123），进入线下购物管理页面。 2、选择一件商品，并填写用户需要购买的数量 1。 3、将商品添加到购物清单。	商品成功添加到购物清单中，显示商品名称、数量和价格。	符合预期结果
2	用户选择线上支付或线下支付	1、用户在购物清单页面选择支付。 2、提供“线上支付”与“线下支付”选项，用户选择其中一种支付方式。	用户能够成功选择支付方式。	符合预期结果
3	支付成功后，系统自动生成商品出入库记录	1、用户成功完成支付（线上或线下）。 2、系统自动生成商品出库记录，并更新库存数量。	系统生成正确的出库记录，商品库存数量减少。	符合预期结果

续表 6-3　用户线下购买测试模块

序号	测试功能	测试步骤	预期结果	实际结果
4	如果库存不足，系统不允许商家将商品	1、商家登录（用户名：admin，密码：123123123），选择一个库存不足的商品。	无法选择数量，禁止商品到添加到	符合预期结果

	添加至购物清单	2、尝试添加该商品到购物清单中，并选择购买数量1。	购物清单。	
5	用户选择多个商品时，系统能够正确计算总价	1、商家为用户添加多个商品到购物清单。 2、用户选择支付方式，系统计算所有商品的总价。	系统正确计算并展示所有商品的总价，用户可看到准确的付款金额。	符合预期结果

表 6-4　出入库记录测试模块

序号	测试功能	测试步骤	预期结果	实际结果
1	系统在商品售出后自动生成出库记录	1、商家登录系统（用户名：admin，密码：123123123），并完成用户支付后，商品售出。 2、系统自动生成出库记录，记录商品名称、数量、出库时间、变动类型等信息。	系统成功生成出库记录，并且记录内容正确（商品名称、数量、时间等）。	符合预期结果
2	商家入库时系统自动生成入库记录	1、商家登录系统（用户名：admin，密码：123123123），完成商品入库操作。 2、系统自动生成入库记录，记录商品名称、数量、入库时间、变动类型等信息。	系统成功生成入库记录，并且记录内容正确（商品名称、数量、时间等）。	符合预期结果

续表 6-4　出入库记录测试模块

序号	测试功能	测试步骤	预期结果	实际结果
3	商家点击导出按钮后，系统生成并下载 Excel 表格	1、商家登录系统（用户名：admin，密码：123123123），进入出入库记录页面。 2、选择时间区间（例如从 2024 年 12 月 1 日到 2024 年 12 月 5 日）以及变动类	系统根据选择的时间区间和变动类型生成 Excel 文件，并开始下载	符合预期结果

	型（如入库或出库）。 3、点击“导出”按钮提交请求。	到浏览器。

6.3 本章小结

本章主要介绍了后端和前端测试环境的配置及功能测试的详细设计。首先，后端环境采用了 Spring Boot 3.3.4，结合 MyBatis-Plus 进行数据库操作，并通过 Spring Security 确保系统的安全性。同时，邮件服务通过 spring-boot-starter-mail 完成，支持发送通知邮件等功能。在后端测试方面，采用了集成测试来验证 API 的正确性，并确保系统各模块协同工作。

在前端部分，使用 Vite 创建了 Vue 3 项目，并集成了 Pinia、Axios 和 Element Plus，配置了环境变量以确保前后端的顺利交互。前端的测试主要采用了手动测试和模拟用户操作的方式，通过测试用例验证各个功能模块的正常运行。通过与后端的集成测试，验证了数据的准确传输，确保了系统的稳定性和功能的实现。

针对具体功能，测试用例覆盖了前台购物车、预订、用户线下购买、出入库记录等模块。每个测试功能都明确了测试过程、预期结果和实际结果，确保系统能够在不同情况下正常工作。例如，购物车模块测试了商品加入、修改数量、支付验证等，预订模块则确保用户在商品缺货时可以预订并支付成功。此外，后台商品管理和出入库记录模块也涵盖了商品添加、库存更新及导出记录等功能。通过这些测试用例，系统的稳定性和功能性得到了有效保障。

第 7 章 总结与展望

7.1 主要工作成果

系统功能模块开发与实现：

购物车模块：成功设计并实现了用户购物车功能，支持商品添加、删除、修改数量、支付流程的控制，以及库存验证。用户可以根据商品的库存状态进行选择，且购物车数据与后端数据库进行同步管理。

预订管理模块：实现了商品预订功能，针对缺货商品提供预订服务，支持用户选择收货方式并进行预付款。系统对已预订商品提供提示，用户有权选择是否继续预订。

线下购物模块：实现了用户线下购物功能，包括商品选择、支付方式选择（线上支付与线下支付）以及购买完成后的出入库记录生成。

出入库记录模块：开发了商品的出入库记录功能，支持商品售出、入库时自动生成记录，并提供导出 Excel 表格的功能，方便商家查看和分析商品库存变动。

系统稳定性与性能优化：

在实现各项功能的过程中，优化了数据库查询与数据存储，确保系统在大数据量操作下依然稳定运行。

完成了前后端的接口对接，确保前端页面与后端数据的无缝连接，实现了用户交互的流畅体验。

用户体验设计与功能完善：

通过与项目组成员的反馈与讨论，持续优化了用户界面与交互流程，提升了用户体验。特别是在购物车、预订管理、订单支付等环节中，用户操作更为直观、便捷。

测试与质量保证：

完成了系统各功能模块的单元测试、集成测试，确保系统的各项功能均能正常工作，并对潜在的 Bug 进行了修复。

对系统进行了负载测试，保证系统能够在高并发情况下稳定运行。

7.2 工作展望

功能扩展与优化：

用户个性化推荐：将考虑引入个性化推荐算法，基于用户的购买历史和浏览行为，推荐适合的商品，提高用户的购买转化率。

性能提升与系统优化：

数据库优化：继续优化数据库的设计，特别是在处理大数据量的库存管理和出入库记录时，进一步提升查询性能和系统响应速度。

缓存机制：考虑引入缓存机制，特别是在用户频繁访问的商品数据、库存信息等，减少数据库的压力，提高系统响应速度。

移动端适配与跨平台支持：

移动端开发：为提高用户的使用便利性，未来将开发移动端应用（Android/iOS）或优化现有的响应式网页，以便用户在各种设备上都能流畅使用系统。

跨平台支付支持：进一步扩展支付方式，支持更多的支付平台，如支付宝、银行卡支付等，满足不同用户的需求。

系统安全与数据保护：

在保证系统功能的同时，增强系统的安全性，特别是对用户个人数据、支付信息、库存数据等敏感信息的保护，确保符合相关法规（如 GDPR）和安全标准。

用户反馈与迭代更新：

通过用户的使用反馈和市场的变化，定期对系统进行版本迭代，加入新的功能，修复已知的问题，持续改进系统的用户体验。

参考文献

[1] 黎健斌,黎子欣,李开彦,等.新零售视角下中国传统手工艺品传承与创新发展——基于肇庆市端砚的路径探索[J].品位·经典,2024,(07):46-48.

[2] 耿庆阳.基于 Spring Boot 与 Vue 的电子商城设计与实现[D].西安石油大学,2020.DOI:10.27400/d.cnki.gxasc.2020.000569.

[3] 姬晓雪,郝运,马国庆,等.传统绢花互联网之路的探索与实践[J].办公自动化,2021,26(13):61-62.

[4] 白富康.竹雕工艺品独特审美价值和开发前景[J].上海工艺美术,2024,(03):61-63.

[5] 陈伟.Spring Boot 框架下年货订购系统的关键技术研究与实现[D].中南大学,2022.DOI:10.27661/d.cnki.gzhnu.2022.006752.

[6] David Fernández Bellver, M. Belén Prados-Peña, Ana M. García-López, Valentín Molina-Moreno. "Crafts as a key factor in local development: Bibliometric analysis." Heliyon 9.1 (2023): e13039. Web. 16 Nov.

[7] 朴明,于湘菲.基于 SSM 框架技术的线上工艺品商城设计与实现[J].造纸装备及材料,2024,53(02):85-87.

[8] 马雪山,张辉军,陈辉,等.前后端分离的 Web 平台技术研究与实现[J].电子技术与软件工程,2022,(08):70-73.

[9] 何贵涛.A 线上购物商城服务营销优化研究[D].广西大学,2021.DOI:10.27034/d.cnki.ggxiu.2021.000355.

[10] 陆婷婷.新媒体时代广西壮族刺绣工艺品传播研究[D].上海师范大学,2021.DOI:10.27312/d.cnki.gshsu.2021.002179.

[11] 潘涛,王柳,董冉冉.基于 Vue.js 框架的网上商城管理系统的设计与实现[J].科技与创新,2023,(13):8-10.DOI:10.15913/j.cnki.kjycx.2023.13.003.

[12] 张玉.基于 Web 平台的购物网站的设计与实现[D].华中科技大学,2020.DOI:10.27157/d.cnki.ghzku.2020.004264.

[13] 刘启伟.基于 Vue.js 框架的 Web 前端开发工具的设计与实现[D].北京邮电大学,2021.DOI:10.26969/d.cnki.gbydu.2021.002714.

[14] 李晨.基于 Spring Boot 的电子商城设计与实现[D].哈尔滨工业大学,2020.DOI:10.27061/d.cnki.ghgdu.2020.002417.

[15] 黑马程序员,Spring Boot 企业级开发教程《第 2 版》,人民邮电出版社,2024:237-248

[16] 刘启伟.基于 Vue.js 框架的 Web 前端开发工具的设计与实现[D].北京邮电大学,2021.DOI:10.26969/d.cnki.gbydu.2021.002714.

[17] 白添予.基于 MyBatisPlus 的数据库框架优化综述[J].电脑与信息技术,2024,32(03):75-77+133.DOI:10.19414/j.cnki.1005-1228.2024.03.021.

致谢

在此，我要衷心感谢我的导师周老师，感谢您在项目中给予的悉心指导和无私帮助。您的专业知识和耐心解答使我受益匪浅，让我不断进步和成长。感谢我的同学们，特别是在项目过程中，你们的支持与分享让我解决了许多问题，大家的团队精神和努力是项目顺利完成的重要保障。最后，我要感谢我的家人，感谢你们一直以来的理解和鼓励，在我忙碌时给予我最大的支持，让我能够专心投入到工作中。你们的关爱是我前进的动力，感谢你们的陪伴与支持！