

成都锦城学院

本科生毕业论文（设计）

题 目 基于 SpringBoot 的游戏商城

设计与实现

二级学院 计算机与软件学院

专 业 人工智能

学生姓名 陈劲儒

学 号 203020429 年级 2020

指导教师 张凌飞

教务处 制表

2023 年 4 月 1 日

基于 SpringBoot 的游戏商城系统设计与实现

专业：人工智能

学生：陈劲儒 指导老师：张凌飞

摘要：随着数字化时代的到来，游戏已经成为了人们日常生活中不可或缺的一部分。为了更好地为玩家提供服务，本项目设计并实现了一个基于 SpringBoot 的游戏商城。该商城主要提供虚拟商品——游戏的在线购买和下载服务。商城的后端使用 Java 进行开发，数据存储采用 MySQL 数据库，同时还利用了 Redis 作为缓存技术，提高了系统的响应速度。前端则采用了现代化的 Web 技术进行设计，提供了友好的用户界面。商城分为前后台两部分，前台为玩家提供游戏浏览、购买、评论等功能，后台则为管理员提供了游戏的上下架、用户管理、订单管理等功能。此外，商城还引入协同过滤算法实现智能推荐功能，根据用户的历史行为和评分，为用户推荐相似或感兴趣的遊戲，进一步提高用户体验。通过本项目的实现，不仅为玩家提供了一个方便快捷的购买游戏的平台，同时也为游戏开发者提供了一个展示和销售自己作品的场所。

。

关键词：Springboot; Java; MySQL; Redis; 游戏商城; 用户管理; 订单管理; 智能推荐

Design and Implementation of a Game Mall System Based on SpringBoot

Major: Artificial Intelligence

Student: Chen Jingru Supervisor: Zhang Lingfei

Abstract: With the advent of the digital age, games have become an indispensable part of people's daily lives. To better serve players, this project designs and implements a game mall based on SpringBoot. The mall primarily offers virtual products - online purchase and download services for games. The backend of the mall is developed in Java, with data storage using the MySQL database. Additionally, Redis is employed as a caching technology, enhancing the system's response speed. The frontend adopts modern Web technologies for design, providing a user-friendly interface. The mall is divided into front and back ends. The frontend offers players game browsing, purchasing, and commenting functionalities, while the backend provides administrators with game listing, user management, and order management functionalities. Furthermore, the mall plans to introduce a collaborative filtering algorithm to implement intelligent recommendation features. Based on users' historical behaviors and ratings, it recommends games that are similar or of interest, further enhancing user experience. Through the implementation of this project, it not only provides players with a convenient platform to purchase games but also offers game developers a venue to showcase and sell their creations.

Key Words: SpringBoot; Java; MySQL; Redis; Game Mall; User Management; Order Management; Intelligent Recommendation

目 录

| | |
|---------------------------|-----------|
| 1 绪论 | 1 |
| 1.1 开发背景及意义 | 1 |
| 1.2 国内外研究现状 | 1 |
| 1.3 主要研究内容 | 2 |
| 1.4 论文结构 | 3 |
| 1.5 本章小结 | 4 |
| 2 使用的相关技术 | 5 |
| 2.1 开发运行环境 | 9 |
| 2.2 SpringBoot 框架 | 9 |
| 2.3 Thymeleaf 模板引擎 | 11 |
| 2.4 MySQL 数据库 | 11 |
| 2.5 Mybatis 数据库映射框架 | 11 |
| 2.6 本章小结 | 12 |
| 3 需求分析 | 错误!未定义书签。 |
| 3.1 系统概述 | 错误!未定义书签。 |
| 3.2 功能性需求分析 | 错误!未定义书签。 |
| 3.2.1 通用功能 | 错误!未定义书签。 |
| 3.2.2 浏览游戏商品功能 | 错误!未定义书签。 |
| 3.2.3 购物车功能 | 错误!未定义书签。 |
| 3.2.4 用户管理功能 | 错误!未定义书签。 |
| 3.2.5 游戏商品管理功能 | 错误!未定义书签。 |
| 3.3 非功能性需求分析 | 错误!未定义书签。 |
| 3.3.1 系统鲁棒性 | 错误!未定义书签。 |
| 3.3.2 系统响应速度 | 错误!未定义书签。 |
| 3.3.3 可扩展性 | 错误!未定义书签。 |
| 3.4 本章小结 | 错误!未定义书签。 |
| 4 系统设计 | 13 |
| 4.1 研究思路 | 13 |
| 4.2 功能结构设计 | 13 |
| 4.3 系统主要功能设计 | 13 |
| 4.3.1 注册功能设计 | 13 |
| 4.3.2 登录功能设计 | 15 |

| | |
|---------------------------|----|
| 4.3.3 游戏商品浏览功能设计 | 15 |
| 4.3.4 游戏商品购买功能设计 | 15 |
| 4.3.5 游戏上下架功能设计 | 15 |
| 4.3.6 用户评论管理功能设计 | 15 |
| 4.3.7 智能推荐功能 | 16 |
| 4.4 数据库设计 | 16 |
| 4.4.1 游戏商城系统 E-R 图 | 16 |
| 4.4.2 数据库表设计 | 17 |
| 4.5 本章小结 | 18 |
| 5 系统开发与实现 | 19 |
| 5.1 具体功能设计与实现 | 19 |
| 5.1.1 用户登录注册及身份认证模块 | 19 |
| 5.1.2 查看已发布的游戏模块 | 23 |
| 5.1.3 查看游戏商品详情模块 | 24 |
| 5.1.7 我的购物车模块 | 26 |
| 5.1.8 添加游戏商品到购物车 | 28 |
| 5.1.9 用户管理模块 | 29 |
| 5.1.10 游戏商品管理模块 | 31 |
| 5.1.11 评论管理模块 | 32 |
| 5.1.12 智能推荐模块 | 32 |
| 5.2 本章小结 | 33 |
| 6 系统测试 | 34 |
| 6.1 测试环境 | 34 |
| 6.1.1 硬件环境 | 34 |
| 6.1.2 软件环境 | 34 |
| 6.2 系统功能点测试 | 34 |
| 6.2.1 用户身份认证测试 | 34 |
| 6.2.2 游戏商城和购物车测试 | 37 |
| 6.2.3 后台管理系统测试 | 39 |
| 6.3 本章小结 | 41 |
| 7 总结与展望 | 42 |
| 参考文献 | 43 |
| 致谢 | 44 |

1 绪论（待修改）

1.1 开发背景及意义

随着数字化时代的到来，游戏行业已经成为全球最大的娱乐产业之一。从手机游戏到大型多人在线角色扮演游戏，游戏的种类和形式多种多样，满足了各种玩家的需求。然而，随着游戏数量的增加，玩家面临的选择也越来越多，如何在众多的游戏中找到自己喜欢的，成为了一个挑战。这也使得游戏商城的存在变得尤为重要。

游戏商城不仅为玩家提供了一个购买和下载游戏的平台，更重要的是，它可以为玩家提供个性化的游戏推荐，帮助玩家在短时间内找到自己喜欢的游戏。然而，目前市面上的游戏商城大多功能单一，缺乏智能推荐系统，导致玩家很难找到自己真正喜欢的游戏。所以开发一个游戏商城将游戏像普通商品一样展示和售卖，保障了游戏开发人员的利益的同时也满足了游戏用户的需求，即一个整合了游戏销售与管理的游戏商城，让游戏用户体验游戏环境下生态性，也满足了游戏用户对于娱乐的追求。

1.2 国内外研究现状

游戏商品本体的形式分为三个阶段：

首先是卡带的年代。这是基于集成电路板的技术，只需将其接入 FC 游戏机，玩家就能立即体验游戏。在这个时代，游戏商店是不存在的，玩家必须亲自前往实体店购买他们想要的游戏卡带。每个卡带代表一个独特的游戏。由于其体积庞大，如果收藏了很多卡带，存储就会成为一个挑战。值得注意的是，我国并未销售官方版本的卡带，市场上流通的大多是非正版。

接下来是光盘的时代。这可以看作是卡带技术的进步。光盘的尺寸比卡带小得多，更为便于储存。而且，光盘的存储能力远超卡带，大约可以达到 25GB。只要将光盘放入游戏机，就可以立即游玩，无需下载。但遗憾的是，我国并没有为游戏光盘设立专门的市场，玩家大多在某些电子商务平台上购买。

然后，进入了数字化的时代。在这个时代，玩家可以在购买游戏后立即下载并开始体验。但下载速度很大程度上取决于玩家的网络环境。除了直接在游戏平台上购买，玩家还可以选择在第三方平台购买激活码，然后在游戏平台上使用。目前，Steam 是最受玩家欢迎的游戏平台，尽管其下载速度相对较快，但由于其服务器不

在我国，所以有时会遇到访问问题。因此，有机会在我国创建一个与 Steam 类似的游戏平台，结合游戏销售和社区功能，为玩家提供更好的体验。

相比我国的游戏平台，国外的平台游戏种类更多，功能也更为完善。其中，Steam 因其巨大的用户基数和长时间的发展而占据了主导地位。尽管 Epic 平台也提供了类似的功能，并且每周都会免费提供游戏以吸引玩家，但其效果并不如人意，大多数玩家仍然选择使用 Steam。事实上，Steam 已经在游戏市场上建立了垄断地位，其他任何平台都难以与之匹敌。

现在，我国的游戏市场上，专门的游戏商城并不多见，它们大多专注于销售游戏激活码。当玩家购买了激活码，他们通常需要转到其他游戏平台进行激活和下载以开始游戏。其中，表现较为出色的游戏平台大多是国外打造的，它们成功地融合了社区论坛和游戏商品销售，并要求玩家通过该平台启动和玩游戏，这极大地方便了游戏的持续维护，为玩家带来了全方位的体验。值得一提的是，国外的游戏内容分级制度做得很好，游戏种类繁多，玩家体验上乘。而在我国，像这样能够为玩家提供全方位体验的平台，腾讯的 WeGame 游戏平台算是其中的佼佼者。但由于我国的游戏分级制度尚待完善，大部分游戏都偏向于年轻玩家，这使得平台难以吸引更多广泛的玩家群体。无论是从玩家的体验还是从商城的整体构建来看，与国外的平台都存在明显的差距。

1.3 主要研究内容

对于本游戏商城系统，研究内容如下：

（1）本系统所需要的功能主要分为前台功能和后台功能，游戏商城前台功能，包括普通用户的认证，账户管理模块（登录注册修改密码），游戏的展示模块（游戏在主页的显示，首页推荐的更新），评论区管理模块（发布，删除，修改评论），后台功能包括游戏管理模块（游戏上架，下架，价格修改，编辑详情信息），用户管理模块（对用户的账号进行管理员管理，对用户发表的评论进行管理），以及智能推荐模块（根据用户的历史购买记录进行推荐）。

（2）本游戏商城的主要功能就是游戏的销售模块以及管理模块，销售模块包括游戏商品管理，智能推荐系统，购买记录以及购物车管理，管理模块包括商品管理，管理用户，管理前端的游戏资讯^{错误!未找到引用源。}。

（3）依据需求评估，对各个系统的实体进行区分，进而完成实体属性的构建。基于这些实体之间的联系，完成数据库表格和系统 ER 模型的搭建。

（4）本人使用 IDEA 工具对该游戏商城系统进行设计和功能模块的开发。系统数据采用 MySQL 数据库进行存储，利用 Java 来处理繁杂的业务流程，而 HTML5 和 JavaScript 则被用于前端页面的构建。

（5）对该游戏商城的功能进行检验，并在测试过程中记录整个系统出现的各种问题，根据报告对相关模块进行修正，直至所有问题得到妥善处理。

1.4 论文结构

本部分主要概述了论文的整体框架，关于基于 SpringBoot 的游戏商城的结构如下所示：

第一章：绪论。本章着重阐述了构建游戏商城的背景及其重要性。通过概述游戏商品经历的三个发展阶段，揭示了游戏商城在国内外的研究进展。结合目前颇受欢迎的 Steam 游戏平台，分析了创建国内游戏商城的潜在价值和优点。此外，还简要描述了该游戏商城预期的功能和所采用的技术手段。章节末尾，对论文的整体布局进行了说明。

第二章：基本理念及相关技术概览。该游戏商城核心上是一个 Java Web 项目，采纳了 SpringBoot 框架。在前端部分，本系统结合了 Bootstrap、Layui 和 jQuery 等多种技术来构建前端，而在数据库方面，选择了 Mysql，并采用 mybatis 作为持久层来管理数据库。

第三章：游戏商城项目需求探讨。首先，强调了需求分析的核心地位，然后将需求细分为功能性和非功能性两大类。功能性需求描述了游戏商城预期要达到的功能，而非功能性需求则涉及响应时间、扩展性和稳定性等方面。

第四章：系统架构设计。开始时，对预期功能进行了分类，并提供了功能结构图，主要包括通用功能、游戏商城和后端管理三大部分。接下来，基于游戏商城的实体，设计了 ER 模型。

第五章：深入的设计与执行。在这一章，详细描述了游戏商城各项功能的实现方式，展示了功能的实际界面，并附上了相关功能的代码片段。

第六章：系统验证。在此章节，针对游戏商城的功能进行了全面测试，选择了 SoapUI 作为测试工具，并通过图表和图片详细展示了各功能的测试结果。

第七章：回顾与前瞻。本章回顾了开发游戏商城的整体流程，并分析了项目中可能的不足之处，同时也提出了未来可能的改进方向。

1.5 本章小结

本部分首要描述了游戏商城项目的起源和重要性、国际与国内的研究进展、核心研究议题以及整篇论文的布局。对国内外流行的游戏平台进行了深入探讨，通过对比各大游戏平台，揭示了当前在我国创建游戏商城的潜在优势。论文的最后，本人概述了整篇文章的框架，主要包括引言、基本概念及相关技术概览、游戏商城项目的需求探讨、系统架构、深入的设计与执行，以及回顾与前景。

2 需求分析

本节主要阐述了游戏商城系统的需求探讨。在项目策划阶段，需求探讨被视为关键步骤，它旨在明确系统应具备的功能特性。此步骤的核心是明确用户期望系统达到的功能目标。基于用户的需求，本人在实际开发中不仅要满足明确的功能要求，还需确保项目的其他关键标准，如稳定性和性能等，都得到满足。只有确保功能和非功能标准都达到高水平，才能构建出高品质的系统，为用户提供卓越的使用体验。

2.1 系统概述

本游戏商城系统分为前台系统和后台系统，前台系统又分为游戏商城模块和用户账号模块，游戏商城模块主要负责普通用户访问的商店主页，游戏详情页，游戏评论区，购物车，用户账号模块负责用户的头像，账号密码修改等，后台系统包括管理员模块：对用户，游戏，评论区的管理，以及对根据普通用户的购买记录进行推荐的智能推荐系统，接下来分别论述各系统。

前台系统：

游戏商城模块：

- （1）可以看到游戏的列表，展示游戏的概图、游戏名称和当前价格；
- （2）下方的“为你推荐模块”就可以看到系统自动推荐的游戏；
- （3）点击游戏名称就可以进入游戏详情页面，在这里可以看到更多游戏信息：查看游戏的CG、实机图片和游戏介绍；
- （4）游戏详情也下方即是该游戏的评论区，用户可自由发表评论；
- （5）添加游戏到购物车即可在用户的购物车页面显示，用户点击已购买的游戏即可显示该游戏的激活码，点击弹出的对话框中的复制按钮即可一键复制激活码，到相关游戏平台输入激活码即可下载该游戏并开始游玩，同时可以直接在游戏详情页面进行修改；

用户账号模块：

- （1）鼠标移动到用户头像是会出现下拉框，可以选择“修改头像”，“修改密码”；
- （2）选择后在相应的弹出框内进行修改即可；

后台管理系统：

管理员模块：

- (1) 用户管理可以对用户进行增加、删除、修改等操作^{错误!未找到引用源。},
- (2) 管理游戏商品的上架和下架,对用户名和所发评论进行审核,如果用户发布了不良信息的帖子,可以对相关评论进行删除。
- (3) 用户管理可以对用户的信息以及发出的评论等进行审核,修改。
- (4) 管理游戏可以对游戏进行新建,管理,删除。
- (5) 查看用户标签可以查看智能推荐系统对用户的标签。

2.2 功能性需求分析

本小节描述本系统的功能性需求分析。管理员登录后可以对用户,游戏,评论进行管理和修改,同时可以查看智能推荐系统对用户的标签,同时也具有和普通用户端一样的修改本管理员账户信息的功能;普通用户登录可以对个人信息,游戏业务,购物车等进行操作。具体如图 3-1 所示:

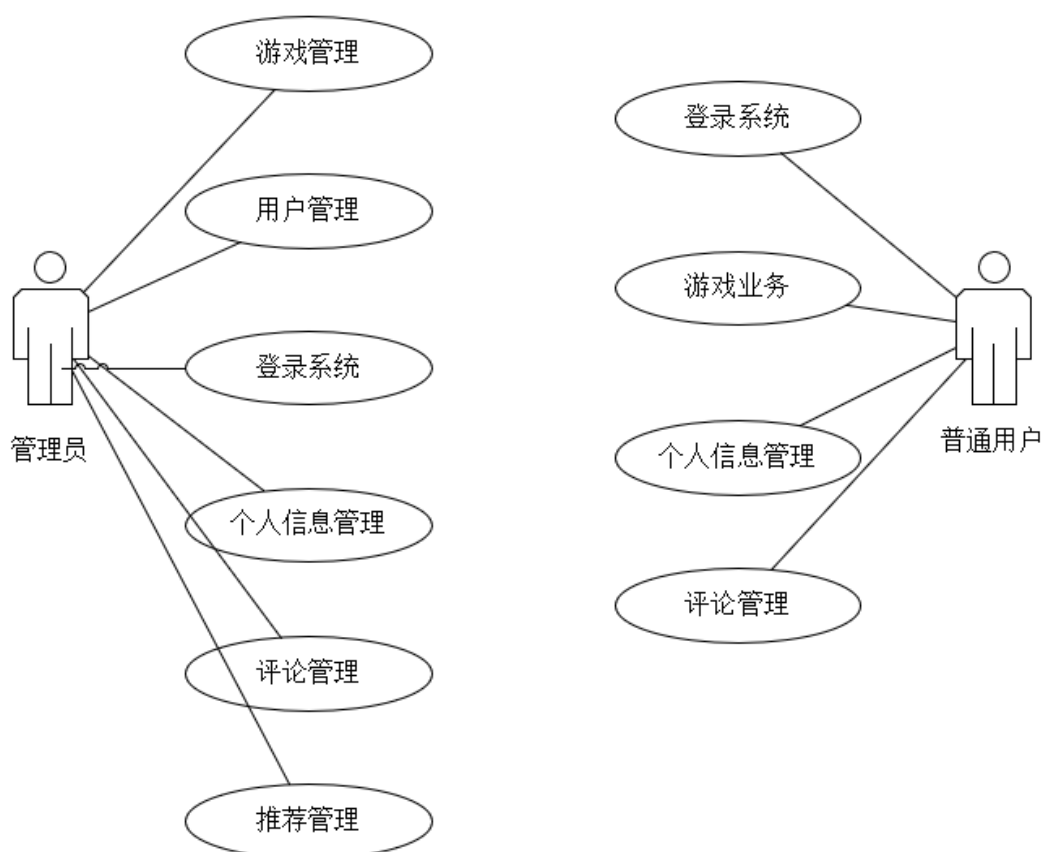


图 3-1 系统用例图

2.2.1 通用功能

本系统前台系统和后台系统都具有的功能。

用户登录与注册（安全模块）

为了抵御非法用户的恶意攻击，本游戏商城系统增设了安全防护模块。当用户尝试进入游戏商城的前端页面时，若未经登录，系统将引导至登录界面进行身份确认。首次使用的用户可以先完成注册后再登录。而对于管理员来说，访问后端管理界面也需进行登录验证，在登录页面输入管理员账号后会自动跳转到管理员页面。没有管理员权限的账户无法进入后台管理系统。

2.2.2 浏览游戏商品功能

本小节描述前台普通用户在游戏商城主页的浏览游戏商品功能。

（1）查看游戏信息

本功能主要在用户在商店界面浏览时会以卡片的形式来展示游戏图片，名字，以及价格。

（2）查看游戏商品详细信息

此功能主要提供了深入的游戏详情查看：用户可以浏览游戏的 CG 与实际游戏截图，将游戏加入购物车（若已经购入，则标注为“已购买”，若已在购物车中，则提供“移出购物车”选项），同时还可以查阅关于该游戏的用户评价。

2.2.3 购物车功能

（1）购物车

此功能允许用户浏览购物车内的游戏商品，并提供购买选项，允许购买购物车中的全部游戏产品。

（2）添加游戏商品到购物车

此功能的核心是将游戏产品放入购物车。当用户在游戏详情页面对某款游戏感兴趣并点击“添加到购物车”时，该游戏将被加入到“我的购物车”中。

2.2.4 用户管理功能

（1）用户管理

本功能可以对用户的账号进行管理，但对普通用户的密码进行了 Md5 加密以确保用户账户的安全性。

2.2.5 游戏商品管理功能

（1）游戏管理

本功能可以对游戏进行上架，删除操作。

（2）游戏信息修改

本功能同时可以修改游戏的基本信息，如游戏名，游戏类型，游戏价格。

2.3 非功能性需求分析

该部分主要介绍系统的非功能性需求，如系统响应速度，系统鲁棒性，服务器负载能力，代码耦合性，这些功能虽然看似无关紧要，但是却是整个系统能否让用户满意的基石，一个响应慢，服务器负载能力弱的系统会导致许多体验问题，因此，该部分也是整个系统需求中重要的一环，接下来分别描述本系统的非功能性需求。

2.3.1 系统鲁棒性

本系统基于业界广受认可的 SpringBoot 框架构建，该框架融合了一套经过优化的配置方案，确保代码的高度鲁棒性和可靠性。通过 Maven 仓库，我们能够高效地管理项目依赖，并轻松添加新的组件。

2.3.2 系统响应速度

本系统前后端交互主要依赖 Ajax 技术，Ajax 基于 JavaScript。在传统 web 中，一次 HTTP 请求对应一个页面。而利用 JavaScript，可以在当前页面发送新请求并接收数据。这使得用户体验连续，页面内容得以实时刷新，但视觉上仍停留在原页面，同时可以异步处理数据，使得设计系统时的选择更多。

2.3.3 可扩展性

本系统主要运用了 Java 作为后端语言，前端也是基于成熟框架，能够非常轻易的添加，删除相关功能和自定义功能，实际上，本系统的前端就运用了多种框架和技术进行复合使用，这样能为开发的选择提供更多可能性。

2.4 本章小结

本章首先对系统的系统需求进行了说明，而后详细介绍了本系统的前台系统和后台系统的部分详细设计，而后，对本系统的功能性需求进行了分析，给出系统用例图便于理解，列出浏览商品，购物车，用户管理等功能。同时对本系统的非功能性需求进行了分析，从鲁棒性，响应速度，可扩展性三个方面进行了论述。

3 使用的相关技术

本系统的开发环境使用了 IDEA，开发语言是 Java，前端页面运用多种前端框架以及相关技术，包括 Thymleaf、Bootstrap、Layui 和 jQuery 等。运用 Mybatis 连接到 MySQL 数据库，运用 Maven 管理包依赖以及整个项目。

3.1 开发运行环境

本项目的开发工具是 IntelliJ IDEA 2023.1，运行环境为 Windows 11，开发工具包围 JDK1.8、使用 Maven 管理包以及项目，运用 Mybatis 链接服务端和数据库，数据库使用 MySQL。

3.2 SpringBoot 框架

Spring Boot 利用众多注解实现了自动配置，使得在搭建 Spring 应用时，开发者只需引入适当的场景依赖，Spring Boot 会根据这些依赖进行自动配置，从而轻松地创建一个独立的 Spring 应用，无需额外的手动配置。本游戏商城项目选择 SpringBoot 框架还基于以下几个优点。

SpringBoot 的优点：

- （1）集成 Tomcat、Jetty 和 Undertow 服务器，避免 WAR 文件部署；
- （2）利用启动器简化依赖管理；
- （3）自动配置 Spring 及其他第三方库；
- （4）为生产环境提供即用的功能；
- （5）最小化代码生成和 XML 配置需求；

SpringBoot 架构主要包括以下几个部分：View、Controller、Service、Mapper 和 pojo（或称为 entity）。

View 层：这是用户界面部分，它根据收到的数据为用户呈现相应的页面。

Controller 层：它是响应用户请求的核心，确定要展示哪个视图以及需要哪些数据。Controller 主要负责处理前端的请求，与 Service 层交互，并将 Service 层的数据返回给前端。

Service 层：这一层可以进一步细分为：

- （1）接口定义：主要声明各种方法。
- （2）接口的继承。

(3) impl: 这是接口的具体实现部分，它整合了 mapper 和 service。

Service 层主要负责处理业务逻辑，虽然它涉及到数据库的操作，但并不直接与数据库交互。它有其接口定义和实现，而在 impl 类中，它需要引入 mapper 类来与数据库交互。

(4) Mapper 层：也被称为 DAO 层，它定义了与数据库交互的接口，而具体的数据库操作语句则在 mapper.xml 文件中定义。

(5) 在 src/main/resource 目录下的 mapper.xml 文件中，定义了与数据库交互的具体 SQL 语句。

(6) Pojo (entity) 层：这里是定义数据实体的地方，它的属性与数据库表的字段相对应。通常，它会包含 getter、setter 和 toString 方法（如果没有使用 lombok 插件的话）。这些实体有时被称为 pojo，有时被称为 model，还有时被称为 domain 或 dto。

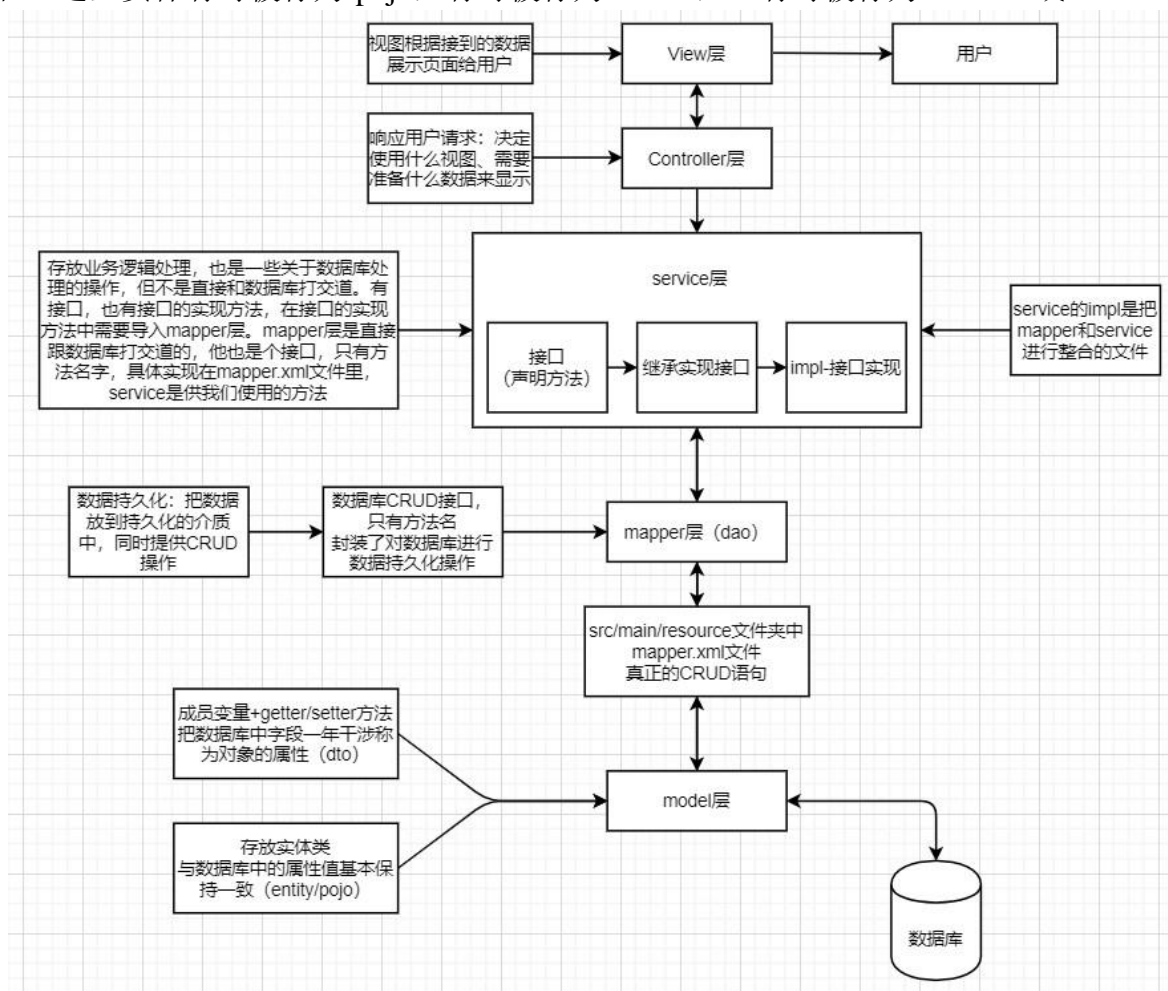


图 2-2-1 SpringBoot 框架图（待修改）

3.3 Thymeleaf 模板引擎

Thymeleaf 是一款为 Web 和独立环境设计的现代服务器端 Java 模板引擎。与其他模板引擎不同的是，Thymeleaf 具有自然模板的能力，这意味着模板可以在浏览器中作为静态文件运行，而不需要任何特定的服务器端修改。Thymeleaf 的核心优势在于它的表达式语法，它可以与 Spring 框架无缝集成，从而实现数据和视图层的紧密交互。通俗地说，Thymeleaf 就像一个精心设计的展示室，与仅提供基础 HTML 标签的传统模板相比，它能够更加直观地展示数据和逻辑[12]；与那些需要额外学习和配置的模板引擎相比，Thymeleaf 提供了一个更为直观和简洁的解决方案，同时保持了高度的灵活性和扩展性，简而言之，Thymeleaf 旨在提供一个简单、高效且功能强大的模板解决方案，而不会增加额外的复杂性。

3.4 MySQL 数据库

MySQL 是全球最受欢迎的开源关系数据库管理系统之一。在众多企业和开发者中，MySQL 因其稳定性、性能和可靠性而受到广泛的认可。作为一个关系数据库，MySQL 提供了一个结构化的方式来存储、组织和检索数据。其特点是拥有高效的数据处理能力，支持大型的数据集，并且具有出色的并发处理性能。MySQL 的另一个显著优势是其灵活性，它可以与各种编程语言和平台无缝集成，从而成为许多网站和应用程序的首选后端存储解决方案。简而言之，MySQL 不仅为开发者提供了一个强大、高效且易于管理的数据库系统，而且其开源特性也使得它在全球范围内得到了广泛的应用和推广。

3.5 Mybatis 数据库映射框架

Mybatis 是一款优秀的基于 Java 平台的持久化框架，它将对象与 SQL 语句之间的映射关系进行了简化和优化。与其他 ORM 框架不同，Mybatis 更加注重 SQL 的原生体验，允许开发者直接编写 SQL 语句，从而实现更为精细的数据操作和查询。这种设计使得 Mybatis 在处理复杂、高性能的数据库操作时，表现出了卓越的效率和灵活性。特别是在需求频繁变动的互联网项目中，如电商、社交平台等，Mybatis 都是一个理想的选择。此外，Mybatis 的动态 SQL 功能也为开发者提供了更大的自由度，可以根据不同的业务逻辑和条件，动态生成并执行相应的 SQL 语句。简而言之，Mybatis 为 Java 开发者提供了一个既强大又灵活的数据库交互解决方案，使得数据操作更为直观、高效和可控。

3.6 本章小结

本章介绍了本游戏商城系统运用的开发环境，运行环境，以及使用的技术。本系统基于 Springboot 框架开发，前端模板引擎运用 Thymleaf 等多种技术，数据区管理为 MySQL，数据库映射框架为 Mybatis。基于以上技术来开发本系统。

4 系统设计

4.1 研究思路

本系统根据前述系统需求来完成整个系统的系统设计，本小节会详述本系统各个功能的设计，并给出流程图等。

本系统的前端设计包括页面的设计与用户体验的功能设计等，两者结合最后给出的设计才能为用户提供良好的体验。本章从概念设计出发，来确保系统符合设计与用户期望，从而创建出符合预期的系统。

4.2 功能结构设计

本小节主要对于本系统的游戏商城功能结构设计进行诠释，本系统的游戏商城部分分为通用功能，游戏商城部分，以及后台管理部分。

通用功能主要包括登录与注册功能，为了保证使用安全，无论是普通用户还是管理员都需要登录后才能进行任何操作。

游戏商城部分主要包括游戏商城主页面的游戏信息，购买游戏功能，购物车功能，我的游戏功能，以及对于用户账号信息进行管理的个人中心部分。

后台部分主要包括管理员的各项功能，对于游戏上下架管理的游戏管理功能，对于用户的用户管理功能，以及对于智能推荐系统的推荐功能。

本系统功能设计结构图见图 4-1：

（待补充）

图 4-1 功能结构设计

4.3 系统主要功能设计

本系统的主要功能包括用户的注册登录，游戏商城的浏览游戏，管理员管理功能，以及智能推荐功能。

4.3.1 注册功能设计

用户未拥有账号时需要首先注册才能使用，点击注册后需要在注册页面中输入信息后，程序判断符合要求后，点击注册按钮，前端的数据发送到后端后，后端将该数据上传到数据库中相应的表中，成功储存后即代表注册成功，前端会提醒用户相应信息。

而当用户输入的信息不符合要求时，不会进行写入数据库操作，系统会提醒用户信

息错误，更改后即可正常注册。

（待补充）图 4-2 用户注册流程图

4.3.2 登录功能设计

用户拥有管理员或者普通用户账号时可以直接登录，用户点击登录后，在登录界面输入管理员或者普通账户后，点击登录，后端会将数据库中的数据与前端输入的数据进行比较，当相同时，根据数据库中的用户属性来决定登录成功后跳转到游戏商城界面或者后台控制界面，当用户名或密码错误时，会提醒用户名或者密码错误，用户输入正确用户名和密码后即可继续。

（待补充）

图 4-3 用户登录流程图

4.3.3 游戏商品浏览功能设计

游戏商城的首页面呈现了游戏的封面图和标题，当点击封面图时，可以深入了解游戏的具体详情。游戏的相关信息存储在游戏数据表里。当用户成功登录后，前端会向后端请求游戏数据，并在前端界面上展示，游戏展示的顺序基于其添加到系统的时间。

（待补充）

图 4-4 游戏浏览流程图

4.3.4 游戏商品购买功能设计

游戏详情页可直接购买游戏或者添加到购物车，购买后即可在“我的游戏”页面中查看拥有的游戏。

（待补充）

图 4-5 游戏购买流程图

4.3.5 游戏上下架功能设计

管理员登录后，在后台界面选择游戏管理，在界面中可以对已有游戏的信息进行修改，在左侧游戏管理下方还可以选择新建游戏，输入游戏相应信息后，会自动将数据保存到数据库中。

（待补充）

图 4-8 游戏上下架流程图

4.3.6 用户评论管理功能设计

当用户发送了不合适或者违法的评价时，管理员可以选择删除该评论，管理员登录后，点击用户管理，然后选择评论管理即可查看用户发送的所有评论，当需要删除时，选择其中一条评论，然后选择删除，即可删除该评论。

（待补充）

图 4-9 用户评论管理流程图

4.3.7 智能推荐功能

我们将使用基于协同过滤的推荐算法。协同过滤算法可以分为两种：

用户-用户协同过滤：这种方法首先会找到与目标用户兴趣相似的用户群体，然后推荐这个群体中的用户喜欢的、但目标用户尚未互动过的物品。

物品-物品协同过滤：这种方法会为用户推荐与他们之前喜欢的物品相似的物品。例如，如果一个用户喜欢了某款 RPG 游戏，系统会推荐其他与这款游戏相似的 RPG 游戏给该用户。

系统推荐流程如下：

用户登录后，系统首先从数据库中获取该用户的购买和评分历史。

使用用户-用户或物品-物品协同过滤算法，计算推荐分数。

根据推荐分数，为用户推荐得分最高的游戏。

将推荐的游戏列表展示给用户。

（待补充）

图 4-10 智能推荐流程图

4.4 数据库设计

4.4.1 游戏商城系统 E-R 图

根据需求分析，本系统开发有以下 4 个实体，以下是实体以及实体的属性：

- （1）用户：用户 ID, 用户密码, 用户头像, 用户分类
- （2）游戏：游戏 ID, 游戏名, 游戏类型, 游戏描述, 游戏图标, 创建时间
- （3）评论：用户 ID, 游戏 ID, 评论
- （4）用户已购游戏：游戏 ID, 用户 ID

4.4.2 数据库表设计

用户表用来保存用户的个人信息，同时可以标记用户类型，同时，该表还为用户修改个人信息以及购买游戏等功能提供了支撑。详细描述如表 4-1：

表 4-1 用户表

| 字段名称 | 字段描述 | 字段类型 | 字段约束 |
|----------|-------|---------------|------|
| id | 用户 id | int | 主键 |
| name | 用户名 | varchar (255) | 非空 |
| pwd | 密码 | varchar (255) | 非空 |
| is_admin | 管理员标签 | char (1) | 非空 |
| userimg | 用户头像 | varcahr | 非空 |

游戏表保存游戏信息，包括游戏名等，该表主要服务游戏商城游戏的显示等，同时，该表也是包括购买游戏，购物车，智能推荐系统的基石，该表详细信息如表 4-2：

表 4-2 游戏表

| 字段名称 | 字段描述 | 字段类型 | 字段约束 |
|-------------------|-------|---------------|------|
| id | 游戏 id | bigint | 主键 |
| dname | 游戏名称 | varchar (255) | 非空 |
| dtel | 游戏类型 | varchar (255) | 非空 |
| description | 游戏价格 | int | 非空 |
| gamelogo | 游戏图片 | varchar(255) | 非空 |
| gameinfo | 游戏简介 | varchar(255) | 非空 |
| establishmentdate | 发布时间 | time | 非空 |

评论表保存用户的评论，该表包括用户 id,游戏 id,以及评论内容，该表负责评论内容，评论管理等。详细描述如表 4-3：

表 4-4 评论表

| 字段名称 | 字段描述 | 字段类型 | 字段约束 |
|---------|-------|--------------|------|
| userid | 用户 id | int | 主键 |
| gameid | 游戏 id | int | 非空 |
| comment | 评论内容 | Varchar(255) | 非空 |

用户游戏表保存用户已购买的游戏，在“我的游戏”页面调用，同时是智能推荐系统的数据源。详细如表 5：

表 4-5 用户游戏表

| 字段名称 | 字段描述 | 字段类型 | 字段约束 |
|--------|-------|--------|------|
| userid | 用户 id | bigint | 非空 |
| gameid | 游戏 id | bigint | 非空 |

E—R 图如下：

（待补充）

图 4-2 E-R 图

4.5 本章小结

这一章详细阐述了游戏商城的设计流程。首先，通过功能结构图明确了要开发的功能；接着，分析了系统中的实体，并为这些实体定义了属性，随后绘制了 E-R 图，基于 E-R 图进行数据库表的设计。经过深入的分析，确定了当前需要的 4 个数据表，它们是：用户表、游戏表、评论表、用户游戏表。

5 系统开发与实现

在这一章中，基于概要设计和需求分析，将深入地设计并阐述游戏商城的具体实施方法。此章节将通过代码及其解释的方式进行展开。

5.1 具体功能设计与实现

本小节主要描述本系统的具体功能设计与实现，本系统的主要功能为用户的登录注册，用户的个人信息管理，用户已购买游戏管理，用户购物车，管理员的游戏上下架，用户管理，智能推荐系统，以下小节将分别论述这些功能。

5.1.1 用户登录注册及身份认证模块

用户没有注册账号登录时，拦截器会拦截用户的部分请求，用户必须登录才能进行部分操作。在用户在登录页面选择注册时，系统会自动将用户跳转到注册页面，用户输入账号和密码后，同时符合要求后，前端会发送一个 Post 请求到后端，后端 Controller 中的相应端点会将该注册信息保存到数据库中

前端相应代码及 Controller 代码如下：

代码 5-1 注册

```
前端: <div class="layui-form">

    <form    role="form"    action="admin/register"    method="post"    onsubmit="return
validatePassword();">

        请输入姓名:<input type="text" name="name" id="name"><br>
        请输入密码: <input type="password" name="password" id="password"><br>
        <input type="submit" value="提交">
    </form>
</div>

Controller:

@RequestMapping(value = "admin/register",method = RequestMethod.POST)
public String signUp(String name,String password){

    auserservice.Insert(name, password);

    return "admin/success";

}
```

用户注册界面如图 5-1 所示：

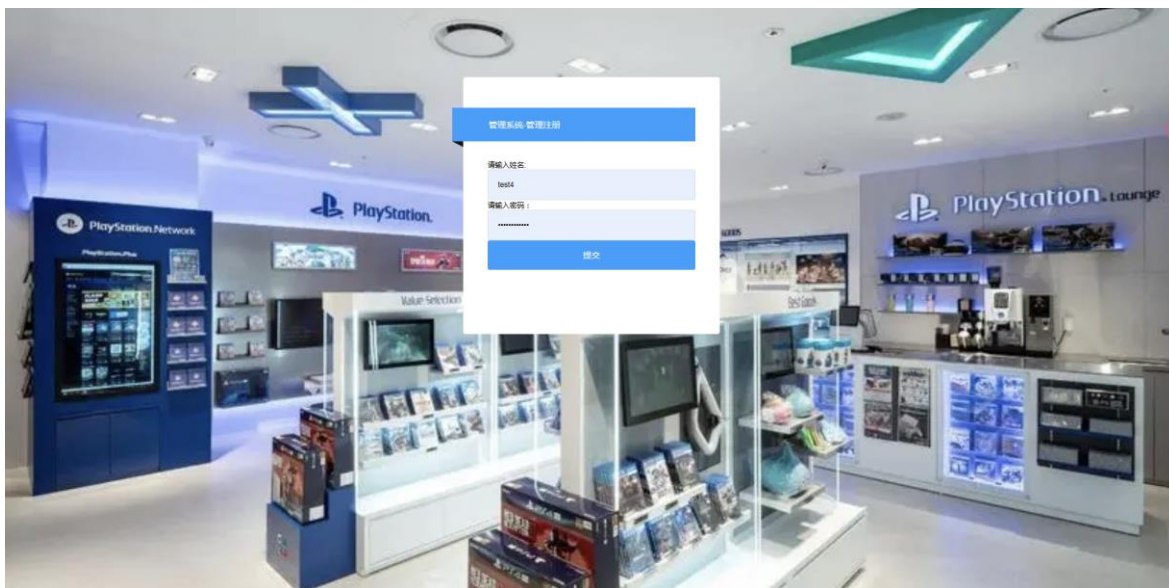


图 5-1 用户注册界面

当用户拥有账号后，在游戏商城页面选择登录或者直接进入登录页面既可进行登录操作。本系统是一个前后端分离的系统，当用户在登录界面输入账号密码并且登录后，本系统前端将通过 Post 方法将相应数据传到 controller 的 login 端点中，该端点将判断传进的账号密码与数据库中的账号密码进行查找，并将相应数据保存到 session 中，而后，系统将判断用户属性，如果用户是管理员，用户将被跳转到后端管理界面中，如果用户是普通用户，用户将被跳转到游戏商城主页

Controller 代码如下：

代码 5-2 登录

```
@GetMapping("/userlogin")
public String login(){
    return "admin/login";//打开 admin 文件夹之下的 login.html 静态文件
}

@PostMapping("/userlogin")
public String login(@RequestParam("username") String name, @RequestParam("password")
String pwd, HttpSession session){ // @RequestParam 是传进来的值在 html 那边的名字
    if(!StringUtils.hasText(name)||!StringUtils.hasText(pwd)){
        session.setAttribute("errorMsg","用户名和密码不能为空");
        return "admin/login";
    }
}
```

```
}

Auser auser = auserservice.login(name, pwd);

if(auser!=null){

    session.setAttribute("name", auser.getName());

    session.setAttribute("pwd", auser.getPwd());

    session.setAttribute("userimg", auser.getUserimg());

    session.setAttribute("userpower",auser.getUserpower());

    session.setAttribute("userId", auser.getId());

    //return "admin/index"; //地址不变， index.html 直接加载到当前页面

    if(auserservice.checkAdminByname(name)){//    如    果    是    管    理    员

//auserservice.checkAdminByname(name)是判断是否是管理员 1 是管理员 0 是普通用户

        return "/admin/index"; //地质发生变化， 变为 admin/sys_index}

    }

    else{//如果是普通用户

        session.setAttribute("name", auser.getName());

        session.setAttribute("pwd", auser.getPwd());

        session.setAttribute("userimg", auser.getUserimg());

        session.setAttribute("userpower",auser.getUserpower());

        session.setAttribute("userId", auser.getId());

        System.out.println(auserservice.checkAdminByname(name));

        System.out.println(auser.getId());

        return "redirect:/gamefront/game";

    }

}

}

else {

    session.setAttribute("errorMsg","用户名或密码错误");

    return "admin/login";

}
```

}

}

用户可以登录游戏商城前台或后台系统，具体演示如图 5-2 所示：

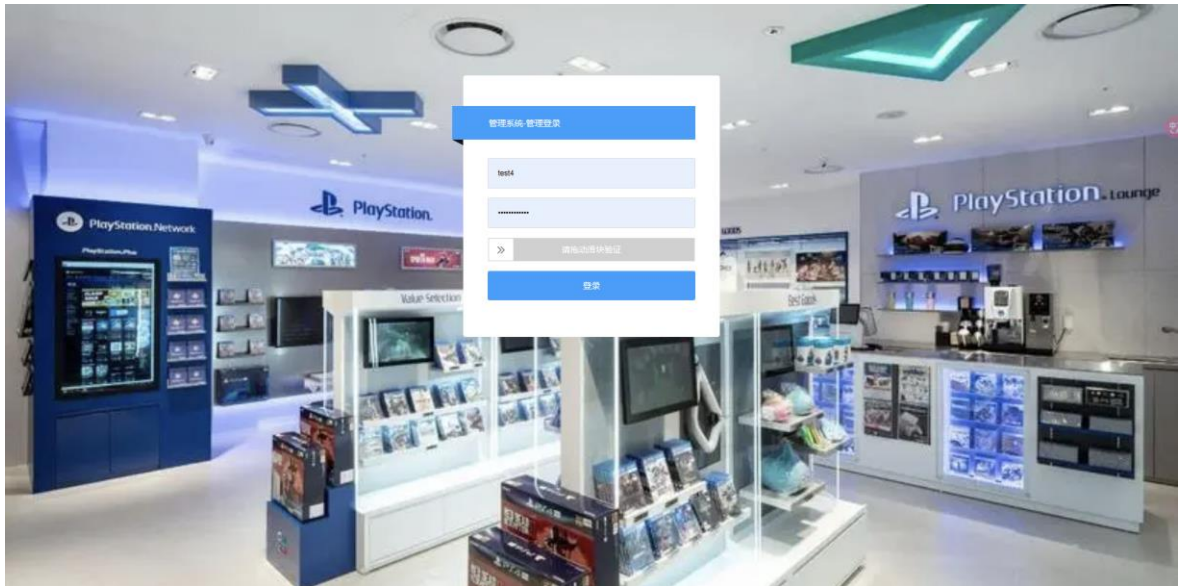


图 5-2 用户登录界面

本系统同时设有拦截器，当用户没有登录时不能进入用户中心，后台管理界面，以及进行购买游戏等操作：

具体拦截器代码如下：

代码 5-3 拦截器

```
@Component

public class AdminLoginInterceptor implements HandlerInterceptor {

    //调用 Controller 方法前执行。当其返回值为 true 时，继续向下执行；当其返回值为 false
    时，会中断后续的所有操作。

    */

    /*  @Override

    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object o)
    throws Exception {

        String requestServletPath = request.getServletPath();
```

```

        if (requestServletPath.startsWith("/admin") && null ==
request.getSession().getAttribute("name")) {
            request.getSession().setAttribute("errorMsg", "请重新登陆");
            response.sendRedirect(request.getContextPath() + "/admin/userlogin");
            return false;
        } else {
            request.getSession().removeAttribute("errorMsg");
            return true;
        }
    }
}

```

5.1.2 查看已发布的游戏模块

本游戏商城的游戏商城页面使用了 Thymeleaf,Ajax 等多项技术，当用户进入游戏商城页面时，后端端点会将数据库中的所有数据通过 session 发送到网页，网页会通过 Thymeleaf 技术动态的将数据库中的数据应用到页面中，同时本页面还运用 Bootstrap 技术来保证本页面元素的统一性

相关代码如下：

代码 5-5 游戏列表

```

@GetMapping("game")
public String showGames(Model model) {
    List<Game> games = gameService.findAllGames();
    model.addAttribute("games", games);
    return "gamefront/game";
}
//前端部分

<div class="container mt-3">
    <div class="row">
        <div class="col-sm-4" th:each="game : ${games}">
            <div class="card">

```

```

<div class="card-body">
    <h4 class="card-title" th:text="${game.dname}"></h4>
    <p class="card-text">Game Type: <span th:text="${game.dtel}"></span></p>
    <p class="card-text">Price: <span th:text="${game.description}"></span></p>
    <a th:href="@{/admin/game/details(id=${game.id})}" class="btn btn-primary"><i
class="fas fa-eye"></i> View Details</a>
</div>
</div>
</div>
</div>
</div>
```

本商店的商店页面如图所示：

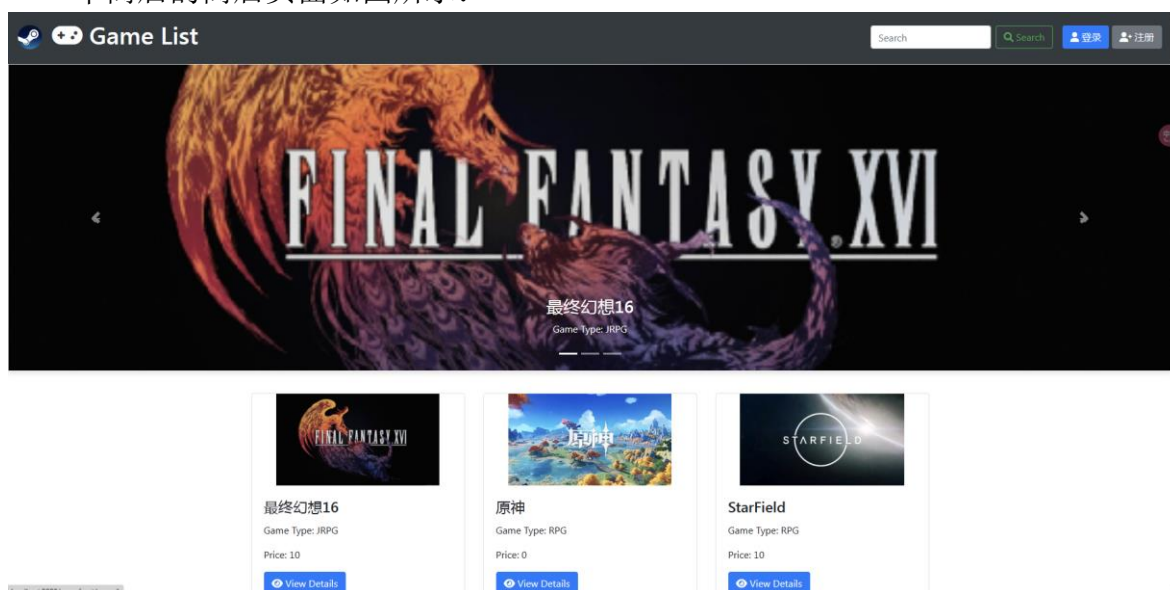


图 5-4 游戏商品列表

5.1.3 查看游戏商品详情模块

在本游戏商城中，当用户在游戏列表界面上点击某个游戏的名称时，系统会自动导航到该游戏的详细页面。这个详细页面精心设计，展示了游戏的名称、价格、简介和概览图像。页面还配备了“购买游戏”和“添加到购物车”的按钮。

为了实现这一流程，前端使用 Ajax 技术局部刷新页面，避免了整个页面的重新加载。当用户进行购买或添加到购物车的操作时，前端会向后端发送游戏的 ID 和用

户的 ID。后端服务接收到这些信息后，使用 Mybatis 框架连接到 MySQL 数据库。它会根据提供的 ID 在游戏表中查询相关的游戏信息，如名称、价格、简介和概图地址。同时，它还会在游戏购买表中统计该游戏的已购买次数，并检查该用户是否已经购买了该游戏。

一旦后端服务完成了这些查询和统计操作，它会使用 Java 的泛型功能将这些数据打包成 JSON 格式，并发送回前端。前端接收到这些数据后，会解析并渲染它们，从而为用户展示最新和最相关的游戏详情。这整个过程旨在为用户提供流畅、高效和直观的购物体验。

具体 Controller 层代码如下：

代码 5-6 游戏商品详情

```
@GetMapping("/game/details")
public String gameDetails(@RequestParam("id") int id, Model model){
    Game game = gameService.GetGameInfoById(id);
    System.out.println(id);
    Comment comment = commentService.showcomment(id);
    model.addAttribute("gameDetails", game);
    model.addAttribute("comment",comment);
    System.out.println(comment);
    return "gamefront/gameDetails";
}
```

游戏详情页面具体展示如下图图 5-4 所示：

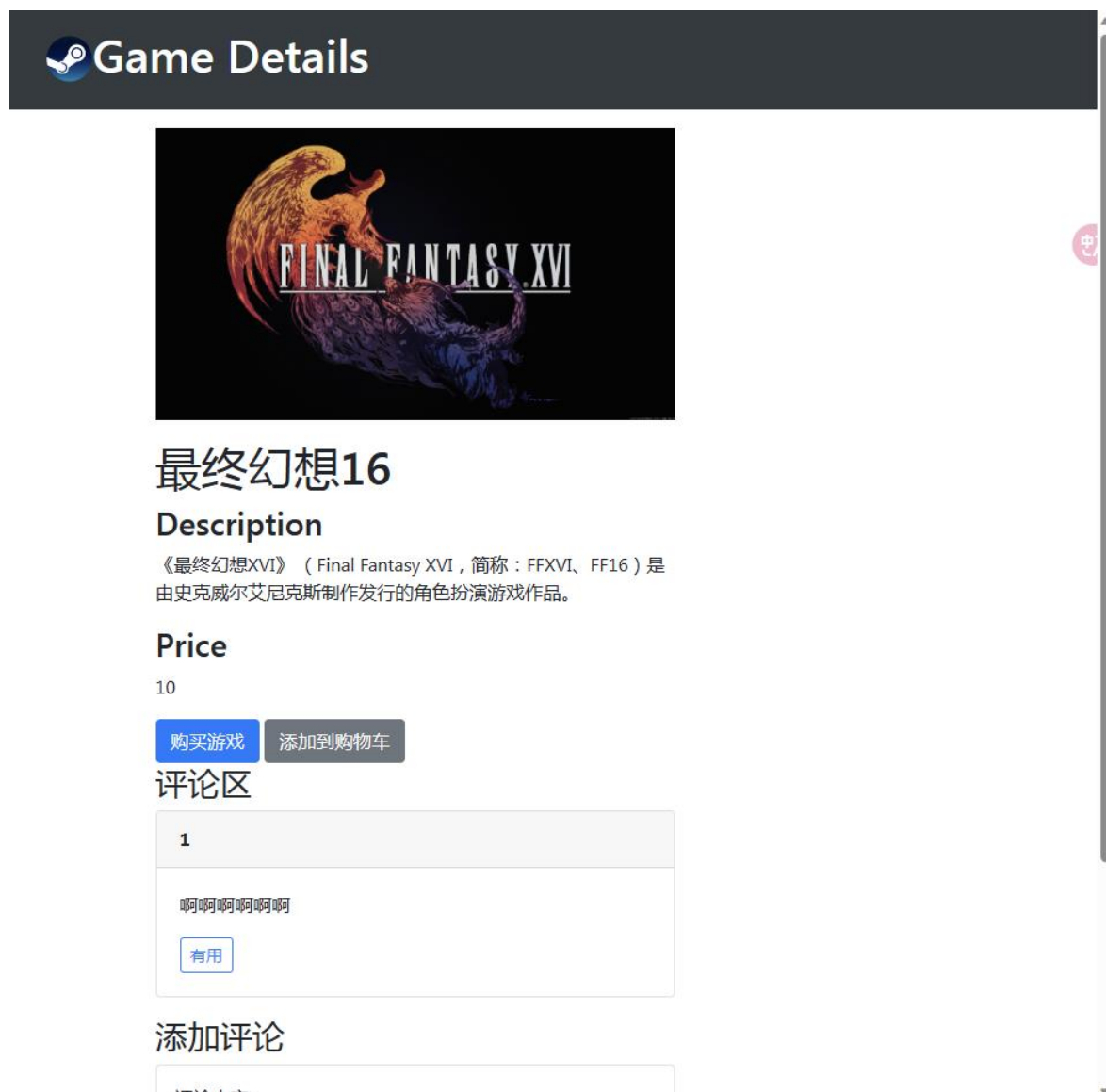


图 5-5 游戏商品详情

5.1.7 我的购物车模块

在“我的购物车”界面，用户可以看到所有已添加到购物车的游戏。每个游戏都以卡片的形式展现，包括游戏的封面图片、名称以及游戏类型。用户可以选择支付虚拟货币购买已添加进购物车的游戏，或者选择从购物车中移除不想购买的游戏。

为了实现这一功能，我们使用了 Vue 和 Ajax 技术来实现前端的页面控制。当用户决定购买某个游戏时，会弹出一个支付窗口。点击支付后，后端会使用 Spring 和 Mybatis 的事务管理来处理购买请求。

事务管理确保了所有操作要么全部成功，要么全部失败。这是通过 MySQL 的事务功能实现的，它可以保证数据的完整性和一致性。在事务开始时，会锁定用户

表，这意味着在事务处理期间，用户表只能被当前事务操作。这样可以防止在读取用户金额时出现的脏读、不可重复读和幻读等问题。一旦用户数据被成功更新，事务就会结束，释放对用户表的锁定。

此外，如果在处理过程中出现任何错误，MySQL 支持事务的回滚功能，可以将数据恢复到事务开始前的状态，确保数据的安全性。

购物车页面如图：

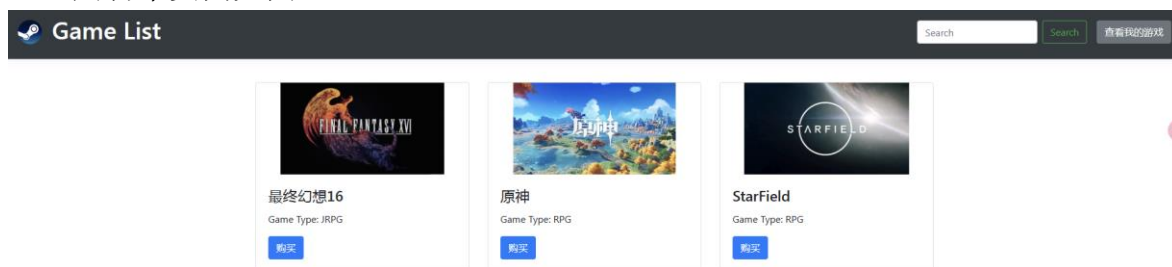


图 5-9 我的购物车

代码 5-9 购物车记录

```
<div class="col-sm-4" th:each="game : ${cart}">
  <div class="card">
    <!-- 假设购物车游戏信息的 Thymeleaf 变量为 entry.value -->
    
    <div class="card-body">
      <h4 class="card-title" th:text="${game.value.dname}"></h4>
      <p class="card-text">Game Type: <span th:text="${game.value.dtel}"></span>
    </p>
      <button class="btn btn-primary" onclick="console.log('${game.value.id}');
buyGame(parseInt('${game.value.id}'))">购买</button>
    </div>
  </div>
</div>
```

5.1.8 添加游戏商品到购物车

当在游戏详情页面点击添加进购物车，相应的信息会写入到 session 中，前端会根据 session 中的数据在购物车页面中显示添加进购物车的游戏，用户可以在购物车页面查看

添加进购物车页面如下：



最终幻想16

Description

《最终幻想XVI》（Final Fantasy XVI，简称：FFXVI、FF16）是由史克威尔艾尼克斯制作发行的角色扮演游戏作品。

Price

10

购买游戏

添加到购物车

图 5-10 添加到购物车

代码 5-10 添加到购物车

```
@PostMapping("/addGameToCart")
@ResponseBody
public Map<String, Object> addGameToCart(@RequestParam("gameId") int gameId,
HttpSession session) {
    Map<String, Object> map = new HashMap<>();
    try {
        // 从数据库中获取游戏信息
        Game game = gameService.getgameById(gameId);
        if (game != null) {
```

```
// 从 session 中获取购物车列表
Map<Integer, Game> cartItems = (Map<Integer, Game>)
session.getAttribute("cartItems");

// 如果购物车列表为空，则创建一个新的购物车列表
if (cartItems == null) {
    cartItems = new HashMap<>();
}

// 将游戏对象存储到购物车列表中，使用游戏 ID 作为键
cartItems.put(gameId, game);

// 将购物车列表重新存储回 session 中
session.setAttribute("cartItems", cartItems);

map.put("success", true);
map.put("message", "游戏已添加到用户购物车");
} else {
    map.put("success", false);
    map.put("message", "无法找到游戏");
}
} catch (Exception e) {
    e.printStackTrace();
    map.put("success", false);
    map.put("message", "服务器错误");
}
return map;
}
```

5.1.9 用户管理模块

管理员从登录页面登录后，可以在后台页面进行用户管理，该页面会显示用户

的用户名，以及被加密的密码，管理员可以在这里进行相应操作，同时，当用户发表不当或非法言论时，管理员可在评论管理中选中该条评论进行删除，当用户屡教不改时，可以对该用户进行删号处理

具体前端展示如下图所示：

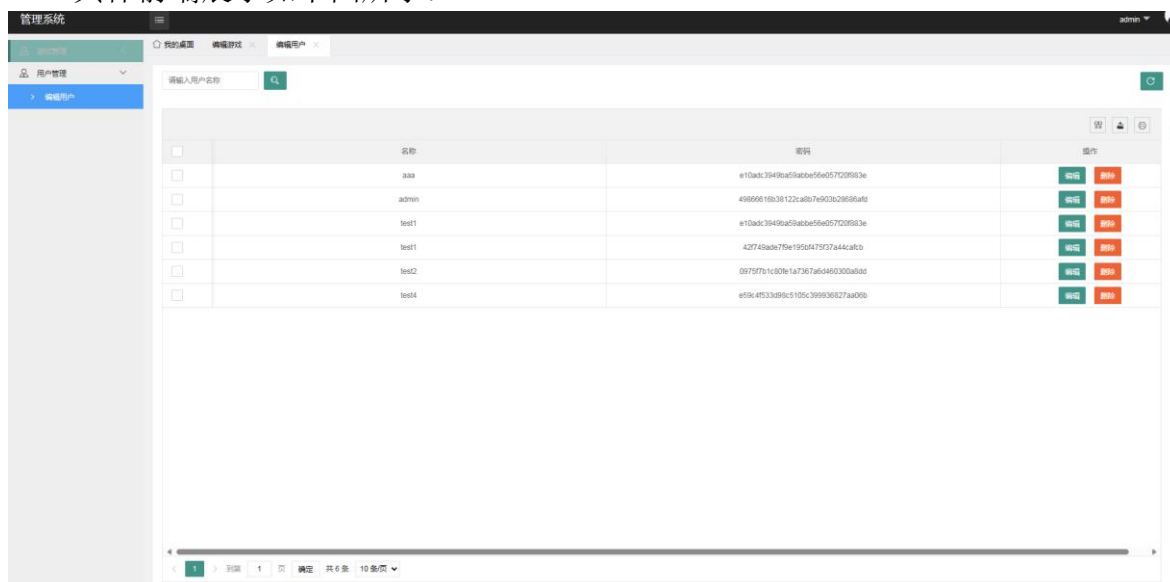


图 5-11 用户管理

代码 5-11 用户管理

```
<body>
<div class="layui-fluid">
  <div class="layui-row">
    <div class="layui-form">
      <div class="layui-form-item">
        <input th:if="${userInfo!=null}" type="hidden" id="id" name="id"
th:value="${userInfo?.id}">
      </div>
      <div class="layui-form-item">
        <label for="name" class="layui-form-label">
          <span class="x-red">*</span> 用户名称</label>
        <div class="layui-input-inline">
          <input type="text" id="name" name="name" lay-verify="required" placeholder="输
```

入用户名称（必填）"

```
autocomplete="off" class="layui-input" th:value="${userInfo?.name}"></div>
```

```
</div>
```

```
<div class="layui-form-item">
```

```
<label for="pwd" class="layui-form-label">
```

```
<span class="x-red">*</span>用户密码</label>
```

```
<div class="layui-input-inline">
```

```
<input type="text" id="pwd" name="pwd" lay-verify="required" placeholder="输入
```

用户密码（必填）"

```
autocomplete="off" class="layui-input" th:value="${userInfo?.pwd}"></div>
```

```
</div>
```

```
<div class="layui-form-item">
```

```
<label class="layui-form-label"></label>
```

```
<button class="layui-btn" lay-filter="submit" lay-submit="">保存</button>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

5.1.10 游戏商品管理模块

为了进行有效管理，管理员可以添加新的游戏商品或因各种原因需要移除的游戏。这里的编辑和删除功能，在添加或编辑游戏时，主要是通过前端弹出一个模态窗口，让管理员填写或更改游戏的详细信息，然后提交。一旦服务器接收到这些游戏数据，它会整理这些信息，并通过 Mybatis 进行处理，从而更新数据库中的相关数据。一旦数据更新成功，前端会利用 Ajax 获取最新数据，并 e 进行快速渲染，从而实时更新游戏商品列表，这大大提高了管理员的工作效率。具体的操作界面如下图所示：

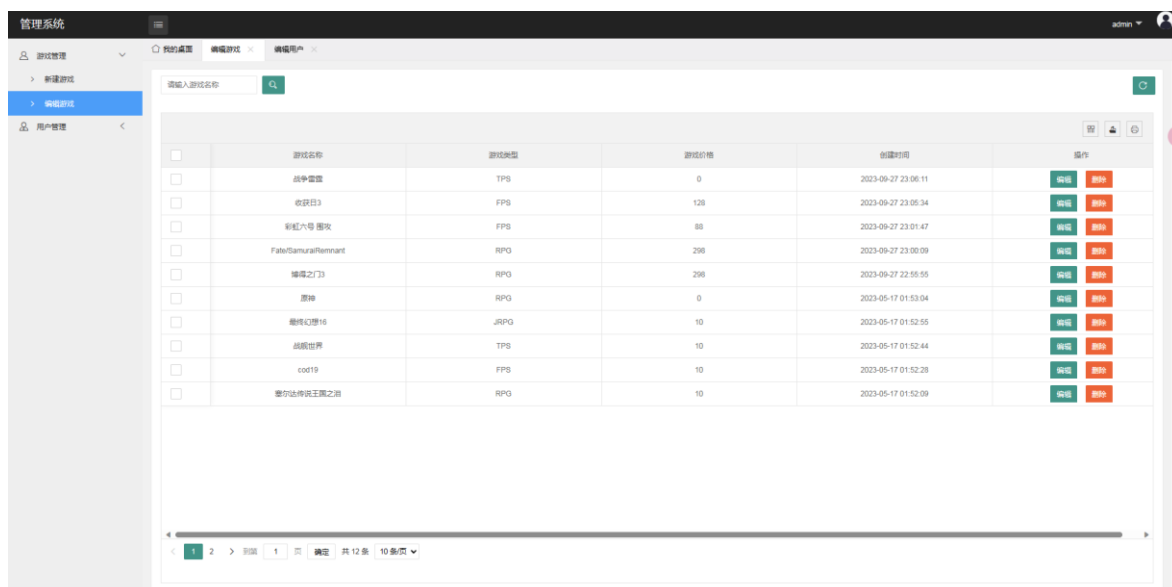


图 5-12 游戏管理

5.1.11 评论管理模块

在游戏详情或帖子页面，用户可以自由地发表自己的评论。然而，为了确保社区的健康和秩序，我们提供了一个评论管理功能，供管理员使用。通过这个功能，管理员可以浏览所有用户发布的评论，并有权删除那些不合适或非法的评论。

当管理员发现某条评论违反了社区规定或法律法规时，他们可以直接在评论旁点击“删除”按钮，将其从系统中移除。此外，为了提高管理效率，我们还为管理员提供了一个搜索功能，使他们可以通过关键词快速找到特定的评论。：

（待添加）

图 5-13 评论管理

代码 5-12 评论管理

待添加

5.1.12 智能推荐模块

随着游戏商城的日益增长，为用户提供个性化的游戏推荐变得尤为重要。智能推荐模块的目标是根据用户的购买历史，为其推荐可能感兴趣的遊戲。为了实现这一目标，本人采用了基于物品的协同过滤算法。简单来说，该算法会考虑用户过去购买的游戏类型（如 RPG、动作、冒险等），并找出与这些游戏类型相似的其他游戏推荐给用户。

当用户登录并浏览游戏商城时，系统会自动为其生成推荐列表。这是通过以下

步骤实现的：

获取用户的购买历史。

根据购买历史中的游戏类型，找出商城中与之相似的其他游戏。

将这些游戏按照某种逻辑（如相似度、新颖度等）排序，并展示给用户。

代码 5-14 智能推荐实现

| |
|-------|
| 待添加代码 |
|-------|

在游戏商城的首页或专门的推荐页面，系统会展示为用户推荐的游戏列表。用户可以直观地看到基于其购买历史为其推荐的游戏，从而更容易找到自己可能感兴趣的

（待添加）

图 5-15 智能推荐界面

5.2 本章小结

本章主要通过相应代码和图片，同时以文字描述了本系统中的核心功能，通过这些功能的实现，本系统基本满足了前述的相关需求分析中所需要的功能，本章给出了登录注册，游戏展示，用户管理，评论区管理，以及智能推荐模块的相应代码，以证明本系统对于核心功能的实现。

6 系统测试

详细的系统测试能够在检测本系统质量的同时，同时还能够检测本系统存在的漏洞，有针对性的对本系统进行详细的测试，不仅能够完成系统错误的检测，还能够进一步优化系统的功能。

6.1 测试环境

测试结果基于固定的电脑型号以及配置来进行测试，以减少在测试中的变量：

6.1.1 硬件环境

表 6-1 硬件环境

| | |
|--------|------------|
| CPU 型号 | I7-10875h |
| CPU 主频 | 2.3GHz |
| 物理内存 | 16GB |
| 物理硬盘 | 1.5tb 固态硬盘 |

6.1.2 软件环境

表 6-2 软件环境

| | |
|---------------|----------------|
| 操作系统 | Windows 11 专业版 |
| IntelliJ IDEA | 2023.1 |
| MySQL 版本 | 8.0.28（64 位） |
| JDK 版本 | 1.8 |

6.2 系统功能点测试

6.2.1 用户登录注册验证功能测试

用户登录注册验证功能主要包括登录，注册，身份验证组成，本测试的目的是为了检测系统功能是否正常工作，测试内容及结果如下所述。

表 6-3 用户身份认证接口测试

| 用例 | 接口地址 | 请求方式 | 接口参数 | 结果 |
|------|-----------------|------|-----------|----|
| 用户登录 | admin/userlogin | POST | userlogin | 正常 |
| 用户注册 | admin/signup | GET | signup | 正常 |

| | | | | |
|----------|--------------------------|------|----------------|----|
| 管理员登录 | admin/userlogin | POST | userlogin | 正常 |
| 获取当前用户信息 | admin/api/getCurrentUser | GET | getcurrentuser | 正常 |
| 用户修改自身信息 | user/edit | GET | edit | 正常 |

表 6-4 用户登录接口测试

| | |
|------|---|
| 测试目的 | 通过普通用户账号登录本系统 |
| 前置条件 | 无 |
| 操作步骤 | 输入普通用户账号密码 |
| 预期结果 | 跳转到游戏商城首页 |
| 实际结果 |  |
| 测试结果 | 跳转到游戏商城首页 |

表 6-5 用户注册接口测试

| | |
|------|--|
| 测试目的 | 用户注册新账号 |
| 前置条件 | 无 |
| 操作步骤 | 填写用户名、密码 |
| 预期结果 | 弹出注册成功 |
| 实际结果 |  |


| | |
|------|---|
| |  |
| 测试结果 | 弹出注册成功 |

表 6-6 管理员登录接口测试

| | |
|------|---|
| 测试目的 | 管理员通过管理员账户登录 |
| 前置条件 | 无 |
| 操作步骤 | 管理员填写管理账号和密码点击登录 |
| 预期结果 | 跳转到游戏商城后台页面 |
| 实际结果 |  |
| 测试结果 | 跳转到游戏商城后台页面 |

表 6-7 获取当前用户信息测试

| | |
|------|----------------------|
| 测试目的 | 用户登录后，可以查看自身的用户信息 |
| 前置条件 | 用户登录 |
| 操作步骤 | 用户点击我的账号 |
| 预期结果 | 查看到用户自身的账号、密码、手机号等信息 |

| | |
|------|----------------------|
| 实际结果 | 待添加 |
| 测试结果 | 查看到用户自身的账号、密码、手机号等信息 |

表 6-8 用户修改自身信息测试

| | |
|------|----------------------|
| 测试目的 | 用户登录后，可以修改自身的用户信息 |
| 前置条件 | 用户登录 |
| 操作步骤 | 用户填写想要修改的内容，然后点击修改 |
| 预期结果 | 用户自身的账号、密码、手机号等信息已修改 |
| 实际结果 | 待添加 |
| 测试结果 | 用户自身的账号、密码、手机号等信息已修改 |

6.2.2 游戏商城功能测试

游戏商城功能包括游戏展示，游戏详情，购物车，用户的已购游戏，本测试结果如下，各端点未显异常。

表 6-9 游戏商城与购物车接口测试

| 用例 | 接口地址 | 请求方式 | 接口参数 | 结果 |
|----------|-------------------------|------|---------------|----|
| 显示全部游戏 | gamefront/game | POST | games | 正常 |
| 查看游戏详情 | game/details | GET | game | 正常 |
| 添加游戏到购物车 | gamefront/addGameToCart | POST | gameId userId | 正常 |
| 查看购物车 | gamefront/cart | POST | userId | 正常 |

表 6-10 查看已发布游戏列表测试

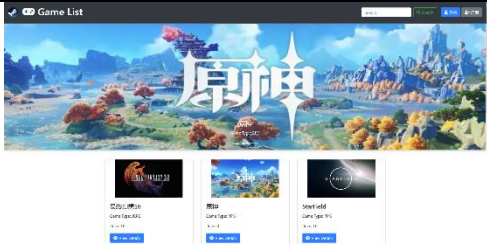
| | |
|------|--|
| 测试目的 | 测试查看商城已有游戏 |
| 前置条件 | 无 |
| 操作步骤 | 进入商城首页 |
| 预期结果 | 看到游戏列表 |
| 实际结果 |  |
| 测试结果 | 看到游戏列表 |

表 6-11 查看游戏详情测试

| | |
|------|--|
| 测试目的 | 查看游戏详情，如简介评论等 |
| 前置条件 | 无 |
| 操作步骤 | 在游戏列表界面点击“view details” |
| 预期结果 | 展示游戏详情 |
| 实际结果 |  |
| 测试结果 | 展示游戏详情 |

表 6-12 添加游戏到购物车测试

| | |
|------|-------------------|
| 测试目的 | 用户将游戏添加进购物车 |
| 前置条件 | 用户登录 |
| 操作步骤 | 在游戏详情页面点击“添加到购物车” |
| 预期结果 | 弹出“已添加到购物车” |

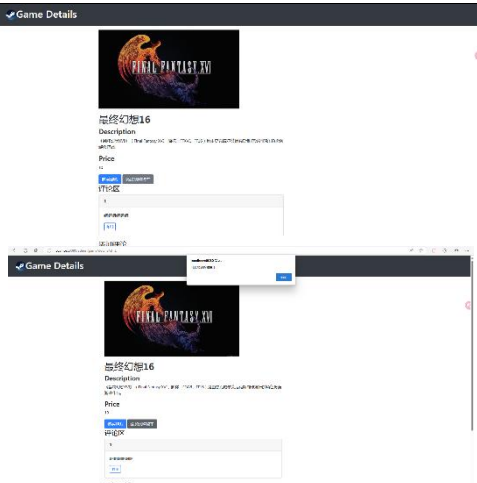
| | |
|------|--|
| 实际结果 |  |
| 测试结果 | 弹出“已添加到购物车” |

表 6-13 查看购物车测试

| | |
|------|--|
| 测试目的 | 用户查看购物车已有的游戏 |
| 前置条件 | 用户登录 |
| 操作步骤 | 用户进入游戏列表界面，点击“查看我的购物车” |
| 预期结果 | 展示添加进入购物车的游戏 |
| 实际结果 |  |
| 测试结果 | 展示添加进入购物车的游戏 |

6.2.3 后台管理系统测试

表 6-14 后台管理系统接口测试

| 用例 | 接口地址 | 请求方式 | 接口参数 | 结果 |
|------|--------------|------|----------|----|
| 用户管理 | user/listall | GET | userList | 正常 |
| 评论管理 | 待添加 | GET | 待添加 | 正常 |

| | | | | |
|------|------------|-----|---|----|
| 游戏管理 | admin/game | GET | 无 | 正常 |
|------|------------|-----|---|----|

表 6-15 用户管理测试

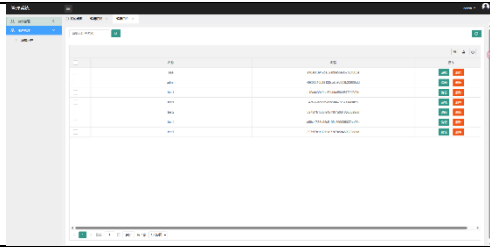
| | |
|------|--|
| 测试目的 | 管理员通过该功能查看所有用户 |
| 前置条件 | 管理员登录 |
| 操作步骤 | 点击用户管理 |
| 预期结果 | 展示系统已有的用户部分信息 |
| 实际结果 |  |
| 测试结果 | 展示系统已有的用户部分信息 |

表 6-16 用户评论管理测试

| | |
|------|-----------------|
| 测试目的 | 管理员管理评论 |
| 前置条件 | 管理员登录 |
| 操作步骤 | 管理员点击评论管理，可删除评论 |
| 预期结果 | 用户评论被删除 |
| 实际结果 | 待添加 |
| 测试结果 | 用户评论被删除 |

表 6-17 游戏管理测试

| | |
|------|--|
| 测试目的 | 管理员能够看到已发布的游戏 |
| 前置条件 | 管理员登录 |
| 操作步骤 | 点击游戏管理 |
| 预期结果 | 展示游戏的信息列表 |
| 实际结果 |  |
| 测试结果 | 展示游戏的信息列表 |

6.3 本章小结

在这一章节中，主要对系统的功能和性能进行了深入的检验，采用了 SoapUI 作为测试工具。经过一系列的验证，本游戏商城的所有功能都能够顺利执行，同时响应速度也相当迅速，达到了预期的标准。此外，本项目展现出了长期的稳定性和可靠性，基本上达到了用户的期望。（待修改）

7 总结与展望

待添加

参考文献

[1] 待添加

致谢

待添加