**校园二手交易APP软件需求说明书**

目录

[1. 引言 2](#_Toc179410143)

[1.1 文件信息 2](#_Toc179410144)

[1.2 使用人员 2](#_Toc179410145)

[1.3 编写目的 3](#_Toc179410146)

[1.4 背景 3](#_Toc179410147)

[1.5 定义与缩写 3](#_Toc179410148)

[1.6 参考资料 3](#_Toc179410149)

[2. 任务概述 3](#_Toc179410150)

[2.1 目标 3](#_Toc179410151)

[2.2 用户特点 4](#_Toc179410152)

[2.3 应用环境 4](#_Toc179410153)

[2.4 假定和约束 5](#_Toc179410154)

[3. 需求规定 5](#_Toc179410155)

[3.1 对功能的规定 5](#_Toc179410156)

[3.1.1 总体说明 5](#_Toc179410157)

[3.1.2 业务流程 5](#_Toc179410158)

[3.1.3 数据字典 6](#_Toc179410159)

[用户信息表（User\_Info） 6](#_Toc179410160)

[商品信息表（Product\_Info） 7](#_Toc179410161)

[订单信息表（Order\_Info） 7](#_Toc179410162)

[3.2 对性能的规定 8](#_Toc179410163)

[3.2.1总体说明 8](#_Toc179410164)

[3.2.2精度 9](#_Toc179410165)

[3.2.3时间特性要求 9](#_Toc179410166)

[3.2.4灵活性 10](#_Toc179410167)

[3.3 输入输出要求 10](#_Toc179410168)

[3.3.1用户信息输入 10](#_Toc179410169)

[3.3.2 输出要求 12](#_Toc179410170)

[3.4 数据库与缓存策略 13](#_Toc179410171)

[3.4.1 MySQL数据库存储 13](#_Toc179410172)

[3.4.2 Redis缓存优化 14](#_Toc179410173)

[4. 运行环境规定 14](#_Toc179410174)

[4.1 设备 14](#_Toc179410175)

[4.2 支持软件 15](#_Toc179410176)

[4.3 接口 15](#_Toc179410177)

[4.4 控制 15](#_Toc179410178)

# 1. 引言

## 1.1 文件信息

文件状态：已完成

文件标识：“校园二手交易APP需求说明书”

当前版本：1.0

作者：陈志浩、马萨里沐、吴仲飞

完成日期：2024年10月7日

版本更新信息：无

团队成员信息：

姓名：[团队成员1姓名]

学号：[团队成员1学号]

团队：校园二手交易APP开发团队

姓名：[团队成员2姓名]

学号：[团队成员2学号]

团队：校园二手交易APP开发团队

## 1.2 使用人员

**学生**

作为主要的用户群体，学生将使用该APP进行二手商品的买卖、浏览和交易管理。

**商家**

提供二手商品的卖家，他们需要利用该APP快速上架商品，并有效管理商品、订单和配送流程。

**管理员**

管理平台用户和商家，负责监督整个交易流程，处理用户投诉，管理用户账户和权限。

## 1.3 编写目的

本软件需求说明书旨在为基于SSM框架（Spring + SpringMVC + MyBatis）开发的校园二手交易APP提供详尽的功能需求和非功能需求说明。其主要目标是确保开发人员、测试人员、项目经理及其他相关人员对系统的功能需求有统一、清晰的理解，并为后续的开发、测试、部署及维护工作提供准确、可靠的依据。

## 1.4 背景

随着校园内二手商品交易需求的日益增长，现有的交易方式往往分散、不便，缺乏一个集中化、智能化的交易平台。本项目旨在开发一个基于SSM框架的Android校园二手交易APP，以满足学生和校园商家在二手商品交易中的实际需求。该系统将提供一个便捷、高效、安全的交易平台，涵盖商品搜索、购物车管理、下单支付、订单和配送管理等核心功能，旨在提升校园二手商品交易的效率和用户体验。

## 1.5 定义与缩写

**SSM**：Spring + SpringMVC + MyBatis框架，用于构建后端服务。

**APP**：Android应用程序，指校园二手交易APP。

**API**：应用程序接口，用于前后端数据交互。

**MySQL**：关系型数据库管理系统，用于存储用户、商品、订单等数据。

**Redis**：高性能缓存数据库，用于提高系统响应速度和数据处理能力。

**Nginx**：高性能的HTTP和反向代理web服务器，用于处理静态资源请求和负载均衡。

**Tomcat**：Apache软件基金会（ASF）开发的Servlet容器，用于部署Java Web应用。

## 1.6 参考资料

《Spring官方文档》：详细介绍Spring框架的使用方法和最佳实践。

《MyBatis用户指南》：全面解析MyBatis框架的功能和用法。

《Android开发文档》：Android官方提供的开发指南和API参考。

# 2. 任务概述

## 2.1 目标

本项目旨在开发一个基于SSM框架的Android校园二手交易APP，实现以下主要功能：

1. 用户注册/登录功能
2. 商品搜索功能
3. 购物车管理功能
4. 下单与支付功能
5. 订单和配送管理功能
6. 商家管理功能
7. 用户管理功能

## 2.2 用户特点

**学生用户**

熟悉手机操作，对界面友好性和易用性有较高要求。

期待一个易于使用的界面和便捷的交易流程。

关注商品的价格、质量、卖家信誉等信息。

**商家用户**

希望快速上架商品，并有效管理商品信息、订单状态和配送流程。

期待系统简洁高效，能够支持批量操作和多商品管理。

关注系统的稳定性和数据安全性。

**管理员用户**

需要一个后台系统来管理用户、商品和订单信息。

负责监督整个交易流程，处理用户投诉和交易异常。

关注系统的权限管理和数据安全控制。

## 2.3 应用环境

**前端设备**

本APP设计用于Android设备，兼容并支持Android 5.0及以上版本的各种智能手机和平板设备。这意味着，只要用户的设备运行的是Android 5.0或更高版本，他们就可以下载并安装本APP，享受二手商品交易的便捷服务。

**服务器端**

服务器端架构采用先进的SSM（Spring + SpringMVC + MyBatis）框架，这是一种流行的Java Web应用框架组合，提供了高度的灵活性和可扩展性。数据库系统选择的是MySQL，它是一款功能强大、易于使用的开源关系型数据库管理系统，能够满足大多数中小型应用的数据存储需求。此外，为了提高系统性能和响应速度，我们还引入了Redis作为缓存系统，它能够有效地存储和读取数据，减少数据库访问次数，从而提升整体性能。

**网络要求**

为了确保用户能够流畅地使用APP的各项功能，需要有一个稳定的网络连接。无论是Wi-Fi还是4G/5G网络，只要能够提供足够的带宽和低延迟，都可以满足APP的运行需求。在网络环境不稳定或中断的情况下，APP将尝试使用本地缓存的数据来保持部分功能的可用性，同时提示用户检查网络连接。

## 2.4 假定和约束

在系统设计和开发过程中，我们基于一些合理的假定和约束来制定决策。首先，我们假定系统必须能够处理高并发下的订单交易，这是因为在校园内，特别是在一些特殊的时期（如学期末、毕业季等），二手商品的交易需求可能会激增，系统需要能够承受这种压力。其次，我们假定所有支付交易都通过安全的第三方支付接口完成，以确保用户资金的安全性和合规性。此外，我们还假定数据必须进行加密存储，以保护用户隐私和敏感信息不被泄露。

在约束方面，我们主要面临的是技术选型、资源限制和法律法规等方面的限制。技术选型方面，我们已经确定了使用SSM框架、MySQL数据库和Redis缓存等关键技术，这在一定程度上限制了我们在技术上的选择。资源限制方面，我们需要在有限的预算和时间内完成系统的开发、测试和部署工作，因此需要在功能需求、性能需求和开发成本之间做出权衡。法律法规方面，我们需要遵守相关的法律法规要求，如用户隐私保护、数据安全和电子支付等方面的规定。

# 3. 需求规定

## 3.1 对功能的规定

### 3.1.1 总体说明

本部分将对APP的核心功能需求进行总体概述，旨在提供一个清晰的功能框架，为后续详细设计提供指导。APP旨在打造一个安全、便捷、高效的二手商品交易平台，满足校园内师生对二手商品交易的需求。通过集成多种功能模块，APP将为用户提供注册登录、商品搜索、购物车管理、下单支付、订单管理、配送追踪以及商家管理等一站式服务。这些功能将共同构成一个完整的交易生态系统，确保用户能够轻松完成从商品浏览到交易完成的整个过程。

### 3.1.2 业务流程

**用户注册与登录**

用户可以通过手机号或邮箱进行注册，填写基本信息并完成验证。

注册成功后，用户可以使用手机号或邮箱及密码进行登录。

为提升用户体验和安全性，APP支持短信验证码登录。

**商品搜索与浏览**

用户可以通过商品分类和关键字进行搜索，快速找到感兴趣的二手商品。

搜索结果将展示商品的基本信息，包括商品名称、价格、图片、卖家信息等。

APP支持模糊搜索和智能推荐功能，提高搜索的准确性和效率。

**购物车管理**

用户可以将感兴趣的商品添加到购物车中，进行批量购买或管理。

购物车支持批量删除和修改数量等操作。

购物车与下单功能无缝对接，用户可以直接从购物车进入下单页面进行支付。

**下单与支付**

用户可以从购物车中选择要购买的商品，填写收货信息并选择支付方式。

系统会对订单信息进行校验和审核，确保订单的有效性和合规性。

支付成功后，系统将生成唯一的订单号，并通知卖家进行发货。

**订单管理**

用户可以在订单管理页面中查看历史订单信息，包括订单号、商品名称、价格、购买时间、物流信息等。

用户可以对订单进行取消、退款等操作（需符合平台规定）。

**配送追踪**

商家可以管理订单的发货状态，包括标记订单为已发货、填写物流信息等。

用户可以通过APP追踪自己的物流信息，实时了解商品的配送进度。

**商家管理**

商家拥有独立的后台管理界面，可以上架新商品、修改商品信息、删除不再出售的商品等。

商家可以管理自己的订单和售后服务，包括处理用户的退款请求和投诉等。

### 3.1.3 数据字典

### 用户信息表（User\_Info）

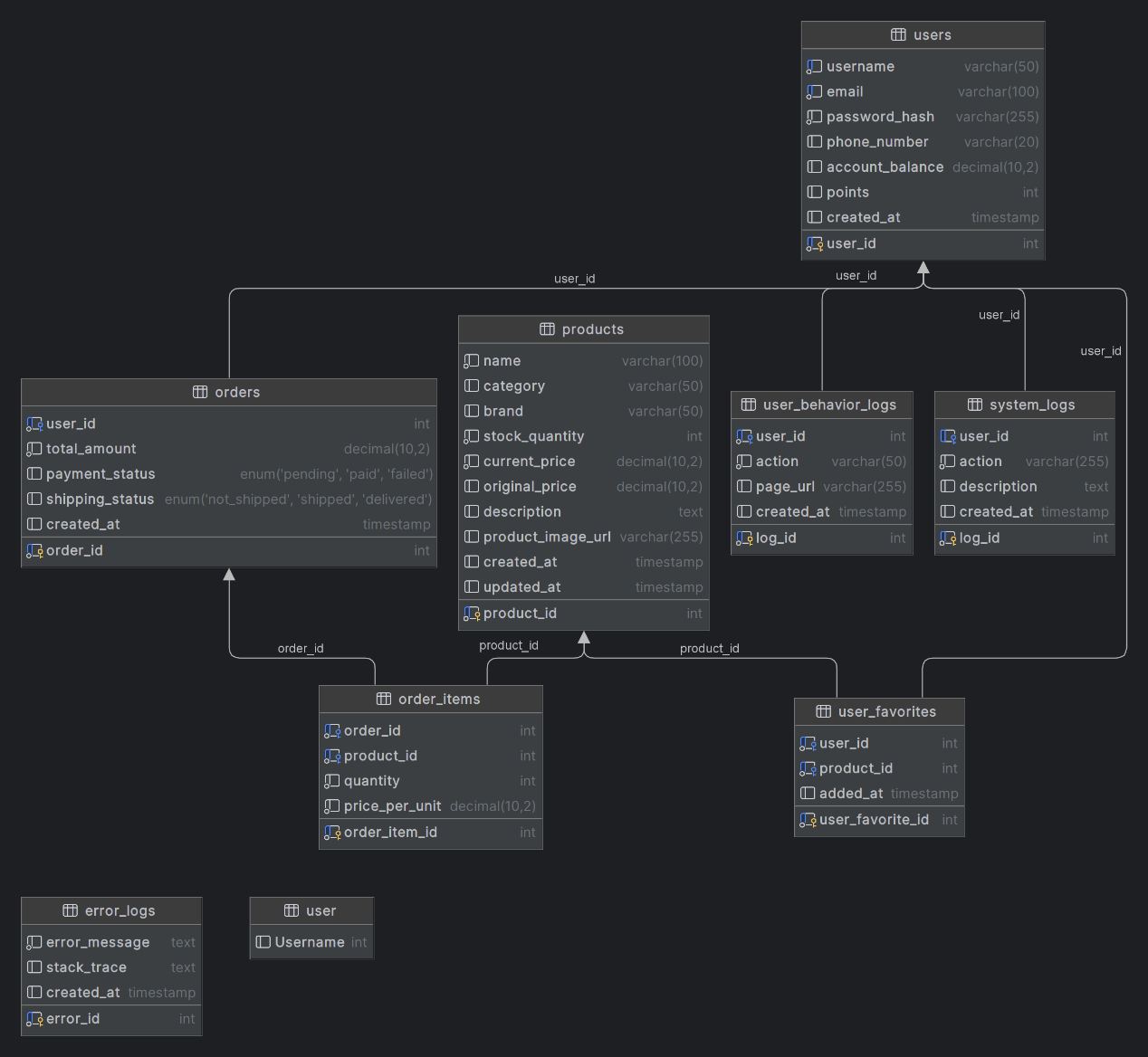
| **字段名称** | **数据类型** | **主键** | **备注** |
| --- | --- | --- | --- |
| 用户ID | INT | 是 | 主键，自增 |
| 用户名 | VARCHAR(255) | 否 | 唯一索引 |
| 密码 | VARCHAR(255) | 否 | 存储加密后的密码 |
| 手机号 | VARCHAR(11) | 否 | 唯一索引 |
| 邮箱 | VARCHAR(255) | 否 | 唯一索引 |
| 注册时间 | DATETIME | 否 |  |
| 最后登录时间 | DATETIME | 否 |  |

### 商品信息表（Product\_Info）

| **字段名称** | **数据类型** | **主键** | **备注** |
| --- | --- | --- | --- |
| 商品ID | INT | 是 | 主键，自增 |
| 商品名称 | VARCHAR(255) | 否 |  |
| 价格 | DECIMAL(10, 2) | 否 | 货币类型 |
| 库存数量 | INT | 否 |  |
| 商品描述 | TEXT | 否 |  |
| 图片链接 | VARCHAR(255) | 否 |  |
| 卖家ID | INT | 否 | 外键，关联用户信息表的用户ID |

### 订单信息表（Order\_Info）

| **字段名称** | **数据类型** | **主键** | **备注** |
| --- | --- | --- | --- |
| 订单ID | INT | 是 | 主键，自增 |
| 用户ID | INT | 否 | 外键，关联用户信息表的用户ID |
| 商品ID | INT | 否 | 外键，关联商品信息表的商品ID |
| 购买数量 | INT | 否 |  |
| 订单金额 | DECIMAL(10, 2) | 否 | 货币类型 |
| 支付状态 | VARCHAR(50) | 否 | 如：已支付、未支付 |
| 发货状态 | VARCHAR(50) | 否 | 如：已发货、未发货 |
| 物流信息 | VARCHAR(255) | 否 | 如：快递单号 |



系统ER图

## 3.2 对性能的规定

### 3.2.1总体说明

在当今快速发展的技术环境中，系统性能是衡量软件质量的关键指标之一。本部分将对系统的性能要求进行全面阐述，旨在确保系统能够满足用户期望，提供高效、稳定、可靠的服务。性能要求涵盖了多个方面，包括精度、时间特性以及灵活性，这些要求将共同构成系统性能评估的基准。

### 3.2.2精度

系统应能够准确识别和处理用户输入的数据，包括数字、字符、日期等，误差率不超过0.01%。

对于模糊输入或异常输入，系统应提供清晰的错误提示，并引导用户进行正确的输入，错误识别率不超过0.05%。

系统在处理数据时，应确保计算结果的准确性，避免由于算法错误或数据溢出等问题导致的精度损失，计算误差率不超过0.001%。

对于需要高精度的计算任务，系统应采用适当的数值计算方法和精度控制策略，如双精度浮点运算、误差累积控制等，以确保计算结果的准确性。

系统输出的数据应清晰、准确，符合用户期望和行业标准，输出误差率不超过0.01%。

对于需要展示大量数据的场景，系统应采用有效的数据可视化方法，如图表、曲线等，以提高数据的可读性和理解性，同时确保数据展示的精度和准确性。

### 3.2.3时间特性要求

时间特性是衡量系统响应速度和处理能力的重要指标。系统应能够快速响应用户请求，并在合理的时间内完成数据处理任务。

**响应时间**

系统应能够在用户提交请求后的合理时间内给出响应，避免用户等待时间过长。对于实时性要求较高的功能，如在线聊天、实时交易等，系统应确保响应时间在50毫秒至500毫秒范围内。

在高负载情况下，系统应能够保持稳定的响应时间，避免响应时间显著延长。

**处理时间**

系统在处理数据时，应确保在合理的时间内完成计算任务，避免用户因等待时间过长而产生不满。对于需要处理大量数据的场景，系统应采用高效的算法和数据结构，以提高数据处理速度，确保处理时间不超过用户可接受的阈值。本系统的处理时间不超过3000ms。

在处理复杂计算任务时，系统应能够合理分配计算资源，确保计算任务在规定时间内完成。

**并发处理能力**

系统应能够支持大量用户同时在线和进行交易，确保在高并发情况下系统的稳定性和性能。系统应采用分布式架构和负载均衡技术，以提高系统的并发处理能力和可扩展性。

在高并发场景下，系统应能够保持稳定的响应时间和处理速度，避免系统崩溃或性能显著下降。

**恢复时间**

在系统发生故障或异常情况下，系统应能够尽快恢复正常运行，以减少对用户的影响。系统应建立完善的故障恢复机制和备份策略，如自动重启、数据恢复等，以确保在发生故障时能够迅速恢复数据和服务。

恢复时间应控制在用户可接受的范围内，以减少对用户业务的影响，本系统的恢复时间为1小时以内。

### 3.2.4灵活性

**解耦程度**

1. **Spring框架**：Spring框架通过依赖注入（DI）和面向切面编程（AOP）提供了高度的模块化和解耦。这使得各个组件之间的依赖关系最小化，便于维护和扩展。
2. **SpringMVC**：作为Spring框架的MVC实现，SpringMVC通过控制器（Controller）将业务逻辑与页面展示分离，进一步增强了系统的解耦性。
3. **MyBatis**：MyBatis是一个半自动化的ORM框架，它允许开发者自定义SQL语句，减少了对JDBC的直接依赖，同时通过Mapper接口与业务逻辑层解耦。
4. **服务层**：在SSM架构中，通常还会有一个服务层（Service Layer），它作为业务逻辑层和数据访问层（DAO）之间的中间层，进一步解耦业务逻辑和数据访问。
5. **模块化设计**：整个系统可以被划分为不同的模块，如用户管理、商品管理、订单管理等，每个模块相对独立，便于单独开发和测试。

**接口灵活性**

1. **RESTful接口**：可以采用RESTful风格设计API接口，这样的接口易于理解和使用，同时也便于与其他系统或前端进行集成。
2. **版本控制**：在接口设计时，可以引入版本号，这样在未来需要对接口进行更新或修改时，可以保持向后兼容性，不影响现有客户端。
3. **参数验证**：通过SpringMVC的验证框架，可以对接口参数进行有效验证，确保接口的健壮性。
4. **异常处理**：通过AOP可以统一处理接口调用过程中的异常，提高接口的稳定性和用户体验。
5. **安全性**：通过Spring Security或其他安全框架，可以为接口提供认证和授权机制，确保接口的安全性。。

## 3.3 输入输出要求

### 3.3.1用户信息输入

**注册信息细化**

**用户名**：要求唯一性，且符合平台命名规则（如不得包含特殊字符、长度限制等）。

**邮箱**：需验证邮箱有效性，确保用户能接收系统通知及找回密码链接。

**密码**：复杂度要求包括大小写字母、数字及特殊字符的组合，且长度需达到一定标准，增强账户安全性。

**手机号码**：可选填写，但需通过短信验证码验证其真实性，用于二次验证、找回密码及接收重要通知。

**个人简介**：用户可自由填写，支持文本及表情符号，展现个性化信息。

**登录信息增强**

**密码找回**：除邮箱重置外，增加手机号码找回途径，提高用户找回账户的效率。

**登录保护**：引入图形验证码、短信验证码等机制，防止暴力破解及恶意登录。

个人信息更新

**头像上传**：支持多种格式（如JPEG、PNG），设定文件大小及尺寸限制，确保图片加载速度。

**收货地址管理**：允许用户添加、编辑、删除多个收货地址，支持设置默认地址，便于快速选择。

**联系方式**：提供紧急联系人信息填写选项，增强用户账户的安全性。

**安全设置升级**

**第三方账号绑定**：简化登录流程，提升用户体验，同时利用第三方平台的实名认证信息增强账户安全性。

**支付密码**：独立于登录密码设置，用于支付、提现等敏感操作，增加一层安全保护。

**双重验证**：提供指纹识别、面部识别等生物识别技术选项，提升账户安全性。

**商品信息输入**

**图片上传**：支持批量上传，自动调整图片尺寸及质量，确保商品展示效果。

**规格参数**：提供模板化输入，便于商家快速填写，同时支持自定义参数添加，满足个性化需求。

**类别与品牌**：采用树状结构分类，便于用户快速定位所需商品；品牌需经过审核，确保商品来源的可靠性。

**价格调整与促销**

**价格历史记录**：系统自动记录商品价格变动历史，便于用户及商家查看价格趋势。

**促销活动管理**：提供丰富的促销工具，如限时折扣、满减券、积分兑换等，支持商家自定义活动规则。

**库存管理智能化**：

**自动补货提醒**：根据销售数据预测库存需求，自动发送补货提醒至商家后台。

**库存预警**：当库存低于设定阈值时，自动触发预警机制，确保商品供应充足。

**订单信息输入**

**商品筛选**：支持按价格、品牌、规格等条件筛选购物车中的商品，便于用户快速找到目标商品。

**批量操作**：提供批量删除、修改数量等功能，简化用户操作流程。

**订单提交优化**

**支付方式多样化**：支持信用卡、借记卡、第三方支付平台、余额支付等多种支付方式，满足不同用户需求。

**发票信息**：提供电子发票及纸质发票选项，支持用户自定义发票抬头及内容。

**订单修改灵活性**

**用户修改**：在订单未支付前，允许用户修改收货地址、商品数量、支付方式等关键信息。

**商家修改**：提供订单状态管理界面，支持商家根据订单实际情况调整订单状态，如备货中、已发货、物流异常等。

**批量数据处理能力**

**数据导入导出**：提供CSV、Excel等格式的数据导入导出功能，便于商家进行数据分析、备份及迁移。

**数据校验**：在数据导入过程中进行严格的校验，确保数据的准确性及完整性。

### 3.3.2 输出要求

**商品展示**

**首页推荐智能化**

**个性化推荐**：基于用户浏览历史、购买记录、兴趣偏好等数据，运用机器学习算法进行个性化商品推荐。

**热门商品展示**：根据商品销量、评价等数据进行排序，展示热销商品，吸引用户关注。

**分类浏览精细化**

**多级分类**：采用多级分类结构，便于用户快速定位所需商品类别。

**筛选条件丰富化**：提供价格区间、品牌、规格、评价等多种筛选条件，满足用户多样化需求。

**商品详情页完善化**

**高清图片展示**：提供多角度、高清的商品图片，支持图片放大、缩小及旋转功能，便于用户查看商品细节。

**用户评价真实化**：展示用户真实评价，包括文字描述、星级评分、上传图片等，为其他用户提供参考。

**售后服务政策清晰化**：明确列出商品退换货政策、保修期限等售后服务信息，增强用户购买信心。

**订单状态**

**订单跟踪实时化**

**订单状态更新**：实时更新订单状态信息，如待支付、已支付、备货中、已发货、运输中、已签收等，便于用户随时查看订单进度。

**物流信息同步**：与物流公司实现数据对接，实时同步物流信息，包括发货时间、预计到达时间、当前位置等。

**支付结果**

**支付成功页友好化**

**支付信息展示**：支付成功后，展示订单号、支付金额、支付方式等关键信息，便于用户核对。

**后续操作提示**：提供返回商家页面、继续购物等后续操作提示，提升用户体验。

**错误提示明确**：若支付失败，提供明确的错误提示信息，如余额不足、网络错误、银行卡信息错误等。

**重新支付便捷**：提供重新支付按钮或链接，便于用户快速进行再次支付操作。

**退款通知及时化**

**退款进度跟踪**：对于已申请退款的订单，提供退款进度跟踪功能，便于用户随时查看退款状态。

**退款结果通知**：通过邮件、短信或APP推送等方式，及时通知用户退款结果及退款金额等信息。

**用户反馈**

**消息通知多样化**

**通知渠道丰富**：提供站内信、邮件、短信、APP推送等多种通知渠道，确保用户能及时接收到重要信息。

**通知内容个性化**：根据用户偏好及需求，提供个性化的通知内容，如订单状态更新、促销信息、系统通知等。

**用户评价激励化**

**评价奖励**：鼓励用户对购买的商品及商家服务进行评价，提供积分、优惠券等奖励措施。

**评价质量监控**：对评价内容进行审核及监控，确保评价的真实性及客观性。

**评价数据分析**：收集并分析用户评价数据，为商家提供改进建议及优化方向。

## 3.4 数据管理能力要求

### 3.4.1 MySQL数据库存储

**用户数据管理**

在MySQL数据库中，用户数据的管理是核心之一。用户基本信息表（如用户ID、用户名、邮箱、手机号码、密码哈希值、注册时间等）构成了用户身份认证的基础。此外，为了提升用户体验和忠诚度，系统还需存储用户的账户余额、积分余额、已领取的优惠券信息、历史订单记录以及多个收货地址等。这些信息不仅有助于用户管理自己的账户和购物历史，也为商家提供了宝贵的用户行为数据，用于精准营销和个性化推荐。

**商品数据管理**

商品数据是电商平台的生命线，其管理同样至关重要。MySQL中需设计商品基本信息表，包含商品ID、名称、类别、品牌、库存量、当前价格、原价（用于显示折扣信息）、价格历史记录（记录价格变动，支持价格趋势分析）、促销活动详情（如打折、满减、赠品等）、商品图片链接及描述等。这些信息的准确性和实时性直接影响到用户的购物体验和商家的销售业绩。

**订单数据管理**

订单数据的管理涉及多个环节，包括订单详情（商品列表、数量、单价、总价）、支付状态（待支付、已支付、支付失败等）、物流信息（发货时间、物流公司、运单号、物流状态）、退款记录（退款申请时间、原因、状态、金额等）以及发票信息（发票类型、抬头、税号等）。MySQL中需建立订单表、订单商品关联表、支付记录表、物流信息表、退款记录表等，以全面记录订单从创建到完成的整个过程，为用户提供清晰的订单跟踪服务，同时为商家提供订单管理和数据分析的工具。

**日志管理**

日志管理对于系统的稳定性和安全性至关重要。MySQL中应设立系统操作日志表、异常日志表和用户行为日志表。系统操作日志记录管理员对系统的操作记录，如数据备份、用户权限变更等；异常日志记录系统运行过程中出现的错误和异常，便于开发人员快速定位问题；用户行为日志记录用户的访问轨迹、点击行为等，为数据分析提供数据支持。这些日志数据不仅有助于故障排查和性能优化，也是安全审计的重要依据。

**索引优化**

为了提高数据库查询效率，针对高频查询的字段（如用户ID、商品ID、订单号）建立索引是必要的。索引可以加速数据检索速度，但也会增加写操作的开销和存储空间的占用。因此，在创建索引时需要权衡利弊，根据实际的查询需求和系统性能进行优化。

**事务处理**

在电商系统中，许多关键操作如订单支付、库存更新等都需要保证数据的原子性、一致性、隔离性和持久性（ACID特性）。MySQL提供的事务处理机制能够确保这些操作在异常情况下也能保持数据的一致性，防止数据丢失或错误。

### 3.4.2 Redis缓存

**热点数据缓存**

Redis作为高性能的内存数据库，非常适合用于缓存频繁访问的热点数据。在电商平台中，商品信息、用户信息、订单状态等数据往往被频繁查询。将这些数据缓存到Redis中，可以显著减少MySQL数据库的访问压力，提高系统的响应速度和吞吐量。同时，Redis支持多种数据类型（如字符串、列表、集合、哈希等），能够灵活满足不同的缓存需求。

**会话管理**

在分布式系统中，用户会话的管理是一个挑战。Redis提供了会话存储的解决方案，通过将会话信息存储在Redis中，可以实现会话的共享和持久化。这样，即使用户在不同的服务器上访问系统，也能保持会话的一致性。此外，Redis还支持会话过期、会话迁移等功能，进一步增强了会话管理的灵活性和可靠性。

**分布式锁**

在并发访问的场景下，如何保证数据的一致性是一个重要的问题。Redis提供了分布式锁的实现机制，通过在Redis中设置一个唯一的锁标识和过期时间，可以确保在同一时间只有一个客户端能够访问共享资源。这样，即使在高并发的情况下，也能保证数据的一致性和系统的稳定性。

**限流与熔断**

在电商系统中，API请求的限流和熔断是保护系统稳定性的重要手段。Redis可以结合令牌桶算法、漏桶算法等实现API请求的限流控制，防止系统因过载而崩溃。同时，当系统检测到某个服务出现异常时，可以自动触发熔断机制，暂时停止对该服务的调用，避免故障扩散。熔断机制可以根据服务的恢复情况进行自动或半自动的恢复操作，确保系统的整体稳定性和可用性。

## 3.5 故障处理要求

**故障响应时间**

**一级故障**：系统瘫痪，严重影响用户使用。响应时间不超过1小时。

**二级故障**：功能模块无法使用，部分用户受影响。响应时间不超过1小时。

**三级故障**：个别功能异常，小部分用户受影响。响应时间不超过4小时。

**故障处理流程**

1. **故障发现**：通过系统监控、用户反馈、测试发现故障。
2. **故障分类**：根据故障影响范围和严重程度进行分类。
3. **故障响应**：根据故障分类，启动相应级别的故障响应流程。
4. **故障定位**：分析日志、复现故障，确定故障原因。
5. **故障修复**：开发修复补丁，进行代码修改。
6. **测试验证**：对修复后的系统进行全面测试，确保故障已解决。
7. **发布上线**：将修复方案部署到生产环境。
8. **用户通知**：通过APP通知、邮件、短信等方式告知用户故障处理进度和结果。
9. **故障总结**：记录故障处理过程，总结经验教训，优化故障处理流程。

**故障处理标准**

**及时性**：故障发生后，运维人员需尽快响应并处理。

**透明性**：故障处理过程中，保持与用户的良好沟通，及时通报故障处理进展。

**有效性**：确保故障处理措施能够有效解决问题，避免二次故障。

**记录性**：详细记录故障现象、处理过程和结果，便于后续分析和回顾。

**故障预防措施**

**系统监控**：建立完善的系统监控机制，及时发现并预警潜在故障。

**定期备份**：定期对系统数据进行备份，确保数据安全。

**代码审查**：定期进行代码审查，提高代码质量，减少故障发生。

**压力测试**：定期进行压力测试，确保系统在高负载下稳定运行。

**故障处理团队**

**负责人**：项目经理崔同鑫负责故障处理的总体协调和决策。

**开发人员**：开发组负责故障的定位和修复。

**测试人员**：测试组负责验证故障是否已修复。

**运维人员**：测试组与开发组共同负责故障的及时发现和初步响应。

**客服人员**：用户组负责与用户沟通，收集用户反馈。

**故障处理工具**

**日志分析工具**：如ELK Stack、Splunk等，用于分析系统日志。

**性能监控工具**：如New Relic、Dynatrace等，用于监控系统性能。

**故障追踪系统**：如JIRA、Bugzilla等，用于跟踪和管理故障处理过程。

**故障处理文档**

**故障处理手册**：详细记录故障处理流程、常见故障及其解决方案。

**故障报告模板**：用于记录故障处理的详细信息，包括故障现象、处理过程、解决方案等。

## 3.6 其他专门要求

**校园定制**

能根据不同的学校定制不同的需求。目前仅需要支持北京信息科技大学的校园二手交易市场标准。二手交易市场标准由学校学生群体共同商定。

# 4. 运行环境规定

## 4.1 设备

**客户端**：APP支持在Android 5.0及以上版本的设备上运行。这确保了大多数现代Android设备都能安装和使用该APP。

**服务器端**：服务器端部署在Linux服务器上，优先选择云平台（如阿里云、AWS等），以确保系统的高可用性和可扩展性。

## 4.2 支持软件

**服务器软件**：使用Nginx或Tomcat作为服务器容器，以支持Java应用的部署和运行。

**数据库**：采用MySQL数据库存储商品、用户、订单等数据，确保数据的可靠性和持久性。

**缓存**：使用Redis进行缓存优化，提高系统响应速度和用户体验。

## 4.3 接口

**API接口**：APP通过RESTful API与服务器端进行交互，实现数据的请求和响应。这有助于保持前后端的分离和独立开发，提高系统的灵活性和可维护性。

**第三方接口**：除了支付接口外，APP还可能需要与其他第三方服务（如物流服务、短信服务等）进行集成，以满足用户的多样化需求。

## 4.4 控制

**权限管理**：通过权限管理功能，确保管理员、商家和用户分别拥有不同的操作权限。这有助于防止未授权访问和操作，保护系统的安全性和稳定性。

**日志记录**：对系统的重要操作进行日志记录，以便在出现问题时进行排查和追踪。同时，日志记录也有助于审计和合规性检查。

综上所述，本APP的运行环境规定涵盖了客户端、服务器端、支持软件、接口和控制等方面。在开发和部署过程中，需严格遵守这些规定，以确保系统的正常运行和用户体验的持续优化。

# 签字

|  |  |
| --- | --- |
| 需求方 | 开发方 |
|  |  |

# 附件

暂无附件