# **电商购物平台 - 软件需求规格说明书 (SRS)**

## **1. 引言 (Introduction)**

### **1.1 目的 (Purpose)**

本文档旨在详细定义“XX电商购物平台”（暂定名）的功能性及非功能性需求。它是项目设计、开发、测试和验收的主要依据，旨在确保所有项目干系人（包括开发团队、测试团队、产品经理、客户等）对系统需求有统一、明确的理解。

### **1.2 项目范围 (Scope)**

包含: 本系统是一个基于JavaWeb技术的B2C电商平台，核心功能包括用户管理、商品浏览与搜索、购物车管理、订单处理（含模拟支付）、以及后台管理（商品、订单、用户）。  
不包含: 本阶段 不包含 以下功能（未来可扩展）：复杂的营销/促销活动（如秒杀、拼团、优惠券系统）、供应商管理、多语言/多货币支持、在线客服系统、详细的财务报表、真实的第三方支付接口集成（仅做模拟或预留接口）、移动App客户端。

### **1.3 定义、首字母缩写词和缩略语 (Definitions, Acronyms, and Abbreviations)**

* **B2C:** Business-to-Consumer (商家对消费者)
* **SKU:** Stock Keeping Unit (库存量单位)
* **UI:** User Interface (用户界面)
* **UC:** Use Case (用例)
* **SRS:** Software Requirements Specification (软件需求规格说明书)
* **HTTPS:** Hypertext Transfer Protocol Secure (安全超文本传输协议)
* **CRUD:** Create, Read, Update, Delete (增删改查)
* **QPS:** Queries Per Second (每秒查询率)
* **NFR:** Non-functional Requirement (非功能性需求)
* **RTO:** Recovery Time Objective (恢复时间目标)
* **RPO:** Recovery Point Objective (恢复点目标)

### **1.4 参考文献 (References)**

* 《项目立项书》V1.0 (假设)
* 《用户界面原型设计》V0.9 (假设) - *建议关联具体的原型文档或链接*
* 《电商购物平台 - 软件系统设计 (SDD) V2.2》

### **1.5 文档概述 (Overview)**

本文档后续章节将分别描述系统的总体情况（包括产品愿景、用户、约束等）、详细的功能需求、接口需求、数据需求以及非功能性需求（质量属性）。

## **2. 总体描述 (Overall Description)**

### **2.1 产品前景 (Product Perspective)**

本系统是一个独立的Web应用程序，旨在成为公司/个人开展在线零售业务的基础平台。  
初期专注于核心购物流程，未来可考虑扩展支持移动端访问（通过响应式Web设计或独立App）、集成更丰富的营销工具、对接ERP系统等。

### **2.2 产品功能概述 (Product Functions Summary)**

* **用户端:** 提供商品展示、分类导航、商品搜索、购物车管理、安全的用户注册与登录、地址管理、订单创建与查看、(模拟)支付等核心购物功能。
* **管理端:** 提供商品管理（分类、商品信息、库存）、订单管理（查看、状态更新）、用户管理（查看、状态控制）等后台维护功能。

### **2.3 用户特征 (User Characteristics)**

* **访客 (Guest):** 未登录用户，可浏览商品、搜索商品、查看商品详情。
* **注册用户/顾客 (Customer):** 已登录用户，拥有访客所有权限，并可进行购物车操作、下单、支付、管理个人信息和地址等。
* 管理员 (Administrator): 负责后台管理，包括商品管理、订单管理、用户管理等。  
  (详细的用户画像和权限设定可参考相关补充文档)

### **2.4 约束 (Constraints)**

* **技术约束:**
  + 必须使用 Java 语言 (JDK 17+) 及 Spring Boot 框架 (3.x) 作为核心后端技术栈。
  + 数据库选用 MySQL 8.x。
  + 数据持久化框架选用 MyBatis 3.x。
  + 缓存选用 Redis 6.x+。
  + 前端技术栈采用 Vue3 (使用 TypeScript) + Vite + Axios，实现前后端分离。
  + Web服务器/反向代理使用 Nginx。
* **运行环境约束:**
  + 系统需能部署在标准的 Linux 服务器环境下。
  + 用户端需兼容主流现代浏览器（Chrome, Firefox, Safari, Edge 最新版本）。
* **法规约束:** （若涉及真实交易和用户数据）需遵守相关的电子商务法规、数据隐私保护条例（如中国的《网络安全法》、《个人信息保护法》，或GDPR等，根据目标市场确定）。
* **开发资源约束:** （若有）需在 X 个月内完成核心功能开发，投入 Y 人月资源。

### **2.5 假设和依赖 (Assumptions and Dependencies)**

* **假设:**
  + 用户具备基本的网络浏览和操作能力。
  + 用户拥有可访问互联网的设备和网络连接。
* **依赖:**
  + 系统正常运行依赖于稳定的服务器环境、数据库服务、Redis服务。
  + （若未来集成真实支付）依赖于第三方支付接口的可用性和稳定性。
  + （若使用CDN）依赖于CDN服务的可用性。

## **3. 具体需求 (Specific Requirements)**

### **3.1 功能需求 (Functional Requirements) - 详细用例描述**

*(注：以下仅展示部分核心用例的详细描述，其他用例（如添加到购物车 UC\_AddToCart, 查看购物车 UC\_ViewCart, 后台管理等）应参照此格式进行补充细化)*

#### **3.1.1 用户注册 (UC-01)**

* **用例名称:** 用户注册
* **简述:** 允许新用户创建平台账户。
* **参与者:** 访客 (Guest)
* **前置条件:** 用户未登录；用户已打开注册页面。
* **后置条件:**
  + **成功:** 创建新的用户账户记录，状态为“活动”；用户自动登录（或跳转到登录页面提示注册成功）；（可选）发送欢迎邮件。
  + **失败:** 用户账户未创建；停留在注册页面并显示具体的错误信息。
* **基本流程 (Main Success Scenario):**
  1. 用户进入注册页面，页面显示输入字段：用户名、邮箱、密码、确认密码，以及“注册”按钮和指向“用户协议”的链接。
  2. 用户输入用户名（例如：john\_doe）。
  3. 用户输入邮箱（例如：john.doe@email.com）。
  4. 用户输入密码（例如：P@sswOrd123）。
  5. 用户再次输入确认密码（与步骤4相同）。
  6. 用户勾选“我已阅读并同意用户协议”（假设有此选项）。
  7. 用户点击“注册”按钮。
  8. 系统 **校验** 输入信息：
     + 所有必填项（用户名、邮箱、密码、确认密码）均已填写。
     + 用户名格式符合规则（例如：长度5-20位，只能包含字母、数字、下划线）。
     + 用户名未被其他用户注册。
     + 邮箱格式符合标准 RFC 5322 规范。
     + 邮箱未被其他用户注册。
     + 密码符合复杂度要求（例如：长度至少8位，包含大小写字母、数字和特殊字符）。
     + 确认密码与密码完全一致。
     + （若有）“同意协议”复选框已被勾选。
  9. 系统将密码进行加盐哈希处理（例如使用 BCrypt）。
  10. 系统在 t\_user 数据库表中创建一条新记录，包含用户名、哈希后的密码、邮箱、注册时间（当前系统时间）、账户状态（“活动”）。
  11. 系统为用户创建会话（Session/JWT），标记为已登录状态（或重定向到登录页面）。
  12. 系统显示“注册成功”信息，并跳转到用户中心或首页。
  13. (可选) 系统异步发送一封欢迎邮件到用户注册邮箱。
* **扩展流程/异常流程 (Extensions/Alternative Flows):**
  + 8a. 若有必填项为空，系统在对应字段旁显示错误信息：“此项不能为空”，注册按钮点击无效或点击后停留在当前页面。
  + 8b. 若用户名格式不符，显示错误信息：“用户名格式无效，请使用5-20位字母、数字或下划线”。
  + 8c. 若用户名已存在，显示错误信息：“用户名已被注册，请尝试其他用户名”。
  + 8d. 若邮箱格式无效，显示错误信息：“请输入有效的邮箱地址”。
  + 8e. 若邮箱已被注册，显示错误信息：“该邮箱已被注册，您是否需要找回密码？”。
  + 8f. 若密码不符合复杂度要求，显示提示信息：“密码至少需要8位，且包含大小写字母、数字和特殊字符”。
  + 8g. 若确认密码与密码不一致，显示错误信息：“两次输入的密码不一致”。
  + 8h. 若“同意协议”未勾选，点击注册按钮时提示：“请先阅读并同意用户协议”。
  + 10/11. 若数据库写入失败或会话创建失败，记录错误日志，向用户显示通用错误信息：“注册过程中发生未知错误，请稍后重试或联系客服”。

#### **3.1.2 查看商品详情 (UC-12)**

* **用例名称:** 查看商品详情
* **简述:** 用户查看单个商品的详细信息。
* **参与者:** 访客, 注册用户
* **前置条件:** 用户已导航到商品详情页面（通常通过点击列表页的商品链接）。
* **后置条件:** 商品的详细信息已展示给用户。
* **基本流程:**
  1. 用户请求访问某个商品的详情页面（URL通常包含商品ID或唯一标识符，如 /products/123）。
  2. 系统根据请求中的商品ID，从数据库 t\_product 表中检索商品信息。
  3. 系统检查商品状态，如果商品状态为 ON\_SALE (上架)。
  4. 系统获取商品详细信息，包括：
     + 商品名称 (例如: “2025新款智能手机X”)
     + 商品主图和多张副图/细节图 (URL列表)
     + 商品当前销售价格 (例如: ¥4999.00)
     + (可选) 市场价/划线价 (例如: ¥5299.00)
     + 商品描述 (支持富文本格式，展示图文详情)
     + 规格参数 (例如: 品牌: ABC, 型号: X25, 颜色: 黑色, 内存: 12GB, 存储: 256GB) - 若商品有多种规格 (SKU)，则展示规格选择器 (如下拉框或按钮组)。
     + 当前库存状态 (例如: “有货”, “仅剩 N 件”, “无货”) - 具体库存数通常不直接显示给普通用户。
     + (可选) 已售数量或用户评分/评价摘要。
     + (可选) 所属分类链接。
  5. 系统将获取到的信息渲染到商品详情页面模板中（或通过API返回给前端）。
  6. 页面展示商品图片轮播/放大镜效果。
  7. 页面展示商品名称、价格、描述、规格参数等信息。
  8. 页面提供数量选择器（默认为1）。
  9. 页面提供“加入购物车”按钮和（可选的）“立即购买”按钮。
  10. (可选) 页面下方展示用户评价列表（分页加载）。
  11. (可选) 页面展示相关商品推荐。
* **扩展流程/异常流程:**
  + 2a. 若根据商品ID未找到商品，系统返回 404 Not Found 错误页面/响应。
  + 3a. 若商品状态为“下架” (OFF\_SALE) 或“删除” (DELETED)，系统返回 404 错误页面/响应或提示“该商品已下架”。
  + 4f. 若商品有多种规格 (SKU)，当用户选择不同规格组合时，页面需要动态更新对应的价格、库存状态和（可选的）SKU图片。如果某个规格组合无货，则“加入购物车”按钮应置灰或提示无货。

#### **3.1.3 提交订单 (UC-20)**

* **用例名称:** 提交订单
* **简述:** 用户选择购物车中的商品，确认信息后生成订单。
* **参与者:** 注册用户 (Customer)
* **前置条件:** 用户已登录；购物车中至少有一件商品。
* **后置条件:**
  + **成功:** 生成新的订单记录（状态为“待支付”）；扣减对应商品库存；清空购物车中已下单的商品；用户被引导至支付页面或订单成功页。
  + **失败:** 订单未生成；库存未扣减；购物车未清空；停留在订单确认页并显示错误信息。
* **基本流程:**
  1. 用户从购物车页面点击“去结算”，进入订单确认页面。
  2. **用户界面 (订单确认页) 清晰展示：**
     + **收货地址区域:** 显示默认地址，提供“切换地址”链接/按钮，允许选择已有地址或“添加新地址”（跳转或弹窗）。选中的地址高亮显示。
     + **商品清单区域:** 列表展示购物车中待结算商品的缩略图、名称、单价、购买数量、小计金额。
     + **(可选) 配送方式区域:** 默认选项（如“普通快递”），可能提供其他选项（如“加急配送”，可能产生额外费用）。
     + **(可选) 优惠券/积分区域:** 显示可用优惠券列表供选择，或输入优惠码的输入框；显示可用积分及抵扣规则。
     + **订单金额区域:** 实时计算并显示：商品总额、运费（若有）、优惠金额（若有）、积分抵扣（若有）、 **应付总额** 。
     + **“提交订单”按钮:** 位于页面底部，醒目。
  3. 用户检查所有信息，点击“提交订单”。
  4. **系统验证 (后端):**
     + **库存校验:** 逐一检查订单中每个商品的实时库存是否 >= 用户购买数量。使用数据库锁（如 SELECT … FOR UPDATE）或乐观锁机制防止并发超卖。
     + **地址校验:** 验证用户选择的收货地址ID是否有效且属于该用户。
     + **价格校验:** (可选，但推荐) 重新获取商品当前价格，与用户提交时的价格进行比对，防止用户在前端篡改价格（允许小范围浮动或按提交时价格为准，策略需确定）。
     + **优惠券/活动校验:** 验证所选优惠券/活动是否依然有效、可用且符合使用条件。
  5. **金额计算 (后端):** 基于后端数据重新精确计算订单各项金额及应付总额。
  6. **订单生成 (数据库事务内):**
     + 在 t\_order 表中插入一条新记录，包含：user\_id, 生成的唯一 order\_no (例如基于时间戳+随机数+用户ID哈希)，total\_amount (商品总额), discount\_amount, freight\_amount, payable\_amount (应付总额), order\_status (‘PENDING\_PAYMENT’), receiver\_info (收货人姓名、电话、地址详情 - 从地址簿冗余或关联), create\_time, update\_time。
     + 对于订单中的每一种商品，在 t\_order\_item 表中插入记录，包含：order\_id, product\_id, product\_name (冗余), product\_image\_url (冗余), current\_unit\_price (下单时单价), quantity, total\_price (小计)。
  7. **库存扣减 (数据库事务内):** 更新 t\_product (或 t\_sku) 表中对应商品的库存数量 (stock = stock - quantity)。**必须** 包含 WHERE stock >= quantity 条件。
  8. **购物车清理 (数据库事务内):** 从用户的购物车数据中（Redis Hash 或数据库表）移除已下单的商品项。
  9. **(事务管理):** 步骤 6, 7, 8 必须在同一个数据库事务中执行，确保原子性。若任一步骤失败，整个事务回滚，数据库状态恢复到下单前。
  10. **响应用户:** 返回订单创建成功的响应，包含订单号、应付金额，并将用户重定向到支付页面（或订单成功提示页）。
* **扩展流程/异常流程:**
  + 4a. 若库存不足，事务不提交，返回错误信息给前端：“抱歉，商品[商品名称]库存不足，请修改数量或移除后再试。”，页面停留在确认页，并标记出库存不足的商品。
  + 4b. 若地址无效，返回错误信息：“收货地址无效，请重新选择。”
  + 4c. 若价格校验失败（超出允许范围），返回错误信息：“商品价格已变动，请刷新页面后重试。”
  + 4d. 若优惠券/活动无效，返回错误信息：“您选择的优惠券/活动已失效或不适用。”
  + 9a. 若事务执行失败（数据库异常等），记录详细错误日志，返回通用错误信息：“系统繁忙，下单失败，请稍后重试。”

### **3.2 接口需求 (Interface Requirements)**

#### **3.2.1 用户界面 (User Interfaces - UI) - 关键页面描述**

*(注：以下描述为文本形式，建议配合 V0.9 的用户界面原型设计文档 [Source 54] 或实际的线框图/高保真设计稿查看)*

* **注册页面 (/register):**
  + **布局:** 页面居中表单，顶部Logo，底部版权信息。
  + **内容:** “用户注册”标题，输入框及标签：用户名 (必填, placeholder: “5-20位字母/数字/下划线”)，邮箱 (必填, placeholder: “请输入有效邮箱”)，密码 (必填, type=“password”, placeholder: “至少8位，含大小写/数字/特殊字符”)，确认密码 (必填, type=“password”, placeholder: “请再次输入密码”)。复选框：“我已阅读并同意用户协议” (必选)。按钮：“注册”。链接：“已有账号？立即登录”。实时输入校验提示显示在输入框下方。
* **商品列表页面 (/products/category/{categoryId}):**
  + **布局:** 顶部面包屑导航 (e.g., “首页 > 电子产品 > 手机”)，左侧筛选/排序区域，右侧商品网格/列表展示区域，底部页码。
  + **筛选/排序区:** 排序选项 (综合、销量、价格升序、价格降序)，(可选) 价格区间输入，(可选) 品牌筛选列表，(可选) 其他属性筛选 (如内存、屏幕尺寸)。
  + **商品展示区:** 网格布局，每个商品单元显示：商品主图、商品名称 (截断或完整显示)、销售价格、(可选) 销量/评价数、“加入购物车”快捷按钮。鼠标悬停可显示更多信息或效果。支持切换到列表视图（显示更多文字信息）。
  + **页码区:** 显示总页数/总条目数，提供上一页、下一页、具体页码跳转。
* **购物车页面 (/cart):**
  + **布局:** 表格形式展示购物车商品。顶部提示信息（如“购物车共有 N 件商品”）。底部汇总区域和结算按钮。
  + **表格列:** 复选框（用于选择结算商品）、商品图片、商品名称（链接到详情页）、单价、数量（带+/-调整按钮或输入框）、小计金额、操作（“删除”链接）。
  + **汇总区:** 显示已选商品总数、已选商品合计金额、(可选) 预估优惠金额。按钮：“去结算”（只有在勾选商品且商品有效时才可点击）、“清空购物车”、“继续购物”链接。
* **后台管理 - 商品列表 (/admin/products):**
  + **布局:** 左侧菜单导航，右侧内容区。内容区顶部为搜索/筛选表单，下方为商品数据表格，底部为分页控件。
  + **搜索/筛选:** 输入框（商品名称/ID/SKU）、下拉框（商品分类、上架状态）。按钮：“搜索”、“重置”、“+ 添加新商品”。
  + **表格列:** 商品ID、主图缩略图、商品名称、所属分类、价格、库存、销量、状态（上架/下架 - 开关按钮）、创建时间、操作（“编辑”、“删除”、（上/下架操作若状态列不是开关））。
  + **分页控件:** 同前台列表页。

#### **3.2.2 软件接口 (Software Interfaces) - RESTful API 示例 (前后端分离)**

* **标准响应格式:**  
  // 成功  
  {  
  “code”: 0, // 0 表示成功  
  “message”: “Success”,  
  “data”: { … } // 业务数据  
  }  
  // 失败  
  {  
  “code”: 4001, // 非0表示错误，需定义全局统一的错误码规范  
  “message”: “用户名已存在”, // 对错误的描述  
  “data”: null  
  }
* **获取商品列表 API:**
  + **Endpoint:** GET /api/v1/products
  + **Query Parameters:** categoryId (Long, 可选), page (Integer, 默认1), pageSize (Integer, 默认10), sortBy (String, 可选, e.g., ‘price’), sortOrder (String, 可选, ‘asc’/‘desc’), keyword (String, 可选)
  + **Success Response (200 OK):** code: 0, data: { “list”: [ { “id”: 1, “name”: “…”, … }, … ], “total”: 120 }
  + **Error Response:** 400 Bad Request (参数错误), 500 Internal Server Error
* **添加商品到购物车 API:**
  + **Endpoint:** POST /api/v1/cart/items
  + **Requires Authentication:** Yes (e.g., JWT in Authorization header)
  + **Request Body (JSON):** { “productId”: 123, “quantity”: 1 }
  + **Success Response (201 Created):** code: 0, data: { “cartItemCount”: 5 } (或其他表示成功的结构)
  + **Error Response:** 400 Bad Request (库存不足, 商品无效等), 401 Unauthorized, 404 Not Found (商品不存在), 500 Internal Server Error
* **提交订单 API:**
  + **Endpoint:** POST /api/v1/orders
  + **Requires Authentication:** Yes
  + **Request Body (JSON):** { “addressId”: 5, “paymentMethod”: “SIMULATED\_PAY”, “items”: [ { “productId”: 123, “quantity”: 1 }, … ] } (或基于购物车项ID)
  + **Success Response (201 Created):** code: 0, data: { “orderId”: “…”, “orderNo”: “…”, “payableAmount”: 5098.00 }
  + **Error Response:** 400 Bad Request (库存不足, 地址无效等), 401 Unauthorized, 500 Internal Server Error

*(注：API 设计细节应在系统设计文档 (SDD) 中进一步明确)*

### **3.3 数据需求 (Data Requirements) - 详细规则与约束**

#### **3.3.1 核心实体与关系 (E-R 描述)**

* 用户 (User) 与 地址 (Address): 一对多
* 用户 (User) 与 订单 (Order): 一对多
* 订单 (Order) 与 订单项 (OrderItem): 一对多 (Composition)
* 订单项 (OrderItem) 与 商品 (Product): 多对一 (引用下单时的商品快照)
* 商品 (Product) 与 分类 (Category): 多对一 (或多对多，取决于业务)
* 用户 (User) 与 购物车 (ShoppingCart): 一对一 (逻辑或物理)
* 购物车项 (CartItem) 与 商品 (Product): 多对一

#### **3.3.2 关键数据表结构 (示例)**

* **t\_user (用户信息表):**
  + id (BIGINT, PK, Auto Increment)
  + username (VARCHAR(50), Unique, Not Null, 校验: 5-20位字母/数字/下划线)
  + password\_hash (VARCHAR(60), Not Null, 存储BCrypt哈希值)
  + email (VARCHAR(100), Unique, Not Null, 校验: RFC 5322 format)
  + phone\_number (VARCHAR(20), Unique, Nullable, 校验: 手机号格式)
  + status (TINYINT, Not Null, Default: 1, **含义: 1:Active, 0:Inactive/Disabled**)
  + created\_at (DATETIME, Not Null, Default: CURRENT\_TIMESTAMP)
  + updated\_at (DATETIME, Not Null, Default: CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP)
* **t\_product (商品信息表):**
  + id (BIGINT, PK, AI)
  + name (VARCHAR(255), Not Null)
  + description (TEXT, Nullable)
  + category\_id (BIGINT, Not Null, FK references t\_category(id))
  + price (DECIMAL(10, 2), Not Null, 校验: > 0)
  + stock\_quantity (INT, Not Null, Default: 0, 校验: >= 0)
  + image\_url (VARCHAR(512), Nullable) // 主图
  + status (TINYINT, Not Null, Default: 1, **含义: 1:On Sale (上架), 0:Off Sale (下架), 2:Deleted (删除)**)
  + created\_at, updated\_at (DATETIME)
* **t\_order (订单信息表):**
  + id (BIGINT, PK, AI)
  + order\_no (VARCHAR(32), Unique, Not Null, 生成规则: e.g., YYYYMMDDHHMMSS + 随机数 + userId hash)
  + user\_id (BIGINT, Not Null, FK references t\_user(id))
  + total\_amount (DECIMAL(12, 2), Not Null) // 商品总额
  + payable\_amount (DECIMAL(12, 2), Not Null) // 应付总额
  + status (VARCHAR(20), Not Null, Default: ‘PENDING\_PAYMENT’, **枚举值: ‘PENDING\_PAYMENT’ (待支付), ‘PAID’ (已支付), ‘SHIPPED’ (已发货), ‘DELIVERED’ (已送达), ‘COMPLETED’ (已完成), ‘CANCELLED’ (已取消)**)
  + receiver\_name (VARCHAR(50), Not Null) // 收货人信息冗余
  + receiver\_phone (VARCHAR(20), Not Null)
  + receiver\_address (VARCHAR(500), Not Null) // 省市区+详细地址冗余
  + created\_at, updated\_at, paid\_at (Nullable), shipped\_at (Nullable), completed\_at (Nullable) (DATETIME)
  + **状态转换逻辑:** (详见 3.5 分析模型 - 状态图)
* **t\_order\_item (订单项表):**
  + id (BIGINT, PK, AI)
  + order\_id (BIGINT, Not Null, FK references t\_order(id))
  + product\_id (BIGINT, Not Null, FK references t\_product(id)) // 关联当时商品
  + product\_name (VARCHAR(255), Not Null) // 商品名称快照
  + product\_image\_url (VARCHAR(512), Nullable) // 商品图片快照
  + unit\_price (DECIMAL(10, 2), Not Null) // 下单时单价快照
  + quantity (INT, Not Null, 校验: > 0)
  + total\_price (DECIMAL(12, 2), Not Null) // quantity \* unit\_price

*(注：其他表如 t\_category, t\_address, t\_cart\_item 等也应类似定义)*

### **3.4 非功能性需求 / 质量属性 (Non-functional Requirements - NFR)**

#### **3.4.1 性能 (Performance)**

* **并发用户数:**
  + **目标:** 系统设计应能支撑 **500** 并发用户同时在线浏览，**100** 并发用户进行下单操作。
  + **验收标准:** 通过压力测试工具（如 JMeter）模拟上述并发场景，系统在 1 小时内保持稳定运行，无严重错误（HTTP 5xx 错误率 < 0.1%），且满足下述响应时间要求。
* **数据量:** 设计需考虑支持至少 **100,000** SKU 和 **1,000,000** 注册用户，年订单量 **500,000**。
* **响应时间:**
  + **目标:**
    - 95% 的页面请求（非API）加载时间 < **3** 秒。
    - 99% 的核心 API 请求（如获取商品详情、提交订单校验、获取购物车）响应时间 < **500ms** (服务端处理时间，不含网络传输)。
    - 数据库核心查询（如商品详情、订单查询）平均执行时间 < **50ms**。
  + **验收标准:** 在性能测试期间，使用监控工具（如 APM 系统）或日志分析测量上述指标，确认达标。
* **资源优化:** 前端静态资源（JS, CSS, Images）应进行压缩合并，并利用浏览器缓存和CDN（若预算允许）。后端应实现数据库连接池，并对热点数据（如商品分类、首页推荐）考虑使用缓存（如Redis）。

#### **3.4.2 安全性 (Security)**

* **认证与会话:** 用户登录使用 HTTPS POST 请求；密码传输全程加密；登录成功后服务端生成安全的 JWT Token；Token 具有合理的过期时间（例如，访问 Token 30分钟，刷新 Token 7天）；提供“记住我”功能时，使用安全的长期令牌机制。
* **密码策略:** 强制用户密码符合复杂度要求（见UC-01）；密码存储使用 BCrypt (cost>=10) 加盐哈希；提供安全的密码重置流程（如通过有时效的邮件链接）。
* **输入验证:** 所有外部输入（URL参数、表单字段、请求体、HTTP头）必须在服务端进行严格的类型、格式、长度、范围校验和净化，以防御 XSS, SQL注入等。使用 Spring Validation 或类似机制进行 DTO 校验。使用 MyBatis 的 # {} 参数化查询防御 SQL 注入。
* **访问控制:** 严格按照 RBAC 模型实现权限控制。使用 Spring Security 的注解（如 @PreAuthorize）或配置来保护 Controller 方法或 URL 路径。确保用户只能访问自己的订单、地址等私有数据。后台管理接口必须有严格的管理员角色校验。
* **依赖安全:** 定期扫描项目依赖库（如使用 OWASP Dependency-Check），及时更新存在已知漏洞的版本。
* **日志审计:** 对关键操作（登录、下单、支付、后台管理操作）记录详细的审计日志，包含用户ID、操作时间、IP地址、操作内容。
* **部署安全:** 生产环境服务器应进行安全加固，关闭不必要的端口和服务，配置防火墙规则。数据库密码等敏感配置信息应加密存储或通过安全方式管理（见 SDD 配置管理部分）。
* **HTTPS:** 全站强制使用 HTTPS。

#### **3.4.3 可用性 & 可靠性 (Availability & Reliability)**

* **目标:** 核心功能（浏览、加购、下单、支付模拟）达到 **99.9%** 可用性（月度计算）。
* **度量方式:** 通过监控系统（如 Prometheus + Grafana）监控核心 API 的健康状态和成功率。可用性 = (总请求数 - 失败请求数) / 总请求数 \* 100%。
* **数据库高可用:** 数据库实现主从复制（Master-Slave Replication）以支持读写分离和故障切换。
* **监控告警:** 关键服务（应用服务器、数据库、Redis）应有监控告警，能在故障发生时（如CPU/内存超限、服务宕机、核心接口错误率骤增）及时通知运维人员。
* **备份恢复:**
  + **目标:** RTO (恢复时间目标) < **4** 小时, RPO (恢复点目标) < **1** 小时。
  + **策略:** 数据库执行每日凌晨全量备份，每小时事务日志备份。备份文件异地存储。需制定并演练数据库恢复计划。
* **容错:** 对外部服务调用（如未来支付接口、短信服务）应有超时、重试和熔断机制（如使用 Resilience4j）。

#### **3.4.4 可维护性 (Maintainability)**

* **编码规范:** 严格遵守《阿里巴巴Java开发手册》或其他选定规范。代码提交前需通过静态代码检查工具（如 Checkstyle, PMD, SpotBugs）扫描。
* **模块化:** 代码按功能模块组织（e.g., com.ecommerce.user, com.ecommerce.product），遵循高内聚低耦合原则。定义清晰的 Service 层接口。
* **日志标准:** 使用 SLF4j + Logback/Log4j2。日志级别分明（DEBUG, INFO, WARN, ERROR）。日志输出包含时间戳、线程名、日志级别、类名方法名、TraceID（用于链路追踪）、具体消息。对请求入口、异常堆栈、关键业务操作进行日志记录。
* **配置管理:** 应用配置（数据库连接、第三方服务密钥等）应外部化（如 application.properties/yml），不同环境（开发、测试、生产）使用不同配置文件。敏感配置加密处理或使用配置中心。
* **版本控制:** 使用 Git 进行版本控制，遵循清晰的分支管理策略（如 Gitflow）。每次代码提交需关联到具体的需求或Bug修复任务。
* **文档:** 提供必要的架构设计文档、API文档（Swagger/OpenAPI自动生成）、部署文档。代码中复杂逻辑需有注释说明。
* **测试:** 单元测试覆盖率目标 > 70%；核心业务流程需要有集成测试覆盖。

### **3.5 分析模型 (Analysis Models) - 描述**

*(注：以下为对分析模型的文字描述，对应的 PlantUML 图见 SDD 附录)*

* **用例图描述:** 图中包含三个主要参与者（访客、注册用户、管理员）。围绕系统边界，绘制出核心用例（如浏览商品、用户注册、添加到购物车、提交订单、管理商品等）。用直线连接参与者和他们能执行的用例。用例之间用 <<include>> (如提交订单必须先登录) 和 <<extend>> 关系表示依赖和扩展。
* **活动图描述 (示例: 完整下单流程):** 图展示了从查看购物车到订单提交成功的完整流程。包括用户操作（查看购物车、点击结算、确认信息、提交订单）、系统判断（购物车是否为空、用户是否登录、库存/地址/价格校验、事务处理结果）和系统活动（显示页面、执行校验、生成订单、扣减库存、清空购物车、跳转支付）。清晰地表达了主流程和异常分支。
* **状态图描述 (示例: 订单状态):** 图定义了订单生命周期中的各个状态（待支付、已支付、已发货、已送达、已完成、已取消）以及触发状态转换的事件或动作（支付成功、管理员发货、用户确认收货、支付超时等）。明确了订单状态流转的规则。
* **领域模型/类图描述:** 图展示了系统核心的业务实体（User, Address, Product, Category, Order, OrderItem, ShoppingCart, CartItem 等）及其属性和它们之间的关系（一对一、一对多、多对多）。它是数据库设计和代码实现中领域对象的基础。