五、详细设计说明书

[1．引言 1](#_Toc506973482)

[1.1编写目的 1](#_Toc506973483)

[1.2项目背景 1](#_Toc506973484)

[1.3定义 2](#_Toc506973485)

[1.4参考资料 2](#_Toc506973486)

[2．总体设计 2](#_Toc506973487)

[2.1需求概述 2](#_Toc506973488)

[2.2软件结构 2](#_Toc506973489)

[3．程序描述 2](#_Toc506973490)

[3.1功能 3](#_Toc506973491)

[3.2性能 3](#_Toc506973492)

[3.3输入项目 3](#_Toc506973493)

[3.4输出项目 3](#_Toc506973494)

[3.5算法 3](#_Toc506973495)

[3.6程序逻辑 3](#_Toc506973496)

[3.7接口 3](#_Toc506973497)

[3.8存储分配 3](#_Toc506973498)

[3.9限制条件 3](#_Toc506973499)

[3.10测试要点 3](#_Toc506973500)

# 1．引言

## 1.1编写目的

本设计说明书的目的是明确系统开发的技术要求与设计规范，确保开发人员在实施过程中有明确的指导，提供各模块的功能与实现方法。同时，它也为项目经理、测试人员、客户和其他利益相关者提供详细的系统设计文档，以保证所有的需求得以落实。设计说明书将对系统的各项功能、模块接口、数据结构等进行详细描述，为最终的软件交付提供支持。

## 1.2项目背景

本项目是为某电子商务平台开发的一个软件商城交易平台的管理系统，旨在为商家、用户和平台管理员提供一个高效、安全的购物与管理系统。系统将帮助商家高效管理商品、订单和销售数据，提供给用户便捷的购物体验，同时为平台管理员提供强大的管理功能。本项目的开发由韩守坤负责，目标是为在线零售行业提供一个完整的电商解决方案，满足用户、商家和管理员的多方面需求。

## 1.3定义

· **商品管理**：指商家在平台上对商品进行上架、下架、编辑、删除等操作的功能。

· **订单管理**：商家、用户和管理员对订单的创建、支付、查询、退款等操作的集合。

· **管理员**：平台负责管理商家和用户的账户信息、商品审核、数据分析等任务的系统角色。

· **API接口**：用于不同模块或系统之间进行数据交换和请求响应的程序接口。

· **RESTful API**：一种基于HTTP协议的API设计风格，通过标准的HTTP方法（如GET、POST、DELETE、PUT）来操作资源。

· **用户**：系统中进行注册、登录、浏览商品、下单购物等操作的个人或实体。

· **状态码**：HTTP协议中用于表示请求处理结果的数字编码，常见的有200（成功）、400（请求错误）、500（服务器错误）等。

## 1.4参考资料

1. 可行性研究报告；
2. 项目开发计划；
3. 需求规格说明书；
4. 概要设计说明书；
5. 测试计划（初稿）；
6. 用户操作手册（初稿）；

# 2．总体设计

## 2.1需求概述

本系统包括三个核心模块：**商家模块**、**用户模块**、**平台管理员模块**。每个模块提供一系列功能，以实现电商平台的完整操作。商家可管理商品、处理订单和进行数据分析；用户可浏览商品、购物和支付；管理员则负责对商家、商品和用户进行管理，并提供数据分析服务。

## 2.2软件结构

本系统采用前后端分离的架构，前端负责用户与系统的交互，后端处理具体的业务逻辑和数据操作。系统采用RESTful API接口规范，通过HTTP协议进行前后端通信。

##### 1. ****前端****：

前端使用 **Vue.js** 框架开发，采用单页面应用（SPA）设计，提升用户体验。Vue.js 提供了响应式的数据绑定和组件化开发，简化了开发过程，且具有较高的性能和灵活性。Vue.js 中，组件化的开发方式使得项目结构更加清晰，能够有效复用组件，提升开发效率。使用 **Vue Router** 进行页面路由管理，采用 **Vuex** 进行全局状态管理，确保各个组件之间的数据流动和一致性。前端通过 **Axios** 库与后端进行 RESTful API 的数据交互，支持异步请求和响应。

##### 2. ****后端****：

后端基于Node.js与Express框架开发，处理各类API请求，管理数据库数据并执行业务逻辑。后端负责处理商家、用户、管理员的请求，支持多用户并发访问，确保高效响应。

##### 3. ****数据库****：

使用MySQL作为关系型数据库管理系统，用于存储商品、用户、订单、支付等核心数据。所有数据通过后端API进行访问与修改。

##### 4. ****系统架构图****：

**用户前端界面**

**（vue.js）**

**后端业务逻辑层**

**（Excepress）**

**API接口层**

**（Node.js）**

**数据层**

**（MySQL）**

#### 2.3 模块设计与API接口

##### 1. ****商家模块****

**商品管理**：

* + **POST /product/add**：添加商品
  + **PUT /product/updateById/{id}**：更新商品信息
  + **DELETE /product/deleteById/{id}**：根据ID删除商品
  + **GET /product/getById/{id}**：根据ID查询商品
  + **GET /product**：查询所有商品

**订单管理**：

* + **GET /order**：查询所有订单
  + **PUT /order/status/{id}**：更新订单状态

**数据统计与分析**：

* + **GET /statistics/sales**：获取销售数据统计
  + **GET /statistics/feedback**：获取用户反馈统计

##### 2. ****用户模块****

**用户注册与登录**：

* + **POST /user/register**：用户注册
  + **POST /user/login**：用户登录

**购物车管理**：

* + **POST /cart/add**：添加商品到购物车
  + **DELETE /cart/remove**：从购物车删除商品
  + **GET /cart**：查看购物车

**订单处理**：

* + **POST /order/create**：创建订单
  + **GET /order/{id}**：查询订单详情

##### 3. ****平台管理员模块****

**管理员管理**：

* + **POST /admin/login**：管理员登录
  + **POST /admin/register**：管理员注册

**商家与商品管理**：

* + **GET /merchant/list**：查看所有商家
  + **GET /product/list**：查看所有商品
  + **PUT /product/approve**：审核商品

**数据统计**：

* + **GET /admin/statistics**：平台总体销售统计
  + **GET /admin/user\_behavior**：用户行为分析

#### 数据库设计

数据库主要包含以下几张表：

* **用户表 (users)**：存储用户信息，包括用户名、密码、邮箱、手机号等。
* **商品表 (products)**：存储商品信息，包括商品名称、价格、库存、描述、图片等。
* **订单表 (orders)**：存储订单信息，包括订单ID、商品ID、用户ID、支付状态、订单状态等。
* **商家表 (merchants)**：存储商家信息，包括商家名称、商家账户信息、商品列表等。
* **支付表 (payments)**：存储支付信息，包括支付状态、支付时间、支付金额等。

#### 2.5 系统接口与数据交互

系统采用RESTful API进行各模块间的交互，所有请求使用标准HTTP方法（GET、POST、PUT、DELETE），并使用JSON格式传递数据。

* **POST /user/login**：用户登录接口，接受用户名和密码，返回JWT Token。
* **GET /product**：获取商品列表，支持分页、筛选和排序功能。

# 3．程序描述

## 3.1功能

1. **用户管理模块**：负责用户注册、登录、密码管理等功能。
2. **商品管理模块**：负责商品的添加、删除、修改、查询等功能。
3. **订单管理模块**：负责订单的生成、支付、查询、取消等功能。
4. **购物车模块**：负责商品的添加、删除、结算等功能。
5. **支付模块**：支持多种支付方式（如信用卡、支付宝、微信支付等）。
6. **评价模块**：允许用户对购买的商品进行评价，查看其他用户评价。

## 3.2性能

1. 系统应能够支持同时1000个用户在线使用。
2. 商品查询响应时间应小于500毫秒。
3. 用户注册过程应在3秒内完成。
4. 支付过程应在5秒内完成，确保高可用性和可靠性。

## 3.3输入项目

1. 用户信息（用户名、密码、邮箱等）
2. 商品信息（商品名称、描述、价格、库存等）
3. 订单信息（用户ID、商品ID、数量、支付信息等）
4. 评价信息（用户ID、商品ID、评分、评论内容等）

## 3.4输出项目

1. 用户注册成功/失败信息
2. 商品列表及详情信息
3. 订单确认信息及状态
4. 支付成功/失败信息
5. 用户评价列表

## 3.5算法

1. 用户验证算法（如：哈希加盐技术存储密码）
2. 商品检索算法（基于索引的快速查询）
3. 订单生成算法（生成订单号及处理并发）
4. 支付流程算法（状态机管理支付状态）

## 3.6程序逻辑

**3.6.1 用户管理模块**

**1. 用户注册逻辑**

输入： 用户名、密码、邮箱

步骤： 1. 检查用户名是否已存在

是 → 返回“用户名已存在” -

否 → 继续下一步

2. 验证邮箱格式是否正确

否 → 返回“邮箱格式不正确”

3. 将密码进行哈希处理（加盐）后存储

4. 将用户信息保存到数据库

5. 返回“注册成功”

输出： 注册状态

1. **用户登录逻辑**

输入： 用户名、密码

步骤： 1. 获取用户信息

查询数据库：输入用户名

2. 验证用户名是否存在 -

否 → 返回“用户不存在”

3. 验证密码

使用存储的哈希值进行对比

否 → 返回“密码错误”

4. 记录用户登录状态并返回“登录成功”

输出： 登录状态、用户信息

* + 1. **商品管理模块**

1. **商品添加逻辑**

输入： 商品信息（名称、描述、价格、库存等）

步骤： 1. 验证商品信息完整性

如：名称、价格和库存不能为空

2. 验证价格是否为非负数

否 → 返回“价格无效”

3. 将商品信息保存到数据库

4. 返回“商品添加成功”

输出： 添加状态

1. **商品查询逻辑**

输入： 查询条件（类别、价格范围等）

步骤： 1. 接受查询条件

2. 根据条件查询数据库

3. 返回查询结果列表

输出： 商品列表

**3.6.3 订单管理模块**

**1、订单生成逻辑**

输入： 用户ID、商品ID、数量

步骤： 1. 验证商品库存是否足够

否 → 返回“库存不足”

2. 生成订单号（UUID或自增ID）

3. 创建订单信息

4. 更新商品库存

5. 保存订单信息到数据库

6. 返回“订单创建成功”及订单信息

输出： 订单状态、订单详细信息

**2、 订单查询逻辑**

输入： 用户ID、订单状态（可选）

步骤： 1. 查询用户的所有订单记录

2. 根据状态过滤订单（如提供）

3. 返回订单列表

输出： 订单列表

**3.6.4购物车模块**

**1、 商品添加到购物车逻辑**

输入： 用户ID、商品ID、数量

步骤： 1. 验证商品是否存在

否 → 返回“商品不存在”

2. 验证库存是否足够

否 → 返回“库存不足”

3. 检查购物车中是否已有该商品

是 → 更新数量

否 → 添加新商品到购物车

4. 返回“添加成功”

输出： 购物车状态

**2、购物车结算逻辑**

输入： 用户ID

步骤： 1. 查询购物车中的所有商品

2. 检查每个商品的库存

如有商品库存不足 → 返回“库存不足”并提示

3. 生成订单并调用订单生成逻辑

4. 清空购物车

5. 返回“结算成功”或订单信息

输出： 订单状态、订单信息

**3.6.5 支付模块**

**1、 支付逻辑**

输入： 订单ID、支付信息

步骤： 1. 获取订单信息

2. 验证订单状态是否允许支付

只有“待支付”状态的订单可支付

3. 调用第三方支付接口

4. 根据支付结果更新订单状态

成功 → 返回“支付成功”及最新订单信息

失败 → 返回“支付失败”

输出： 支付状态、订单信息

**3.6.6 评价模块**

**1、 添加评价逻辑**

输入： 用户ID、商品ID、评分、评论内容

步骤： 1. 验证用户是否已购买该商品

否 → 返回“您未购买该商品，无法评价”

2. 验证评价内容的有效性

按字数或禁用词检查

3. 将评价信息保存到数据库

4. 更新商品评分信息

5. 返回“评价成功”

输出： 评价状态

**2、 查询评价逻辑**

输入： 商品ID

步骤： 1. 查询该商品的所有评价信息

2. 返回评价列表

输出： 评价列表

## 3.7接口

**3.7.1商品管理**

**1、POST 添加商品**

|  |
| --- |
| 请求URL: /product/add  请求参数:  name: string  description: string  price: number  image: string  category\_id: integer  url: string  状态码: 200 OK |

**2、DELETE 根据ID删除商品**

|  |
| --- |
| 请求URL: /product/deleteById/{id}  路径参数:  id: integer  状态码: 200 OK |

**3、GET 根据ID查询商品**

|  |
| --- |
| 请求URL: /product/getById/{id}  路径参数:  id: integer  状态码: 200 OK |

**4、GET 查询所有商品**

|  |
| --- |
| 请求URL: /product  状态码: 200 OK |

**5、PUT 更新商品信息**

|  |
| --- |
| 请求URL: /product/updateById/{id}  路径参数:  id: string  请求体:  name: string  description: string  price: number  image: string  category\_id: integer  url: string  状态码: 200 OK |

**3.7.2用户管理**

**1、POST 管理员登录**

|  |
| --- |
| 请求URL: /admin/login  请求参数:  username: string  password: string  状态码: 200 OK |

**2、POST 管理员注册**

|  |
| --- |
| 请求URL: /admin/register  请求参数:  username: string  password: string  rePassword: string  状态码: 200 OK |

**3、POST 用户登录**

|  |
| --- |
| 请求URL: /user/login  请求参数:  username: string  password: string  状态码: 200 OK |

**4、POST 用户注册**

|  |
| --- |
| 请求URL: /user/register  请求参数:  username: string  password: string  rePassword: string  状态码: 200 OK |

**5、GET 查询所有用户**

|  |
| --- |
| 请求URL: /user  状态码: 200 OK  GET 根据id查询用户  请求URL: /user/{id}  路径参数:  id: string  状态码: 200 OK |

**3.7.3订单管理**

**1、POST 生成订单**

|  |
| --- |
| 请求URL: /order/add  请求参数:  order\_id: integer  product\_id: string  user\_id: string  payment\_time: string  order\_status: string  状态码: 200 OK |

## 3.8存储分配

1. 用户表：存储用户基本信息
2. 商品表：存储商品信息
3. 订单表：存储订单信息
4. 购物车表：临时存储用户购物车信息
5. 评价表：存储用户评价内容

## 3.9限制条件

1. 用户名长度限制：1-20个字符
2. 密码复杂度要求：至少8个字符，包含字母和数字
3. 商品价格：非负数
4. 支付方式需符合第三方支付接口规定

## 3.10测试要点

1. 验证用户注册、登录功能的完整性和安全性
2. 测试商品的增加、删除、修改是否有效
3. 测试订单生成和处理过程，包括支付状态的变化
4. 测试购物车功能的完整性和用户体验
5. 测试评价功能的有效性与真实性
6. 进行负载测试，确保系统在高并发情况下的稳定性