

## 本科实验报告

|        |                  |
|--------|------------------|
| 课程名称   | B/S 体系软件设计       |
| 项目名称   | 商品比价网站开发         |
| 姓    名 | 朱子乔              |
| 学    号 | 3220103468       |
| 学    院 | 计算机科学与技术学院       |
| 专    业 | 软件工程 2202        |
| 指导老师   | 胡晓军              |
| 实验日期   | 2024 年 11 月 14 日 |

# 目录

|  |           |
|--|-----------|
| <b>1 引言</b>  | <b>3</b>  |
| 1.1 背景介绍   | 3         |
| 1.2 实验目的与意义  | 3         |
| 1.3 系统功能概述   | 3         |
| <b>2 系统需求分析</b>  | <b>4</b>  |
| 2.1 用户需求分析   | 4         |
| 2.2 功能需求分析   | 4         |
| 2.3 非功能需求分析  | 4         |
| 2.4 条件与限制  | 6         |
| <b>3 系统设计</b>  | <b>6</b>  |
| 3.1 基本设计流程和概念  | 6         |
| 3.2 前端技术框架   | 7         |
| 3.2.1 Vue.js   | 7         |
| 3.2.2 Element Plus                                     | 7         |
| 3.2.3 ECharts  | 8         |
| 3.2.4 npm 包管理工具  | 8         |
| 3.3 后端技术框架   | 8         |
| 3.3.1 Python   | 8         |
| 3.3.2 Flask  | 9         |
| 3.3.3 Selenium   | 9         |
| 3.3.4 MySQL  | 9         |
| 3.3.5 Docker 部署  | 10        |
| 3.3.6 部署   | 10        |
| <b>4 数据库设计和 ER 图</b>                                   | <b>10</b> |
| 4.1 数据库概述  | 10        |
| 4.2 数据表设计  | 11        |
| 4.2.1 用户表 ('user')                                     | 11        |
| 4.2.2 商品表 ('product')                                  | 11        |
| 4.2.3 商品历史价格表 ('price_history')                        | 11        |
| 4.2.4 商品分类表 ('category')                               | 12        |
| 4.2.5 降价提醒表 ('price_alert')                            | 12        |
| 4.2.6 触发器 ('update_product_latest_price_after_insert') | 12        |
| 4.3 ER 图   | 13        |
| <b>5 接口设计</b>  | <b>14</b> |
| 5.1 用户模块接口   | 14        |
| 5.1.1 用户登录接口   | 14        |

|          |                            |           |
|----------|----------------------------|-----------|
| 5.1.2    | 发送验证码接口 . . . . .          | 14        |
| 5.1.3    | 用户注册接口 . . . . .           | 15        |
| 5.2      | 邮箱提醒个人设置模块接口 . . . . .     | 15        |
| 5.2.1    | 设置价格提醒接口 . . . . .         | 15        |
| 5.2.2    | 获取用户的价格提醒记录接口 . . . . .    | 15        |
| 5.2.3    | 删除价格提醒记录接口 . . . . .       | 16        |
| 5.3      | 商品模块接口 . . . . .           | 16        |
| 5.3.1    | 获取最新商品接口 . . . . .         | 16        |
| 5.3.2    | 获取商品历史价格接口 . . . . .       | 17        |
| 5.4      | 验证码模块接口 . . . . .          | 17        |
| 5.4.1    | 发送验证码接口 . . . . .          | 17        |
| 5.4.2    | 验证验证码接口 . . . . .          | 18        |
| <b>6</b> | <b>UI 设计</b>               | <b>18</b> |
| 6.1      | 首页 . . . . .               | 18        |
| 6.2      | 用户登录界面 . . . . .           | 19        |
| 6.3      | 用户注册界面 . . . . .           | 19        |
| 6.4      | 商品搜索界面 . . . . .           | 20        |
| 6.5      | 商品详情页面 . . . . .           | 20        |
| 6.6      | 价格走势图页面 . . . . .          | 20        |
| 6.7      | 降价提醒设置页面 . . . . .         | 21        |
| 6.8      | 商品分类页面 . . . . .           | 22        |
| 6.9      | 手机适配页面 . . . . .           | 24        |
| <b>7</b> | <b>系统出错设计</b>              | <b>24</b> |
| 7.1      | 错误分类 . . . . .             | 24        |
| 7.2      | 错误处理机制 . . . . .           | 25        |
| 7.2.1    | 全局错误处理 . . . . .           | 25        |
| 7.2.2    | 常见错误处理 . . . . .           | 25        |
| 7.3      | 错误响应设计 . . . . .           | 26        |
| 7.4      | 错误日志记录 . . . . .           | 26        |
| 7.5      | 示例 . . . . .               | 26        |
| 7.5.1    | 用户登录失败 (401 错误) . . . . .  | 26        |
| 7.5.2    | 商品查询失败 (404 错误) . . . . .  | 27        |
| 7.5.3    | 数据库连接失败 (500 错误) . . . . . | 27        |
| 7.6      | 总结 . . . . .               | 27        |
| <b>8</b> | <b>附录</b>                  | <b>27</b> |
| 8.1      | 时间安排 . . . . .             | 27        |
| 8.2      | 备注 . . . . .               | 28        |

# 1 引言

## 1.1 背景介绍

随着电子商务的飞速发展，消费者在购买商品时常常需要比对不同平台上的价格，以获得最优惠的购买选择。然而，不同平台之间的价格差异可能会影响消费者的购买决策。为了帮助消费者快速比较商品价格并作出决策，本实验通过开发一个商品价格比较网站，提供实时商品价格查询与比较服务，提升用户的购物效率。

## 1.2 实验目的与意义

本项目是 2024-2025 秋冬学期《B/S 体系软件设计》的课程项目，旨在设计并实现一个商品价格比较网站，通过综合运用前后端技术、数据库管理和数据爬取技术，为用户提供多平台商品价格查询与比较的服务。实验的主要目的如下：

- 实现商品名称的搜索功能，能够在主流电商平台（如淘宝、京东）查询商品价格。
- 提供商品价格历史走势图，帮助用户更好地分析商品价格变化趋势。
- 支持用户注册、登录及设置降价提醒功能，确保用户能够根据自己的需求及时获得价格变化信息。
- 为了提高系统的响应速度与用户体验，本实验将使用前端分离的架构，实现跨平台支持（如 PC 端、移动端）。

本实验将帮助加深对 Web 开发、数据库管理以及 API 接口设计的理解，由一人独立完成。

## 1.3 系统功能概述

本商品价格比较网站的基本功能包括：

1. **用户注册与登录**：实现用户注册与登录功能，确保用户可以通过唯一的用户名和 Email 注册，并对注册信息进行格式验证。
2. **商品查询与价格比较**：通过商品名称在主流电商平台（如淘宝、京东）查询该商品的实时价格，并展示多个平台的价格信息，供用户进行比较。
3. **商品信息管理**：将商品信息和商品价格保存在数据库中，支持商品名称、多级品类、规格、条码和图片等信息的管理。
4. **历史价格走势图**：展示商品的历史价格变化，帮助用户了解商品的价格趋势。
5. **降价提醒**：用户可设置降价提醒，当商品价格发生变化时，通过邮件或 App 推送等方式提醒用户。
6. **响应式设计**：网站支持在移动端和 PC 端的友好展示，确保用户在各种设备上都能获得良好的体验。
7. **商品扫描功能（增强功能）**：用户可以使用相机扫描商品条码或拍摄商品图片，自动查询商品信息和价格。

## 2 系统需求分析

### 2.1 用户需求分析

本系统的目标用户为日常网购的消费者。用户主要需求包括：

- **商品价格对比**: 用户希望能够快速、准确地对比不同电商平台的商品价格，从而做出最优购买决策。
- **实时价格更新**: 用户期望系统能够实时获取商品在多个平台上的价格，并及时更新。
- **历史价格查询**: 用户希望能够查看商品的历史价格趋势，以便做出更具决策力的购买选择。
- **降价提醒功能**: 当商品价格发生变化时，用户希望系统能够推送价格提醒，以便抓住降价时机。
- **多平台支持**: 用户希望系统能够支持至少两个电商平台（如淘宝、京东）的价格查询和比较。
- **简易的注册与登录流程**: 用户期望系统提供简单易用的注册和登录功能，以便快速使用系统。
- **支持移动端使用**: 用户希望系统能在手机、平板等移动设备上流畅运行，并且提供友好的 UI 界面。

### 2.2 功能需求分析

本系统的功能需求包括以下几个方面：

- **用户注册与登录**: 系统需要支持用户注册和登录功能，用户在注册时需要输入有效的用户名、密码和邮箱，并进行格式验证。同时，系统需保证用户名和邮箱的唯一性，避免重复注册。
- **商品查询与价格比较**: 用户能够通过输入商品名称，系统从多个主流电商平台（如淘宝、京东等）查询商品的实时价格，并展示查询结果，支持价格的横向对比。
- **历史价格展示**: 系统需要提供商品历史价格走势图，展示商品在不同时间段的价格变化，帮助用户分析价格趋势。
- **降价提醒功能**: 用户可以选择指定商品并设置降价提醒功能，当商品价格低于某个阈值时，系统会通过邮件或 App 推送等方式提醒用户。
- **商品管理与数据库**: 系统需要建立一个商品数据库，保存商品的详细信息，如商品名称、品类、规格、条码、图片等，并与平台的商品价格实时同步。
- **支持多个平台查询**: 系统必须能够从至少两个电商平台（如淘宝、京东）获取商品价格信息，进行价格比较。
- **移动端适配**: 系统需要支持手机端浏览器以及 App 的使用，适配不同分辨率的设备，保证良好的用户体验。

### 2.3 非功能需求分析

#### • 性能需求

- **查询响应时间**: 系统在获取商品价格时，能够在 3 秒内完成对主流电商平台（淘宝、京东等）的查询，并显示价格结果。

- **高并发支持**: 系统应能支持至少 1000 个用户同时进行商品价格查询，且系统响应不应超过 5 秒。
- **价格查询优化**: 为提高查询效率，系统应支持异步查询多个电商平台价格，并在平台响应后合并结果展示给用户。
- **数据库查询性能**: 系统在查询商品信息时，数据库查询响应时间不应超过 2 秒，且能处理大规模商品数据的存取。

- **安全性**

- **用户密码加密**: 用户密码必须经过加密存储，使用如 bcrypt 等安全的哈希算法。不得以明文存储密码，防止密码泄露。
- **防 SQL 注入**: 系统需要防止 SQL 注入攻击，使用参数化查询或 ORM 框架来避免直接在 SQL 语句中拼接用户输入的内容。
- **身份验证**: 用户登录后，系统需要为每个用户分配唯一的会话标识符，并通过 JWT (Json Web Token) 或 Session 验证用户身份。
- **数据传输安全**: 所有的敏感数据（如用户密码、支付信息等）必须通过 HTTPS 协议加密传输，防止数据在传输过程中被窃取。
- **XSS 防护**: 需要防止跨站脚本攻击 (XSS)，对用户输入的数据进行转义，确保用户输入不会执行恶意脚本。
- **敏感信息保护**: 系统存储的敏感信息（如邮箱、手机号等）应加密存储，并定期审查系统的安全性。

- **数据管理**

- **定期数据更新**: 系统应定期从电商平台获取商品价格数据，保证价格信息的及时更新。建议每天或每小时进行一次更新。
- **备份与恢复**: 系统应支持数据库定期备份，并能在数据丢失时进行恢复。备份周期可以设定为每日一次，保存最新一周的数据备份。
- **数据清理**: 对于长时间未查询的商品信息或价格数据，系统应进行定期清理，避免数据库存储过多无用数据，影响查询性能。

- **可扩展性**

- **电商平台扩展**: 系统应能支持更多电商平台的接入，例如拼多多、苏宁等，且系统架构需要支持模块化扩展。
- **商品数据扩展**: 随着商品数量的增加，系统应能支持更大规模的数据存储和查询。数据库应使用分库分表或其他优化方案来处理大规模数据。
- **负载均衡**: 随着用户量和查询请求的增加，系统应能够进行负载均衡，保证高并发情况下仍能保证系统的高可用性。

- **用户体验**

- **界面友好**: 系统前端界面需要简洁、直观，并能快速响应用户操作。价格对比、图表展示等功能应清晰明了，便于用户理解。

- **移动端适配**: 系统应支持在手机端和桌面端都能流畅运行，确保不同设备的用户均能获得一致的体验。
- **页面加载时间**: 系统的页面加载时间应尽量减少，理想情况下，页面加载时间应小于 2 秒，避免用户等待过久。
- **操作反馈**: 用户每次操作后应得到即时反馈，如商品价格查询后，页面应快速展示价格结果，若无结果则给出提示。

## 2.4 条件与限制

- **平台限制**:

- 当前版本仅支持淘宝、京东平台的商品查询功能，未来可考虑扩展到拼多多、苏宁等平台。扩展时需考虑不同平台的 API 或抓取方式。
- 某些电商平台（如淘宝、京东）对爬虫和 API 访问有限制，系统可能需要解决登录验证、验证码等问题，避免被平台封锁。

- **技术限制**:

- 后端开发使用 Node.js，数据库使用 MySQL。前端使用 Vue.js 与 CSS，可能会遇到部分旧版浏览器或设备的兼容问题。
- 在进行商品价格查询时，部分平台需要使用 API 密钥或登录验证才能访问数据，这要求开发者具备该平台的开发者权限或通过网页抓取方式获取数据。

- **代码规范**:

- 本系统开发采用统一的代码风格，前端 Vue 代码采用 ESLint 进行代码检查，后端 Node.js 代码使用 Prettier 和 ESLint 进行代码格式化。
- 采用 Git 进行版本控制，每次开发应先创建分支，开发完成后提交 PR 并进行代码审查，确保代码质量。
- 所有接口（API）必须加上清晰的注释，文档应保持更新，接口返回结果应符合统一的格式。

- **工具版本限制**:

- **Node.js 版本**: 开发环境使用 Node.js v16.0.0 版本，确保与项目依赖兼容。
- **MySQL 版本**: 数据库使用 MySQL 8.0 版本，支持更高效的查询和更强的事务支持。
- **Vue.js 版本**: 前端开发使用 Vue.js v3.0.0 及以上版本，确保支持最新的功能和优化。
- **Webpack 版本**: 前端打包工具使用 Webpack v5.0 及以上版本，保证高效打包和模块化。

## 3 系统设计

### 3.1 基本设计流程和概念

在本项目中，我们的设计目标是实现一个商品价格比较网站，该系统应具备用户管理、商品价格查询、价格走势图显示以及降价提醒等核心功能。设计流程大致如下：

- **需求分析**: 首先，根据项目要求对功能进行需求分析，明确系统需要实现的基本功能，如用户注册与登录、商品价格查询、价格图表显示等。
- **技术选型**: 根据需求和技术可行性，选择合适的前端与后端技术栈。前端使用 Vue.js 与 Element Plus 组件库，后端使用 Node.js 和 Express 框架，数据库使用 MySQL 进行数据存储。
- **系统架构设计**: 设计系统的前后端架构，明确数据流向，如何进行 API 请求，如何展示数据等。前端与后端通过 RESTful API 进行通信，确保模块化与可扩展性。
- **实现与测试**: 进行系统的开发与测试，包括前端界面的实现、后端接口开发、数据库设计、功能模块测试与调试等。
- **部署与维护**: 将完成的系统部署到生产环境，确保系统稳定运行，并进行日常维护和性能监控。

系统架构是基于现代 Web 开发常用的前后端分离模式，前端与后端通过 RESTful API 进行数据交互，前端负责用户界面的展示和交互，后端负责业务逻辑处理和数据存储。

## 3.2 前端技术框架

项目前端使用了 **Vue.js + Element Plus** 组件库 + **ECharts** 图表组件 + **npm** 包管理工具作为前端开发的技术栈。

### 3.2.1 Vue.js

**Vue.js** 是一款用于构建用户界面的 JavaScript 框架。它基于标准 HTML、CSS 和 JavaScript 构建，并提供了一套声明式的、组件化的编程模型，帮助开发者高效地开发用户界面。无论是简单还是复杂的界面，Vue.js 都可以胜任。Vue.js 的两个核心功能包括：

- **声明式渲染**: Vue.js 基于标准 HTML 拓展了一套模板语法，使得我们可以声明式地描述最终输出的 HTML 和 JavaScript 状态之间的关系。开发者只需定义数据与界面的关系，Vue.js 会根据数据的变化自动更新视图。
- **响应性**: Vue.js 会自动跟踪 JavaScript 状态并在其发生变化时响应式地更新 DOM，从而确保用户界面的即时更新和高效渲染。

Vue.js 以其简洁的语法和强大的功能在前端开发中得到了广泛应用，尤其在构建单页应用和组件化开发方面具有显著优势。

### 3.2.2 Element Plus

**Element Plus** 是一个基于 Vue 3 的高质量 UI 组件库，提供了丰富的组件和扩展功能，例如表格、表单、按钮、导航、通知等，能够帮助开发者快速构建高质量的 Web 应用。Element Plus 的设计理念是：

- 提供开箱即用的 UI 组件和扩展功能，帮助开发者快速构建应用程序。
- 提供详细的文档和教程，帮助开发者更好地掌握和使用 Element Plus。
- 高度可定制，支持开发者根据需求调整样式和功能。

Element Plus 是 Vue 3 的官方推荐 UI 组件库，兼容性良好，并且得到了社区的广泛支持，是构建现代化 Web 应用的优秀工具。

### 3.2.3 ECharts

**ECharts** 是一款基于 JavaScript 的数据可视化图表库，提供直观、动态、可交互、可个性化定制的数据可视化图表。ECharts 最初由百度团队开源，并于 2018 年捐赠给 Apache 基金会，成为 ASF 孵化级项目。ECharts 提供了多种类型的图表，例如折线图、柱状图、散点图、饼图、K 线图、热力图、关系图、漏斗图等，适用于各种数据展示需求。

- **直观且生动的图表展示：** ECharts 提供了丰富的图表类型和自定义样式，能够帮助开发者生动地展示数据。
- **交互性：** ECharts 支持图表的交互操作，例如点击、缩放、提示框等，使得用户可以更加便捷地查看数据。
- **个性化定制：** ECharts 提供强大的配置项，开发者可以根据具体需求定制图表的样式、功能等。

ECharts 在数据可视化领域具有广泛的应用，尤其适用于价格走势图、业务数据分析等场景。

### 3.2.4 npm 包管理工具

**npm** (Node Package Manager) 是 Node.js 的包管理工具，用于管理 JavaScript 项目的依赖包。npm 不仅支持包的安装、更新和卸载，还可以管理项目中的各种工具与库。

- **依赖管理：** npm 可以帮助开发者快速安装所需的第三方库和工具，解决了库版本依赖问题。
- **包的发布和共享：** 开发者可以通过 npm 将自己的包发布到公共仓库供他人使用，也可以从公共仓库中下载别人编写的包。
- **命令行工具：** npm 提供了功能强大的命令行工具，帮助开发者高效管理项目依赖、执行构建、部署等任务。

npm 在 JavaScript 项目中扮演着核心角色，它简化了库管理和自动化任务执行。

## 3.3 后端技术框架

项目后端使用了 **Python + Flask + Flask-JWT-Extended + Flask-Bcrypt + BeautifulSoup4** 或 **Scrapy + MySQL** 作为技术栈，同时使用 Docker 进行容器化部署。

### 3.3.1 Python

**Python** 是一门易于学习和使用的高级编程语言，广泛应用于 Web 开发、数据分析、机器学习等领域。Python 拥有丰富的库和框架，使其成为构建后端 API 服务和自动化脚本的理想选择。在本项目中，Python 用于构建后端 API 服务和处理爬虫数据采集任务。

- **简洁的语法：** Python 提供了简洁且可读性强的语法，开发者可以快速上手并高效开发应用。
- **丰富的库支持：** Python 拥有大量的开源库，可以满足 Web 开发、数据处理、爬虫抓取等多种需求。
- **跨平台：** Python 是跨平台的，可以在 Windows、Linux 和 macOS 等多种操作系统上运行，方便部署。

Python 的简洁性和丰富的生态使其在 Web 后端开发中广泛应用，特别适用于快速开发 API 服务和爬虫任务。

### 3.3.2 Flask

**Flask** 是一个轻量级的 Python Web 框架，适合用来构建小型到中型的 Web 应用和 API 服务。Flask 提供了灵活且简洁的 API，可以让开发者更加专注于业务逻辑的实现。Flask 的优势包括：

- **轻量级**: Flask 非常轻便，不捆绑过多功能，开发者可以根据需要灵活选择中间件和扩展。
- **快速开发**: Flask 提供了丰富的工具和插件，开发者能够快速构建出功能完备的 Web 服务。
- **可扩展性**: Flask 通过插件机制，支持扩展各种功能，如认证、数据库连接、表单处理等。

Flask 在本项目中作为核心 Web 框架，用于构建商品查询 API、用户认证、价格比较等服务。

### 3.3.3 Selenium

**Selenium** 是一个强大的 Python 库，通常用于自动化浏览器操作。与 BeautifulSoup4 和 Scrapy 不同，Selenium 通过模拟用户操作来控制浏览器，适合抓取动态加载的网页内容。尤其在需要获取懒加载图片或模拟用户登录才能访问数据时，Selenium 展现出了其独特的优势。

- **动态网页抓取**: Selenium 可以自动化浏览器操作，模拟点击、滚动等用户行为，获取页面中通过 JavaScript 动态加载的数据。对于电商平台中许多通过懒加载显示的商品图片，Selenium 能够有效抓取。
- **模拟用户登录**: 很多电商平台的商品信息和价格数据在未登录状态下无法完全展示，Selenium 允许模拟用户登录，获取登录后的页面内容，确保抓取到完整的商品信息。
- **页面交互**: Selenium 能模拟用户在页面上的交互，如选择下拉菜单、点击按钮等，帮助实现更复杂的数据抓取场景。

Selenium 是处理需要模拟用户行为、获取动态加载内容等复杂情况时的理想选择，因此在本项目中，特别适合用于抓取需要登录后的电商平台商品价格、图片以及其他相关信息。

### 3.3.4 MySQL

**MySQL** 是关系型数据库管理系统，在本项目中用于存储用户数据、商品信息、历史价格数据等。MySQL 的特点包括：

- **高性能**: MySQL 在处理大量数据时能够保持较好的性能，适合用于大规模 Web 应用。
- **易于使用**: MySQL 提供了简单易懂的 SQL 语言，使得开发者能够快速上手并高效操作数据库。
- **可靠性**: MySQL 提供了事务支持、数据完整性约束等功能，确保数据的可靠性和一致性。

MySQL 用于存储系统中的用户数据、商品信息、价格历史记录等关键数据。

### 3.3.5 Docker 部署

为确保项目的可移植性和简化部署过程，整个项目将容器化并使用 **Docker** 部署。Docker 可以将应用及其所有依赖项打包成一个容器，便于在任何支持 Docker 的平台上运行。

- **一致的开发和生产环境：** 使用 Docker 容器，可以确保开发环境、测试环境和生产环境的一致性，避免环境差异导致的问题。
- **便捷的部署和扩展：** Docker 容器支持快速的应用部署和横向扩展，方便项目在不同的服务器或云平台上进行部署。
- **自动化管理：** 通过 Docker Compose 等工具，可以实现容器的自动化管理、编排和扩展。

通过 Docker 部署，项目可以在任何支持 Docker 的平台上快速上线，并且能够通过容器化管理轻松扩展和维护系统。

### 3.3.6 部署

在部署过程中，系统的前端应用、数据库和爬虫将分别容器化并通过 Docker 部署。部署步骤如下：

- **构建 Docker 镜像：** 根据应用的 Dockerfile 文件构建前端服务、数据库和爬虫的镜像。
- **Docker Compose 编排：** 使用 Docker Compose 来定义和管理多容器应用，包括前端、后端、数据库和爬虫等服务。
- **容器化数据库：** MySQL 数据库将运行在 Docker 容器中，并与其他容器（如前端和爬虫）进行网络连接。
- **运行和监控：** 容器部署后，可以通过 Docker 和容器编排工具（如 Kubernetes）来监控系统的运行状态，确保高可用性和稳定性。

这种容器化部署方式不仅提高了应用的可移植性，还简化了开发、测试和生产环境的管理，使得项目能够更加高效地进行维护和扩展。

## 4 数据库设计和 ER 图

### 4.1 数据库概述

为了支持商品价格比较、用户注册和登录、价格历史记录、降价提醒等功能，数据库将存储以下关键数据：

- **用户信息：** 用于存储注册用户的基本信息。
- **商品信息：** 存储商品的详细信息，包括名称、规格、图片、分类等。
- **价格历史记录：** 记录商品在不同平台上的历史价格，用于展示价格趋势图。
- **降价提醒：** 存储用户设置的降价提醒信息。

本系统使用 MySQL 数据库来存储这些数据。数据表之间通过外键进行关联，确保数据一致性与完整性。

## 4.2 数据表设计

根据项目需求，设计以下主要数据表：

### 4.2.1 用户表 ('user')

存储用户信息，用于注册、登录和验证。

| 字段名称            | 数据类型         | 约束条件                       | 备注                 |
|-----------------|--------------|----------------------------|--------------------|
| <b>id</b>       | INT          | PRIMARY KEY AUTO_INCREMENT | 用户的唯一标识符           |
| <b>username</b> | VARCHAR(255) | NOT NULL                   | 用户名，不能为空           |
| <b>email</b>    | VARCHAR(255) | NOT NULL                   | 用户的电子邮件地址，<br>不能为空 |
| <b>password</b> | VARCHAR(255) | NOT NULL                   | 用户密码               |

表 1: 用户表设计

### 4.2.2 商品表 ('product')

存储商品的详细信息。

| 字段名称                | 数据类型          | 约束条件                       | 备注                    |
|---------------------|---------------|----------------------------|-----------------------|
| <b>id</b>           | INT           | PRIMARY KEY AUTO_INCREMENT | 商品的唯一标识符              |
| <b>name</b>         | VARCHAR(255)  | NOT NULL                   | 商品名称                  |
| <b>category</b>     | VARCHAR(255)  | NOT NULL                   | 商品品类                  |
| <b>platform</b>     | VARCHAR(100)  | NOT NULL                   | 商品平台                  |
| <b>shop_url</b>     | VARCHAR(255)  | NOT NULL                   | 商店 URL                |
| <b>img</b>          | BLOB          | NULL                       | 商品图片 (BLOB 类型存储二进制数据) |
| <b>latest_price</b> | DECIMAL(10,2) | NOT NULL                   | 商品最新价格                |
| <b>created_at</b>   | TIMESTAMP     | DEFAULT CURRENT_TIMESTAMP  | 商品创建时间，默认为当前时间        |

表 2: 商品表设计

### 4.2.3 商品历史价格表 ('price\_history')

记录商品在不同平台上的历史价格。

| 字段名称              | 数据类型          | 约束条件   | 备注                      |
|-------------------|---------------|--|-------------------------|
| <b>id</b>         | INT           | PRIMARY KEY AUTO_INCREMENT                           |                         |
| <b>product_id</b> | INT           | FOREIGN KEY REFERENCES product(id) ON DELETE CASCADE | 价格记录的唯一标识符商品 ID, 关联到商品表 |
| <b>price</b>      | DECIMAL(10,2) | NOT NULL   | 历史价格                    |
| <b>created_at</b> | TIMESTAMP     | DEFAULT CURRENT_TIMESTAMP                            | 记录时间, 默认为当前时间           |

表 3: 商品价格历史表设计

#### 4.2.4 商品分类表 ('category')

存储商品的分类信息, 用于多级分类。

| 字段名称             | 数据类型         | 约束条件                       | 备注             |
|------------------|--------------|----------------------------|----------------|
| <b>id</b>        | INT          | PRIMARY KEY AUTO_INCREMENT | 分类的唯一标识符       |
| <b>name</b>      | VARCHAR(255) | NOT NULL                   | 分类名称           |
| <b>parent_id</b> | INT          | DEFAULT NULL               | 父分类 ID, 支持多级分类 |

表 4: 商品分类表设计

#### 4.2.5 降价提醒表 ('price\_alert')

存储用户设置的商品降价提醒信息。

| 字段名称                   | 数据类型 | 约束条件   | 备注            |
|------------------------|------|--|---------------|
| <b>id</b>              | INT  | PRIMARY KEY AUTO_INCREMENT                           | 降价提醒的唯一标识符    |
| <b>user_id</b>         | INT  | FOREIGN KEY REFERENCES user(id)<br>ON DELETE CASCADE | 用户 ID, 关联到用户表 |
| <b>product_id</b>      | INT  | FOREIGN KEY REFERENCES product(id) ON DELETE CASCADE | 商品 ID, 关联到商品表 |
| <b>price_threshold</b> | INT  | NOT NULL   | 用户设置的价格阈值     |

表 5: 降价提醒表设计

#### 4.2.6 触发器 ('update\_product\_latest\_price\_after\_insert')

当商品价格历史表 ('price\_history') 插入新记录时, 自动更新商品表 ('product') 的最新价格字段。

DELIMITER

```
CREATE TRIGGER update_product_latest_price_after_insert AFTER INSERT ON price_history FOR EACH ROW
```

- 获取该商品的最新价格 SELECT price INTO latest\_price FROM price\_history WHERE product\_id = NEW.product\_id ORDER BY created\_at DESC LIMIT 1;
- 更新商品表中的最新价格 UPDATE product SET latest\_price = latest\_price WHERE id = NEW.product\_id;

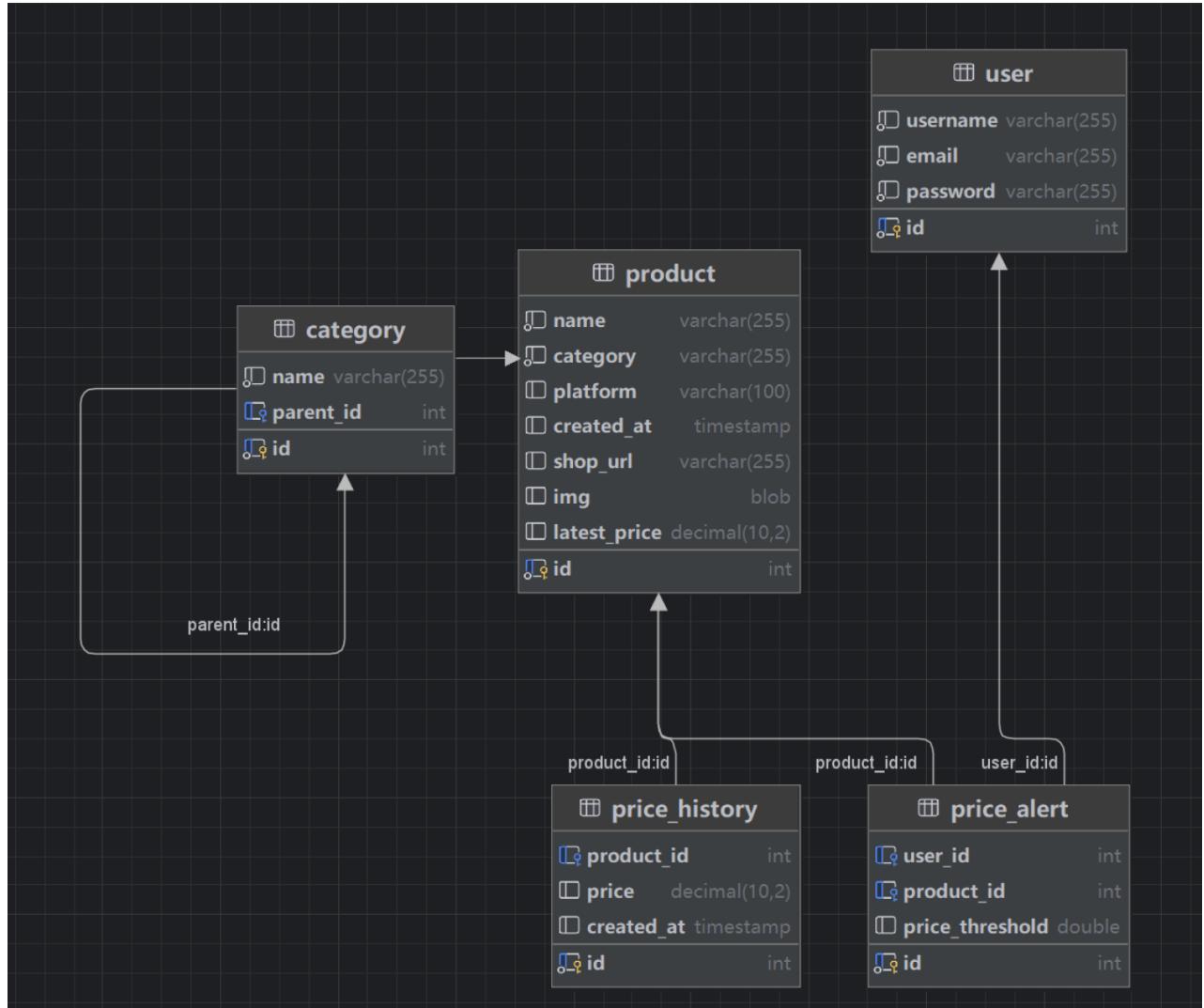
---

END

*DELIMITER;*

### 4.3 ER 图

以下是简化版的 ER 图示意：



该 ER 图展示了各数据表之间的关系：

- \*\*Users\*\* 表与 \*\*Price Alerts\*\* 表之间有一对多关系（一个用户可以有多个降价提醒）。
- \*\*Products\*\* 表与 \*\*Price History\*\* 表之间有一对多关系（一个商品有多个价格历史记录）。
- \*\*Products\*\* 表与 \*\*Categories\*\* 表之间通过外键建立关联，商品归属于一个分类，分类支持多级。
- \*\*Price Alerts\*\* 表与 \*\*Products\*\* 表之间有多对一关系（每个提醒与一个商品相关）。

该设计可以有效地支持商品价格查询、价格历史展示以及降价提醒功能。

## 5 接口设计

本节设计了项目的各个模块接口，包括用户模块、商品价格爬取模块、商品查询模块、降价提醒模块等。每个模块下面列出了对应的接口，以便实现项目所需的功能。当前 api 为实际工作使用的 api。

### 5.1 用户模块接口

用户模块包括用户的注册、登录、验证码发送、价格提醒设置及管理的接口设计。

#### 5.1.1 用户登录接口

- **接口路径:** ‘/api/users/login’
- **请求方法:** POST
- **请求参数:**
  - ‘email’: 用户邮箱，字符串。
  - ‘password’: 用户密码，字符串。
- **返回参数:**
  - ‘status’: 操作状态，成功为 ‘200’，失败为 ‘400’。
  - ‘message’: 登录提示信息，成功时返回用户名，失败时返回错误原因（如” 用户未找到” 或” 密码错误”）。
- **功能描述:** 该接口用于用户登录，通过邮箱和密码进行身份验证，成功则返回用户名，失败返回相应的错误信息。

#### 5.1.2 发送验证码接口

- **接口路径:** ‘/api/users/send-captcha’
- **请求方法:** POST
- **请求参数:**
  - ‘email’: 用户邮箱，字符串。
  - ‘username’: 用户名，字符串。
- **返回参数:**
  - ‘status’: 操作状态，成功为 ‘200’，失败为 ‘400’。
  - ‘message’: 返回验证码发送的提示信息，成功时为” 验证码已发送到邮箱”，失败时为错误信息。
- **功能描述:** 该接口用于发送注册验证码到用户邮箱，验证邮箱是否已注册，如已注册则返回提示。

### 5.1.3 用户注册接口

- **接口路径:** '/api/users/register'
- **请求方法:** POST
- **请求参数:**
  - 'email': 用户邮箱, 字符串。
  - 'username': 用户名, 字符串。
  - 'password': 用户密码, 字符串。
  - 'captcha': 用户输入的验证码, 字符串。
- **返回参数:**
  - 'status': 操作状态, 成功为 '200', 失败为 '400'。
  - 'message': 注册提示信息, 成功时为"注册成功", 失败时为错误信息 (如"验证码无效或已过期")。
- **功能描述:** 该接口用于用户注册, 首先校验验证码的有效性, 如果验证码正确, 保存用户信息到数据库。

## 5.2 邮箱提醒个人设置模块接口

### 5.2.1 设置价格提醒接口

- **接口路径:** '/api/users/set\_alarm'
- **请求方法:** POST
- **请求参数:**
  - 'username': 用户名, 字符串。
  - 'productId': 商品 ID, 整数。
- **返回参数:**
  - 'status': 操作状态, 成功为 '200', 失败为 '500'。
  - 'message': 设置价格提醒的提示信息, 成功时为"价格提醒设置成功", 失败时为错误信息。
- **功能描述:** 该接口用于设置用户的商品价格提醒。通过用户名和商品 ID 为用户设置价格提醒。

### 5.2.2 获取用户的价格提醒记录接口

- **接口路径:** '/api/users/personal\_alert'
- **请求方法:** POST
- **请求参数:**

- ‘username’: 用户名，字符串。

- **返回参数:**

- ‘status’: 操作状态，成功为 ‘200’，失败为 ‘404’。
- ‘alerts’: 用户的价格提醒记录，列表。每个记录包含商品信息及其提醒条件。
- ‘message’: 返回获取提醒记录的提示信息，成功时返回价格提醒数据，失败时返回“没有找到提醒记录”。

- **功能描述:** 该接口用于获取用户的价格提醒记录，返回用户所有的价格提醒，并包括相应商品的详细信息。

### 5.2.3 删除价格提醒记录接口

- **接口路径:** ‘/api/users/delete\_alert’
- **请求方法:** DELETE
- **请求参数:**

- ‘id’: 提醒记录 ID，整数。

- **返回参数:**

- ‘status’: 操作状态，成功为 ‘200’，失败为 ‘400’。
- ‘message’: 删除价格提醒的提示信息，成功时为“提醒已删除”，失败时为“删除失败，未找到记录”。

- **功能描述:** 该接口用于删除用户的价格提醒记录，通过提醒 ID 删除相应的提醒记录。

## 5.3 商品模块接口

商品模块包括商品数据获取和商品历史价格查询的接口设计。

### 5.3.1 获取最新商品接口

- **接口路径:** ‘/api/products/shop\_display’
- **请求方法:** POST
- **请求参数:**
  - ‘page’: 页码，整数。
  - ‘limit’: 每页数量，整数。
  - ‘searchQuery’: 搜索关键词，字符串，支持商品名或其他相关描述。
  - ‘priceMin’: 最低价格，浮动值，默认为 ‘null’。
  - ‘priceMax’: 最高价格，浮动值，默认为 ‘null’。
  - ‘platform’: 商品所在的平台，如‘淘宝’、‘京东’，字符串。

- ‘priceSort’: 价格排序，‘asc’ 表示升序，‘desc’ 表示降序。
- ‘isLocalSearchEnabled’: 是否启用本地搜索，布尔值。若为 ‘false’，则启用爬虫搜索。
- **返回参数:**
  - ‘products’: 商品列表，包含商品 ID、名称、价格等信息。
  - ‘totalCount’: 总商品数，整数。
  - ‘searchCount’: 符合搜索条件的商品数量，整数。
- **功能描述:** 该接口根据传入的分页、筛选条件以及是否启用爬虫进行商品查询。若启用爬虫，系统会调用外部爬虫服务获取商品数据。

### 5.3.2 获取商品历史价格接口

- **接口路径:** ‘/api/products/history\_price’
- **请求方法:** POST
- **请求参数:**
  - ‘productId’: 商品 ID，整数，表示需要查询历史价格的商品。
- **返回参数:**
  - ‘historyPrices’: 商品的历史价格记录，列表。每个记录包含历史价格和时间戳。
- **功能描述:** 该接口根据传入的商品 ID 查询商品在不同时间点的历史价格，并返回所有历史记录。

## 5.4 验证码模块接口

用户模块包括用户的注册、登录和相关功能的接口设计。

### 5.4.1 发送验证码接口

- **接口路径:** ‘/api/email/send’
- **请求方法:** POST
- **请求参数:**
  - ‘email’: 用户邮箱，字符串，格式必须符合邮件格式。
- **返回参数:**
  - ‘status’: 操作状态，成功为 ‘200’，失败为 ‘400’。
  - ‘message’: 返回的提示信息。
- **功能描述:** 该接口接收用户的邮箱地址，并发送一个验证码到指定邮箱，验证码用于后续的验证操作。

### 5.4.2 验证验证码接口

- 接口路径: ‘/api/email/validate’
- 请求方法: POST
- 请求参数:
  - ‘email’: 用户邮箱, 字符串, 格式必须符合邮件格式。
  - ‘captcha’: 验证码, 字符串, 长度为 6 位字符。
- 返回参数:
  - ‘status’: 操作状态, 成功为 ‘200’, 失败为 ‘400’。
  - ‘message’: 返回的提示信息。
- 功能描述: 该接口接收用户的邮箱和验证码, 验证验证码是否正确, 若验证成功, 返回相应状态信息; 若验证失败, 返回错误信息。

## 6 UI 设计

### 6.1 首页

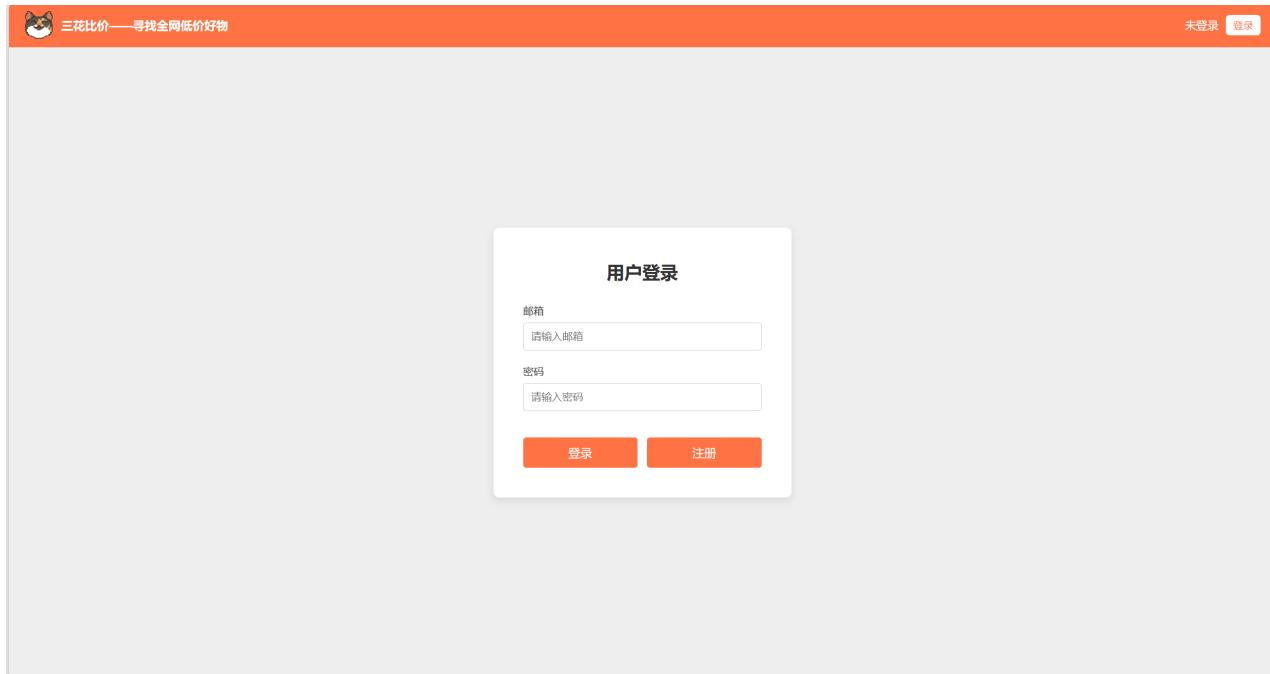
该界面展示了网页的基础界面, 可在此处选择登陆、检索功能, 并带有历史记录展示, 便于用户浏览。

The screenshot shows the homepage of a price comparison website. At the top, there is a navigation bar with a logo, the text '三花比价——寻找全网低价好物', and user status indicators ('未登录' and '登录'). Below the navigation bar is a search bar containing the text '猫粮'. To the left of the search bar is a sidebar with several filters: '价格筛选' (Price Filter) with dropdowns for '20' and '50'; '价格排序' (Price Sort) with a dropdown set to '降序'; '平台选择' (Platform Selection) with a dropdown set to '苏宁易购'; and '本地搜索模式' (Local Search Mode) and '点击筛选' (Click to Filter) buttons.

The main content area displays a search result for '猫粮' (Cat Food). It shows a summary: '商品库存 共有 542 个 本次找到 15 个'. Below this, there are 15 product cards arranged in three rows of five. Each card includes a thumbnail image, the product name, a brief description, the price, and purchase options ('苏宁易购', '猫粮', '立即购买', '评价', '收藏'). A decorative banner for '麦富迪三文鱼鲜肉夹心全价全期猫粮2kg' is also visible on the right side of the page.

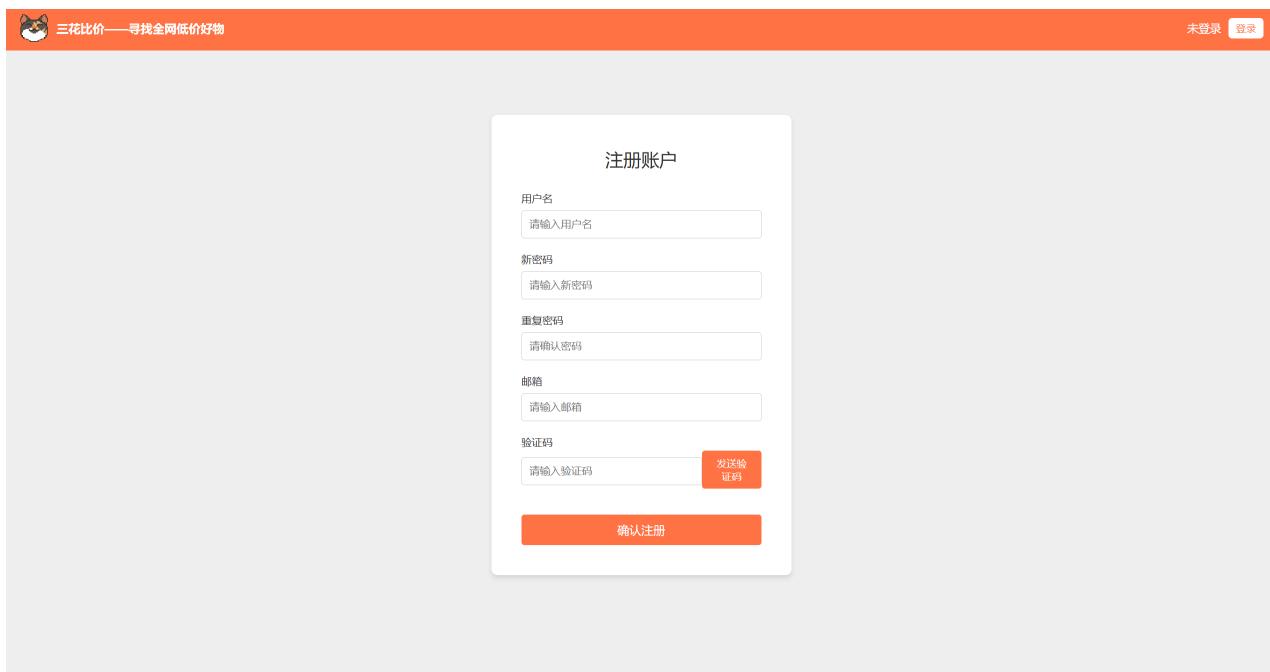
## 6.2 用户登录界面

该界面展示了用户登录功能，用户可以输入用户名和密码进行登录操作。



## 6.3 用户注册界面

该界面用于新用户注册，用户需要填写用户名、邮箱和密码，并进行验证。



## 6.4 商品搜索界面

该界面提供商品搜索功能，用户可以输入商品名称并查看相关商品列表。此界面每个项目展示来自不同电商平台的同一商品最低价格，帮助用户快速选择最优价格。

The screenshot shows a search interface for '猫粮' (Cat Food). On the left, there's a sidebar with '价格筛选' (Price Filter) buttons for '最低价格' (Lowest Price) and '最高价格' (Highest Price), and a '升序' (Ascending Order) button. Below that are '平台选择' (Platform Selection) dropdowns and '本地搜索模式' (Local Search Mode) and '点击筛选' (Click to Filter) buttons. The main search area has a search bar with '猫粮' and a '搜索' (Search) button. It displays a message '商品库存 共有 542 个 本次找到 85个'. Below this are several product cards, each with an image, name, price, and a '苏宁易购' (Suning) link. One card for '凯悦思猫粮冻干' is highlighted with a red border.

## 6.5 商品详情页面

点击跳转到对应最低价的原平台商品的详细信息，包括商品名称、图片、价格、规格、分类等。

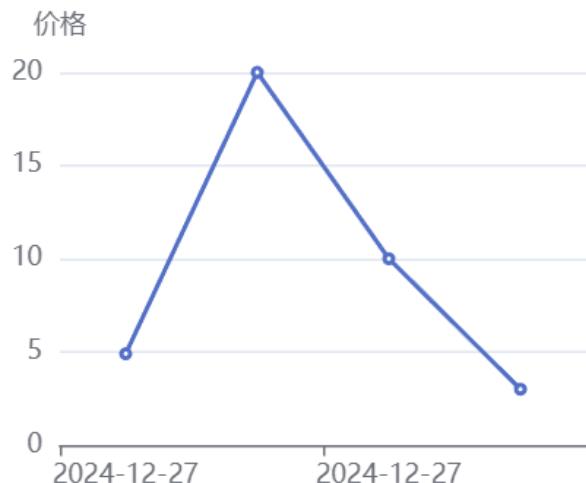
The screenshot shows a product detail page for a '大型3D游戏机' (Large 3D Game Console) on JD.com. At the top, there's a navigation bar with '京东首页' (JD.com Home), '浙江' (Zhejiang), and various user account links. The main product image shows hands holding a white game console connected to a television screen displaying a game. The title '大型3D游戏机' is prominently displayed. Below the image, there's a promotional banner for '11.11大促火热进行中！不等预售，现货低价！最低五折起！还有惊喜券！'. The price is listed as '京东价 ¥6?? 登录查看更多优惠' (JD Price ¥6?? Log in to view more discounts). Further down, there are sections for '增值业务' (Additional Services), '配送至' (Shipped to), '京东物流' (JD Logistics), and service guarantees like '3年全保换新' (3-year full replacement warranty). A large red '加入购物车' (Add to Cart) button is centered at the bottom.

## 6.6 价格走势图页面

展示商品价格的历史数据，并以图表形式展示价格波动情况。

# 价格走势图

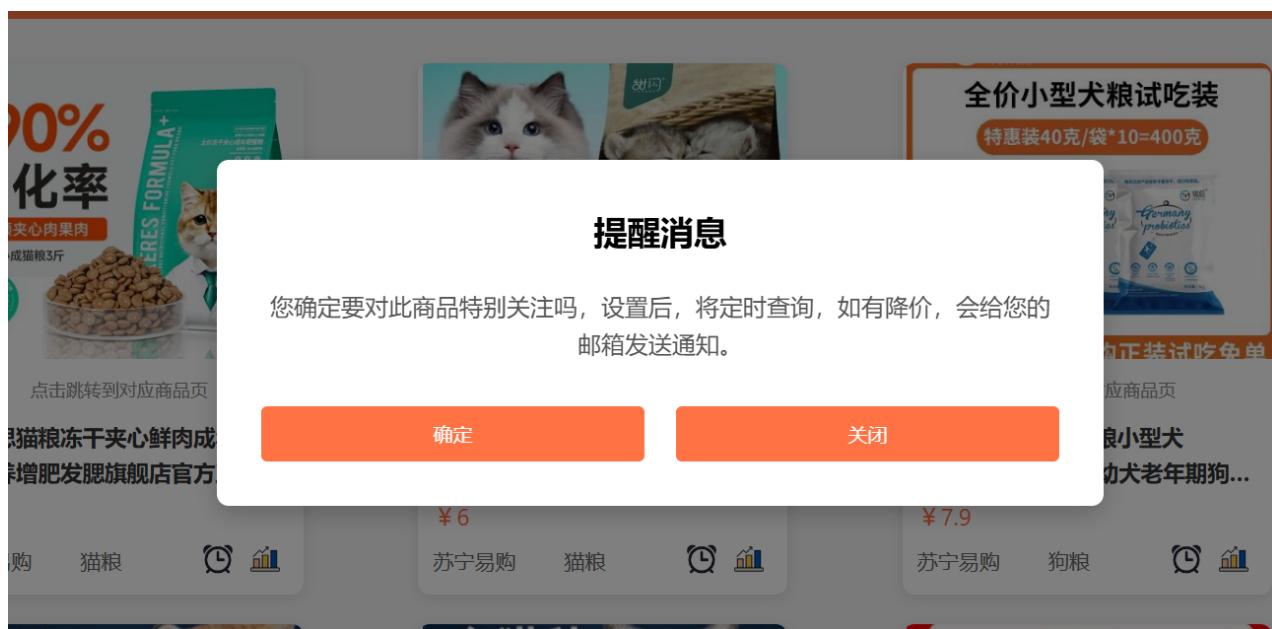
## 商品价格走势图



[关闭](#)

## 6.7 降价提醒设置页面

用户可以设置降价提醒，输入商品信息和目标价格，一旦价格下降则会收到提醒。



| 降价提醒列  |          |                       |
|--|----------|-----------------------|
| 商品名  | 提醒价格     | 操作                    |
| 凯锐思猫粮冻干夹心鲜肉成年期专用营养增肥发腮旗舰店官方正品 鸡肉味 100g91%消化率冻干夹心鲜肉猫粮 | ¥ 3      | <button>取消提醒</button> |
| 狗粮试吃装诺旦狗粮小型犬400g(无冻干)成犬幼犬老年期狗粮、猫粮、猫砂                 | ¥ 10.9 起 | <button>取消提醒</button> |

## 6.8 商品分类页面

展示商品的筛选功能，用户可以按类别筛选商品。



## 价格筛选

最低价格

最高价格

## 价格排序

升序

## 平台选择

苏宁易购



本地搜索模式

点击筛选

## 6.9 手机适配页面

UI 会随着屏幕宽度自行调节商品摆放。

## 7 系统出错设计

系统错误处理模块的设计是保证系统稳定性和可维护性的关键部分。本节将描述如何设计和处理系统中的常见错误，包括用户请求错误、服务器错误、数据库错误等。通过错误捕获、分类和日志记录，确保系统能够正确响应用户，并为开发人员提供足够的调试信息。

### 7.1 错误分类

在系统中，我们将错误分为以下几类：

- **客户端错误 (4xx)**: 由用户请求引起的错误，例如请求缺少必要参数、参数格式不正确等。

- **服务器错误 (5xx)**: 由系统内部错误引起的错误，例如数据库连接失败、服务器异常等。
- **未知错误 (未知码)**: 无法预期和处理的错误，这些错误需要进行日志记录和错误监控。

## 7.2 错误处理机制

### 7.2.1 全局错误处理

全局错误处理是在整个应用中捕获所有未处理的错误，避免系统崩溃，保证错误可以被统一管理。我们使用 Flask 的 `@app.errorhandler` 装饰器处理全局错误。

- **中间件设计**: 通过 Flask 的 `@app.errorhandler` 装饰器，捕获所有未处理的错误，返回统一格式的错误响应给用户。
- **错误日志记录**: 所有系统级错误（如 5xx 错误）将被记录到日志系统中，供开发人员后续排查。
- **返回格式**: 统一的错误返回格式，包括错误代码和错误描述。例如：

```
{"status": 500, "message": "Server internal error, please try again later."}
```

### 7.2.2 常见错误处理

- **用户请求错误 (4xx)**:

- 400 - Bad Request: 请求参数无效或缺失。例如：

```
{"status": 400, "message": "Missing required parameter: Username cannot be empty."}
```

- 401 - Unauthorized: 用户未授权或认证失败。例如：

```
{"status": 401, "message": "The user is not logged in or the login credentials are invalid."}
```

- 404 - Not Found: 请求的资源不存在。例如：

```
{"status": 404, "message": "The requested item does not exist."}
```

- **服务器错误 (5xx)**:

- 500 - Internal Server Error: 服务器内部错误。例如：

```
{"status": 500, "message": "Server internal error, please try again later."}
```

- 502 - Bad Gateway: 网关错误，通常由后端服务无法访问或响应超时引起。例如：

```
{"status": 502, "message": "The request timed out, and the commodity price data could not be obtained."}
```

- 503 - Service Unavailable: 服务不可用，通常是由于服务器过载或停机维护引起。例如：

```
{"status": 503, "message": "The service is temporarily unavailable, please try again later."}
```

- **未知错误 (未知码)**:

- 系统捕获到无法预料的错误时，返回一个通用错误消息和错误代码。例如：

```
{"status": 500, "message": "An unknown error occurred, please contact the administrator."}
```

### 7.3 错误响应设计

每个错误响应应遵循以下格式，确保前端能够快速解析并进行相应处理：

- **状态码 (status)**: HTTP 响应状态码，4xx、5xx 等。
- **错误消息 (message)**: 简洁明了的错误描述。
- **错误详情 (details)**: 可选字段，提供更详细的错误描述，特别是开发阶段。
- **错误代码 (error\_code)**: 唯一的错误代码，便于前端和运维人员快速定位问题。

### 7.4 错误日志记录

系统错误日志是保证系统稳定性和调试的重要工具。在错误处理时，所有的错误信息将记录在日志系统中，包括但不限于：

- 错误的时间戳。
- 错误的类型（例如：400、500 错误）。
- 错误的请求路径和请求参数。
- 错误堆栈信息（如果是代码内部错误）。
- 错误发生时的用户信息（如果适用）。

所有的错误日志将保存到本地日志文件或者集中式日志管理系统（如 ELK、Loggly 等），并定期进行审查和清理。

## 7.5 示例

### 7.5.1 用户登录失败 (401 错误)

当用户登录时，用户名或密码不匹配，系统返回如下错误：

- **请求：**

*POST /api/login*

- **请求体：**

*{ "username": "john\_doe", "password": "incorrect\_password" }*

- **响应：**

*{ "status": 401, "message": "Username or password is wrong.", "error\_code": "AUTH\_01" }*

### 7.5.2 商品查询失败 (404 错误)

当用户查询的商品不存在时，系统返回如下错误：

- **请求：**

*GET /api/product/query?product\_id = 999*

- **请求体：**

无请求体

- **响应：**

{ "status" : 404, "message" : "The requested item does not exist.", "error\_code" : "PRODUCT01" }

### 7.5.3 数据库连接失败 (500 错误)

当数据库连接失败时，系统返回如下错误：

- **请求：**

*POST /api/product/add*

- **请求体：**

{ "name" : "newitem", "category\_id" : 1, "price" : 99.9 }

- **响应：**

{ "status" : 500, "message" : "connect fail", "error\_code" : "DB01" }

## 7.6 总结

系统错误设计的核心在于能够快速、准确地捕获错误并反馈给用户，同时确保开发人员能够通过日志追踪到问题根源。通过规范化的错误分类、统一的错误响应格式和详细的日志记录，系统能够在发生错误时保持良好的可维护性，并且避免由于系统错误导致的用户体验下降。

## 8 附录

### 8.1 时间安排

项目的开发和实施遵循一定的时间安排，以确保按时交付并满足项目需求。以下是项目的时间安排：

- **需求分析阶段：**2024 年 11 月 1 日 - 2024 年 11 月 7 日

- 完成需求收集与分析
- 明确系统功能模块与技术栈

- **系统设计阶段：**2024 年 11 月 8 日 - 2024 年 11 月 14 日

- 数据库设计与 ER 图
- 系统架构设计与技术方案确认

- **前后端开发阶段:** 2024 年 11 月 15 日 - 2024 年 11 月 30 日
  - 完成前端页面开发与接口设计
  - 实现后端逻辑和 API 接口
- **集成与测试阶段:** 2024 年 12 月 1 日 - 2024 年 12 月 7 日
  - 进行系统集成和功能测试
  - 修复 BUG 并优化性能
- **部署与交付阶段:** 2024 年 12 月 8 日 - 2024 年 12 月 10 日
  - 部署系统到生产环境
  - 完成系统交付与文档撰写

## 8.2 备注

- 所有时间安排为预估时间，实际进度可能根据开发过程中遇到的问题进行调整。
- 系统设计阶段将与开发阶段并行进行，确保开发过程中的问题可以尽早识别并解决。
- 在测试阶段，可能会根据实际情况进行多轮回归测试，以确保系统稳定性。
- 部署阶段的测试将特别注意生产环境的性能和安全性，确保用户在使用过程中不会受到影响。