浙江大学

本科实验报告

课程名称：	B/S 体系软件设计
姓　　名：	秦际州
学　　院：	计算机科学与技术学院
系：	软件工程
专　　业：	软件工程
学　　号：	3220105929
指导教师：	胡晓军

2024 年　12 月　25 日

浙江大学实验报告

课程名称：B/S 体系软件设计 实验类型：综合

实验名称：多平台商品比价系统

学生姓名：秦际州 专业：软件工程 学号：3220105929

同组学生姓名：无 指导老师：胡晓军

实验地点：线上 实验日期：2024 年 12 月 25 日

多平台商品比价系统 —— Price Sleuth 软件开发心得

目录

1 引言

本文档旨在全面概述比价平台项目的软件开发流程及其个人心得体会。该项目的核心目标是打造一个多平台商品比价系统，使用户能够在不同电商平台间轻松进行价格比较。本项目遵循 B/S 架构体系，涵盖了前端界面设计、后端服务开发、数据库管理等多个技术领域。

在项目实施过程中，我负责了从需求分析到系统部署的整个软件开发生命周期。通过对项目需求的深入分析，我确定了系统架构，并选择了合适的技术栈以实现项目目标。在开发过程中，我遇到了多项技术挑战，包括但不限于用户界面的优化、系统性能的提升、数据安全与隐私保护等。

本文档将详细介绍项目的技术实现细节、遇到的技术难题及其解决方案，并分享我在完成这门课程作业过程中的心得体会。

2 需求分析和系统设计

当我最初接到这个比价平台项目时，我内心认为这个项目并不具有太大的挑战性。这种自信来源于我在之前课程中积累的 B/S 架构设计经验。老师提供的需求表中列出的功能点，在我看来，大多数都是可以实现的。然而，随着项目的逐步推进，我逐渐意识到自己的初步判断过于简单化了。

在项目的需求分析阶段，我迅速地进行了初步的资料搜集，以确定如何实现比价系统的核心功能——通过官方 API 获取各大主流电商平台的商品价格信息。我认为，这一功能是整个项目成功的关键所在。因此，在没有彻底考虑清楚所有细节和潜在挑战的情况下，我便急切地开始了设计工作。

这种急躁的态度导致了在商品查找功能实现上的巨大时间浪费。我大约花费了两天的时间在寻找和研究相关的 API，但由于网络上缺乏明确的指导，我不得不在多个电商平台的技术文档中进行深入的搜索和探索。遗憾的是，尽管付出了巨大的努力，最终却收效甚微。

这次经历给了我一个深刻的教训：在编程工作开始之前，必须进行全面而细

致的需求分析。这一步骤对于避免项目开发过程中的停滞不前至关重要，同时也能减少编写那些最终无法使用的代码逻辑的风险。一个详尽的需求分析可以帮助我们预见潜在的问题，从而在项目初期就制定出更加切实可行的开发计划。

通过这次项目，我学会了在着手编码之前，要花更多的时间去理解项目需求，评估技术可行性，并制定出合理的开发策略。这不仅能够提高开发效率，还能确保项目能够顺利进行，最终达到预期的目标。

3 前端设计

在之前的课程中，我已经积累了一些使用 Vue3 框架进行前端开发的经验。因此，在本次项目中，我得以将这些经验付诸实践，并致力于打造一个精致的前端界面。如今，我对这个界面的设计感到非常满意。

这次项目对我来说是一个全新的挑战，因为我首次尝试从零开始搭建前端框架。以往的经历大多是在课程已经提供的框架基础上进行开发，所以这次我格外小心谨慎，生怕因为框架搭建不当而引发后续难以修复的问题。为此，我投入了大量时间学习如何搭建一个稳固且高效的前端框架。

在开发过程中，我深刻体会到了前端代码规整的重要性。以往，我常常将 style 属性直接写在 HTML 语句中，并根据实际位置进行调整。然而，这种做法会导致后续元素都需要遵循同样的设置方式，且在界面大小发生变化时，之前设置的固定值可能会导致错位。因此，在本次项目中，我统一采用了 CSS 格式进行书写，并主要使用相对位置（即宽度和长度的百分比）来表示界面布局。这样一来，即使界面大小发生变化，也不会出现错位的情况。

然而，本次项目还需要适配手机端。由于电脑和手机在屏幕长宽比上存在显著差异，即使采用了相对位置表示法，仍然会影响最终的呈现效果。因此，我学习了如何前端读取截面宽度，并将其放在界面的钩子函数中，以实时获取界面的宽度信息。同时，我也利用了 CSS 的媒体查询功能，根据不同的宽度匹配不同的 style，从而轻松实现了手机端的适配。

图 1 手机界面部分展示

在解决了页面布局问题后，我又遇到了新的难题。我使用了 ElementPlus 组件库，但发现许多组件的 style 不能直接修改，否则会被组件自身的风格覆盖。为了解决这个问题，我采取了一些折衷的方法。对于非必要的组件，我采用普通方式实现，并手动设置 CSS。而对于一些集成度较高的信息展示组件，我则通过更加深层的 style 修改来覆盖组件本身的风格。

在解决了这些基础设计问题后，我开始发挥创意进行个性化设计。根据要求，我需要设计降价提醒功能。虽然要求只通过邮箱通知用户，但我在前端界面中添加了额外的通知方式，如主界面右上角的小弹窗和用户界面的收件箱。此外，我还设计了一些细节来增强用户体验。例如，在用户界面的降价提醒中，我根据降价幅度使用不同的颜色进行标注：降价幅度很大的商品用绿色通知，降价适中的

商品用黄色框标注，而一般降价的商品则使用灰色框表示。这样可以帮助用户更直观地区分不同的降价区间，从而选择合适的商品。

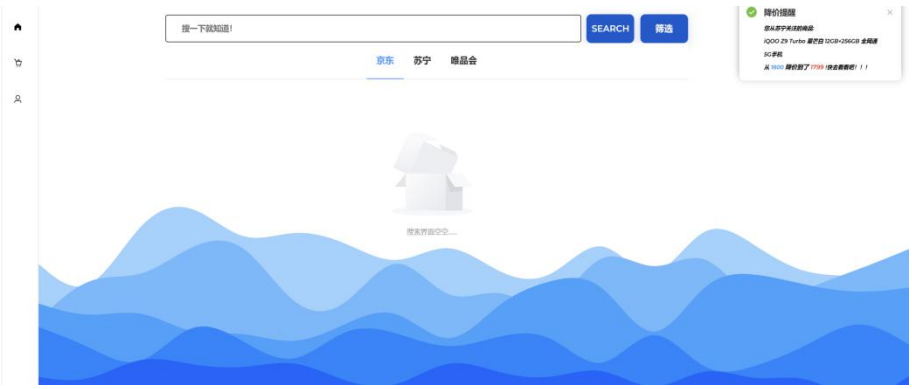

图 2 主界面降价展示

图 3 降价提示界面并进行颜色区分

除了上述细节外，我还在购物车的商品卡片界面上做了一些小设计。为了区分不同平台收藏的产品，我为它们设置了不同的背景颜色。例如，从京东收藏的商品背景是红色的多边形，苏宁的商品背景是黄色，而唯品会的商品背景则是粉色。这些设计不仅让界面更加美观，也提高了用户的使用体验。

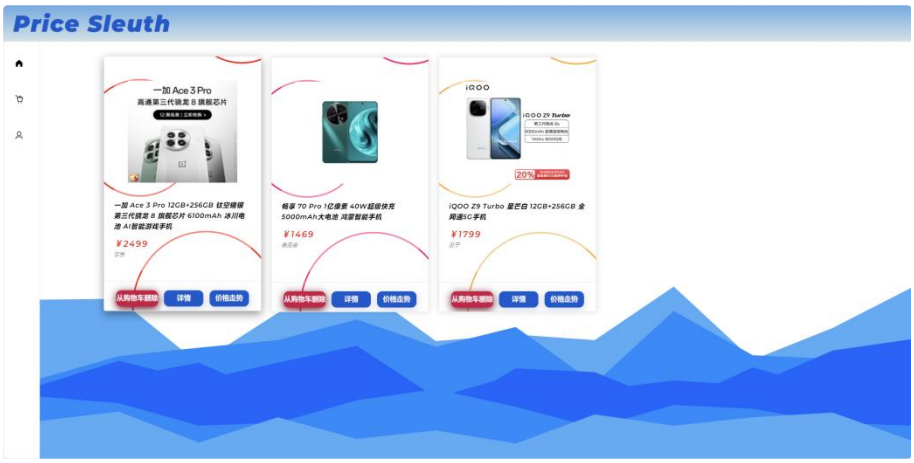

图 4 用户购物车界面

通过这次项目经历，我的前端能力得到了显著的提升。我不仅掌握了 Vue3 框架的深入应用，还学会了如何从零开始搭建一个稳固且高效的前端框架。在这个过程中，我深刻体会到了前端代码规整的重要性，学会了如何使用 CSS 进行灵活的布局设计，并成功解决了适配手机端的问题。

同时，我也深刻认识到了组件库在前端开发中的重要作用，以及如何在不直接修改组件 style 的情况下实现个性化设计。这些经验和技能不仅提升了我的前端开发能力，也为我在未来的项目中更加高效地完成任务打下了坚实的基础。

展望未来，我计划继续深入学习和探索前端领域的新知识。其中，React 框架是我非常感兴趣的一个方向。我计划在寒假，使用 React 来完成一个个人主页的开发。我期待在 React 的学习过程中，能够遇到新的挑战和机遇，不断锻炼自己的解决问题的能力，并在这个过程中收获新的知识和经验。我相信，通过不断的学习和实践，我将成为一名更加优秀的前端开发者。

4 后端设计

后端方面，我选择了 Django 这一框架。之前我的后端开发经历主要基于 Java，而这次实验正是一个学习 Django 后端逻辑，并用 Python 完成整个后端编写的绝佳机会。

Django 框架确实非常方便实用，特别是在数据库操作、用户密码加密和用户 token 返回等方面表现尤为出色。通过调用 Django 的统一接口，就能轻松实现常

规的前后端交互，如登录注册等操作，使用起来相当便捷。

本次项目的后端核心在于爬虫的实现。我采用了 selenium 库来爬取三个平台的数据。由于是初次接触爬虫，我花费了数天时间学习爬虫的逻辑和实现方法。最初，我尝试使用 request 库，但发现它只能爬取静态网页内容，无法获取实时搜索的动态结果。抓包的方法也因主流电商平台设有防抓包机制而行不通。最终，我选择了 selenium 库来模拟用户操作，获取网页的 HTML 文件。

在实现爬虫时，我注意到了一些关键点。例如，京东和唯品会需要登录后才能查找商品，而苏宁则可以直接搜索。因此，我针对不同平台设计了不同的爬虫逻辑。此外，各平台加载数据的逻辑也各不相同。京东可以在搜索结果出来后直接获取数据，而苏宁和唯品会则需要下拉页面并等待数据加载完毕后再进行爬取。

除了商品爬取，登录逻辑也各不相同。唯品会可以直接爬取网站的二维码进行登录，而京东和苏宁的二维码是带有时间戳的，会绑定浏览器，因此在不同浏览器中打开同一个二维码 URL 的结果也会不同。为此，我转变思路，尝试爬取微信的登录二维码，这一方法实现起来相对简单。

在爬取二维码时，还有一些逻辑需要注意。例如，前端发送请求获取二维码时，后端爬取到二维码后需要返回给前端，但不能关闭后端逻辑，因为需要判断是否登录成功。我设计了一个逻辑：前端先发送一个获取二维码的请求，然后定时向后端发送请求以获取当前二维码的状态。第一个请求会保持后端运行并检测登录状态，第二个请求则不断获取当前二维码的 URL。这种异步操作有效地解决了问题。

在后端开发过程中，我收获颇丰。现在我已经能够熟练使用爬虫，并且对平时较少使用的 Python 也愈发得心应手。

5 数据库使用

在这次实验中，我选择了继续使用熟悉的 MySQL 数据库。凭借之前丰富的使用经验，我操作起 MySQL 来已经是得心应手。无论是数据库的创建、表的设计，还是数据的增删改查，我都能够迅速且准确地完成。这种熟悉感不仅提高了我的工作效率，也让我能够更加专注于实验的其他部分，如 Django 框架的应用

和爬虫逻辑的实现。总的来说，MySQL 的熟练使用为我的实验顺利进行提供了有力保障。

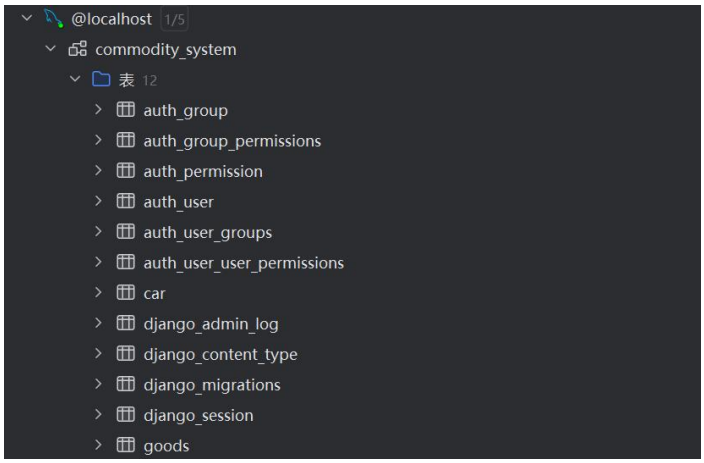

图 5 数据库界面

6 Docker 部署

本次实验中，我认为最具挑战性的部分是 Docker 部署。由于这是我首次接触 Docker，因此倍感陌生。在部署之初，我首先深入学习了 Docker 的基本概念和运行机制，随后根据老师的建议，决定采用 Docker Compose 来整合前后端以及数据库，实现一体化部署。这要求我分别为前端和后端编写 Dockerfile，并在项目根目录下编写 docker-compose.yml 文件，以串联起整个项目。

在尝试部署后端 Django 时，我较为顺利地实现了与数据库的连接。然而，在前端部署方面，我遇到了诸多困难。网络上关于前端部署的方法各异，有的推荐使用 npm 直接运行 Vue 项目，有的则建议使用 Nginx 提供服务。经过多次尝试，我最终选择了 Nginx。但在初次使用时，我发现每次刷新页面都会出现 404 错误，无法找到对应的页面。经过排查，我发现这是因为前端项目经过 npm build 打包后生成的 dist 文件夹中的文件路径问题。于是，我修改了 Nginx 的配置文件 nginx.conf，添加了重写规则，确保在路径不存在时能够重定向到 index.html，从而解决了刷新页面的问题。

虽然前后端都成功部署在了 Docker 中，但它们之间的交互却成了新的难题。由于前端和后端在 Docker 容器中相对独立，我查阅了相关资料后，决定通过端

口映射来实现它们之间的通信。我将前端容器的 80 端口映射到本地的 8080 端口，后端容器的 8000 端口则直接映射到本地的 8000 端口。这样，前端就可以通过向本地的 8000 端口发送请求来与后端进行交互了。

然而，问题并未就此结束。由于我的爬虫程序需要浏览器的驱动，而我在 Windows 本机中使用的 Edge 驱动在 Docker 容器中无法正常工作，因此我不得不更换了浏览器驱动，并修改了后端代码以适应新的驱动。同时，我还在 Dockerfile 中添加了下载和安装新驱动的逻辑。

最后一个难题是关于数据库和后端启动顺序的问题。尽管我在 docker-compose.yml 文件中明确指定了后端的启动依赖于数据库，但有时仍会出现后端启动完成而数据库尚未启动的情况。为了解决这个问题，我编写了一个脚本来检测数据库是否已启动。只有当数据库确实已启动时，才会进行数据库迁移和后端服务的启动。

通过这次 Docker 部署的经历，我不仅深入理解了 Docker 的运行逻辑，还掌握了简单的部署技巧，深刻体会到了 Docker 在开发和部署中的便捷性。

7 总结致谢

这是我首次独立圆满完成的一个前后端开发项目，整个过程中，我倾注了大量的心血，无论是理论学习还是实际操作，都全力以赴。这次经历让我受益匪浅，收获颇丰。

在此，我要衷心感谢自己在这个学期里所付出的努力与坚持，正是这些，让我得以完成一个令自己倍感自豪的项目。同时，我也要深深感谢老师的悉心指导与无私奉献，您的付出让我得以在知识的海洋中畅游，收获满满！

展望未来，我衷心祝愿 BS 这门课程能够越来越好，帮助更多像我一样热爱编程、追求技术进步的同学们实现梦想，成就未来！