开发体会

1. 系统设计阶段

1.1 技术栈的选择

1. 前后端分离：我采用了前后端分离的方法，这是一种现代的主流Web开发模式，提高了系统的灵活性和可维护性。
2. 前端框架选择：我选择React作为前端框架。它具备这样的特点：

- 组件化开发-模块化设计：React鼓励将UI分解为独立的、可重用的组件。每个组件都有自己的状态和逻辑，这使得代码更易于维护和测试。
- 虚拟DOM-性能优化：React使用虚拟DOM来优化渲染性能。虚拟DOM是一个轻量级的内存中的DOM表示，React通过比较虚拟DOM和实际DOM的差异，只更新需要更新的部分，从而减少了对实际DOM的操作，提高了应用的性能。

3. 后端框架选择：我选择Spring Boot作为后端框架。它具备以下特点：

- 快速开发-自动配置：Spring Boot提供了自动配置功能，能够根据类路径中的依赖自动配置Spring应用。开发者无需手动配置大量的XML文件或Java配置类，从而大大减少了配置工作量。
- 简化依赖管理：版本管理：Spring Boot的依赖管理机制确保了各个库之间的版本兼容性，减少了版本冲突的可能性。
- 嵌入式服务器-内置服务器：Spring Boot内置了Tomcat、Jetty和Undertow等服务器，开发者无需单独部署应用服务器，只需通过简单的命令即可启动应用。

1.2 系统设计的挑战与调整

- 在系统设计阶段，我进行了深入的需求分析。这包括了用户分类、登录注册功能、个人信息管理、商品信息搜索、降价提醒等方面。基于这些需求，我设计了相应的数据模型和后端API。这个过程中，我意识到设计是需要在实际开发过程中进行调整的，需要不断地迭代和改进，以适应项目的实际情况。
- 在这个过程中，我明白了技术选型的重要性：选择合适的技术和工具对于项目的成功至关重要。这不仅影响了开发效率，也直接关系到最终产品的性能和用户体验。面对新技术，我不仅要学习它们的基础，还要学会如何将它们应用于实际项目中。这个过程提高了我的技术能力和问题解决能力。

2. 开发阶段

2.1 前端

在前端开发中，我从如下几个角度对前端开发进行了完善：

- 组件化开发：我充分利用了 React 的组件化特性，将界面分解为可重用的组件。这种方法提高了代码的可维护性，也使得功能模块更加清晰。在组件设计时，我注重界面的用户体验和交互设计，确保每个组件不仅功能完备，而且易于操作。
- 状态管理：对于复杂的应用状态，我采用了 Redux 进行状态管理。这帮助我在组件间有效地共享和管理状态，特别是在处理用户认证和设备数据时非常有用。通过 Redux，我能够更方便地跟踪和调试应用状态，特别是在应用规模变大时，它显得尤为重要。
- 响应式和动态数据绑定：React 的响应式系统使得数据的显示和更新变得非常简单。我只需关注数据本身，React 会自动处理 DOM 的更新。动态数据绑定极大地减少了 DOM 操作的代码，提高了开发效率。
- 样式和布局：我在 React 项目中使用了现代 CSS 框架（如 Bootstrap 或 Tailwind CSS），这加速了响应式布局的开发，并提供了一致的界面样式。对于复杂的布局和动画，我利用了 React 的过渡和动画系统，增强了用户交互体验。
- 生态系统：React 拥有丰富的生态系统，我使用了 React Router 进行路由管理，确保应用的导航流畅且易于维护。此外，我还利用了 Axios 进行 HTTP 请求，简化了与后端服务的交互。
- 性能优化：在开发过程中，我特别关注了性能优化。通过使用 React.memo 和 useMemo 等工具，我减少了不必要的渲染，提升了应用的性能。虚拟 DOM 的使用也确保了应用在更新时的流畅性。

2.2 后端

在开发阶段，我主要使用 Spring Boot 作为后端框架，以下是我在开发过程中的一些心得体会：

- 异步处理：Spring Boot 提供了多种异步编程模型，例如基于 @Async 注解的异步方法调用和基于 CompletableFuture 的异步编程。我主要使用 CompletableFuture 来处理异步任务，例如数据库操作和外部 API 调用。
- RESTful API 设计：我使用 Spring Boot 的 @RestController 注解来设计和实现 RESTful API，这些 API 为前端 Vue 应用提供了所需的数据服务。在 API 设计中，我遵循 RESTful 设计原则，例如使用 HTTP 动词表示操作类型、使用资源路径表示资源、使用状态码表示操作结果等。我还注重 API 的可扩展性和安全性，例如使用版本控制、参数校验、权限控制等机制来确保 API 的健壮性和安全性。
- 数据库交互：我选择了适合项目需求的数据库（例如 MySQL），并使用 Spring Data JPA 作为 ORM 框架与数据库交互。Spring Data JPA 提供了强大的功能来简化数据库操作，例如自动生成 SQL 语句、支持分页查询、支持事务管理等。我特别注意了数据库操作的效率和安全性，例如使用索引、避免 N+1 查询、使用参数化查询等。

3. 测试阶段

- 测试计划与用例设计：

我首先制定了详细的测试计划，确定了测试的范围和目标。这包
括了对所有功能点的覆盖，如用户登录、商品获取、数据展示等。 为每个功能点编写了详细的测试用例，包括正常场景和边缘情况。

- 手动测试与自动化测试：手动测试与自动化测试：对于一些关键的用户界面和交互，我进行了手动测试，以确保用户体验的流畅性和直观性。同时，我也实施了自动化测试。
- 错误处理和边缘情况：我特别关注了错误处理和边缘情况的测试，如输入验证、异常处理等。这帮助我发现并修复了可能导致系统崩溃或不正常行为的问题。 通过这些测试，我增强了系统的鲁棒性和用户体验。

4. 总结

事实上，在完成该项目之前，我并没有使用过React以及Spring boot作为前、后端框架。通过这个项目，我巩固了软件工程的系统设计知识，还熟悉了一套技术栈，对前后端分离的B/S设计模式也有了深入的了解，收获巨大。