一、Web 性能分析与调优

1. 后端性能优化（Spring Boot）

• 使用异步处理任务：如邮件通知、支付结果回调等，使用 @Async 提高响应速度。

• 分页加载数据：对图书列表、订单列表等内容，采用分页查询避免一次性加载大量数据，使用 PageHelper 或 JPA 的分页支持。

• 缓存机制：
---使用 Redis 缓存热门图书数据、首页轮播图信息、分类目录等，减少数据库访问压力。
---Spring Cache 注解方式配合 Redis 使用，自动处理缓存。

• 数据库连接池调优：
---使用 HikariCP（Spring Boot 默认）；
---调整 maximumPoolSize 等参数以支持高并发。

一、Web 性能分析与调优

1. 后端性能优化（Spring Boot）

• 减少数据库请求：
--- 合理使用 @Transactional 控制事务；
---避免 N+1 查询，优化 JPA 或 MyBatis 的查询语句；
---批量处理插入和更新。

• 接口限流：
---使用 Bucket4j 或 Sentinel 对特定接口进行限流处理，防止恶意请求或突发流量导致系统崩溃。

一、Web 性能分析与调优

2. 前端性能优化（Vue）

• 懒加载（Lazy Load）：
---图片懒加载，减少页面初始加载资源;
---Vue 路由懒加载（使用动态 import()）减少首页加载体积。

• 打包优化:
---使用 Webpack 的代码拆分（Code Splitting）；
---压缩 JavaScript 和 CSS;
---Tree Shaking 移除未使用的代码。

• 组件缓存:
---使用 <keep-alive> 对路由组件进行缓存，提升返回页面速度。

• 减少 HTTP 请求数:
---合并资源文件（CSS、JS）；
---使用 HTTP/2 提升请求并发效率。

一、Web 性能分析与调优

3. 网络与部署优化

- 前后端分离部署，通过 Nginx 做静态资源分发和反向代理;
- GZIP 压缩 静态资源;
- CDN 加速 静态资源（如图书封面图片）;
- 部署集群 + 负载均衡（如 Nginx+Spring Boot 多实例）以实现高可用性;
- 服务监控与日志分析：如使用 Spring Boot Admin、Prometheus+Grafana、ELK 进行服务性能与异常监控。

二、Web 可用性分析与优化

1. 用户界面与交互设计（前端）

•响应式设计:
---支持 PC 端本地浏览;

• 友好的错误提示与加载状态:
---页面加载中显示 Spinner;
---操作失败/成功给出明确提示（Toast、Snackbar）;
---对 404、500 页面进行美化，提供返回主页按钮。

• 导航清晰:
---分类导航清晰可见;
---面包屑导航增强用户定位;
---表单验证与提示:
---对购物车、支付、注册等表单进行前端验证;
---提供实时错误反馈（如“邮箱格式错误”）。

二、Web 可用性分析与优化

2. 功能可用性（后端）

• 支付模拟稳定性:
---支付宝沙箱环境异常情况的兜底处理;
---对支付状态做幂等性处理，避免重复下单;

• 搜索功能优化:
---支持拼音模糊匹配。

• 访问控制与权限管理:
---卖家/买家分权限显示功能;
---防止未授权用户访问管理页面。