浙江大学

# 本科实验报告

课程名称: B/S 体系软件设计

姓　　名: Li Anjing

学　　院: 计算机科学与技术学院

　　系:

专　　业: 软件工程

学　　号: 3230104388

指导老师: 胡晓军

2026 年 1 月 4 日

# 目录

本次 “B/S 体系软件设计” 课程的大作业为我增加了一次全栈开发经历。从最初的需求构思，到技术选型，再到前后端的编码实现，直至最终的 Docker 容器化部署，我独立完成了一个集成了基于 DeepSeek 和百度智能云 AI 引擎的智能图片管理系统。这是一次代码的学习与实践，也是一次对软件工程全生命周期的实践。在此过程中，我遇到了各种各样的问题，但最终都得以冤难解决。以下是我对整个开发过程的心得体会与小结。

# 1 技术选型与系统架构设计阶段

在项目启动初期，我面临的首要问题是如何设计这个图片管理系统。这个系统具有挑战的点就在于，传统的图片管理系统往往只是简单的增删改查，而这门课程的系统开发要求我们引入 AI，实现一个智能化的视觉资产管理平台。

在前端选型上，根据先前数据库、软件工程基础等课程的经验，我选择了 Vue 3 搭配 Element Plus。Vue 3 的组合式 API 能够让代码逻辑更加聚合，配合 TypeScript 使用时，类型推断可以使前端点开发更加高效；而 Element Plus 提供的丰富组件可以降低 UI 开发的复杂程度。为了实现响应式布局，我学习研究了 CSS 的 Media Query，确保系统在 PC 端和移动端都能有良好的展示效果。

在后端选型上，我选择了 Spring Boot 3 作为技术栈。数据持久层则采用了 MyBatis Plus，提供的通用 Service 和 Mapper 接口，同时运用分页插件和条件构造器实现了较为复杂的检索功能。数据库选择了 MySQL 8.0，其稳定性和对 JSON 数据的支持符合我们该项目的需求。

在 AI 选择上，我引入了 2 个 AI 引擎：

- 视觉层：我引入百度智能云通用物体识别 API，调用成熟的云服务 API 能在低成本下获得较高的识别准确率，实现根据图片 EXIF 信息在上传图片时自动添加标签，并且可以智能识别图片内容新增 AI 智能识别的标签。
- 语义层：引入 DeepSeek-V3 大语言模型，通过 Prompt Engineering，让大模型将用户的自然语言转化为结构化的 SQL 查询条件，从而实现问答检索功能。

# 2 核心开发与 Debug 阶段

## 2.1 后端开发

后端的开发过程中尤其是 AI 接口的对接遇到了不少问题。比如在对接百度智能云时，我遇到了大文件上传超时的问题。当用户上传 10MB 以上的高清图时，网络传输加上 AI 识别的时间容易超时。再比如，百度智能云的标签识别精准度较差，识别的标签与图片实际内容差别较大。最后也都通过相关参数、置信度等的调节，最大程度上优化了性能。

DeepSeek 的语义解析也是一个难点。起初，AI 经常出现幻觉。比如我发现如果提问 “找到所有 2026 年上传到图片”，后端输出筛选的时间范围没有问题，但是回答是 “找不到相关图片”；而如果我单纯输入 “2026 年”，回答可以精准找到所有 2026 年上传到图片。通过了解我发现，AI 会机械地将 “图片” 等词提取为关键词，导致数据库查询条件变成 filename LIKE ’% 图片%’，结果返回为空。为了解决这个问题，我进行了 Prompt 调优，在系统提示词中显式加入负面约束，例如 “删除‘图片’、‘查找’等无意义词汇”，并在 Java 代码层增加了关键词黑名单的过滤逻辑，最终实现了更加理想的 Text-to-SQL 转换成功率。

此外，MyBatis Plus 的分页插件也给我带来了一点困扰。在开发图片轮播功能时，我通过控制台信息发现虽然前端传了 size=10，但后端返回了数据库中所有数据，导致轮播了所有图片库中的图片，

而不是预先设计的轮播最近上传的 10 张图片。经过排查代码以及相关内容的学习才发现，MyBatis Plus 需要配置 PaginationInnerInterceptor 拦截器才能开启物理分页，否则会降级为内存分页。修复这一 Bug 后，不仅能够修复轮播的问题，还能够在图片库缩略图展示页面实现分页，让前端界面得以美化。

### 2.2 前端开发

前端开发遇到的最大困难就是手机移动端适配的问题。起初 PC 端的侧边导航栏在手机上会占据一半屏幕，导致右侧图片库等区域拥挤不美观。根据日常系统界面样式习惯，我设计在手机移动端引入 el-drawer 抽屉组件，结合 CSS 的 Media Query，实现了 PC 端与手机端不同的响应式设计。PC 端保持原先界面不用变，而手机移动端的导航栏设计成可收起可展开的抽屉样式，不占据右侧主页面空间。

最初版本的前端功能以及交互非常简单基础。为了追求更好的用户使用体验，我根据平时对比较成熟的系统界面的观察，增加丰富了前端的 UI 交互设计。比如实现了拖拽改变标签展示顺序，实现标签管理；拖拽上传图片文件；增加“全选当前页”按钮，避免繁杂的选择操作。

### 2.3 Docker 容器制作与部署

因为先前缺少相关经验，Docker 容器的制作部署过程中，我也新学习了不少知识。我编写了 Dockerfile 和 docker-compose.yml，实现了应用容器与数据库容器的编排。在本地 Windows 环境调试通过后，部署容器却发现图片上传报错。而用户注册和登录功能并没有出错，说明数据库连接没有问题。经过排查以及查询相关资料，我意识到使代码里硬编码了 Windows 的路径，而 Docker 的 Linux 容器中不存在 D 盘。随后，我通过引入 Spring Boot 配置文件占位符 ${FILE_STORAGE_-PATH}，并在 Docker Compose 中注入环境变量，配合 Volume 挂载，解决了跨平台路径映射问题。

## 3 测试与收尾阶段

在功能开发完成后，我参照课程作业的要求以及先前的中期设计报告进行了多方面的测试。

- 功能测试：覆盖了注册登录、图片增删改查、标签管理、AI 搜索等核心流程，确保逻辑闭环。
- 非功能测试：测试了系统的浏览器兼容性以及手机移动端适配性。
- 安全性测试：针对 SQL 注入和 XSS 攻击进行了验证。MyBatis 的预编译机制和 Vue 的自动转义使系统能够具有良好的安全性。此外，我也关注了 JWT Token 的安全性，确保密码等敏感信息在传输和存储时都经过了加密处理。

在收尾阶段，我完成了用户使用手册、测试报告等项目文档的撰写，编写了 README.md、导出数据库 SQL 脚本、整理接口文档。在文档撰写过程中，我一次次重新复盘了整个项目的开发流程，有了更加清晰的思路，巩固了学习到的新知识，反思了遇到过的各类问题，并且在复盘过程中进一步对系统进行了微调，使其更加优化。

## 4 小结

通过这次图片管理系统的开发经历，我实现了从理论知识到工程实践的开发过程。

在技术层面，我深入理解了 Vue 3 组件化开发、Spring Boot 微服务架构以及 MySQL 数据库设计，能够独立完成前后端联调。对于 AI 的使用也不再停留在调用 API 的表面，而是学会了如何处理 AI 模型的边界情况，学会了 Prompt Engineering。我还更好地掌握了 Docker 常用命令及 Compose 编排，学会了制作与部署一个 Docker 容器。

在工程层面，我深刻体会到了规范的重要性。无论是接口规范、代码中的异常处理规范，还是 git 版本控制规范，都是保证项目长期可维护的关键。我也更好掌握了通过日志快速定位问题、利用 Postman 进行接口隔离测试，这些对于我以后开发其他项目的 dubug、测试等都有帮助。

当然，这个系统仍有改进空间。例如，目前图片缩略图是实时生成的，对于超大并发访问可能会有性能瓶颈，未来可以引入 Redis 缓存热点数据，或者引入 CDN 加速。此外，AI 功能目前依赖公网 API，未来我可以尝试部署本地轻量级模型以保护用户隐私，进一步我还可以学习训练一个大模型并接入。