Edu-Hub 在线教育平台

课程设计个人总结与心得体会

班级：软件 233

学号：32306300071

姓名：杨竣淇

2026 年 1 月 3 日

1 项目概况

本次课程设计中，我独立完成了“Edu-Hub 在线教育平台”的全栈开发工作。该项目旨在构建一个轻量级、高互动性的在线教学系统，解决传统教学中资源管理分散、师生互动不足的问题。通过本次实践，我成功实现了基于 RBAC 的用户权限体系、递归章节树管理、视频流媒体播放以及基于部门的课程分发复用等核心功能。

2 技术收获

2.1 前端工程化能力的提升

在前端开发中，我深入实践了 **Vue 3 + Vite** 的现代开发模式。通过使用 **Composition API**，我学会了如何将复杂的业务逻辑（如课程进度追踪）抽离为可复用的 Composable 函数，极大地提高了代码的可维护性。同时，**Pinia** 的使用让我对状态管理有了更清晰的认识，摒弃了 Vuex 中繁琐的 Mutation 概念，使数据流向更加直观。此外，**Tailwind CSS** 的原子化设计理念让我摆脱了传统 CSS 命名的困扰，能够快速构建出响应式且风格统一的界面。

2.2 国际化与主题定制

在提升用户体验方面，我实现了前端的国际化与主题切换功能。对于国际化（i18n），我认识到这不仅仅是简单的文本替换，更需要维护一套独立的 JS 语言包变量，将所有界面文案进行抽离和管理。而在主题切换方面，为了支持明暗模式的平滑过渡，我构建了一套基于 CSS 变量（Custom Properties）的样式系统。通过改变根节点的 CSS 变量

值，系统能够实时响应主题变化，这比传统的加载多套 CSS 文件方式更加高效且易于维护。

2.3 后端架构与数据库设计

在后端层面，我掌握了基于 **Node.js (Express)** 构建 RESTful API 的规范。特别是在数据库设计方面，通过 **Sequelize ORM** 进行模型定义和关联查询，我深刻理解了关系型数据库中“一对多”、“多对多”关系的实际应用。例如，在设计“课程-章节”的树形结构时，我对比了“邻接表”与“嵌套集”模型的优劣，最终选择了适合本项目的邻接表方案，并通过递归算法实现了高效的树形数据组装。

3 问题解决与挑战

3.1 递归组件与数据结构

在开发“章节编辑器”时，我遇到了多级目录渲染和拖拽排序的难题。初期尝试直接在模板中循环渲染，导致代码极其臃肿且难以维护。后来通过查阅资料，我采用了 Vue 的**递归组件**技术，配合后端返回的树形 JSON 数据，优雅地实现了无限层级的目录展示。这一过程让我明白了数据结构在前端视图渲染中的决定性作用。

3.2 权限控制的细粒度实现

如何确保“学生只能看已选课程”、“教师只能管理自己创建的课程”是系统安全的重点。我设计了基于 **JWT** 的无状态认证机制，并编写了多个 Express 中间件（如 `requireTeacher`, `isEnrolled`）进行请求拦截。在接口开发初期，由于系统采用了严格的 JWT 鉴权，后续接口的调试必须依赖有效的 Token。为此，我利用 **Apifox** 的环境变量功能，配置了自动提取 Token 的后置脚本，实现了登录后自动将 Token 注入全局变量。这一配置打通了接口调用的鉴权链路，极大地提高了开发效率，也帮助我及时发现并修复了潜在的越权漏洞。

3.3 需求变更与开发范围的权衡

在单人全栈开发的过程中，我深刻体会到了需求分析与实际开发之间的差距。起初我认为设计好的数据库表结构是完美的，但在后续开发中，随着业务逻辑的深入，我不得不频繁地变更表结构。这让我明白了数据库设计是一个持续迭代的过程，而非一劳永逸。

此外，受限于单人开发的时间和精力，我不得不对原定的功能版图进行大幅裁剪。原计划包含“课程系统”、“论坛系统”和“账户管理系统”三大模块。最终，为了保证核心业务的质量，我完全舍弃了独立的“账户管理系统”，将其简化为基础的个人信息修改；对

于“论坛系统”，我也做出了妥协，取消了全站自由发帖的“自由论坛”功能，转而专注于与教学紧密相关的“课程章节评论”功能。这种“做减法”的决策过程虽然痛苦，但却保证了项目核心功能的完整性和稳定性。

4 心得体会

4.1 全栈思维的重要性

本次项目让我从单一的前端或后端视角跳脱出来，建立了全局的**全栈思维**。在定义 API 接口时，我会下意识地考虑前端调用的便利性；在设计前端页面时，我也会考虑到后端数据的查询效率。这种换位思考的能力，极大地降低了前后端联调时的沟通成本。

4.2 规范化与复用的价值

在项目后期，我尝试基于 Edu-Hub 快速衍生出“企业培训系统”。得益于前期良好的代码规范和模块化设计（如封装通用的 `CourseCard` 组件、后端的 `Scope` 查询作用域），我仅用了极短的时间就完成了新系统的原型。这让我深刻体会到，软件工程中的“高内聚、低耦合”不仅仅是一句口号，更是提升开发效率和软件生命周期的金科玉律。

4.3 结语

总的来说，这次课程设计不仅是一次技术的练兵，更是一次工程实践的洗礼。它让我明白了，优秀的代码不仅要能运行，更要易于阅读、易于维护、易于扩展。在未来的学习和工作中，我将继续保持对技术的热情，不断探索，精益求精。