软工个人报告

个人项目总结与课程反思报告

1. 项目概述

1.1 个人角色与职责

在小组课程项目中，我担任**前端开发工程师**的角色，主要负责项目的前端功能实现、UI/UX优化以及与后端的接口对接。具体职责包括：

- 使用 Vue.js 框架进行组件化开发。
- 实现目标管理模块（ObjHub）、AI相关模块及总览模块的核心交互逻辑。
- 基于 Element Plus 和 ECharts/ApexCharts 实现 UI 组件和数据可视化。
- 参与需求分析、原型设计，并协助制定技术方案。

1.2 个人技术栈与工具

技术栈：

- **框架与语言**：Vue.js (v3)、Vue Router (v4)、Vuex (v4)、TypeScript、JavaScript、HTML5、CSS3。
- **UI 组件库**：Element Plus、Lucide Icons、ECharts (v5)、ApexCharts (v4)。
- **Markdown 支持**：`vue3-markdown-it`、`markdown-it-katex`、`@kangc/v-md-editor`。
- **网络请求**：Axios + Axios Interceptors。
- **构建工具**：Vite (v6)、Webpack。
- **动画与交互增强**：GSAP (GreenSock Animation Platform)、墨水波纹动画。

开发工具：

- **IDE**：VS Code。
- **版本控制**：Git / GitHub。
- **协作工具**：Teambition（任务分配与进度跟踪）。
- **软件工程方法**：敏捷开发（Scrum 迭代开发）、UML 设计建模（用于系统结构设计）。

---

2. 项目实施过程

2.1 需求分析与设计

需求调研：

- 通过团队讨论、竞品分析（如 Todoist、Notion），明确了核心功能点：
  - 目标与任务管理（支持单次任务与周期任务）。
  - 社交互动（好友系统、动态发布与评论）。

- 数据分析（日历视图、热力图展示完成情况）。
- AI 引导建议（基于目标内容生成学习路径）。

设计思想与方法：

- **组件化设计**：将页面拆分为多个可复用组件（如 `AiGuidanceBox.vue`，`TaskShow.vue`）。
- **状态管理**：使用 Vuex 管理全局状态（如用户信息、目标列表、任务完成状态）。
- **API 接口封装**：每个业务模块对应独立 API 文件（如 `objhub-api.ts`，`peer-api.js`），提高代码可维护性。
- **响应式布局**：结合 `el-scrollbar`、`flexbox`、`grid` 实现跨设备兼容性。

2.2 编码与实现

核心模块开发：

- **目标管理模块（ObjHub）**
  - 实现了目标创建、编辑、删除、子目标关联等功能。
  - 采用 Axios 封装 RESTful API 请求，处理错误统一提示（ElNotification）。
  - 任务完成状态更新通过 `completeTask()` 和 `cancelCompleteTask()` 方法实现。
- **AI 引导建议（AiGuide）**
  - 利用 `getAiGuidanceForObjective()` 接口从后端获取建议文本。
  - 实现逐字打字效果（逐字显示 + 动画过渡）。
- **数据分析模块（Analysis）**
  - 使用 ECharts 和 ApexCharts 实现日历视图、热力图等图表。
  - 通过 `vuex` 同步目标与任务数据，实现数据驱动视图。

关键技术难点与解决方案：

- **跨域问题**：使用 Vite 的代理配置解决本地开发环境下的跨域限制。
- **性能优化**：对资源加载、无限滚动、高频事件（如鼠标移动）使用防抖节流策略。
- **组件通信**：父子组件通过 props/$emit 通信，全局状态通过 vuex 管理。
- **样式隔离**：使用 `<style scoped>` 防止样式污染。

代码规范与最佳实践：

- 所有变量命名遵循语义化命名规范（如 `taskId`，`isCompleted`）。
- 所有异步操作统一 try-catch 错误处理，避免未捕获异常。
- 使用 TypeScript 类型定义接口，提高类型安全性和可读性。
- 所有组件文件命名统一为 PascalCase（如 `AiGuidanceBox.vue`）。

2.3 测试与迭代

测试方法：

- **单元测试**：使用 Jest 对部分核心函数进行单元测试（如日期格式化、数据过滤）。

- **集成测试**：通过浏览器手动验证各模块功能是否符合预期（如任务提交、目标关联、数据图表渲染）。
- **测试覆盖率**：约覆盖了 70% 的关键路径，部分复杂逻辑因时间限制未能完全覆盖。

2.4 项目管理

项目计划：

- 采用 Scrum 敏捷开发模式，每两周一个迭代周期。
- 初期制定了详细的任务分解表，但由于需求变更导致部分功能延期。
- 例如原计划中的“多层级目标树”最终简化为“父子目标关联”，以适应开发进度。

---

3. 贡献与达成度

3.1 个人贡献度

- 负责前端 60% 的模块开发（共编写约 2000 行 Vue 代码）。
- 实现了大部分核心功能模块的交互逻辑与视觉呈现。
- 优化了 AI 引导模块的用户体验（增加逐字动画 + 提示反馈）。
- 修复了多个关键 Bug，提升整体稳定性。

3.2 目标达成度

功能模块	完成情况	备注
目标管理	完全实现	包括创建、编辑、删除、关联
AI 引导建议	完全实现	后端返回建议，前端渲染
数据分析模块	部分实现	交与另一位前端开发同学
用户社交模块	部分实现	交与另一位前端开发同学
多媒体资源中心	❌未完成	时间限制，功能未完整实现

---

4. 问题与改进

4.1 技术层面

- **未解决的技术难点**：
  - 图表数据异步加载时的空白状态处理不完善。
- **未来学习计划**：
  - 学习更深入的 Vue Composition API 使用技巧。
  - 掌握 GraphQL 替代 RESTful API 提高前后端解耦能力。
  - 深入研究 ECharts 高级图表定制技巧。

4.2 协作层面

- **团队协作痛点**：
  - 需求频繁变更导致返工较多。

- 后端接口文档更新不及时，影响联调效率。

- **改进建议**：
  - 引入接口文档自动化工具（如 Swagger/OpenAPI）。
  - 增加前期原型评审环节，减少后期变更。

---

5. 课程反思

5.1 课程收获

- **知识体系构建**：
  - 掌握了现代前端开发流程（Vue + Vite + TypeScript）。
  - 理解了软件工程中需求分析、架构设计、编码实现、测试交付的全流程。
- **能力提升**：
  - 提升了组件化开发、状态管理、异步编程等硬技能。
  - 提高了团队协作、任务规划、问题沟通等软技能。
- **思维转变**：
  - 从“写代码”转向“做产品”，关注用户价值与体验。
  - 从“个人开发”转向“团队合作”，学会如何高效协同。

5.2 课程改进建议

- **教学内容**：
  - 可增加更多实际案例分析，帮助学生理解企业级开发流程。
- **实践项目**：
  - 建议设置阶段性里程碑，引导学生逐步完成项目。
- **考核方式**：
  - 增加代码质量评估维度（如代码审查、文档完整性）。
  - 增设答辩环节，鼓励学生表达自己的设计思路与思考。

结语

本次课程项目是我第一次参与完整的软件工程项目，不仅提升了我的前端开发能力，也让我深刻体会到团队协作与工程化思维的重要性。虽然过程中遇到不少挑战，但每一次解决问题的过程都让我成长。未来我将继续深化技术积累，同时提升项目管理与沟通协作能力，努力成为一名优秀的全栈开发者。