**项目总结报告**

日期：2025/7/17

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 5 | 项目名称 | 类chatshare多模态平台 |
| 编程语言 | JavaScript/Java/Python | 开发平台和框架 | React/Spring Boot |

|  |
| --- |
| **项目工作小结** |
| 1. 采用哪种架构风格？哪些设计模式？  * 架构风格   本系统采用典型的**分层逻辑架构**，主要分为表现层、控制层、服务层、外部服务层、数据模型层以及数据访问层等核心模块。  表现层是系统与用户或其他外部系统直接交互的顶层模块，为用户提供直观的界面或接口，同时隐藏底层业务逻辑和数据处理细节。  控制层负责处理来自前端的API请求，并将其转发至业务逻辑层。控制层还承担参数验证、异常捕获与路由分发的职责。  服务层负责系统核心功能的实现，如模型路由、微调任务构造、权限校验等，封装了主要的应用逻辑。  外部服务层是系统与外部第三方服务、API或数据存储交互的模块，解耦核心业务逻辑与外部依赖，确保系统能够安全、高效地集成外部资源。  数据模型层定义了系统的核心数据结构，例如User表示用户实体，包含用户名、邮箱、订阅等级等。这些模型类通常与数据库表结构一一对应。  数据访问层封装了对数据库的CRUD操作，通过对应的Repository类访问持久化数据，基于ORM工具实现，与数据库保持良好的结构映射。   * 设计模式   + 桥接模式   用户管理系统引入桥接模式，将用户的角色与生成权限分离。用户的角色维度是继承自抽象基类Person的子类。抽取生成权限维度，并使之成为独立的类层次Permission，这样就可以在Person类中引用这个新层次的对象， 从而使其拥有相应的生成权限。该设计提高了系统的可扩充性，在两个变化维度中任意扩展一个维度，都不需要修改原有系统。   * + 门面模式   AI生成系统通常涉及多个复杂的子系统，用户使用系统的难度提高，系统逻辑变得复杂，维护成本增加。门面模式非常适合用于封装复杂的AI生成系统，为客户端提供一个简化的接口。客户端只需要与门面类交互，无需了解复杂的子系统，子系统变化不会影响客户端代码。开发人员可以在不改变门面接口的情况下优化内部实现，并针对不同场景提供多个门面类。这种设计使得AI生成功能的复杂性被隐藏在一个简单接口后面，同时保持了系统的灵活性和可扩展性。   1. 技术方案有哪些亮点？  核心技术<多模态整合> 文本生成图像、图像再生成文字/3D内容，实现自然语言与视觉数据双向融合<功能新颖> 集成了文生3D、数据微调等功能，满足更多样化的创作需求<多模型选择> 支持切换不同生成模型，API调用多样化生成控制<预设模块> 内置AI续写、角色扮演、高清放大、线稿填充等预设模板<风格预设> 生图模块内置油画、水彩、赛博朋克等典型艺术风格，支持自定义风格描述<参数控制> 生图模块可调节分辨率、生成数量、采样器、随机种子等多种参数部署与性能<灵活部署> 用户可选择在线调用或下载静态模型在本地运行，提升系统弹性和可控性<训练加速> 基于Ray/Deepspeed的分布式微调，加速微调模型训练用户体验<多端适配> UI设计兼顾浏览器与移动设备，交互体验一致，适合多终端展示和使用<智能聊天> 上下文感知的对话系统，支持多轮对话记忆<历史记录> 保存对话记录历史，可以按时间或关键词筛选定制化<自由定制> 为个性化训练和私有数据场景提供接口与系统结构预设，满足专业场景输出要求<模板库> 常用提示词模板一键保存/应用  1. 是否做了单元测试？是否做了系统的功能测试、性能测试、兼容性测试和易用性测试等？   是。单元测试覆盖了系统后端Service层、Security层、Exception层、Entity层、DTO层、Controller层、Config层，整体语句覆盖率达到约90%；  是。合计16个测试用例。   1. 是否采用大模型来辅助开发？对开发生产率约有百分之几的提升？   是。大约60%。 |
| **项目组成员对项目的贡献度（%）** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 姓名 | 需求 | 设计 | 编码 | 测试 | 项目管理\* | 小计 | | 温子墨 | 20% | 20% | 40% | 25% | 20% | 25% | | 杨可凡 | 20% | 25% | 30% | 40% | 10% | 25% | | 蒋宸 | 30% | 30% | 15% | 15% | 35% | 25% | | VITOR JIALE ZHNANG | 30% | 25% | 15% | 20% | 35% | 25% |   注： 项目管理包括项目计划、报告、沟通与协调等。 |

|  |  |
| --- | --- |
| **软件规模** | |
| 前端的代码行数（不包括注解行、空行和复用代码）： | 27714 |
| 后端的代码行数（不包括注解行、空行和复用代码）： | 13539 |

|  |
| --- |
| **经验、教训和建议** |
| **经验**   1. *文档驱动开发与持续重构*  * 建立完善的文档体系，确保开发与文档评审同步推进。 * 每次重大改动需同步更新文档（API文档、操作指南、架构说明）。   *2.渐进式响应式设计*   * 实现智能布局切换，根据屏幕尺寸自动适配最佳视图。   *3.模块化架构设计*   * 前后端分离，采用RESTful API设计，通过稳定接口连接不同层次。 * 组件化开发，确保功能模块独立可复用，制定清晰的项目分层结构。   *4.迭代与协作*   * 制定细粒度迭代计划，任务由成员自主认领，提高效率。   **教训与建议**  **样式管理混乱**：过度使用!important，样式重叠，缺乏统一管理。  **建议**：制定样式规范，依据组件/页面划分样式文件，通过设计系统统一维护。  **技术架构规划不足：**初期选型不完善，后期频繁调整；数据库设计僵化。  **建议：**迭代计划制定阶段进行充分技术调研，明确架构设计。引入灵活的数据层（ORM）和统一错误处理中间件。  **功能与用户体验失衡：**重功能轻体验，移动端适配滞后。  **建议：**早期建立用户体验标准（如交互原型、响应式设计规范）。将用户体验优化纳入迭代任务，邀请志愿者上手测试。  **协作冲突**：多人修改同一组件导致冲突。  建议：采用原子化设计模式，拆分基础组件并统一维护；通过加强沟通协调和GitHub仓库的PR审核机制减少冲突。  **AI辅助开发的局限性：**AI生成代码可能不符合实际需求。  **建议：**将AI作为辅助工具（如生成模板代码），核心逻辑仍需人工验证。建立AI生成代码的审查流程，确保与项目规范一致。 |

项目组各成员签字：*温子墨、杨可凡、蒋宸、VITOR JIALE ZHANG*