基于 LLM 的选课管理系统项目开发方案

程智镝

2024 年 4 月 26 日

目录

1 项目概述

2 关键问题

2.1 项目执行问题

TODO: 对应作业要求中的关键问题

2.2 团队能力和信任问题

2.3 风险管理

2.4 用户体验和反馈问题

2.5 数据安全和隐私问题

2.6 持续维护和升级问题

3 需求分析

3.1 需求规格说明描述

3.2 需求中的主要特点和挑战

3.3 系统的稳定性、性能、安全性和可扩展性

4 解决方案概要

4.1 技术栈选择

前端技术栈：

- 前端框架：使用流行的前端框架，Vue.js，以构建用户友好的界面和提供丰富的交互体验。
- HTML/CSS：使用 HTML 和 CSS 来设计和布局网页，确保页面加载速度快且兼容性好。

- JavaScript：使用 JavaScript 编写客户端逻辑，实现用户操作的动态效果和交互功能。
- UI 库：使用 UI 库，Bootstrap，以便快速构建具有一致性和响应性的界面组件。

后端技术栈：

- 后端框架：使用适合 Web 应用程序的后端框架，SpringBoot。
- 数据库：选择适当的数据库管理系统，MySQL 主从复制或分布式数据库系统以存储和管理学生选课数据。
- API 和数据格式：使用 RESTful API 来处理数据传输和交互，并采用 JSON 或 XML 等标准数据格式。
- 身份验证和安全性：使用身份验证库和安全性框架来确保用户数据和系统的安全性。

其他技术组件：

- 服务器：部署应用程序的服务器，可以选择云托管服务提供商阿里云。搭建多个相同配置的后端服务器以应对高并发请求
- 消息队列：使用消息队列来处理邮件通知和异步任务，提高系统的可靠性和性能。
- 缓存：使用缓存技术 Redis，以提高数据检索和响应速度。
- 负载均衡：负责将传入的请求分发到多个后端服务器，实现负载均衡和高可用，采用软件负载均衡器 Nginx 实现
- 测试工具：选择适当的测试工具和框架，确保代码的质量和稳定性。

4.2 日志和监控

日志记录：

- 选择日志库：选择适当的日志库 Log4j 以便在应用程序中记录日志。
- 设置日志级别：配置不同级别的日志记录，如调试、信息、警告和错误，以便根据需要过滤和查看日志。

- 日志格式：定义日志格式，包括时间戳、日志级别、消息内容以及源代码位置等信息。
- 日志存储：将日志存储在可访问的位置，例如本地文件、数据库或日志管理平台。

监控工具：

- 应用性能监控（APM）：使用 APM 工具 New Relic，来监测应用程序的性能，包括响应时间、事务追踪和错误追踪。
- 基础设施监控：使用基础设施监控工具 Prometheus，监测服务器资源利用、网络流量和数据库性能。
- 日志管理平台：集成日志管理平台 ELK Stack，用于集中存储、搜索和可视化日志数据。

实时警报：

- 设置警报规则：配置警报规则，以便在关键事件发生或性能达到临界值时触发警报
- 通知方式：集成通知方式，如电子邮件、短信、Slack 消息等，以便在触发警报时及时通知相关人员。

日志分析和仪表板：

- 创建仪表板：使用仪表板工具 Grafana，可视化监控数据和日志记录，以便实时查看系统状态。
- 日志分析：使用搜索和查询功能来分析日志数据，以便排查问题、监测趋势和提取有用的信息。

定期审查和优化：

- 定期审查：定期审查监控数据、日志记录和警报历史，以发现潜在问题和性能瓶颈。
- 性能优化：根据监控数据的反馈，优化系统性能，可能包括代码优化、资源扩展和配置调整。

4.3 架构设计

系统整体架构概述选课系统采用了典型的三层架构，包括前端、后端（包含 LLM）和数据库层。这种架构将系统的不同部分清晰地分离，以实现模块化、可扩展和易维护的设计。

前端组件前端是用户与系统互动的界面，负责呈现用户界面、处理用户输入和与后端通信。前端组件包括以下关键特点：

- 用户界面（UI）：使用现代前端框架构建用户友好的界面，以提供直观的用户体验。
- 用户认证和授权：实施用户登录和身份验证机制，根据用户角色授权不同的操作权限。
- 学生选课和 LLM 管理：前端负责展示
- 通知和警报：通过前端界面向用户发送通知和警报，包括考试密码和考试结果的邮件通知。
- 性能优化：前端应具备性能优化策略，包括资源缓存、异步加载和响应式设计，以确保系统在不同设备上表现良好。

后端组件：

- 应用服务器：使用后端框架构建应用服务器，处理前端请求并执行业务逻辑
- 数据库管理：使用适当的数据库系统来存学生选、大模型会话信息、课程信息和用户信息。
- API 接口：提供 RESTful API 接口，用于前端和后端之间的数据传输和交互，包括课程计划管理、选课管理、大语言会话信息处理等。
- 身份验证和安全性：实施用户身份验证和授权，保护用户数据和系统安全。
- 性能优化和缓存：优化后端代码以提高性能，使用缓存来减轻数据库负载。
- 消息队列：集成消息队列，以异步处理邮件通知和其他后台任务。

4.4 数据库设计

- 数据库管理系统：使用合适的关系型或非关系型数据库管理系统来存储数据。
- 数据模型设计：设计合适的数据库表结构，以支持数据的高效检索和存储。
- 数据备份和恢复：实施定期的数据备份策略，以确保数据的安全性和可用性。

5 开发计划

5.1 项目时间表

第 1 周：项目开发和集成

- 周一至周三：进行各个模块的开发，确保每个团队成员按照分好的工进行开发。
- 周四至周五：开始模块集成，确保各个模块能够协同工作，处理接口的问题。

第 2 周：测试和优化

- 周一至周三：进行考试系统整体测试，包括功能测试、性能测试、安全测试等。
- 周四：修复测试中发现的问题，进行考试系统性能优化。
- 周五：完成剩余的优化工作，确保考试系统在各种条件下都能够正常运行。

第 3 周：上线前准备

- 周一至周三：部署系统到预上线环境，进行最后的测试和调优。
- 周四：准备上线所需的文档、备份和监控系统。
- 周五：上线发布项目，进行线上监控和备份。

第 4 周：维护和反馈

- 周一至周三：监控选课系统，处理可能出现的问题和 bug。
- 周四：与用户（考生，老师，管理员等）进行反馈交流，收集用户意见和建议，做好用户满意度调查。
- 周五：根据用户反馈，进行必要的调整和修复。

5.2 项目风险管理

技术风险:

- **风险**: 选择的技术栈可能不够成熟或无法满足系统需求。
- **应对策略**: 在项目前期进行技术评估，验证所选技术的适用性。建立备选方案以备不时之需。

安全风险:

- **风险**: 数据泄露、漏洞和未经授权的访问可能导致系统安全问题。
- **应对策略**: 实施强大的身份验证和授权机制，定期进行安全审计和漏洞扫描，及时修复发现的漏洞。

人员风险:

- **风险**: 项目团队中的关键成员可能离开或出现能力不足的问题。
- **应对策略**: 确保团队有足够的人员资源，建立知识共享和培训计划，减轻对个别成员的依赖。

范围风险:

- **风险**: 需求变更或误解可能导致范围蔓延。
- **应对策略**: 建立严格的变更控制流程，确保每项需求变更都经过评审和批准。与利益相关者进行积极的沟通。

时间风险:

- **风险**: 项目进度可能受到延误，导致无法按计划上线。

- **应对策略**：制定详细的项目计划，设定里程碑并进行定期的进度追踪。提前识别并解决延误问题。

资源风险：

- **风险**：资金、人力和硬件资源可能不足。
- **应对策略**：在项目启动前进行资源规划，与高层管理层协商项目预算，确保有足够的资源支持项目需求。

集成和性能风险：

- **风险**：不同组件之间的集成问题可能导致系统性能下降或故障。
- **应对策略**：进行持续的集成测试，确保各个组件协同工作，同时进行性能测试以发现并解决性能问题。

管理风险：

- **风险**：不良的项目管理实践可能导致项目控制失效。
- **应对策略**：使用有效的项目管理工具和方法，建立明确的沟通渠道，定期审查项目进展。