

基于 LLM 的选课管理系统项目开发方案

程智镒

2024 年 4 月 28 日

目录

1	项目概述	4
2	关键问题	4
2.1	项目执行问题	4
2.2	团队能力和信任问题	4
2.3	风险管理	4
2.4	用户体验和反馈问题	4
2.5	数据安全和隐私问题	4
2.6	持续维护和升级问题	4
3	需求分析	4
3.1	需求规格说明描述	4
3.2	需求中的主要特点和挑战	4
3.3	系统的稳定性、性能、安全性和可扩展性	4
4	解决方案概要	4
4.1	技术栈选择	4
4.2	日志和监控	5
4.3	架构设计	7
4.4	数据库设计	8
5	开发计划	8
5.1	项目时间表	8
5.2	项目风险管理	9

6	功能特点	10
6.1	登录和用户管理	10
6.1.1	用户注册	10
6.1.2	用户登录	11
6.1.3	角色权限管理	11
6.2	课程管理	11
6.2.1	课程导入	11
6.2.2	课程调整和停止	12
6.3	选课策略设置	12
6.3.1	定制化课程参数设置	12
6.3.2	学生名单导入	12
6.4	选课流程	12
6.4.1	自主课程选择	12
6.4.2	选课过程支持	13
6.4.3	防外挂功能	13
6.5	智能化体验	13
6.5.1	智能调配	13
6.5.2	实时反馈	13
7	用户体验和界面设计	14
8	测试和质量保证	14
8.1	测试计划	14
9	部署和维护	14
9.1	部署计划	14
9.2	维护策略	14
10	预算和资源	14
10.1	硬件需求	14
10.2	软件需求	14
10.3	人力资源需求	14
11	推进计划	14

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。
- gpt3.5api: 使用新型大语言 api 接口对选课疑问进行解答，并且根据已有的课程资料对模型进行调整。

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：前端负责展示可选课程和课程计划等信息，提供选课助手（LLM）对话界面，后端对请求进行回复
- 通知和警报：通过前端界面向用户发送通知和警报，包括密码和选课结果的邮件通知。
- 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课信息、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作进行开发。
- 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：进行选课系统整体测试，包括模型测试、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（学生，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。

6 功能特点

6.1 登录和用户管理

6.1.1 用户注册

- **用户信息收集:** 在注册页面，要求用户提供必要的个人信息，如姓名、学号、邮箱地址等。确保用户信息的完整性和准确性。
- **用户角色选择:** 用户在注册时选择其角色，通常分为学生、教师和管理员。每个角色有不同的权限。
- **密码安全:** 强制要求用户创建强密码，包括字母、数字和特殊字符，并确保密码加密存储。
- **邮箱验证:** 发送验证邮件到提供的邮箱地址，包括一个唯一的确认链接。用户需要点击确认链接以完成注册。
- **防止重复注册:** 确保同一邮箱地址或学号不能多次注册。

6.1.2 用户登录

- 用户名密码登录：用户输入已注册的邮箱地址和密码，系统验证这些信息后允许用户登录。
- 记住我：提供“记住我”选项，以便用户在下次访问时免除重新登录。
- 单点登录（可选）：如果系统需要与其他系统集成，可以实施单点登录（SSO）以简化用户登录流程。

6.1.3 角色权限管理

- 角色定义：定义不同角色的权限，如学生、教师和管理员。每个角色有不同级别的访问和操作权限。
- 审计日志：记录用户的登录和操作，以便审计角色权限的使用情况。
- 动态权限管理：允许管理员根据需要更改角色权限，以反映实际情况。
- 用户角色切换（如果适用）：如果用户具有多个角色，例如一个教师也可以是学生，实现用户角色切换功能。
- 密码重置和账户锁定：提供密码重置功能，以及在多次登录失败后锁定账户以提高安全性。
- 权限分配：将权限与角色关联，以便更轻松地为用户分配权限。
- 角色验证：在系统的各个部分使用角色验证，确保用户只能执行其具有权限的操作。

6.2 课程管理

6.2.1 课程导入

- 灵活性：系统应具有足够的灵活性，能够处理不同类型的课程，例如公选课，必修课，通识课等等
- 数据验证：在课程导入过程中，进行数据验证，确保课程信息的正确性和完整性。例如，检查每个课程是否包含课程描述、老师等必要信息

6.2.2 课程调整和停止

- 灵活性: 系统应具有足够的灵活性, 可以在允许范围内让教师对课程进行任意修改
- 兜底限制: 应做好对课程修改的兜底限制, 例如: 时间不能调制周末等

6.3 选课策略设置

6.3.1 定制化课程参数设置

- 课时设置: 允许管理员或教师根据课程的需要来设定课程的课时
- 考试方式设置: 允许教师根据课程本身的特性和需求来设定课程考核的形式
- 选课方式设置: 允许教师根据课程特性设定课程的选课方式, 例如: 指选, 抽签等

6.3.2 学生名单导入

- 数据格式: 允许管理员或教师导入学生名单数据, 数据格式可以是常见的 Excel 格式或 CSV 格式。
- 数据验证: 在导入过程中, 进行数据验证, 以确保学生名单数据的准确性。例如, 检查每个学生是否包括必要的信息, 如姓名、学号、班级等。
- 批量导入: 支持批量导入, 以便一次性导入多个学生名单, 减少手动输入的工作量。
- 重复数据处理: 处理可能存在的重复数据或重复学生记录, 以避免重复导入。

6.4 选课流程

6.4.1 自主课程选择

- 选课时间设定: 管理员可以为不同类型的课程设定不同的选课时间

- 课程呈现：在选课开始后，系统应呈现课程列表给同学，以便他们选择课程。
- 课程信息预览：提供教师和学生查看课程信息的功能，以确保选择时可以看见课程的内容。

6.4.2 选课过程支持

- 备选课程展示：学生应可以对课程加标注，学生可以进行备选课程列表快速选择
- 汇总页面：系统应提供一个汇总页面，显示学生所有选择的课程。这样考生可以了解自己的选课情况。
- 超链接导航：系统应提供超链接，允许学生在不同类型课程之间轻松跳转，并返回到汇总页面。

6.4.3 防外挂功能

- 反脚本技术：系统应该做好对学生操作的监控与限流，防止学生使用脚本选课
- 账号锁定：如果选课期间出现可疑指标，管理员可以远程锁定学生账户，防止进一步作弊。

6.5 智能化体验

6.5.1 智能调配

- 课程冲突检测：系统可以自动检测学生所选择的课程是否存在时间上的冲突，并提供解决方案，如调整课程时间或寻找替代课程。
- 课程平衡建议：基于学生的学业规划和课程安排，系统可以智能地建议学生选择一定比例的核心课程、选修课程和兴趣课程，以保持学业平衡。

6.5.2 实时反馈

- 选课结果预测：系统可以根据学生的选课情况和历史数据，预测学生最终选择的课程，并提供相应的反馈和建议。

- 选课建议优化：系统可以根据学生的反馈和实际选课结果，不断优化推荐算法，提供更加准确和个性化的选课建议。

7 用户体验和界面设计

8 测试和质量保证

8.1 测试计划

9 部署和维护

9.1 部署计划

9.2 维护策略

10 预算和资源

10.1 硬件需求

10.2 软件需求

10.3 人力资源需求

11 推进计划

12 结论