校园问答式RAG知识库项目开发计划书

一、项目概述

1.1 项目背景

校园场景中存在大量分散的文件资源（官网公告、课程PPT、FAQ、规章制度、考试资料等），师生获取关键信息时需手动翻阅多个文件、筛选海量文本，效率低下。本项目旨在开发一款轻量化校园问答式RAG（检索增强生成）知识库，支持PDF/TXT文件上传，快速构建向量索引，实现用户问题与校园资源的精准匹配，返回可追溯的段落证据与简洁回答，解决校园信息检索碎片化、低效化的痛点。

1.2 核心目标（MVP版本）

- 功能目标：支持PDF、TXT文件批量上传，自动完成文本处理与向量索引构建；提供问答交互界面，用户输入问题后，快速返回关联段落证据及精简回答。
- 性能目标：单文件上传处理（100页内PDF/TXT）耗时≤30秒；单条问题检索+回答生成耗时≤5秒；检索命中率≥85%，回答准确率≥80%。
- 落地目标：1-2周完成基础版开发，支持本地部署，适配校园内网环境，师生可快速上手使用。
- 扩展目标：预留多格式支持（PPT、Word）、权限管理、批量导入官网数据的扩展接口，为后续迭代提供基础。

1.3 适用场景

- 学生场景：查询课程重点、考试安排、校规校纪、奖学金申请流程、常见教务问题等。
- 教师场景：检索教学规范、科研通知、会议纪要、学生管理相关文件等。
- 行政场景：快速调取各类公告、流程文件，辅助解答师生咨询，提升办公效率。

二、模型与技术栈优化方案

基于原有需求的模型与技术栈，结合校园场景的轻量化、精准化需求，进行针对性优化，兼顾性能与落地难度，避免过度依赖高算力资源。

2.1 模型优化（核心优化点）

2.1.1 文本分段优化（提升检索精准度）

原有方案未明确分段逻辑，优化为「语义分段+固定长度兜底」策略，适配校园文件的结构化（公告、PPT）与非结构化（FAQ、课程笔记）特点：

- 核心工具：使用LangChain的RecursiveCharacterTextSplitter，结合校园文本特征自定义参数。
- 分段规则：优先按章节、标题拆分（适配PPT、规章制度），无明显结构时按字符长度拆分（单段500-800字符），重叠长度100-150字符（避免拆分语义完整的句子，如课程知识点、通知条款）。
- 优化优势：减少无关文本干扰，提升检索命中率，尤其适配校园公告的条款式文本、课程PPT的知识点拆分。

2.1.2 嵌入模型优化（平衡速度与语义表现力）

针对原有Sentence-BERT/OpenAI Embeddings，结合校园场景轻量化需求，提供两套适配方案，优先推荐本地部署方案：

- 方案一（优先选择）：Sentence-BERT轻量化模型 + 中文微调优化。
- 模型选型：all-MiniLM-L6-v2（轻量，推理快，显存占用低），基于校园文本（公告、课程资料）进行简单微调，提升中文语义匹配度（无需复杂训练，使用少量校园文本做对比学习微调，1-2小时可完成）。
- 优势：本地部署无API调用成本，适配校园内网环境，推理速度快（单段文本嵌入耗时≤1ms），满足批量文件处理需求。
- 方案二（备选）：OpenAI Embeddings（text-embedding-3-small）。
- 适用场景：若具备外网访问权限，且追求更高语义表现力，可选用该方案，嵌入维度512，兼顾速度与精度，适合问答场景的短文本检索。
- 避坑点：避免使用大维度嵌入模型（如text-embedding-3-large），增加索引体积与检索耗时，校园场景无需过高维度。

2.1.3 索引工具优化（适配批量文件与快速检索）

基于原有Faiss，优化索引构建策略，提升批量处理与检索性能：

- 索引选型：优先使用Faiss的IVF_FLAT索引（平衡检索速度与精度），适合中小规模数据集（校园场景预计1000-10000条文本片段，完全适配）；若后续扩展至十万级文本，可升级为IVF_PQ索引（量化压缩，减少内存占用）。
- 优化操作：构建索引时，设置nlist=100（聚类数量，适配文本片段规模），同时保存文本片段与文件来源的映射关系（文件名、页码/段落位置），便于后续返回证据溯源。
- 额外优化：添加索引增量更新功能（原有方案未提及），支持新增文件时无需重建全量索引，仅增量添加新文本片段的嵌入向量，提升用户体验。

2.1.4 生成模型优化（轻量化优先，兼顾回答质量）

摒弃原有通用小型模型，选用适配中文、轻量易部署的模型，同时提供两种生成策略，适配不同算力场景：

- 方案一（本地部署，优先选择）：ChatGLM3-6B（量化版）或Qwen-1.8B-Chat。

- 模型优势：中文表现优于T5/BART，量化后（4-bit/8-bit）可在普通PC上部署（显存占用4-8GB），支持快速生成简洁回答，适配校园问答的口语化、精准化需求（如“如何申请助学金”“课程考核方式是什么”）。
- 生成策略：采用「检索证据+提示词模板」生成，提示词强调“基于提供的校园文本，简洁回答问题，不编造信息，若证据不足则说明无法回答”，避免幻觉。
- 方案二（备选，低算力场景）：模板抽取+轻量化T5微调。
- 适用场景：若设备算力不足（无独立显卡），可选用T5-small中文微调版，仅做文本摘要抽取，结合关键词匹配生成回答，速度快（单条回答生成≤2秒），但回答灵活性略低。

2.2 技术栈优化（提升开发效率与落地性）

基于原有Python、Hugging Face、Faiss、FastAPI/Streamlit，补充依赖工具与优化选型，形成完整技术栈，降低部署难度：

2.2.1 后端技术栈（核心）

- 核心语言：Python 3.9-3.11（兼容性好，避免使用3.12以上版本，部分依赖未完全适配）。
- Web框架：FastAPI（优先选择，异步性能好，自动生成接口文档，便于后续扩展与测试；对比Flask，更适合文件上传与并发请求场景）。
- 文本处理：LangChain（整合文本分段、嵌入、检索、生成全流程，减少重复开发；补充PyPDF2/PyMuPDF用于PDF文本提取，Tika用于复杂格式兼容，提升文件解析成功率）。
- 向量存储：Faiss（核心索引）+ Chroma（备选，轻量级向量数据库，支持增量更新与元数据过滤，比Faiss更易管理，适合新手开发）。
- 模型部署：Transformers（Hugging Face核心库，加载嵌入与生成模型）+ bitsandbytes（实现模型4-bit/8-bit量化，降低部署算力要求）。
- 辅助工具：python-multipart（FastAPI文件上传依赖）、python-dotenv（环境变量管理）、loguru（日志记录，便于问题排查）。

2.2.2 前端技术栈（轻量化，快速落地）

- 方案一（优先选择）：Streamlit + Streamlit-uploader（批量文件上传组件）。
- 优势：开发速度极快，无需前端框架基础（纯Python编写），可快速实现文件上传、问答交互、结果展示界面，适配MVP版本的快速落地；补充streamlit-chat组件，优化问答对话体验，支持历史记录回溯。
- 方案二（备选，追求界面美观）：Vue3 + Element Plus。
- 适用场景：若有前端开发基础，且希望界面更贴合校园风格（适配校园官网色调、布局），可选用该方案，通过Axios调用FastAPI接口，实现更灵活的界面定制。

2.2.3 数据处理与测试工具

- 数据采集：Scrapy（抓取校园官网公告、FAQ，批量获取测试数据；设置爬取频率限制，避免触发官网反爬）；BeautifulSoup4（解析网页文本，提取有效内容）。
- 测试工具：pytest（后端接口测试、模型性能测试）、Postman（接口调试）。
- 部署工具：Docker + Docker Compose（打包前后端、模型、依赖，实现一键部署，避免环境配置问题，适配校园服务器或本地PC部署）。

2.3 优化总结（核心亮点）

- 轻量化：所有核心组件支持本地部署，无需高算力服务器，普通PC即可运行，适配校园内网环境。
- 精准化：针对校园文本优化分段、嵌入、检索策略，提升问答准确率与证据匹配度。
- 易落地：选用成熟工具链，简化开发流程，1-2周可完成基础版，同时预留扩展接口。
- 低成本：无API调用成本，量化模型降低硬件要求，Docker打包简化部署，适合校园项目快速落地。

三、项目开发规划（1-2周，按天拆分）

3.1 开发周期总览

总周期：10天（可根据开发熟练度压缩至7天，或延长至14天完善细节），分为5个阶段：需求梳理与环境搭建、核心模块开发、前后端联调、测试优化、交付部署。

3.2 详细开发计划（按天拆分）

阶段一：需求梳理与环境搭建（第1天）

- 上午：明确细化需求，确定文本分段参数、模型选型（最终确认嵌入与生成模型）、界面原型（简单绘制Sketch或纸笔原型，明确文件上传、问答交互、结果展示的核心布局）。
- 下午：搭建开发环境，完成以下操作：
    - 创建Python虚拟环境，安装核心依赖（FastAPI、LangChain、Transformers、Faiss、PyPDF2等），验证依赖兼容性。
    - 下载选定的嵌入模型（all-MiniLM-L6-v2）与生成模型（ChatGLM3-6B量化版），完成本地加载测试。
    - 搭建Git仓库，初始化项目结构（划分后端、前端、数据、脚本、文档目录），设置.gitignore文件。
- 产出物：环境配置文档、项目目录结构、界面原型草图。

阶段二：核心模块开发（第2-6天，核心阶段）

第2天：文本处理模块开发

- 开发文件解析功能：支持PDF、TXT文件上传，通过PyPDF2/PyMuPDF提取文本，处理PDF中的图片文本（可选，集成pytesseract实现OCR识别，提升解析完整性）。
- 开发文本分段功能：基于LangChain实现语义分段，调试分段参数（字符长度、重叠长度），适配校园公告、PPT文本，生成标准化文本片段。
- 开发文本清洗功能：去除冗余空格、换行、特殊字符，过滤无效文本（如页眉页脚、重复内容），提升后续嵌入与检索精度。
- 测试：上传3-5个校园文件（PDF公告、TXT课程资料），验证文本提取与分段效果，优化清洗规则。
- 产出物：文本处理工具类、测试报告。

第3天：嵌入与索引模块开发

- 开发嵌入功能：集成选定的嵌入模型，实现文本片段的向量转换，添加批量嵌入接口（支持多文件文本片段批量处理）。
- 开发索引功能：基于Faiss构建向量索引，实现索引的创建、保存、加载功能；开发索引增量更新接口，支持新增文件无需重建全量索引。
- 开发元数据管理：为每个文本片段绑定元数据（文件名、页码、段落位置、上传时间），便于后续证据溯源与检索过滤。
- 测试：批量处理10个校园文件，验证嵌入速度、索引构建成功率，测试增量更新功能是否正常。
- 产出物：嵌入与索引工具类、索引管理接口。

第4天：检索与生成模块开发

- 开发检索功能：实现基于向量的相似性检索，设置检索阈值（如相似度≥0.7），返回Top5关联文本片段及元数据；优化检索速度，确保单条检索耗时≤3秒。
- 开发生成功能：集成选定的生成模型，设计提示词模板（适配校园问答场景），实现基于检索证据的回答生成；添加回答过滤规则，避免编造信息、输出无关内容。
- 开发证据关联功能：将生成的回答与检索到的文本片段绑定，确保每个回答都有对应的证据支撑，便于用户验证。
- 测试：输入10-15个校园常见问题（如“奖学金申请时间”“课程重修流程”），验证检索命中率与回答准确率，优化检索阈值与提示词模板。
- 产出物：检索接口、生成接口、证据关联工具类。

第5天：后端接口开发（FastAPI）

- 开发文件上传接口：支持批量上传PDF/TXT文件，接收文件后调用文本处理模块，返回上传结果（成功/失败、处理进度）。

- 开发问答接口：接收用户问题，调用检索与生成模块，返回回答内容、Top5关联证据（含元数据）、检索相似度。
- 开发索引管理接口：支持索引的查询、删除、重建、增量更新，便于管理员维护知识库。
- 开发辅助接口：文件列表查询、问答历史记录查询、接口健康检查。
- 接口文档生成：利用FastAPI自动生成接口文档，编写接口调用说明，便于前后端联调。
- 测试：用Postman调试所有接口，验证接口稳定性、参数合法性，修复接口报错与逻辑漏洞。
- 产出物：后端接口、接口文档。

第6天：前端界面开发

- 基于Streamlit开发核心界面（优先方案）：
    - 文件上传界面：支持批量选择PDF/TXT文件，显示上传进度、处理状态，提供文件列表查看功能。
    - 问答交互界面：支持用户输入问题，显示回答内容、关联证据（标注文件名、页码），支持证据展开/收起，展示问答历史记录。
    - 索引管理界面（简化版）：支持索引重建、增量更新、删除，显示索引状态与文本片段数量。
    - 基础设置界面：支持切换生成模型、调整检索阈值（可选，供管理员使用）。
- 界面优化：适配校园风格（简洁、清晰），优化响应速度，确保文件上传、问答交互无卡顿；添加错误提示（如文件格式错误、检索失败）。
- 测试：模拟用户操作流程，验证界面功能完整性、交互流畅度，修复界面布局问题与逻辑漏洞。
- 产出物：前端界面（Python脚本）、界面测试报告。

阶段三：前后端联调（第7天）

- 完成前后端对接：前端通过请求调用后端接口，确保文件上传、问答、索引管理等功能正常联动。
- 联调测试：模拟完整用户场景（上传文件→等待处理→输入问题→查看回答与证据），排查联调过程中的接口调用错误、数据格式不匹配等问题。
- 性能优化：针对联调中发现的卡顿、延迟问题，优化接口响应速度、检索速度、生成速度（如调整批量处理大小、优化模型推理参数）。
- 兼容性测试：在不同浏览器（Chrome、Edge、Firefox）、不同设备（PC、笔记本）上测试界面与功能，确保兼容性。
- 产出物：联调测试报告、优化后的前后端代码。

阶段四：测试优化与数据准备（第8-9天）

第8天：全面测试与问题修复

- 功能测试：覆盖所有核心功能（文件上传、文本处理、索引构建、问答、索引管理），确保无功能漏洞。
- 性能测试：测试文件处理速度、检索速度、回答生成速度，验证是否达到预设目标（单文件处理≤30秒，问答≤5秒）；测试批量上传（10个文件）的稳定性。
- 评估指标测试：统计检索命中率、回答准确率（邀请5-10名师生参与，输入20-30个问题，人工评分），若未达到目标，优化分段参数、检索阈值、模型提示词。
- 异常测试：测试无效文件上传（如非PDF/TXT、损坏文件）、超长问题输入、高频并发请求（5-10人同时问答），验证系统容错能力。
- 问题修复：针对测试中发现的漏洞、性能瓶颈、体验问题，逐一修复，迭代优化代码。
- 产出物：全面测试报告、优化后的代码。

第9天：数据准备与文档编写

- 数据采集与导入：通过Scrapy抓取校园官网公告、FAQ（100-200条），整理课程PPT、考试资料（10-20份），批量导入知识库，构建测试用数据集。
- 编写使用说明：详细说明系统部署步骤、功能操作流程（文件上传、问答、索引管理）、常见问题排查、注意事项（如文件大小限制、模型部署要求）。
- 编写技术文档：说明项目结构、核心模块逻辑、接口参数、模型选型依据、优化思路，便于后续迭代与维护。
- 录制演示视频：演示系统完整操作流程（从部署到文件上传、问答交互），时长5-10分钟，清晰展示核心功能。
- 产出物：测试数据集、使用说明文档、技术文档、演示视频。

阶段五：交付部署（第10天）

- Docker打包：编写Dockerfile与Docker Compose配置文件，将前后端、模型、依赖打包，实现一键部署。
- 部署测试：在本地PC或校园服务器上部署打包后的系统，验证部署成功率，确保所有功能正常运行。
- 交付整理：整理所有交付物（代码、文档、演示视频、测试数据集、Docker镜像），归档保存。
- 交付验收：梳理交付清单，向需求方演示系统功能，讲解使用方法与技术细节，收集反馈并进行最后微调。
- 产出物：Docker镜像、交付清单、验收报告。

四、数据准备方案

4.1 数据来源

- 校园官网公告：抓取学校官网“通知公告”“教务通知”“科研通知”“学生工作”等栏目内容（用Scrapy爬取，设置爬取间隔1-2秒，避免反爬）。
- 课程资料：收集公开的课程PPT、教案、知识点总结（可从学校教务平台、教师分享的公开资料中获取，确保无版权问题）。
- FAQ常见问题：整理师生高频咨询问题及答案（如教务流程、校规校纪、奖学金申请、宿舍管理等），手动录入或从学校官网FAQ栏目抓取。
- 补充数据：若公开数据不足，手动编写部分校园相关文本（如课程考核规则、社团报名流程），确保数据集覆盖核心场景。

4.2 数据处理规范

- 格式规范：统一整理为PDF或TXT格式，避免使用加密PDF（无法提取文本）；PPT文件需转换为PDF后上传（便于文本提取）。
- 内容规范：过滤无效信息（如广告、无关链接），修正文本错别字、语句不通顺的问题；确保文本内容准确（如通知时间、流程步骤），避免错误信息进入知识库。
- 数量要求：MVP版本数据集至少包含50份文件（20份公告、20份课程资料、10份FAQ），文本片段总量≥1000条，确保检索与生成模型有足够的数据支撑。

4.3 数据导入流程

1. 数据采集与整理：按来源收集数据，统一格式与内容，形成标准化数据集。
2. 批量上传：通过系统的批量文件上传功能，将整理好的文件上传至系统，自动完成文本处理与索引构建。
3. 数据验证：上传完成后，通过问答测试验证数据覆盖度与准确性，补充缺失的核心数据，删除无效数据。

五、评估方案（量化+主观）

围绕核心目标，建立三维评估体系，确保系统满足校园问答场景需求，MVP版本需达到预设评估阈值。

5.1 量化评估指标

评估指标	定义	MVP预设阈值	评估方法
检索命中率	用户问题检索到的Top5证据中，存在与问题相关的有效文本片段的比例	≥85%	人工构造20-30个校园常见问题，统计每个问题检索到的有效证据数量，计算比例

回答准确率	生成的回答与原文证据一致、无编造信息、准确解答问题的比例	≥80%	人工评分（1-5分，3分及以上为准确），统计准确回答的比例；排除证据不足导致的无法回答场景
文件处理速度	单份100页内PDF/TXT文件从上传到完成索引构建的耗时	≤30秒	批量上传10份不同页数的文件，记录平均处理耗时
问答响应速度	用户输入问题到返回回答与证据的总耗时	≤5秒	输入20个问题，记录平均响应耗时，排除网络延迟影响
文件解析成功率	上传的PDF/TXT文件成功提取文本的比例	≥95%	上传50份不同格式、不同内容的文件，统计解析成功的数量

5.2 主观评估指标

- 用户主观满意度：邀请10-20名师生（学生、教师、行政人员）使用系统，从界面易用性、回答相关性、证据清晰度、操作流畅度4个维度评分（1-5分），平均得分≥4分为合格。
- 易用性评估：评估系统部署难度、操作复杂度，确保非技术人员可快速上手（参考使用说明，10分钟内完成文件上传与问答操作）。

5.3 评估迭代机制

评估完成后，针对未达到阈值的指标，制定优化方案：

- 检索命中率低：优化文本分段参数、调整嵌入模型微调数据、提升检索阈值。
- 回答准确率低：优化提示词模板、微调生成模型、增加回答过滤规则。
- 响应速度慢：优化索引结构、压缩模型量化精度、优化接口并发处理。

六、交付物清单

6.1 代码交付物

- 后端代码：FastAPI接口代码、文本处理、嵌入、检索、生成核心模块代码（含注释）。
- 前端代码：Streamlit界面代码（或Vue3代码）、交互逻辑代码。
- 脚本文件：索引构建脚本、增量更新脚本、数据采集脚本（Scrapy）、模型微调脚本。
- 配置文件：依赖配置文件（requirements.txt）、Docker配置文件（Dockerfile、docker-compose.yml）、环境变量配置文件。

6.2 文档交付物

- 使用说明：系统部署步骤、功能操作流程、常见问题排查、注意事项。
- 技术文档：项目结构、核心模块逻辑、接口文档、模型选型与优化思路、数据处理规范。
- 测试报告：功能测试报告、性能测试报告、评估指标测试报告、联调测试报告。

6.3 其他交付物

- 测试数据集：整理好的校园公告、课程资料、FAQ数据集（PDF/TXT格式）。
- 演示视频：系统完整操作演示视频（5-10分钟，含部署、文件上传、问答交互）。
- Docker镜像：打包好的系统Docker镜像，支持一键部署。

七、风险与应对方案

潜在风险	风险影响	应对方案
模型部署算力不足	无法本地部署生成模型，导致问答功能无法使用	优先使用4-bit量化版模型；若算力极有限，切换为模板抽取式生成方案；提供云端部署备选方案（如阿里云轻量服务器）。
校园官网反爬，无法抓取数据	测试数据集不足，影响模型优化与功能测试	降低爬取频率，模拟浏览器请求；手动整理公开资料、编写测试文本；联系学校行政部门获取授权数据。
PDF文本提取不完整（如图片文本、公式）	检索命中率下降，回答缺失关键信息	集成OCR工具（pytesseract）提取图片文本；过滤无法提取的公式、图片内容，标注缺失部分；提醒用户上传可编辑的PDF/TXT。
回答出现幻觉（编造信息）	影响回答准确率，误导用户	优化提示词模板，强制模型基于证据回答；添加回答校验逻辑，若证据不足则返回“无法回答”；人工筛选高质量证据数据，提升模型训练效果。
开发周期超时（超过2周）	无法按时交付，影响项目落地	优先完成核心功能（文件上传、检索、基础问答），预留扩展功能后续迭代；简化非核

		心模块（如界面美化）；合理分配每日开发任务，及时排查问题。

八、后续迭代规划（MVP之后）

MVP版本落地后，可基于用户反馈与校园场景需求，逐步迭代以下功能，提升系统实用性：

- 功能扩展：支持Word、PPT、Excel等多格式文件上传；添加权限管理（学生/教师/管理员角色，区分数据访问权限）；支持批量导入校园官网数据，自动更新知识库。
- 模型优化：基于校园用户问答数据，持续微调嵌入与生成模型，提升回答准确率；支持多模型切换（本地/云端），适配不同用户需求。
- 体验优化：增加问答历史记录导出功能；支持关键词检索、模糊检索；优化界面设计，贴合校园官网风格；添加多语言支持（适配国际学生）。
- 性能扩展：升级向量数据库（如Milvus），支持大规模数据集（十万级文本片段）；实现分布式部署，提升并发处理能力，适配全校师生使用。

九、备注

- 开发过程中，优先保证核心功能（文件上传、检索、问答）的稳定性，再优化界面与细节功能。
- 所有数据需确保合规性，避免使用涉密、侵权文本，抓取校园官网数据前需确认是否允许公开爬取。
- MVP版本建议本地部署或校园内网部署，避免外网访问带来的安全风险与算力消耗。
- 可根据开发团队人数、技术熟练度，调整每日开发任务与周期，确保项目按时交付。

（注：文档部分内容可能由 AI 生成）