项目背景与问题定义
——为什么要做这个 RAG Copilot
1. 业务背景
在 XR 教学编辑器产品中，核心用户是 课程设计人员 / 教学内容编辑人员。
他们的日常工作主要包括：

在编辑器中编排教学流程
上传并管理大量课程素材（文档 / PPT / 教学说明）
根据已有课程内容，快速生成或修改教学说明文本
随着课程数量增加，逐渐暴露出几个明显问题：
内容分散：课程知识分布在多个文档中，查找成本高

重复劳动多：相似课程反复编辑说明，效率低
对新内容不友好：新成员难以快速理解已有课程结构
这些问题并不是「模型能力不够」，而是信息无法被有效利用。

2. 为什么不是“直接用大模型对话”
在早期尝试中，直接使用通用大模型存在明显限制：
模型 不了解具体课程内容
回答容易“编得通顺，但不符合真实教学逻辑”
无法对回答来源进行解释或校验
结论：
问题不在“生成能力”，而在 知识注入与可控性。

3. 项目目标（最小可行目标）
本项目的目标不是做一个“万能 AI”，而是：
做一个能基于课程资料，稳定回答课程相关问题的 RAG Copilot

明确约束如下：
只回答 课程相关内容
回答必须基于 已有文档

最小方案设计
——RAG Copilot 的工程边界

4. 整体方案概览
本项目采用 最小 RAG 架构，而非复杂 Agent 系统：
核心流程：
用户在编辑器中输入问题
系统基于问题进行文档检索（Retrieval）
将检索结果作为上下文注入大模型
模型基于上下文生成回答
明确不做的事情：
不做自动决策链（不引入复杂 Agent Loop）
不做模型微调
不做平台化能力

5. 数据来源与约束
数据来源：课程说明文档 / 教学资料（PDF / 文本）
数据特点：
结构不统一
语言偏业务描述
存在大量相似表述
因此在 MVP 阶段，仅关注：
文档可被检索
检索结果与问题相关
模型回答不脱离文档内容

6. RAG 边界定义（非常关键）
在设计中刻意强调 RAG 的边界：
RAG 只负责：
“把相关内容找出来”
大模型只负责：
“基于给定内容进行组织与表达”

这样做的好处是：
问题可以被拆分测试
出错时能明确责任归因（检索 vs 生成）

测试视角与可验收标准
——这个 Copilot 如何被“测出来”
这一页是面试时最有杀伤力的一页。

7. 核心测试关注点
从测试角度，本项目重点关注三类问题：

7.1 检索是否有效
是否能命中正确课程文档
是否存在明显漏检 / 错检

7.2 生成是否受控
回答是否基于检索内容
是否出现“脱离文档的编造”

7.3 行为是否稳定

相同问题多次提问，结果是否一致
文档不变时，回答是否可复现

8. 最小验收标准（MVP）

本项目定义的最小可验收标准为：
在固定课程文档集下
针对核心课程问题
回答 不出现明显事实错误
回答内容 可追溯到文档
这是一个 “工程可接受” 的标准，而非学术指标。

9. 为什么这个项目适合持续迭代

在当前结构下，后续可以清晰扩展：
增加更多课程文档 → 测检索鲁棒性
引入结构化切分 → 提升命中率
引入测试集 → 做离线回归验证
但这些 都建立在 MVP 跑通的前提之上。

总结一句话：
这个项目不是在证明模型有多强，而是在验证：
在真实业务约束下，RAG 能否被稳定、可测试地用起来。