AI复习平台的开发思路：

我现在有这样一个教育平台需要开发：功能1）智能识别视频的转录内容，然后为转录的内容进行知识点整合和标签，基于视频内容和上传的笔记文档（PDF、WORD、PPT等）生成一个Summary，这个Summary里的知识点自带时间戳，点击时间戳可以跳转到视频对应的地方自动播放；2）Summary的知识点可以匹配我们上传的题库里的题目，3）点击related quesitons可以跳出一个对话框，学生可以在这个对话框里进行题目解答，并且跟LLM交互。

我目前的计划是：

功能一（基于视频和笔记生成Summary并且智能跳转对应的视频内容，已经有一个初稿了）

方案一：

0）针对整体的视频和文本进行Summary

1）对视频轴打tag 想要匹配的字段

2） 再去跟Summary做匹配

方案二：

1）对视频轴打tag 想要匹配的字段

2）基于这个切片的内容整理Summary

功能二：Summary和题目匹配

1）LLM理解Summary知识点，并且给Summary里生成的知识点打标签，这个标签是我们规定好的，比如从\*\*\*这些标签里选择一个最合适的标签

2）rule-based匹配这个标签和题库里题目的知识点标签

功能三：问答式题目交互

1）寻找现成的LLM chatbox框架，进行开发

2）调整LLM的Prompt，让LLM可以实现分步引导、scaffolding、hint、苏格拉底式对话等功能，引导学生一步步解题，从而检测学生是否真的弄懂了这个知识点。

我目前想到的项目开发的要求是：

模块式开发，便于修改和测试；

每个步骤都需要能够evaluate和测试，

我需要不断测试和调整system的propmt以便测验哪一种Prompt能够让生成的效果更好

我需要测试不同的模型的处理任务的效果，综合考虑Accuracy、latency、cost等多方面，选择一个最优的解

在让大模型基于视频和笔记内容尽心Summary的时候，可以先把转录下来的带有时间戳的文本存在本地，这样每次进行Summary功能测试的时候可以自动调用这个转录的文本，而不是每次都得重新转录一遍。

请帮我考虑所有其他可能有问题的地方，然后写一个文档，告诉我每一个功能的实现步骤、使用什么技术，并且考虑到实际开发中的测试问题等等。

main\_v3\_1.py是目前这个版本的后端，目前这个版本可以首先基于视频和笔记生成各种格式的内容，比如完整笔记，Summary等等，其次可以将视频切片并且实现点击对应切片进行跳转播放的功能，video\_processor是针对视频时间戳处理的文本，html是这个完整平台的一个前端页面，仅供参考

prompt是单独放在一个文件夹里，方便后期的管理和调整

## 项目环节

### 环节1：生成高质量Summary文档

**现状**: ✅ 您已经有了基础

* 视频转录 → **直接生成完整Summary** → 包含知识点+时间戳
* **您的目标**: Summary要像教科书一样可读，不是碎片化的片段

### 环节2：从Summary中提取知识点标签

**现状**: ✅ 匹配了

**逻辑**: Summary文档 → 识别其中的知识点 → 打标签(A1, G1等)

* 比如Summary中提到"线性方程的解法"，就标记为A1
* **这是连接视频和题库的桥梁**

### 环节3：题目匹配

**现状**: ❌ 还没有

* 根据知识点标签 → 从AMC题库中找相关题目 → 推荐给用户
* **依赖环节2的标签**

### 环节4：交互式答题

**现状**: ✅ 前端界面有了，后端逻辑没有，Prompt调试

* 弹出题目对话框 → 学生答题 → LLM引导对话 → 判断理解程度
* 后端逻辑可以使用网络上现成的模块？

## 项目架构

project/

├── app.py # 主入口，保持现有Flask结构

├── config.py # 配置管理（开发/生产模式切换）

├── cache/

├── data/ # 数据文件夹

│ ├── cache/ # 缓存文件

│ ├── questions/ # 题库Excel文件

│ └── knowledge/ # 知识点映射

├──prompts/ # prompts

│ ├── ....md

│ ├── ....md

├── core/ # 核心功能模块

│ ├── \_\_init\_\_.py

│ ├── prompt\_manager.py

│ ├── cache\_manager.py # 缓存管理器

│ ├── video\_processor.py # 视频处理（改进版）

├── 现有：分段分析 ✅

└── 新增：整合成完整Summary ⭐ 重点

│ ├── knowledge\_tagger.py # 知识点标记

│ ├── question\_matcher.py # 题目匹配

│ └── tutor\_bot.py # 对话机器人

├── tests/ # 测试文件

│ ├── test\_data/ # 测试数据

│ └── test\_modules.py # 模块测试

└── utils/ # 工具函数

├── data\_loader.py # 数据加载器

└── mock\_data.py # Mock数据生成器

## 环节一 带时间戳的高质量Summary

**时间戳匹配的挑战**:

* 如果您要生成**完整连贯的Summary**，而不是片段式的
* 那么如何知道Summary中的某个知识点对应视频的哪个时间段？

**两种可能的方案**:

### 方案A：先Summary，再匹配时间戳

1. 基于整个转录生成完整Summary
2. 然后想办法把Summary中的知识点和视频时间段关联起来
3. **难点**: 如何精确匹配？

### （选择这个）方案B：先分段，再整合Summary

1. 先把视频按知识点分段（您现有的方法）
2. 然后把这些段落整合成一个流畅的Summary文档
3. **优点**: 时间戳天然准确

第一次：转录视频 → 分析分段 → 保存到文件 后续测试：直接加载文件 → 调用Summary整合器 ✨ 快很多

### 完整的方法列表（需要保留的）

在更新video\_processor.py时，**这些方法都要保留**：

#### 核心处理方法（必须保留）

* extract\_audio\_from\_video() - 音频提取
* \_format\_timestamp() - 时间戳格式化
* transcribe\_video\_with\_timestamps() - 视频转录
* \_find\_matching\_timestamps() - 时间戳匹配
* analyze\_video\_content() - 视频内容分析 **（修改为使用PromptManager）**
* \_parse\_structured\_response() - **结构化响应解析（保留不变）**

#### 缓存相关方法（已添加）

* \_get\_file\_hash()
* \_get\_cache\_file\_path()
* \_save\_analysis\_cache()
* \_load\_analysis\_cache()

#### 工具方法（保留）

* \_print\_processing\_report()
* \_log\_processing\_info()
* clear\_cache()
* list\_cache\_files()

#### 新增Prompt管理方法

* \_validate\_prompts() - **新增**
* \_create\_default\_video\_analysis\_prompt() - **新增**
* update\_analysis\_prompt() - **新增**
* reload\_prompts() - **新增**

### 第一步：优化video processor，添加缓存功能

**优化1-2**: 引入模块化设计

* 导入SummaryIntegrator
* 在\_\_init\_\_中初始化整合器

**优化3**: 缓存系统基础设施

* 创建缓存目录 ./data/cache
* 自动创建目录结构

**优化4**: 核心缓存方法

* \_get\_file\_hash() - 生成文件唯一标识
* \_get\_cache\_file\_path() - 构建缓存文件路径
* \_save\_analysis\_cache() - 保存转录+分析结果
* \_load\_analysis\_cache() - 加载缓存结果

**优化5-9**: 重写主流程

* **优化6**: 先检查缓存，存在则直接加载
* **优化7**: 第一次处理后保存到缓存
* **优化8**: Summary始终重新生成（支持prompt调优）
* **优化9**: 返回完整的结果结构

**优化10**: 缓存管理工具

* clear\_cache() - 清理缓存文件
* list\_cache\_files() - 查看缓存状态

### 第二步 prompt manager

### 阶段测试

第2步：验证缓存效果（今天，10分钟）

第一次运行: 看到转录过程，最后生成缓存文件

第二次运行: 应该直接从缓存加载，跳过转录

检查缓存: 看data/cache/文件夹里是否有JSON文件

## 环节二 从Summary中提取知识点标签

第3步：测试Summary生成（今天，剩余时间）

验证整合Summary: 检查生成的Summary是否像您的样本那样高质量

调试prompt: 如果Summary质量不满意，调整summary\_integrator.py中的prompt

测试时间戳映射: 确认知识点能对应到正确的视频时间

🚨 遇到问题时的检查点

如果报错：

检查是否正确导入了summary\_integrator

确认data/cache目录存在

查看是否有API key相关错误

如果缓存没生效：

检查data/cache/文件夹是否有JSON文件

看控制台是否显示"从缓存加载"信息

如果Summary质量不好：

重点调整summary\_integrator.py中的prompt模板

可以多次运行测试（有缓存后很快）

📋 完成第3步后的下一步

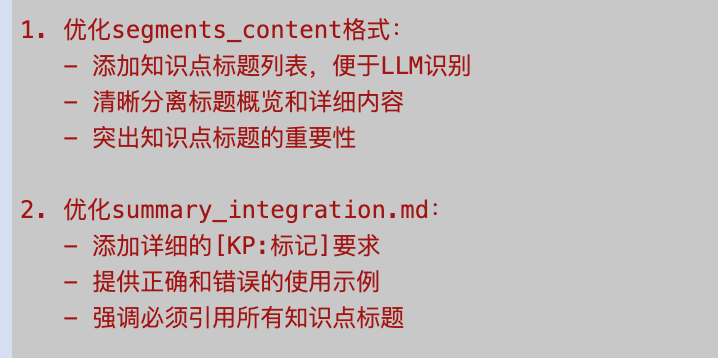
当缓存系统工作正常，Summary质量满意后，我们继续：

创建knowledge\_tagger.py（读取您的mapping.csv，给Summary标记知识点标签）

创建题目匹配功能

Summary 中识别segments\_content的方式不好，怎么优化的？  
- 明确segments\_content的结构

- 调整Prompt



## 环节三 Summary知识点和题目知识点匹配

流程：

Summary 知识点

匹配

题目知识点：