智能体开发：智能培训记录助手

输入一段培训或会议录音，智能体自动生成结构化的培训记录。与普通语音转文字不同，它需要理解课程逻辑、提炼重点、整理知识脉络，并能输出多种形式的成果，实现培训内容的自动沉淀与知识复用。

> 需求：
>
> - 从录音中生成层次化的大纲或思维导图，展示培训内容结构。（已生成大纲）
> - 根据大纲生成上课笔记，格式和风格可自定义。（关键点提取，待实现思维导图）
> - 对讲述中出现的专有名词、案例或典故，自动检索生成“名词解释”与“扩展阅读”。（专有名词解释与示例）
> - 输出排版美观可阅读的 Word 或 PDF 文档，可直接归档或分享（生成markdown格式文件，可导出为pdf）

1. 定义解析

结构化记录：生成具备不同标题项的文本，将内容拆分

课程逻辑：整体表达结构，内容解耦规划

2. 整体流程

- 了解大模型api调用流程
- 了解智能体项目文件结构
- 构造智能体
  - 语音转录
  - prompt与上下文融合
  - 大模型语义分析
- 编写前端
  - 处理流程进度可视化
  - 输出结果展示

3. 🧠 核心文件结构及流程

```
my_llm_agent_project/
├── agent/                  # 智能体核心模块
|   ├── __init__.py
|   ├── agent.py            # 智能体决策流程主逻辑
|   ├── tools.py            # 自定义工具集（搜索/计算/API调用）
|   └── memory.py           # 记忆管理（短期/长期记忆存储）
├── config/
|   ├── __init__.py
|   ├── settings.py         # 参数配置（API密钥/模型参数）
|   └── prompts.py          # 提示词模板管理
├── data/                   # 知识库与数据存储
|   ├── vector_db/          # 向量数据库存储
|   └── knowledge/          # 本地知识文档
├── utils/                  # 工具函数
|   ├── __init__.py
|   ├── logging.py          # 日志记录
|   └── file_loader.py      # 文件加载器（PDF/Word等）
├── routes/                 # API路由（若提供Web服务）
|   ├── __init__.py
|   └── api.py              # FastAPI/Flask路由定义
├── tests/                  # 单元测试
|   ├── test_agent.py
|   └── test_tools.py
├── main.py                 # 应用启动入口
├── requirements.txt        # Python依赖库列表
└── .env                    # 环境变量（敏感信息隔离）
```

3.1 🔧 关键文件详解

文件路径	核心职责	典型内容示例
`agent/agent.py`	智能体决策中枢	• 定义Agent执行流程（ReAct, Plan-and-Execute等) • 集成LLM调用（OpenAI/Claude/Local LLM） • 处理工具调用路由
`agent/tools.py`	扩展智能体能力	python @tool def web_search(query: str): # 调用SerpAPI搜索 return results
`agent/memory.py`	记忆管理	• 短期记忆：`ConversationBufferWindowMemory` • 长期记忆：`VectorStoreRetrieverMemory`

文件路径	核心职责	典型内容示例
`config/settings.py`	集中配置管理	python LLM_MODEL = "gpt-4-turbo" MAX_ITERATIONS = 10 `
`config/prompts.py`	提示工程模板	python SYSTEM_PROMPT = """你是一个专业客服，用简洁语言回答问题...""" `
`data/vector_db/`	知识持久化	• 存放FAISS/Pinecone向量数据库索引 • 存储嵌入后的知识片段
`utils/file_loader.py`	多格式文件支持	python def load_pdf(file_path): # 使用PyPDF2或LangChain文本分割 return chunks
`routes/api.py`	Web服务接口	Python @app.post("/chat") async def chat_endpoint(request: Request): # 调用agent处理请求
`main.py`	程序入口	python from agent import SalesAgent if **name** == "**main**": agent = SalesAgent() agent.run()

3.2 流程

- 了解智能体处理流程
- 明确任务流程：
  - 语音识别模块（语音转文字）
  - 会议纪要生成模块（文字生成纪要）
- 缺陷与优化
  - whisper1(OpenAI语音转录模型)通过request调用---已更改
  - 将utils中负责与大模型沟通的部分完善---所有与大模型的沟通均通过utils.api_client进行
  - 专有名词解释与扩展---已实现
  - 前端编写---已完成

    - 改用讯飞进行流式语音转录替换whisper1---已更改
    - 前端界面优化
        - 实时显示处理结果
        - 阶段分栏
- 将项目转入内网，使用内网训练资料。(待实现)

4. 功能实现

- 语音转文字
- 摘要提取
- 要点提取
- 关键词提取与解析