Knowledge-point-tutor设计

### 知识点专用对话（新功能）

* **使用场景**：深入理解单个知识点
* **对话范围**：严格限制在一个知识点内
* **入口位置**：每个知识点旁边的专用按钮
* **处理器**：knowledge\_point\_dialogue.py

### 步骤1：创建独立的知识点对话系统

具体文件结构：

core/

├── knowledge\_point\_dialogue.py # 知识点专用对话处理器

├── ai\_dialogue\_handler.py # 现有通用对话处理器（不变）

└── prompt\_manager.py # 现有Prompt管理器（不变）

prompts/

├── knowledge\_point\_focused.md # 知识点专用Prompt

├── ai\_dialogue\_templates.md # 现有通用对话Prompt（不变）

└── ...

templates/

├── knowledge\_point\_chat.html # 知识点专用对话页面

├── interactive\_summary\_player.html # 现有Summary页面（添加入口）

└── ...

**目标**：建立完全独立的知识点专用对话功能

### 步骤2：知识点专用对话处理器

├── knowledge\_point\_dialogue.py # 知识点专用对话处理器

**目标**：基于Prompt的智能相关性判断和回复

### 步骤3：专用Prompt设计

├── knowledge\_point\_focused.md # 知识点专用Prompt

**目标**：让LLM自动判断相关性并生成聚焦回复

### 步骤4：独立的前端页面

├── knowledge\_point\_chat.html # 知识点专用对话页面

**目标**：专门的知识点对话界面，与通用对话完全分离

**页面特性**：

* 显著的知识点信息卡片
* 实时相关性状态指示
* "回到焦点"快捷按钮
* 知识点相关的建议问题
* 对话聚焦程度统计

### 步骤5：在Summary页面添加入口

**目标**：为每个知识点提供独立对话入口，在确保其他所有功能不受影响的前提下，添加一个知识点对话的入口，针对这个知识点进行提问。页面也是独立跳出来的。

### 步骤6：后端API路由设计

**目标**：为知识点对话提供专用API

## 实现优先级

1. **第一步**：创建 knowledge\_point\_dialogue.py 处理器
2. **第二步**：设计知识点专用Prompt模板
3. **第三步**：添加后端API路由
4. **第四步**：创建独立的对话页面
5. **第五步**：在Summary页面添加入口按钮