# **AI医疗助手 - 项目计划与路线图**

**文档目的：** 本文档旨在为项目团队所有成员提供一个清晰、统一的项目全景，包括我们的最终目标、开发理念、总体架构和分阶段的实施路线图，以确保团队协作高效，方向一致。

### 1. 项目背景与愿景

#### 1.1 问题陈述

当前，医院医生在诊疗罕见病时，常需花费大量时间查阅医学教科书或最新研究论文。此过程不仅效率低下，还可能因信息检索不全面而增加误诊风险。

#### 1.2 项目愿景

我们旨在开发一款基于大语言模型（LLM）的AI医疗助手应用，为医生提供一个能够即时响应、信息准确的智能问答工具，帮助他们快速获取罕见病的诊疗方案与前沿研究成果。

### 2. 开发理念与总体架构

#### 2.1 开发理念

为了确保项目条理清晰、易于协作，我们将遵循以下四大核心开发理念：

* **模块化 (Modularization):** 我们将系统构建为一套独立但可协作的模块，如同搭建乐高积木。这使得各部分可以独立开发和测试，也便于未来进行功能扩展。
* **迭代开发 (Iterative Development):** 我们不会一次性构建完整的系统，而是采用分阶段、逐步完善的方式推进。这个过程能让我们在每个阶段都获得明确的成果，并及时调整方向。
* **与AI协作 (Working with AI):** 我们的心态是与AI成为合作伙伴，而不仅仅是把它当作一个被动的工具。我们应主动让AI深度参与编码、设计和优化等各个环节，实现人机协同的最大化。
* **拥抱敏捷速度 (Embracing Agile Speed):** 我们要充分意识到，AI驱动的应用开发周期是以“天”为单位计算的，远快于传统模式。团队应保持快速迭代、快速验证的心态，充分利用这一速度优势。

#### 2.2 总体架构

我们的系统将由以下三大核心层级构成：

* **数据层 (Data Layer)**
* **智能核心 (Intelligence Core)**
* **应用层 (Application Layer)**

**层级详解：**

1. **数据层 (Data Layer):**
   * 作为项目的基础，负责管理所有的知识来源，包括结构化的**医学教科书 (Text)和非结构化的最新研究论文 (PDF)**。
2. **智能核心 (Intelligence Core):**
   * 这是项目的大脑。它会利用 **GraphRAG** 技术将原始文本转化为知识图谱，并搭建一个 **Agent System (智能体系统)**。该系统能智能地理解用户问题，并调度不同的知识源来生成最合适的答案。
3. **应用层 (Application Layer):**
   * 这是用户与系统交互的窗口。我们将通过 **Streamlit** 快速构建一个功能性前端界面，如果时间允许，再根据专业UI设计进行优化。

#### 2.3 核心开发模块

为了实现上述架构，我们的开发工作将聚焦于以下三个关键技术模块，后续团队分工也将围绕它们展开：

* **GraphRAG模块:** 对应“智能核心”的知识处理部分，负责将原始数据转化为可检索的知识图谱。
* **Agent System模块:** 对应“智能核心”的决策调度部分，是整个系统的大脑。
* **Streamlit前端模块:** 对应“应用层”，是用户与系统交互的界面。

#### 2.4 系统信息流

当用户提出一个问题时，信息在系统内部的流动过程如下：

1. **用户提问：** 用户通过 **Streamlit前端** 界面输入问题。
2. **请求分发：** 前端将用户的请求发送给 **Agent System (智能体系统)**。
3. **意图理解与规划：** 主控Agent分析问题，理解用户意图，并规划解决步骤（例如，决定是需要查教科书还是查论文）。
4. **工具调用：** Agent调用 **GraphRAG** 工具，向其发出检索指令。
5. **知识检索：** GraphRAG引擎在其知识图谱中进行检索，找出与问题最相关的上下文信息。
6. **信息增强：** 检索到的上下文信息被返回给Agent。
7. **答案生成：** Agent将原始问题和检索到的上下文信息整合成一个更丰富的提示（Prompt），并提交给底层的大语言模型（LLM）以生成最终答案。
8. **结果返回：** 生成的答案经由Agent返回给Streamlit前端。
9. **结果展示：** 前端将最终答案清晰地展示给用户。

#### 2.5 工作流示意图

**第1步: 用户 -> 前端**

* 用户在 Streamlit 前端 界面输入问题。

**第2步: 前端 -> 智能体**

* Streamlit 前端 将请求发送给 Agent System。

**第3-6步: 智能体 <-> 检索器**

* Agent System 分析问题，调用 GraphRAG 引擎 进行检索，并接收返回的相关上下文。

**第7-8步: 智能体 <-> 大模型**

* Agent System 整合信息，向 底层LLM 提问，并接收生成的答案。

**第9-10步: 智能体 -> 前端 -> 用户**

* Agent System 将最终结果返回给 Streamlit 前端，并最终展示给用户。

### 3. 项目实施路线图

我们的开发过程将严格遵循“迭代开发”的理念，分为以下三个清晰的阶段：

#### 第一阶段：PoC (概念验证 - Proof of Concept)

* **释义:** PoC阶段是一系列小型的技术实验，其唯一目的是验证我们的核心技术构想是否可行，而不是开发一个可用的产品。
* **本阶段目标:** **验证我们能否成功使用GraphRAG技术处理医疗文本和PDF，并实现基础的问答功能。**
* **预期成果:** 一系列可独立运行的Python脚本，用以证明核心技术路径已打通，并为下一阶段提供技术选型依据。

#### 第二阶段：MVP (最小可用产品 - Minimum Viable Product)

* **释义:** MVP是产品的最简化版本，它包含足以解决核心问题的功能，并能够让真实用户进行交互和测试。
* **本阶段目标:** **基于PoC的结论，整合各个核心模块，搭建一个可以端到端运行的最小化原型系统。**
* **核心任务:**
  + **模块与接口设计:** 在大规模编码前，为各核心模块（数据层、智能核心、应用层）定义清晰的接口（API）和数据结构。撰写简要设计文档，保证各模块可独立开发、测试和更新。
  + **数据源确认:** 明确MVP阶段需要接入的具体数据来源（例如，某本教科书的特定章节，或几篇精选的论文）。
  + **GraphRAG架构搭建:** 根据PoC结果，最终确定使用LlamaIndex还是LangChain作为核心框架，并选定一个基准大语言模型，搭建出能满足最低要求的后端架构。
  + **Agent System设计:** 细化Agent的设计方案（例如，是单个通用Agent，还是多Agent组合），并完成核心Prompt的初步设计与测试。
* **预期成果:** 一个基于Streamlit的Web应用。用户可以在界面上输入问题，系统能够调用后端的智能核心给出答案。

#### 第三阶段：成品第一版 (Version 1.0)

* **释义:** 这是我们在MVP基础上，经过优化和打磨，最终作为课堂项目交付的正式版本。
* **本阶段目标:** **全面提升系统的稳定性和问答效果，并准备好所有最终交付物。**
* **预期成果:** 一个功能完善、界面友好的应用Demo，一份详尽的项目报告，以及最终演示所需的材料。

### 4. 团队角色与职责 (供6月25日会议讨论)

为确保MVP阶段高效协作，我们将开发任务分解为以下几个核心技术模块。建议团队围绕这些模块进行分工，并在会议中确定具体负责人。

| **开发模块** | **核心任务** | **建议人数** | **负责人 (待填写)** |
| --- | --- | --- | --- |
| **GraphRAG模块** | 负责数据的处理、知识图谱的构建与核心RAG流程的实现。 | 1-2人 |  |
| **Agent System模块** | 负责智能体的设计、工作流编排以及核心Prompt的优化。 | 1-2人 |  |
| **Streamlit前端模块** | 负责搭建用户交互界面，实现前端与后端API的对接。 | 1人 |  |
| **总体设计与协调** | 负责整体技术架构设计、模块接口定义、项目优化、文档撰写及进度协调。 | 1人 | (您) |