# AI 鉴宝师项目：核心开发任务重构

- **多模态理解**：AI 能同时处理图片和文字输入。
- **Agent 决策**：AI 能智能地根据用户需求，选择合适的工具。
- **RAG 增强**：AI 能从外部知识库中检索专业信息，生成更可靠的报告。

---

## 1. 项目模块与开发职责（简化版）

我们将项目精简为三个核心模块，并明确每个模块的最小化开发要求。

### 模块 A：前端开发 (React)

**核心目标**：提供一个简洁、功能单一的 Web 界面，作为 AI 核心模块的**输入和输出窗口**。

**最小化功能列表：**

- **单一页面应用**：仅需一个主页面，无需历史记录、搜索等复杂功能。
- **UI/UX**：
  - 一个图片上传区域，支持单张图片上传。
  - 一个文本输入框，用于用户提问。
  - 一个“开始鉴赏”按钮。
  - 一个用于实时显示加载状态的区域。
  - 一个专门的报告展示区域，用于渲染 AI 返回的 Markdown 格式报告。
- **API 调用**：
  - 仅需调用后端**唯一的**鉴赏接口 (POST /api/v1/appraisal)。
  - 将后端返回的 JSON 数据直接渲染到报告展示区。

**验收标准**：

1. 用户能成功上传一张图片并输入问题。
2. 点击按钮后，能看到加载动画。
3. AI 返回的鉴赏报告能正确、美观地显示。

**开发建议**：可以跳过复杂的路由、状态管理、用户鉴权等，所有逻辑都集中在一个 React 组件中完成即可。

### 模块 B：后端服务开发 (Flask)

**核心目标**：构建一个**轻量级 RESTful API 服务**，只负责接收前端请求，并将请求转发给 AI 核心模块。

**最小化功能列表：**

- **单一 API 接口**：
  - **接口**：POST /api/v1/appraisal
  - **处理逻辑**：
    - 接收前端传来的图片文件和文本问题。
    - 将图片文件保存到一个临时目录或直接转为 Base64 编码，并传递给 AI Agent 核心服务。
    - 调用 AI 核心模块的 Agent，传递图片数据和问题。
    - 将 AI Agent 返回的 Markdown 文本报告，封装成简单的 JSON 格式返回给前端。
    - **无需**历史记录持久化、用户管理等。

**验收标准**：

1. POST /api/v1/appraisal 接口能够正常工作。
2. 接口能够将图片和问题正确地传递给 AI 核心模块。
3. 能够接收并返回 AI 核心模块的最终结果。

**开发建议**：

- 后端逻辑尽量简单，不处理任何业务逻辑。
- **不强制**使用对象存储（OSS），可以直接将图片暂存在本地文件系统。
- **不强制**使用数据库，历史记录功能完全可以省略。

## 模块 C：AI 核心模块（Agent & RAG）

**核心目标**：这是项目的**重中之重**。所有开发资源都应集中于此，确保多模态处理、Agent 和 RAG 功能的可靠性与效果。

**关键功能与技术要求（详细版）：**

1. **多模态 RAG 知识库构建**
   a. **要求**：**必须**包含至少 500 条高质量的图文数据，涵盖不同年代、器型、纹饰的古董。
   b. **实现**：编写脚本，使用 **CLIP** 模型对这些图文数据进行编码，并批量存入 **Milvus** 向量数据库。此知识库的质量直接决定鉴宝的专业性。
2. **Agent 智能体**
   a. **要求**：使用 **LangChain** 或类似框架，构建一个能够决策的 Agent。
   b. **能力**：Agent 必须具备以下两个核心工具调用能力：
      i. multimodal_rag_retriever：接收图片和文本，从 Milvus 中检索最相关的图文资料。

ii. **llm_generator**：接收检索到的上下文和用户问题，调用 **LLaVA** 模型生成报告。

c. **Prompt 工程**：Agent 的 Prompt 必须精心设计，明确其“鉴宝师”的角色定位，确保其能根据不同输入做出正确决策（例如，如果用户只问“什么是青花瓷？”，Agent 不必调用 RAG，直接用大模型回答即可）。

3. **大模型与微调**

a. **要求**：能成功加载 **LLaVA** 或其他主流多模态模型。

b. **可选但强烈推荐**：为了体现微调能力，可以准备一个小规模的**鉴宝领域数据集**（例如 20-50 个高质量问答对），并编写脚本对 LLaVA 进行 **LoRA 微调。**最终项目应使用微调后的模型来提供服务。

**验收标准**：

1. **Agent 决策**：能通过一系列测试用例（例如只传图片、只传问题、图文并传），验证 Agent 能够正确调用不同的工具。

2. **RAG 效果**：上传一张知识库中没有的图片，但其风格与知识库中某类物品相似，RAG 系统能检索到相关的图文资料。

3. **报告质量**：生成的鉴赏报告应结构化、有理有据，且能准确识别出图片中的物品特征。

4. **微调效果**：如果进行微调，能通过对比微调前后模型的表现，证明微调在鉴宝领域的有效性。

## 2. 部署与运维（简化版）

所有模块都应进行 **Docker 容器化。**

- **Dockerfile**：为后端和 AI 核心服务分别编写 Dockerfile。
- **docker-compose.yml**：使用一个简单的 docker-compose.yml 文件，将前端、后端、Milvus、以及可选的 MinIO（用于模拟 OSS）服务编排起来，实现**一键启动。**

最终交付物：
一个包含所有源代码、Dockerfile 和 docker-compose.yml 的代码库，并附带详细的 README.md 文档。该文档需清晰说明如何启动项目，并简要解释 Agent、RAG 和微调的实现原理。

---

# 多模态鉴宝 AI 助理：大模型实战课程大纲（40课时升级版）

本课程分为五大阶段，旨在深入剖析多模态大模型的理论、微调、RAG 和 Agent 四大核心技术，最终完成一个功能强大的商业级项目原型。

---

# 第一阶段：理论基石与工具准备（8个课时）

本阶段重点讲解多模态大模型的底层原理，并搭建整个项目所需的开发环境。

- **第1课：AIGC 时代的变革：多模态大模型概述**
  - 多模态大模型的定义、发展历程与应用前景。
  - **课程项目简介**：为何选择“AI 鉴宝师”作为实战项目。
- **第2课：从 CV 到多模态：计算机视觉核心模型回顾**
  - CNN、Transformer 在图像处理中的应用。
  - 图像编码器与文本编码器的作用。
- **第3课：多模态核心：CLIP 与 LLaVA 深度剖析**
  - **CLIP**：对比学习与图文对齐的原理。
  - **LLaVA**：视觉指令微调与多模态对话能力。
- **第4课：环境搭建：Python、PyTorch 与核心库安装**
  - 搭建 Anaconda/Miniconda 环境。
  - 安装 Transformers、LangChain、Milvus-SDK 等关键库。
- **第5课：容器化魔法：Docker 与 Docker Compose 基础**
  - 理解容器化概念与 Docker 的作用。
  - 编写 Dockerfile 与 docker-compose.yml。
- **第6课：向量数据库的秘密：Milvus 快速上手**
  - 理解向量数据库的原理与优势。
  - Milvus 的安装与基本操作（Collection 创建、数据插入、查询）。
- **第7课：Prompt Engineering 精讲：高效指令设计原则**
  - 掌握 Prompt 的设计技巧，如角色设定、任务拆解。
  - 实战：Few-shot Prompting 与 Chain-of-Thought。
- **第8课：项目架构总览：前后端分离与核心模块设计**
  - 梳理项目的整体技术架构。
  - 讲解前后端、Agent、RAG 和微调模块之间的协作关系。

---

## 第二阶段：多模态大模型微调实战（9个课时）

本阶段将深入讲解大模型微调的理论与实践，赋予模型专业的“鉴宝”能力。

- **第9课：大模型微调理论：为什么需要微调？**
  - 深入理解微调的意义，解决领域知识缺乏与指令遵循问题。
  - 全量微调与参数高效微调（PEFT）的对比。
- **第10课：参数高效微调（PEFT）详解：LoRA 核心原理**
  - LoRA 的数学原理讲解：低秩矩阵分解的直观理解。
  - LoRA 的优势：内存占用小、训练速度快。
- **第11课：多模态微调数据集准备**
  - 讲解高质量鉴宝图文问答对的收集与标注方法。
  - 实战：编写脚本，将原始数据转换为模型可用的微调格式。
- **第12课：LLaVA 模型加载与微调环境配置**
  - 使用 Transformers 库加载 LLaVA 模型。
  - 配置 GPU 环境，准备微调所需的训练脚本。
- **第13课：微调实战：使用 LoRA 对 LLaVA 进行微调（一）**
  - 编写 PyTorch 训练脚本，集成 PEFT 库。
  - 参数设置与训练过程详解。
- **第14课：微调实战：使用 LoRA 对 LLaVA 进行微调（二）**

- 处理训练中的常见问题，如显存溢出、梯度爆炸。
- 使用 wandb 等工具进行训练可视化。

- **第15课：模型评估：微调效果如何量化？**
  - 介绍常用的微调模型评估指标。
  - 实战：编写脚本，对比微调前后模型在鉴宝任务上的表现。
- **第16课：模型保存与加载：如何使用微调后的模型？**
  - 保存微调后的 LoRA 适配器权重。
  - 讲解如何在推理阶段加载基础模型与 LoRA 权重。
- **第17课：微调模型 API 封装**
  - 编写一个独立的 Python 函数，封装微调后的 LLaVA 模型推理服务。

---

## 第三阶段：多模态 RAG 与知识库构建（9个课时）

本阶段将深入讲解 RAG 技术，并带领学员从零开始构建一个多模态鉴宝知识库。

- **第18课：RAG 核心思想：大模型为什么需要知识库**
  - 深入理解 RAG 的工作原理，解决大模型的“幻觉”问题。
  - RAG 在知识密集型任务中的优势。
- **第19课：多模态 Embedding：CLIP 模型深度剖析**
  - CLIP 模型原理讲解：如何将图片和文本映射到同一向量空间。
  - 实战：使用 CLIP 模型对图文数据进行编码。
- **第20课：知识库数据收集与清洗**
  - 讲解鉴宝图录、拍卖记录等资料的收集方法。
  - 实战：编写脚本，批量处理原始图文数据。
- **第21课：数据工程实战：图文数据批量编码与向量化**
  - 编写脚本，批量调用 CLIP 模型进行向量编码。
  - 解决数据批处理中的常见问题。
- **第22课：Milvus 实践：创建与管理多模态向量集合**
  - 使用 Milvus-SDK 创建 Collection，定义字段 Schema。
  - 将编码后的向量与元数据批量导入 Milvus。
- **第23课：向量检索策略：ANN 与精确搜索**
  - 理解近似最近邻搜索（ANN）算法原理。
  - 优化检索性能，如索引选择、参数调优。
- **第24课：RAG 检索工具封装：为 Agent 打造“千里眼”**
  - 编写一个独立的 Python 函数，实现图片与文本的混合检索。
  - 返回检索到的 top-k 相似图文片段。
- **第25课：检索结果的质量评估与优化**
  - 如何评估检索结果的相关性？
  - 讲解 Re-ranking（重排序）技术的应用。
- **第26课：RAG 端到端流程串讲**
  - 回顾并梳理从用户输入到检索结果的全流程。

---

## 第四阶段：Agent 智能体开发（8个课时）

本阶段将独立讲解 Agent 智能体的核心思想、框架与实战，使其能够智能地调用 RAG 和微调后的模型。

- **第27课：Agent 思想：从大模型到智能体**
  - Agent 的概念、核心组件（规划、工具、记忆）。
  - LangChain Agent 框架的原理与架构。
- **第28课：工具箱打造：为 Agent 赋能**
  - 如何为 Agent 封装自定义工具？
  - 实战：封装多模态 RAG 检索工具和微调模型生成工具。
- **第29课：Agent 决策链条构建**
  - 理解 ReAct（Reasoning and Acting）框架。
  - 如何设计 Agent 的 Prompt，引导它根据用户问题做出正确决策。
- **第30课：Agent 核心逻辑实现与测试**
  - 使用 LangChain AgentExecutor 框架，连接大模型与工具。
  - 编写测试用例，验证 Agent 在不同场景下的表现。
- **第31课：端到端集成：Agent、RAG 与微调模型的融合**
  - 如何将 RAG 检索结果作为上下文，传递给 Agent。
  - 如何让 Agent 最终调用微调后的模型生成报告。
- **第32课：Agent 记忆与对话管理**
  - 为 Agent 添加历史记忆功能。
  - 处理多轮对话中的上下文依赖问题。
- **第33课：Agent 工具箱扩展**
  - 为 Agent 增加 Web 搜索工具，用于获取最新拍卖信息。
- **第34课：Agent、RAG、微调三大技术栈总结**
  - 回顾并梳理三大技术栈的融合与协同工作方式。

---

## 第五阶段：项目上线与未来优化（6个课时）

本阶段将把 AI 核心能力与 Web 应用相结合，完成项目部署，并探讨其商业化前景。

- **第35课：Flask 后端服务开发**
  - Flask 框架入门，构建 RESTful API。
  - 编写鉴赏接口，处理文件上传与请求转发。
- **第36课：React 前端开发**
  - React 基础组件与状态管理。
  - 实现图片上传、提问、报告渲染的界面。
- **第37课：前后端联调与 API 测试**
  - 使用 Postman 或 curl 对后端接口进行测试。
  - 解决跨域请求等常见问题。
- **第38课：一键部署：Docker Compose 服务编排**
  - 编写完整的 docker-compose.yml 文件。
  - 实战：在云服务器上使用 Docker Compose 启动整个应用。
- **第39课：部署优化与性能调优**
  - 如何为容器配置 GPU 资源。
  - 探讨服务性能瓶颈与优化方案（如模型量化、批处理）。
- **第40课：项目总结与商业化思考**
  - 回顾项目开发全流程。
  - 探讨“AI 鉴宝师”项目的商业模式与未来发展方向。