武汉大学计算机学院
本科生课程报告

个人项目总结与课程反思报告

专 业 名 称　：软件工程

课 程 名 称　：软件工程

指 导 教 师　：何璐璐 讲师

学 生 学 号　：2023302111029

学 生 姓 名　：何思成

二〇二五年六月

一、项目概述

1．个人角色与职责

作为"得了奖去南二门摆一桌队"的组长，我在团队中扮演着多重角色：首先是项目的算法工程师兼后端开发工程师，同时还承担着团队管理和技术统筹的责任。我们团队开发的项目叫"闻道"。这个名字取自《论语·里仁》中的"朝闻道，夕死可矣"，寓意着知识对人的珍贵与重要性；还有“闻道有先后，术业有专攻”之意，蕴含着我们希望闻道平台能帮助其用户用他们自己的节奏掌握知识的设计理念。

在技术层面，我主要负责四个核心模块的设计与开发工作。教学内容预处理模块是整个系统的基础，负责将原始的教学视频内容转化为可供 AI 处理的结构化数据。AI 总结模块则基于预处理的结果，提供智能化的内容摘要服务。AI 对话模块是用户交互的核心，实现了基于 RAG 的多轮对话功能。知识图谱构建和展示模块则将碎片化的知识点组织成结构化的图谱，并在前端以比较美观的样式展示，提升学习效率。

作为组长，我还需要协调团队成员的工作进度，制定技术方案，并确保各个模块之间的有效集成。这种多重身份让我在项目中既要深入技术细节，又要统筹全局，是一次非常宝贵的实践经验。

2．个人技术栈与工具

在开发技术方面，后端我选择了 Python+Flask+SQLAlchemy+MySQL 的技术栈。Python 的生态系统非常适合 AI 相关的开发工作，Flask 框架轻量级且灵活，SQLAlchemy 提供了优秀的 ORM 功能，MySQL 则保证了数据的稳定存储。

视频预处理部分是技术含量最高的模块之一。我使用 FFmpeg 进行 I 帧提取，这比传统的全帧提取效率更高。为了过滤重复的关键帧，我采用了 Image Phash 相似度算法。语音识别方面选择了 Whisper，它在中文识别上表现优秀。文字处理使用了基于 Paddle OCR 的 CNOCR 包装库。前面这些处理都在本地完成，云端部分则使用在线向量索引服务。特别值得一提的是基于语义相似度的视频分段技术，这得益于前面充分的预处理工作。为了控制并发，我还使用了线程池进行限流处理。

AI 总结模块调用硅基流动 API，支持全文总结、分段总结和关键词提取功能。AI 对话模块使用了 LangChain 框架，支持基于 RAG 的多轮对话，并将对话记录持久化保存到 MySQL 中。知识图谱构建模块通过 LLM 对知识点进行分类并建立关系，由于数据量不大，直接使用 MySQL 存储。

开发工具方面主要使用 VSCode，它的 Python 开发体验很好，插件生态也很丰富。项目管理使用头歌 Git 进行版本控制。在软件工程方法上，我们采用了敏捷开发模式，并使用 PlantUML 语言进行 UML 建模，这让系统设计更加清晰和规范。

二、项目实施过程

1．需求分析与设计

需求分析阶段，作为组长我组织团队成员与邻组进行了头脑风暴会议。我们采用 KJ 法对收集到的需求进行分析和归类，这种方法帮助我们从纷繁复杂的想法中提炼出核心需求。随后，我们与何老师进行了深入的访谈，最终确定了两大核心需求方向：个性化学习和三方数据导入。

个性化学习需求主要体现在系统能够根据用户的学习进度和理解程度，提供定制化的学习内容和问答服务。三方数据导入则要求系统具备良好的扩展性，能够接入不同来源的教学资源。在此基础上，我们进一步提炼出具体的功能需求和非功能需求，特别关注了视频处理性能和课程数据安全性等关键指标。

在设计思想方面，我严格遵循了单一职责原则。视频处理模块被细分为关键帧提取、OCR、ASR、向量索引生成、全文总结和分段总结等独立的处理过程，每个过程都由单一的类来完成，避免了功能重叠和职责不清的问题。这种设计让系统的维护和扩展变得更加容易。

软件架构设计上采用了经典的 MVC 架构+SOA 架构，确保了业务逻辑、数据处理和用户界面的有效分离。设计方法结合了原型化和结构化设计，先用 AI 生成前端界面 demo 快速验证想法，然后通过结构化设计尽可能地实现各模块间的解耦。

2．编码与实现

视频预处理模块

这个模块的开发是整个项目最具挑战性的部分。最初我的设计比较理想化，想要实现完整的功能链条，但在实际开发中发现需要先确保核心的 AI 对话功能能够跑通，所以采用了渐进式的开发策略。

在具体实现上，所有的处理过程都用功能类来实现，采用过程模型的设计思路。考虑到厂商选择和数据合规的需求，OCR 和 ASR 模块都支持抽象类加具体实现的模式，这样后续可以灵活切换不同的服务提供商，目前我们提供了 OCR 处理器为 CNOCR（本地）、腾讯云和阿里云的 OCR 服务，ASR 处理器为 Whisper（本地）和 json 格式字幕导入，用于已有转录字幕的场景。为了提升用户体验，我最后还加上了线程池和实时日志更新功能，日志信息会写入 MySQL，前端可以随时在任务监控界面查看处理进度。

开发过程中遇到了几个重要的技术难点。第一个是关键帧提取技术的选择。最初我使用了自己写的基于 SSIM 相似度的关键帧提取算法，但发现处理速度太慢。改用 FFmpeg 的 I 帧提取后，虽然速度提升了，但重复帧太多，过于敏感。最终的解决方案是在 I 帧提取的基础上，增加 phash 阈值过滤，既保证了速度又控制了重复率。

第二个难点出现在批量处理功能测试时。为了立即给前端返回，我使用了线程后台处理视频，接口立即返回。我发现每个处理线程在后台获取 app context 时会导致数据库连接数爆炸，整个系统崩溃。解决方案是改用线程池排队处理，可以灵活设置并发数，既保证了处理效率又控制了资源消耗。

AI 总结模块

这个模块的核心是分段算法。我使用 Jaccard 相似度对关键帧分割成的区间，嵌入区间内展示过的文字信息，对整个视频进行分段处理，并设置了一系列精心调优的参数：最小区间时长，最大区间时长，每个区间最少多少个关键帧，最多多少个关键帧，内容相关性阈值，区间数量上限。这些参数主要基于 OCR 文本进行调优，因为 ASR 文本比较零散，分段效果不理想。参数的确定主要靠经验和大量测试，这是这个模块最大的挑战。

AI 对话模块

这个模块的功能比最初设想的更加丰富。根据传入参数，用户可以选择对整个课程提问、对单个视频提问、对整个平台内容提问，或者不使用平台数据进行通用对话，而不是我们最初设想的，只能针对某个视频进行对话。对于全平台提问，系统会智能筛选用户有权限访问的课程内容。技术实现上使用 LangChain 框架，结合相似度检索和 BM25 算法进行内容召回，注入聊天记录和包含课程基本信息的系统提示词。系统还会返回引用文档信息，与前端同学合作实现了点击 AI 引用源直接跳转到对应课程时间点的功能。

这个模块的主要难点是召回质量的优化。最初只使用相似度召回，准确率不够理想。后来在原有基础上增加了 BM25 算法，效果有了明显提升。另一个挑战是从单视频问答扩展到整个课程问答的需求变更，需要重新设计数据结构和召回逻辑。在实现流式传输时，还遇到了前端渲染问题，表现为换行前后引用符号未正确渲染，以及换行符被吞噬。解决方案是传输特殊换行符，与前端协作进行替换，同时通过正则表达式解析引用符号，先替换成特殊占位符，最后再进行渲染，最后修复了视频播放界面的 AI 对话换行问题。

知识图谱构建模块

这个模块相对简单一些，主要通过 LLM 对课程所有视频的关键词进行去重和三级分类，然后优先建立上下级关系，酌情建立同级关系。虽然技术含量不高，但在实际调用中也遇到了一些问题。最初 AI 容易生成不存在的关键词，结构化响应也不稳定。通过降低模型温度参数，这些问题得到了有效改善。

除了这些核心模块，我还搭建了整个 Flask 框架的基础架构，包括路由设计、数据库模型定义、API 接口规范等，为整个后端系统提供了稳固的基础。

3．代码规范与最佳实践

在代码规范方面，我严格遵循 Python 社区的 PEP 8 标准，使用下划线命名法。我坚持"代码即文档"的原则，对于函数名和参数名能够清楚表达功能的函数，不添加冗余的文档字符串，但对于参数含义不直观的函数，会提供简要的文档说明。对于逻辑复杂或不直观的代码段，都会添加清晰的注释说明。此外，我还用大模型整理了整个

项目的接口文档等通用的技术文档，降低前后端交流的信息差。

架构设计上采用了 SOA（面向服务架构）和 MVC 的混合模式。SOA 主要针对后端的各种 AI 服务，实现了服务间的充分解耦。MVC 则针对后端数据的展示和管理。通过第三方 API 调用和多线程技术，进一步减少了调用过程中的耦合性，提升了系统的可维护性和扩展性。

作为组长，我制定了明确的代码提交规范：所有代码必须在本地环境完全测试通过后才能提交，同组成员间实行互相审查。这种机制有效保证了代码质量，减少了集成时的问题。

4. 测试与迭代

测试策略方面，我们主要采用了三层测试方法。单元测试是基础，每个模块在提交前都要求通过诸如 if __name__ == "__main__"的方式进行功能验证。系统测试在集中开发结束前或重要里程碑完成前进行，确保各模块间的集成效果。验收测试则在最终展示前执行，模拟真实用户场景。

我们的测试相对比较充分，特别是同组同学还进行了接口测试等灰盒测试，这对系统安全性非常重要。测试的充分性主要体现在覆盖了主要的功能路径和异常情况，能够及时发现和解决潜在问题。

缺陷管理采用了轻量级的方式，主要通过 QQ 群快速@对应的代码负责人，问题解决后在群内通报。这种方式虽然简单，但在小团队开发中非常高效。对于自己发现的 bug，我坚持快速解决并向团队通报可能影响的功能模块。

开发过程中最大的感受是"魔鬼藏在细节里"，有时候一个很小的错误就能让整个功能表现异常。这让我深刻认识到良好编程习惯的重要性，包括充分的日志记录、各种值的校验等防御性编程技术。

5. 项目管理

项目计划制定基于前期访谈和 KJ 法分析的结果，我们根据需求优先级和课后开发时间有限的现实情况，在《项目任务书》中制定了分阶段的开发计划。原计划第二学

期内完成课堂数据抓取清洗、智能问答系统和课堂智能总结功能，最终不仅完成了这些目标，还超额完成了原定于暑期实践的动态知识图谱创建功能，实现了通过关键词定位知识图谱节点的功能。整个项目没有遇到明显的进度延误，基本按照预定计划推进。

团队协作主要通过 QQ 群进行日常沟通，这种方式比较高效和轻量。团队分工协作整体比较顺利，大家都能按时完成分配的任务。对于 AI 工具的使用，我们采取了开放的态度，只要最终能保证质量就可以。偶尔会遇到某个组员完成任务比较敷衍的情况，但通过友善的提醒和沟通，最终都能得到很好的解决。

团队氛围总体比较和谐，没有出现明显的分歧和冲突。这主要得益于前期充分的需求分析和明确的分工，以及日常及时有效的沟通机制。

三、贡献与达成度

1．个人贡献度

根据 Git 仓库的统计数据，我在整个项目中的代码贡献量为 72,247 行（新增 48,243 行，删除 24,004 行）。结合其他方面的统计数据，我在团队中贡献度为 30%。

从功能模块的角度看，我实现了后端的基本架构和所有与 AI 相关的核心功能，实现了知识图谱功能的前后端逻辑。同时，与前端同学的密切合作，成功实现了点击跳转到任意视频任意时间点的功能，这大大提升了用户体验。实现对话功能，并将对话功能从最初只能对单个视频提问扩展到可以对整个课程进行对话，显著增强了系统的实用性。同时增加了 AI 思考过程的可视化显示，让用户能够更好地理解 AI 的收集资料、给出答案的过程。

在系统性能方面，通过优化视频预处理流程和引入线程池机制，显著提升了批量处理的效率和稳定性。在代码质量方面，建立的模块化架构和规范的接口设计，为后续的功能扩展提供了良好的基础。

2．目标达成度

从项目整体来看，我们不仅完成了原计划的所有任务，还超额完成了知识图谱相

关的功能。原本计划在暑期实践中实现的动态知识图谱创建功能，在正常学期内就已经完成。唯一的小遗憾是在搜索功能中直接跳转到视频时间戳的功能，我们最终改为在 AI 对话中实现，目前搜索接口使用视频标题来进行匹配。

核心功能的实现程度都达到甚至超过了预期。教学内容预处理模块不仅能够处理视频内容，还实现了智能分段和关键信息提取。AI 总结模块提供了多层次的总结服务，从全文概览到分段详解都有涵盖。AI 对话模块实现了多场景的智能问答，支持不同粒度的知识检索。知识图谱构建模块成功将碎片化的知识点组织成结构化的图谱，提升了学习效率。

四、问题与改进

1．技术层面

回顾整个开发过程，还有一些技术问题值得在未来项目中改进。首先是视频存储方案的问题。由于开发成本的考虑，我们没有使用云对象存储，导致视频文件同步比较困难，最终还出现了服务器存储空间不足的问题。未来的实训项目中，考虑使用阿里云 OSS 或腾讯云 COS 等云存储服务。

数据库选择方面也有优化空间。知识图谱的存储没有使用 Neo4J 等专用图数据库，而是直接用 MySQL 存储，这在数据量增大后可能会遇到性能瓶颈。未来考虑根据实际需求选择更合适的图数据库。

OCR 模型的选择也可以进一步优化。虽然现有的 CNOCR（飞桨 V3）模型在大多数场景下表现良好，但在某些特殊字体或复杂背景下还有改进空间，可以考虑使用更先进的 OCR 模型。

个人学习计划方面，我准备深入学习更多 RAG 相关的知识，特别是在召回质量和效率优化方面。同时计划学习优秀的多模态材料 RAG 框架，为未来处理更复杂的教学内容做准备。

2．协作层面

团队协作整体比较顺利，但也发现了一些可以改进的地方。进度跟踪不够直观是

一个比较明显的问题，虽然通过 QQ 群能够及时沟通，但缺乏对整体项目进度的可视化管理。有时候需要询问才能了解其他模块的开发进度，这在项目规模更大时可能会成为瓶颈。

另一个问题是任务分配的细化程度还可以提升。虽然大的模块分工比较清楚，但在具体的开发任务上，有时候边界不够明确，导致某些功能的集成需要额外的协调工作。

未来的改进策略包括：引入更正式的项目管理工具，建立定期的进度汇报机制，制定更详细的任务分解和时间节点控制。同时加强代码集成的规范性，减少因为接口不匹配导致的返工。

3．项目管理

工具使用方面，我们主要依赖 Git 进行版本控制，QQ 群进行日常沟通。这种轻量级的组合在小团队开发中比较高效，能够快速响应问题和变更。Git 的分支管理和版本控制功能很好地支持了并行开发，QQ 群的即时通讯特性保证了沟通的时效性。

但这种方案也有明显的局限性。缺乏对项目整体进度的量化管理，难以进行风险预警和资源调配。任务分配和跟踪主要依赖人工记忆，在项目复杂度增加时容易出现遗漏。代码 review 的流程还不够规范，主要依赖开发者的自觉性。

下次项目开发我倾向于使用 Scrum（Leangoo）等专业的项目管理工具。这些工具能够提供更直观的进度可视化，更规范的任务管理流程，以及更完善的团队协作机制。特别是在多人协作的复杂项目中，专业工具的优势会更加明显。

五、课程反思

1．课程收获

这门软件工程课程给我带来了全方位的成长，不仅仅是技术技能的提升，更重要的是软件工程思维的建立。

在知识体系构建方面，我对软件工程的整个生命周期有了深入的理解。从需求分

析、系统设计、编码实现到测试部署，每个阶段都有其特定的方法论和最佳实践。特别是在需求分析阶段，学会了使用 KJ 法等结构化的方法来分析和归类需求，这比之前凭感觉做需求分析要科学得多。在系统设计方面，学会了运用各种设计模式和架构思想，让代码更加规范和可维护。在测试方面，以前我关注的更多是接口测试（逻辑、是否能 SQL 注入等），主要从安全和技术的角度出发，学习了软件工程后，我了解了还有很多测试种类，像系统测试、冒烟测试等，这些测试更多的是从工程的角度构建的。

我的能力提升体现在多个维度。硬技能方面，Python 开发能力有了质的飞跃，特别是在 AI 相关技术栈的应用上，我学习了大语言模型时代的基础架构——Langchain 框架。后端方向上，Flask 框架的使用、数据库设计、API 接口开发、多线程编程等技能都在项目中得到了充分的锻炼。软技能方面，项目管理和团队协作能力得到了很大提升。作为组长，学会了如何制定项目计划、分配任务、协调资源，以及如何在技术选择和时间压力之间找到平衡。

思维模式的转变可能是最重要的收获。以前写代码更多是实现功能就行，现在会从软件工程的角度去思考问题。比如在设计模块时会考虑单一职责原则，在写代码时会考虑可维护性和可扩展性，在测试时会考虑覆盖率和边界条件。这种工程化的思维方式不仅适用于编程，对其他工作也很有帮助。

另外一个重要的收获是对 AI 技术在实际工程中应用的理解。通过这个项目，我深刻体会到 AI 不是万能的，需要结合具体的业务场景和工程实践，才能发挥真正的价值。如何设计合理的数据流程、如何处理 AI 的不确定性、如何优化 AI 服务的性能等，这些都是书本上学不到的宝贵经验。

团队协作方面的成长也很明显。学会了如何在保持团队和谐的同时推进项目进度，如何处理技术分歧，如何平衡个人贡献和团队目标。特别是作为组长，需要在技术深度和管理广度之间找到平衡，这对个人的综合能力是很好的锻炼。

2. 课程改进建议

考核方式方面，我建议在半开卷的前提下，能否增加更多实际分析题的比重，相应减少纯概念题的数量。软件工程更多是一门实践性的学科，死记硬背的概念在实际

工作中用处有限，而分析和解决实际问题的能力才是最重要的。比如可以给出一个真实的项目场景，让学生分析需求、设计架构、选择技术方案等，这样的考核方式更能体现学生的真实水平。

实践项目方面，这学期的华为云活动非常好，希望课程组能为学弟学妹争取更多这样的合作机会。云服务、AI 接口等资源对学生项目的质量提升很有帮助，但这些服务的费用对学生来说往往是负担。如果能有更多企业提供教育支持，学生就能接触到更先进的技术和工具，做出更有实用价值的项目。

另外建议增加一些行业前沿技术的介绍，比如微服务架构、云原生应用等。在当前的技术环境下，了解这些趋势对我们的职业发展很有帮助。

这门课程给我带来了很大的收获，不仅学到了知识和技能，更重要的是培养了工程思维和解决问题的能力，在此祝愿我们院的软件工程课程能不断创新，蒸蒸日上！