에이전트 기반 교육 콘텐츠 생성 서비스를 위한 프레임워크 비교 분석 및 아키텍처 설계 제안서

제 1 부: 교육 콘텐츠 생성 워크플로우의 해체

에이전트 기반 서비스 개발을 위한 최적의 프레임워크를 선정하기에 앞서, 해결하고자 하는 문제의 본질을 명확히 정의하는 것이 필수적입니다. 본 보고서에서 다루는 '교육 콘텐츠 생성 서비스'는 단순한 단일 태스크가 아닌, 상호 의존적인 여러 단계로 구성된 복합적인 워크플로우입니다. 이 워크플로우를 구성하는 핵심 태스크들을 기술적, 교육학적 관점에서 심도 있게 분석함으로써, 프레임워크 평가를 위한 명확하고 구체적인 기준을 수립할 수 있습니다.

1.1. 기본 태스크: 교육학적 수업 계획(Lesson Plan) 생성

핵심 과제

수업 계획은 단순한 텍스트 덩어리가 아니라, 명확한 구조와 교육학적 논리를 갖춘 문서입니다. 따라서 첫 번째 핵심 과제는 주어진 주제에 대해 교육학적으로 타당하고 일관성 있는 수업 계획을 생성하는 것입니다. 이 계획은 후속 태스크의 신뢰성 있는 입력값으로 사용되어야 하므로, 그 품질과 구조적 완결성이 전체 시스템의 성패를

좌우합니다.

태스크 분해

교육 이론과 실제 구현 사례에 따르면 [1], 성공적인 수업 계획 생성 태스크는 다음과 같은 하위 요소들로 분해될 수 있습니다.

- **명확한 학습 목표(Learning Objectives) 정의:** 학생이 수업을 통해 무엇을 배우고 성취해야 하는지를 구체적으로 명시합니다.
- **교수 및 도입(Teaching and Initiation, TI) 활동 설계:** 학습 목표 달성을 위한 교사의 설명, 강의, 시연 등의 활동과 학생의 참여를 유도하는 도입부를 구성합니다.[1]
- **심층 토론(In-depth Discussion, ID) 요점 및 질문 구성:** 개념 이해를 돕기 위한 심화 토론 주제, 질의응답 항목을 구조화합니다.[1]
- **평가 및 피드백 메커니즘 제안:** 학생의 학습 성취도를 평가하고 성장을 촉진할 수 있는 구체적인 피드백 방법을 제시합니다.[2]
- **핵심 어휘 및 보조 자료 식별:** 수업에 필요한 주요 용어와 참고 자료(예: 동영상, 이미지, 참고 문헌) 목록을 정의합니다.[3]

구조화된 출력(Structured Output)의 필요성

후속 에이전트가 생성된 수업 계획을 안정적으로 사용하기 위해서는, 그 출력이 예측 가능하고 기계가 읽을 수 있는(machine-readable) 형식이어야 합니다. 예를 들어, JSON 객체나 Pydantic 모델과 같은 스키마를 따르는 출력이 필수적입니다. 이는 LLM 응답의 신뢰성을 높이고, 종종 발생하는 할루시네이션(hallucination)을 줄이며, 후속 애플리케이션에서 데이터를 즉시 활용할 수 있게 합니다.[5] 따라서 프레임워크가 구조화된

출력을 얼마나 안정적으로, 그리고 쉽게 생성할 수 있는지가 핵심 평가 기준이 됩니다.[6]

계획(Planning) 및 추론(Reasoning) 능력의 필요성

수업 계획의 각 구성 요소(학습 목표, 활동, 평가 등)는 논리적인 순서로 배열되어야 합니다. 에이전트는 단순히 각 요소를 나열하는 것을 넘어, 학습 목표에서 시작하여 평가로 끝나는 일관된 흐름을 계획하고 추론할 수 있어야 합니다. 이는 프레임워크가 명시적인 계획 수립 기능이나 연쇄적 사고(Chain-of-Thought) 추론을 지원해야 함을 시사합니다.[2]

1.2. 계획에서 발표로: 슬라이드 콘텐츠 생성

핵심 과제

이 태스크의 핵심은 앞서 생성된 구조화된 수업 계획을 시각적 매체인 프레젠테이션 슬라이드에 적합한 콘텐츠로 변환하는 것입니다. 이는 단순한 복사-붙여넣기가 아닌, 정교화(elaboration), 요약(summarization), 그리고 형식화(formatting)의 과정입니다.

주요 하위 태스크

- **콘텐츠 정교화:** 수업 계획의 각 요점을 학생들이 이해하기 쉬운 상세한 설명으로 확장합니다.

- **요약 및 청킹(Chunking):** 복잡하고 긴 정보를 슬라이드에 적합한 간결한 글머리 기호(bullet point)나 짧은 문단으로 요약하고 분할합니다.[8]
- **다중 모드(Multi-modal) 통합:** 텍스트 기반 학습을 넘어 학습 효과를 높이기 위해 이미지, 다이어그램, 차트 등의 시각 자료 삽입을 제안하는 기능이 필요합니다. 이는 현대 교육 도구의 핵심 트렌드이며 [9], 에이전트 프레임워크가 다중 모드 모델이나 관련 도구와 원활하게 통합될 수 있어야 함을 의미합니다.

상태 의존성(Stateful Dependency)

슬라이드 콘텐츠 생성은 전적으로 수업 계획 생성 태스크의 출력에 의존합니다. 따라서 전체 워크플로우는 이전 단계의 상태(완성된 수업 계획)가 다음 단계의 에이전트에게 불변적(immutable)이고 접근 가능한 형태로 전달되는 것을 보장해야 합니다.

1.3. 개인화 벡터: 교사 맞춤형 콘텐츠 생성

핵심 과제

이 단계는 일반적인(generic) 콘텐츠를 특정 교사에게 최적화된 개인화된 콘텐츠로 변환하는 과정입니다. 이는 전체 워크플로우에서 가장 복잡한 태스크로, 에이전트 시스템이 교사의 고유한 자료, 교수 스타일, 그리고 교육적 맥락을 통합하는 능력을 요구합니다.

검색 증강 생성(RAG)의 핵심 역할

개인화의 주요 기술적 수단은 검색 증강 생성(Retrieval-Augmented Generation, RAG)입니다. 프레임워크는 교사의 개인 데이터(예: 과거 강의 노트, PDF, 교육 과정 문서)를 효율적으로 수집(ingest)하고 인덱싱(indexing)하여, 이를 기반으로 콘텐츠 생성을 증강하는 데 탁월해야 합니다.[2]

메모리 및 컨텍스트 관리

시스템은 여러 세션에 걸쳐 교사의 선호도와 맥락을 기억해야 합니다. 예를 들어, 특정 교사가 특정 산업 분야와 관련된 예시를 선호한다면, 에이전트는 이 정보를 기억하고 향후 콘텐츠 생성에 적용해야 합니다. 이를 위해서는 강력한 단기 및 장기 메모리 기능이 필수적입니다.[2] 학술 연구에서도 LLM 에이전트가 "학생의 학습 습관에 대한 장기 지식과 실시간 상호작용의 단기 컨텍스트를 모두 유지"하여 "개인화된 학습 경험"을 만드는 능력이 강조됩니다.[2]

LangChain-Teacher 와 같은 실제 프로젝트는 채팅 메모리를 사용하여 명시적으로 컨텍스트를 유지합니다.[22]

제 1 부의 핵심 결론 및 시사점

워크플로우 분석을 통해 도출된 몇 가지 중요한 시사점은 후속 프레임워크 평가의 방향을 결정합니다.

첫째, **이 워크플로우는 본질적으로 순차적이며 상태 의존적입니다.** 수업 계획이 없으면

슬라이드 콘텐츠를 생성할 수 없고, 콘텐츠가 없으면 개인화를 수행할 수 없습니다. 이는 각 단계에서 생성된 결과물(상태)이 다음 단계로 안정적으로 전달되어야 함을 의미합니다. 따라서 자유로운 대화형 상호작용을 위해 설계된 프레임워크보다, 상태 기계(state machine)나 유향 그래프(directed graph)처럼 명시적인 상태 관리와 결정론적 제어 흐름을 제공하는 프레임워크가 구조적으로 더 유리합니다.

둘째, **이 프로젝트는 생성(generation), RAG, 그리고 계획(planning)이 결합된 하이브리드 시스템입니다.** 단순히 챗봇이나 RAG 시스템을 구축하는 것이 아니라, 각기 다른 에이전트 기능이 각 단계에서 요구되는 통합 시스템을 만드는 것입니다. 1 단계(수업 계획)는 계획과 구조화된 생성에, 2 단계(슬라이드 콘텐츠)는 요약과 변환에, 3 단계(개인화)는 RAG 와 메모리에 중점을 둡니다. 성공적인 프레임워크는 이러한 기능들을 개별적으로 제공하는 것을 넘어, 단일하고 일관된 워크플로우 내에서 우아하게 조합할 수 있어야 합니다.

셋째, **다중 에이전트 아키텍처가 가장 자연스러운 해결책입니다.** 각 태스크에 요구되는 뚜렷하게 구분되는 기술 집합(계획가, 작가, 검색 전문가)은 전문화된 에이전트 팀 구성에 직접적으로 부합합니다. 교육 AI 분야의 연구들은 "수석 튜터", "연습문제 설계자", "학습 평가자"와 같은 전문화된 역할을 가진 다중 에이전트 시스템을 명시적으로 사용합니다.[1] AutoGen 이나 CrewAI 와 같은 프레임워크는 바로 이러한 다중 에이전트 협업이라는 전제 위에 구축되었습니다.[11] 따라서 프레임워크 평가는 단순히 태스크 수행 가능 여부를 넘어, 문제를 여러 협력 에이전트로 얼마나 잘 분해하고 지원하는지를 고려해야 합니다.

제 2 부: 프레임워크 심층 분석

제 1 부에서 도출된 평가 기준에 따라, 네 가지 주요 에이전트 프레임워크(LangChain/LangGraph, LlamaIndex, Microsoft AutoGen, CrewAI) 각각의

아키텍처, 핵심 기능, 그리고 교육 콘텐츠 생성 워크플로우에 대한 적합성을 심층적으로 분석합니다.

2.1. LangChain & LangGraph: 오케스트레이션 강자

2.1.1. 핵심 아키텍처

LangChain 은 LLM 애플리케이션을 구성하기 위한 모듈식 프레임워크로 시작했습니다.[24] 그 핵심적인 진화는 LangGraph 의 등장입니다. LangGraph 는 그래프 기반 표현을 사용하여 상태 저장(stateful), 다중 행위자(multi-actor) 애플리케이션을 구축하기 위한 라이브러리입니다.[27] 이 그래프에서 노드(node)는 함수나 에이전트를 나타내고, 엣지(edge)는 제어의 흐름을 정의합니다. 이를 통해 순환(cycle)을 포함한 복잡하고 장기 실행되는 워크플로우를 설계할 수 있습니다.[30] 이러한 아키텍처는 제 1 부에서 분석한 상태 의존적이고 다단계적인 문제의 본질에 직접적으로 부합합니다.

2.1.2. 계획 및 오케스트레이션

LangGraph 는 워크플로우에 대한 명시적이고 세분화된 제어를 제공합니다. 개발자는 노드 간에 전달되는 공유 메모리인 상태 객체(예: AgentState [32])를 직접 정의합니다. 이는 수업 계획 파이프라인에 이상적인 방식입니다. 특히 조건부 엣지(conditional edge)를 생성하는 기능은 동적 라우팅을 가능하게 합니다. 예를 들어, '평가자(evaluator)' 노드를 두어 생성된 계획의 품질이 기준에 미치지 못할 경우, 수정을 위해 이전 노드로 되돌려 보내는 로직을 구현할 수 있습니다.[7] 이러한 '계획가(Planner) -> 실행가(Executor) ->

평가자(Evaluator)' 패턴은 LangGraph 에서 공식화된 모범 사례입니다.[7]

2.1.3. 도구 사용 및 구조화된 출력

LangChain 은 방대한 도구 통합 생태계를 자랑합니다.[25] 특히

.with_structured_output() 메서드를 통해 Pydantic 모델이나 JSON 스키마를 사용하는 구조화된 출력 생성을 기본적으로 지원하는 것이 큰 강점입니다.[5] 이 기능은 기계가 읽을 수 있는 수업 계획을 안정적으로 생성하는 데 매우 중요합니다.

2.1.4. 메모리 관리

LangGraph 는 정교하고 명시적인 메모리 개념을 가지고 있습니다. 단기 메모리(스레드 범위의 상태), 장기 메모리(세션 간 공유), 의미론적 메모리(사실), 일화적 메모리(경험), 절차적 메모리(규칙)를 구분하여 관리합니다.[15] 또한, 체크포인터(checkpointer) 메커니즘은 상태를 영속적으로 저장하여, 장시간 실행되는 콘텐츠 생성 프로세스를 중단했다가 다시 시작할 수 있게 해줍니다.[28]

2.2. LlamaIndex: 데이터 중심 에이전트 프레임워크

2.2.1. 핵심 아키텍처

LlamaIndex 는 RAG 를 위한 데이터 프레임워크로 시작하여 데이터 수집, 인덱싱, 검색 분야에서 탁월한 성능을 보입니다.[12] 이후 에이전트 기능을 추가하며 "컨텍스트 증강 LLM 애플리케이션(Context-Augmented LLM Applications)"을 위한 프레임워크로 발전했습니다.[36] 그 아키텍처는 데이터 커넥터, 인덱스, 그리고 쿼리 엔진을 중심으로 구축되어 있습니다.[36]

2.2.2. 계획 및 오케스트레이션

LlamaIndex 는 여러 단계를 연결하기 위한 이벤트 기반 추상화인 Workflows 를 도입했습니다.[36] 워크플로우는

@step 데코레이터가 붙은 함수들로 구성되며, 각 스텝은 특정 이벤트를 처리하고 새로운 이벤트를 발생시킵니다.[40] 이는 복잡한 RAG 흐름이나 ReAct 에이전트와 같은 에이전트 프로세스를 구축하는 방법을 제공합니다.[40] 이는 강력한 기능이지만, 핵심 데이터 처리 기능에 비해 상대적으로 최근에 추가된 기능입니다.

2.2.3. 도구 사용 및 데이터 처리

이 분야는 LlamaIndex 의 독보적인 강점입니다. LlamaHub 를 통해 160 개 이상의 데이터 소스를 지원하는 데이터 커넥터와 벡터, 트리, 키워드, 지식 그래프 등 고급 인덱싱 전략을 제공합니다.[8] 특히

LlamaParse 도구는 표와 이미지가 포함된 복잡한 PDF 문서를 파싱하는 데 특화되어 있어, 교사의 개인 자료를 수집하는 데 매우 유용합니다.[36]

2.2.4. 메모리 관리

LlamaIndex 는 단기 메모리(토큰 제한 기반 채팅 기록)와 장기 메모리 블록을 모두 지원하는 Memory 모듈을 제공합니다.[13] 특히

StaticMemoryBlock(변하지 않는 정보용)이나 FactExtractionMemoryBlock(대화에서 자동으로 사실을 추출)과 같은 독특한 장기 메모리 블록은 교사 맞춤형 콘텐츠 생성에 매우 유용할 수 있습니다.[13]

2.3. Microsoft AutoGen: 대화형 협업 엔진

2.3.1. 핵심 아키텍처

AutoGen 은 다중 에이전트 간의 협업을 "대화(conversation)"로 모델링하는 프레임워크입니다.[24] 에이전트는 태스크를 해결하기 위해 메시지를 주고받는 "대화 가능한(conversable)" 개체입니다.[45] 핵심적인 오케스트레이션 메커니즘은 대화의 흐름을 관리하는

GroupChat 또는 GroupChatManager 입니다.[47]

2.3.2. 태스크 분해 및 오케스트레이션

AutoGen 에서 복잡한 태스크는 대화를 통해 동적으로 분해됩니다. UserProxyAgent 가

태스크를 시작하면, GroupChatManager 는 대화 기록과 에이전트의 역할을 바탕으로 "다음 발언자(next speaker)"를 결정합니다.[48] 이는 해결 경로가 정해지지 않은 창발적인(emergent) 문제 해결에 탁월합니다. 하지만 본 프로젝트처럼 구조화된 워크플로우의 경우, 이러한 방식이 원치 않는 비결정론(non-determinism)을 야기할 수 있습니다. 물론 유한 상태 기계(Finite State Machines)와 같은 패턴을 적용하여 제약을 가할 수는 있습니다.[46]

2.3.3. 도구 사용

에이전트는 함수 형태의 도구를 장착할 수 있습니다. 한 에이전트(예: AssistantAgent)가 도구 사용을 제안하고, 다른 에이전트(예: UserProxyAgent)가 이를 실행하는 패턴은 관심사의 분리(separation of concerns)와 인간 참여형(human-in-the-loop) 승인 메커니즘을 자연스럽게 구현합니다.[44] 도구를 위한 확장(extension) 생태계도 성장하고 있습니다.[43]

2.3.4. 메모리 관리

AutoGen 의 메모리는 본질적으로 대화 기록과 밀접하게 연결되어 있습니다.[46] 최근 버전에서는 Mem0 과 같은 외부 메모리 시스템과의 통합을 통해 세션 간 영속성과 RAG 패턴을 지원하는 등 보다 명시적인 메모리 기능이 추가되었습니다.[18]

ListMemory 와 RAG 패턴을 통해 에이전트는 대화 중에 관련 정보를 검색할 수 있습니다.[18]

2.4. CrewAI: 역할 기반 프로세스 프레임워크

2.4.1. 핵심 아키텍처

CrewAI 는 역할 놀이(role-playing)를 하는 자율 AI 에이전트들을 오케스트레이션하기 위해 설계된 고수준 프레임워크입니다.[57] 이 프레임워크는 특정 역할(role), 목표(goal), 배경 이야기(backstory)를 가진

Agents, 개별 Tasks, 그리고 정의된 Process 에 따라 이를 실행하는 Crew 라는 개념을 중심으로 구축되었습니다.[11] LangChain 과는 독립적으로 개발된 점이 특징입니다.[57]

2.4.2. 계획 및 오케스트레이션

CrewAI 는 두 가지 주요 프로세스 모델을 제공합니다: Sequential(순차적)과 Hierarchical(계층적).[58]

Sequential 프로세스는 정해진 순서대로 태스크를 실행하므로, 본 프로젝트의 첫 두 단계에 직접적으로 부합합니다. Hierarchical 프로세스는 태스크를 계획, 위임, 검토하는 "관리자(manager)" 에이전트를 도입하여 더 복잡한 협업을 가능하게 합니다.[21] 태스크 위임(delegation)은 CrewAI 의 핵심 개념으로, 에이전트에게 특정 동료에게 작업을 위임할 수 있는 권한을 부여할 수 있습니다.[61]

2.4.3. 도구 사용

에이전트는 사용자 정의 도구는 물론 LangChain 및 LlamaIndex 의 도구를 포함한 광범위한 도구를 장착할 수 있습니다.[63] 프레임워크는

BaseTool 을 상속하여 사용자 정의 도구를 만드는 명확한 구조를 제공합니다.[64]

2.4.4. 메모리 관리

CrewAI 는 memory=True 옵션으로 활성화할 수 있는 정교한 메모리 시스템을 갖추고 있습니다.[21] 단기(현재 컨텍스트를 위한 RAG 기반), 장기(세션 간 통찰력을 위한 SQLite 기반), 그리고 엔티티 메모리를 포함합니다.[21] 그러나 문서와 커뮤니티 논의에 따르면, 대화 기록이나 다중 사용자 컨텍스트를 관리하기 위해서는 내장 메모리를 외부 추적 시스템으로 보완하는 등 신중한 구현이 필요할 수 있습니다.[20]

제 2 부의 핵심 결론 및 시사점

프레임워크 분석을 통해 두 가지 중요한 점이 드러납니다.

첫째, **프레임워크들은 서로 다른 출발점에서 시작했지만 유사한 개념으로 수렴하고 있습니다.** LangChain 은 체인(chain)의 한계를 느끼고 LangGraph 라는 상태 기계를 만들었습니다. LlamaIndex 는 데이터에서 시작하여 그 데이터를 활용할 방법을 모색하다 에이전트와 워크플로우를 구축했습니다. AutoGen 은 대화형 에이전트에서 시작하여 대화에 구조가 필요함을 깨닫고 FSM 과 같은 패턴을 추가했습니다. CrewAI 는 '팀'이라는 고수준 개념에서 시작하여 팀에 프로세스가 필요함을 인지하고 순차/계층 모드를 만들었습니다. 이는 대부분의 태스크가 어떤 프레임워크로든 구현될 수 있음을 의미하지만, 가장 '자연스럽고(idiomatic)' '쉬운' 방법은 프레임워크의 핵심 철학이

문제의 철학과 얼마나 일치하는지에 따라 극명하게 달라질 것임을 시사합니다.

둘째, **추상화 수준(level of abstraction)과 제어 수준(level of control) 사이의 트레이드오프가 핵심적인 주제입니다.** CrewAI 는 가장 높은 수준의 추상화를 제공합니다. '역할'과 '배경 이야기'로 에이전트를 정의하는 것은 개념적으로 매우 간단합니다.[57] 반면, LangGraph 는 가장 낮은 수준의 추상화와 가장 높은 수준의 제어를 제공합니다. 개발자는 모든 상태 전이와 노드 함수를 명시적으로 정의해야 합니다.[28] AutoGen 과 LlamaIndex 는 그 중간에 위치합니다. 본 프로젝트처럼 특정하고 신뢰성 높으며 잠재적으로 복잡한 워크플로우를 위해서는 CrewAI 의 빠른 프로토타이핑 능력보다 LangGraph 의 높은 제어 수준이 더 가치 있을 수 있습니다.

제 3 부: 비교 분석 및 유스케이스 매핑

이 장에서는 각 프레임워크를 서로 직접 비교하고, 교육 콘텐츠 생성 태스크의 특정 요구사항에 맞춰 그 적합성을 평가합니다.

3.1. 아키텍처 및 철학적 차이점

네 가지 프레임워크는 근본적으로 다른 오케스트레이션 모델을 채택하고 있으며, 이는 문제 해결 방식에 직접적인 영향을 미칩니다.

- LangGraph: 결정론적 상태 기계 (Deterministic State Machine)
  제어는 명시적이며 그래프 기반입니다. 상태 객체가 노드 사이를 이동하며, 엣지는 다음 단계를 결정합니다. 이는 예측 가능하고 감사 가능하며 복잡한 파이프라인에 가장 적합합니다.30 워크플로우가 사전에 정의된 구조를 따를 때 강력한 힘을

발휘합니다.

- AutoGen: 비결정론적 대화 모델 (Non-deterministic Conversational Model)
  제어는 대화의 흐름에서 창발적으로 나타납니다. 에이전트들은 메시지를 교환하며 다음 행동을 결정합니다. 이는 해결 경로가 미리 알려지지 않은 동적 문제 해결에 가장 적합합니다.11
- CrewAI: 위임 기반 태스크 계층 (Delegated Task Hierarchy)
  제어는 에이전트의 역할과 미리 정의된 프로세스(순차적/계층적)에 기반합니다. 이는 인간 팀의 구조를 모방하는 시스템을 신속하게 개발하는 데 가장 적합합니다.24 추상화 수준이 높아 개발 속도가 빠릅니다.
- LlamaIndex: 이벤트 기반 데이터 흐름 모델 (Event-driven Data-Flow Model)
  제어는 이벤트가 발생하고 스텝에 의해 처리되는 방식에 기반합니다. 이는 근본적으로 데이터 처리 및 검색을 중심으로 하는 워크플로우에 가장 적합합니다.40 데이터 파이프라인 구축에 강점을 보입니다.

3.2. 프레임워크 기능 비교 매트릭스

아래 표는 제 2 부의 분석을 요약하여 각 프레임워크의 핵심 기능을 명확하고 구조적으로 비교합니다. 이 표는 기술 리더십이 각 프레임워크의 강점과 약점을 한눈에 파악하고, 직접적인 트레이드오프를 이해하여 정보에 입각한 결정을 내리는 데 도움을 주기 위해 설계되었습니다.

기능	LangChain / LangGraph	LlamaIndex	Microsoft AutoGen	CrewAI
핵심 추상화	그래프 / 상태 기계 [28]	데이터 / 워크플로우 [36]	대화 / 메시지 [45]	크루 / 프로세스 [57]

계획 및 오케스트레이션	매우 높은 제어 수준, 결정론적, 순환/분기 완벽 지원 [7]	이벤트 기반, 데이터 흐름 중심, 워크플로우로 복잡한 로직 구현 [40]	대화 기반, 동적, 비결정론적, FSM 으로 제약 가능 [46]	고수준 추상화, 순차/계층 프로세스, 역할 기반 위임 [59]
태스크 분해	명시적 계획(Planner) 에이전트 패턴을 통해 사전 정의 [7]	쿼리 계획(Sub-Questions) 및 워크플로우 스텝으로 분해 [41]	대화를 통한 동적 분해, GroupChatManager 가 다음 역할 결정 [48]	계층적 프로세스에서 관리자 에이전트가 위임 [21]
도구 통합 및 사용	가장 큰 생태계, Pydantic 기반 쉬운 사용자 정의, 명시적 실행 [25]	LlamaHub 를 통한 방대한 데이터 커넥터, LlamaParse 등 강력한 내장 도구 [8]	제안(Assistant) 과 실행(Proxy) 분리 패턴, 성장하는 확장 생태계 [44]	LangChain/LlamaIndex 도구 통합 가능, BaseTool 상속으로 쉬운 사용자 정의 [63]
메모리 및 상태 관리	명시적 상태 객체, 체크포인터를 통한 영속성, 다양한 메모리 유형 지원 [15]	단기/장기 메모리, FactExtraction 등 독특한 메모리 블록, RAG 에 최적화 [13]	대화 기록 중심, Mem0 등 외부 메모리 통합으로 영속성 확보 [18]	단기/장기/엔티티 메모리, RAG 통합, 다중 사용자 컨텍스트는 추가 구현 필요 [20]
RAG 및 데이터 처리	다양한 통합 제공, LangChain Expression Language(LCEL)로 유연한 파이프라인 구축 [25]	업계 최고 수준. LlamaParse, 고급 인덱싱 전략 등 데이터 처리에 독보적 [12]	RetrieveUserProxyAgent 등 RAG 패턴 지원, 커뮤니티 확장 기능 활용 [69]	RAG 도구 통합 가능, 내장 메모리 시스템이 RAG 활용 [21]

구조화된 출력 지원	Pydantic/JSON 스키마를 통한 네이티브 지원, 신뢰성 높고 사용 용이 [5]	LlamaExtract 등 구조화된 데이터 추출 도구 제공, 워크플로우 출력 제어 [36]	JSON 모드 지원, 응답 형식을 위한 프롬프팅 필요, 추가 파싱 필요 가능성 [70]	태스크의 expected_output 필드로 출력 형식 가이드, 스키마 강제는 약함 [71]
생태계 및 커뮤니티	가장 성숙하고 규모가 큼, 방대한 문서와 예제, 활발한 커뮤니티 [24]	데이터 중심 커뮤니티, LlamaHub 를 통한 강력한 확장성, 빠르게 성장 중 [42]	Microsoft 지원, 엔터프라이즈 중심, 연구 기반 커뮤니티 [30]	빠르게 성장하는 커뮤니티, 사용자 친화적, 역할 기반 설계로 인기 [24]
확장성 및 프로덕션	LangGraph Platform 등 배포 도구, 엔터프라이즈에서 검증된 사용 사례 [27]	LlamaCloud 등 관리형 서비스, KPMG 등 엔터프라이즈에서 신뢰 [36]	엔터프라이즈 준비 완료, 분산 에이전트 지원, 확장성 우수 [30]	프로토타이핑에 강점, 단순한 use case 는 확장 용이, 복잡한 경우 추가 노력 필요 [30]
학습 곡선 및 사용 편의성	LangGraph 는 제어 수준이 높아 학습 곡선이 가파름 [12]	데이터 중심 접근으로 RAG 구축은 용이, 에이전트 워크플로우는 학습 필요 [12]	개념이 독특하여 학습 필요, 문서와 예제가 풍부함 [30]	가장 사용자 친화적, 고수준 추상화로 빠른 프로토타이핑 가능 [30]

3.3. 프레임워크의 교육 태스크 적합성 매핑

수업 계획 생성

이 태스크는 강력한 계획 수립 능력과 신뢰성 있는 구조화된 출력을 요구합니다. **LangGraph** 는 명시적인 상태 관리와 스키마 강제 능력 덕분에 이 분야에서 가장 뛰어납니다. **CrewAI** 의 Hierarchical 모드에서 '계획가' 에이전트를 사용하는 것도 강력한 대안입니다. **AutoGen** 은 대화적 특성 때문에 일관된 구조의 결과물을 생성하는 데 있어 신뢰성이 다소 떨어질 수 있습니다.

슬라이드 콘텐츠 생성

이는 수업 계획이라는 상태를 입력받아 변환하는 비교적 직선적인 태스크입니다. 모든 프레임워크가 이 작업을 처리할 수 있지만, 핵심은 상태가 얼마나 깨끗하게 전달되느냐입니다. **LangGraph** 와 **CrewAI(Sequential)**는 바로 이러한 선형적인 상태 전달 워크플로우를 위해 설계되었습니다.

교사 맞춤형 콘텐츠 생성

이것은 RAG 중심의 태스크입니다. **LlamaIndex** 는 기반이 되는 데이터 처리(수집, 인덱싱, 검색)에서 논쟁의 여지가 없는 선두 주자입니다.[12] 여기서 중요한 아키텍처적 결정은 LlamaIndex 를 주 프레임워크로 사용할 것인지, 아니면 다른 프레임워크 내에서 전문화된 도구로 활용할 것인지입니다(예: LangGraph 나 CrewAI 에이전트 내에서 LlamaIndex 기반 쿼리 엔진을 도구로 사용).[31]

제 3 부의 핵심 결론 및 시사점

비교 분석을 통해 두 가지 중요한 아키텍처적 고려사항이 명확해졌습니다.

첫째, **'최고의 RAG 프레임워크'가 반드시 '최고의 오케스트레이션 프레임워크'는 아니라는 점입니다.** LlamaIndex 는 개인화 태스크의 데이터 수집 및 검색 부분에서 우월하지만 [31], 전체 프로젝트는 단순한 RAG 애플리케이션이 아닌 복잡한 다단계 *워크플로우*입니다. 반면 LangGraph 는 상태 저장 및 순환이 가능한 복잡한 워크플로우를 오케스트레이션하는 데 더 뛰어납니다.[28] 따라서 핵심적인 아키텍처 결정은 동급 최고의 RAG 도구를 기반으로 할 것인지, 아니면 동급 최고의 오케스트레이션 도구를 기반으로 할 것인지의 문제입니다. 프레임워크들 스스로도 서로를 통합하는 기능을 제공함으로써 이 점을 인정하고 있습니다.[31]

둘째, **'계획가-실행가(Planner-Executor)' 패턴이 이러한 유형의 문제에 대한 지배적이고 검증된 접근 방식이라는 점입니다.** 사용자의 워크플로우는 하나의 계획(수업 계획)과 그에 따른 실행(슬라이드 생성, 개인화)으로 모델링될 수 있습니다. LangChain/LangGraph 문서는 이 패턴을 명시적으로 지지하며 [7], CrewAI 의 계층적 모드는 관리자 에이전트를 계획가로 두어 이를 직접 구현합니다.[59] AutoGen 예제들 또한 관리자나 사용자 프록시가 다른 에이전트들이 실행할 계획을 시작하는 유사한 구조를 보여줍니다.[53] 이러한 수렴은 선택된 프레임워크가 이 아키텍처 패턴을 일급 기능으로 지원해야 함을 강력하게 시사합니다.

제 4 부: 아키텍처 설계 및 최종 권고

앞선 모든 분석을 종합하여, 교육 콘텐츠 생성 서비스를 위한 구체적이고 실행 가능한

아키텍처와 프레임워크 권고안을 제시합니다.

4.1. 제안된 다중 에이전트 서비스 아키텍처

본 프로젝트를 위해 다음과 같은 역할을 수행하는 다중 에이전트 팀 또는 그래프 기반 아키텍처를 제안합니다.

- **에이전트 역할 정의:**
  - CurriculumPlannerAgent (교육과정 계획가 에이전트):
    - **역할:** 주어진 주제에 대해 구조화된 수업 계획을 생성합니다.
    - **핵심 도구:** 구조화된 출력(Pydantic/JSON)을 강제하는 LLM 호출.
  - ContentSynthesizerAgent (콘텐츠 종합가 에이전트):
    - **역할:** 구조화된 계획을 입력받아 각 섹션을 상세한 설명과 내용으로 확장합니다.
    - **핵심 도구:** 텍스트 생성 및 요약 LLM 호출.
  - SlideDesignerAgent (슬라이드 디자이너 에이전트):
    - **역할:** 종합된 콘텐츠를 슬라이드 크기에 맞는 글머리 기호로 분할하고, 다중 모드 요소(이미지, 차트 등) 삽입을 제안합니다.
    - **핵심 도구:** 텍스트 청킹 및 다중 모드 모델 API 호출.
  - PersonalizationTunerAgent (개인화 튜너 에이전트):
    - **역할:** RAG 를 기반으로 작동합니다. 일반 슬라이드 콘텐츠와 교사의 프로필/문서 세트를 입력받아 맞춤형 콘텐츠를 생성합니다.
    - **핵심 도구:** 교사의 개인 데이터로 구축된 **LlamaIndex 쿼리 엔진**.
  - QualityEvaluatorAgent (품질 평가자 에이전트):
    - **역할:** (선택 사항이지만 강력히 권장됨) 각 단계의 결과물을 사전에 정의된 평가 기준(루브릭)에 따라 검토합니다. 품질이 미달일 경우, 해당 태스크를

수정하도록 이전 에이전트에게 되돌려 보냅니다. 이는 그래프에서 순환(cycle) 로직을 필요로 합니다.

4.2. 최종 프레임워크 권고: LangChain & LangGraph

본 분석을 통해, 교육 콘텐츠 생성 서비스의 주 오케스트레이션 프레임워크로 **LangChain 과 LangGraph** 를 사용할 것을 최종적으로 권고합니다.

4.3. 권고 정당성 및 구현 전략

LangGraph 를 선택하는 이유는 다음과 같은 명확한 기술적 우위에 근거합니다.

1. **최고 수준의 제어 및 결정론:** 교육 콘텐츠 생성 워크플로우는 자유로운 대화가 아닌 구조화된 프로세스입니다. LangGraph 의 상태 기계 모델은 신뢰성과 일관성을 보장하는 데 필요한 정밀하고 감사 가능한 제어 기능을 제공합니다.[28] 개발자는 워크플로우의 모든 단계를 명시적으로 정의하고 예측할 수 있습니다.
2. **명시적인 상태 관리:** 그래프를 통해 전달되는 명확한 State 객체를 정의하는 기능은 '계획 -> 슬라이드 -> 개인화된 슬라이드'로 이어지는 파이프라인 형태의 태스크에 완벽하게 부합합니다.[32] 각 단계의 산출물이 다음 단계의 입력으로 어떻게 전달되는지 명확하게 관리할 수 있습니다.
3. **하이브리드 아키텍처를 위한 탁월한 유연성:** LangGraph 의 모듈성은 다른 생태계의 동급 최고 도구를 통합하는 데 가장 적합합니다. 예를 들어, PersonalizationTunerAgent 는 LangGraph 의 노드로 구현하되, 그 내부에서는 LlamaIndex 쿼리 엔진을 도구로 호출할 수 있습니다. 이는 '오케스트레이션을 위한 LangGraph'와 'RAG 를 위한 LlamaIndex'라는 두 세계의 장점을 모두 취하는 가장

효과적인 아키텍처입니다.[31]

4. **복잡한 로직(순환) 지원:** QualityEvaluatorAgent 가 태스크를 수정하도록 되돌려 보내는 기능은 순환 워크플로우를 필요로 합니다. LangGraph 는 이러한 순환 그래프를 명시적으로 처리하도록 설계되었으며, 이는 다른 프레임워크에서는 덜 성숙하거나 명시적이지 않은 기능입니다.[30]
5. **성숙도와 생태계:** LangChain 은 가장 큰 생태계를 보유하고 있으며 가장 성숙한 프레임워크입니다. 이는 가장 광범위한 통합 기능과 가장 활발한 커뮤니티 지원을 받을 수 있음을 의미합니다.[24]

4.4. 대안 프레임워크 분석

- **CrewAI 를 선택하지 않는 이유:** 계층적 프로세스는 좋은 대안이지만, LangGraph 보다 세분화된 제어를 제공하지 않습니다. 높은 신뢰성과 QualityEvaluatorAgent 와 같은 맞춤형 로직이 필요한 프로덕션 시스템의 경우, CrewAI 의 높은 추상화 수준이 오히려 제약이 될 수 있습니다.[30] 빠른 프로토타이핑에는 탁월하지만, 이 특정하고 복잡한 워크플로우를 미세 조정하는 데는 덜 이상적입니다.
- **AutoGen 을 선택하지 않는 이유:** 핵심 강점은 동적이고 대화적인 태스크 분해에 있습니다.[11] 이는 본 프로젝트의 고도로 구조화되고 예측 가능한 파이프라인과는 맞지 않습니다. AutoGen 을 사용한다는 것은 비결정론적 패러다임에 결정론적 프로세스를 강제하는 것을 의미하며, 이는 비효율적이고 디버깅을 더 어렵게 만듭니다.
- **LlamaIndex 를 선택하지 않는 이유:** RAG 기능은 타의 추종을 불허하며 필수적이지만, 오케스트레이션 도구인 Workflows 는 LangGraph 에 비해 상대적으로 새롭고 덜 성숙합니다.[40] 가장 효과적인 아키텍처는 LlamaIndex 를 데이터 처리라는 가장 잘하는 역할에 집중시키고, 이를 더 강력하고 유연한

오케스트레이션 셸인 LangGraph 내의
*도구*로 활용하는 것입니다.

제 5 부: 결론

본 보고서는 에이전트 기반 교육 콘텐츠 생성 서비스 개발을 위한 최적의 프레임워크를 선정하기 위해, 요구사항 분석, 주요 프레임워크 심층 비교, 그리고 아키텍처 설계를 수행했습니다. 분석 결과, 해당 서비스의 워크플로우는 본질적으로 **구조적이고, 순차적이며, 상태 의존적**이라는 점이 명확해졌습니다. 이는 각 단계의 결과물이 다음 단계의 입력으로 안정적으로 전달되어야 하는 파이프라인 형태의 프로세스임을 의미합니다.

이러한 특성에 기반하여, 네 가지 주요 프레임워크(LangChain/LangGraph, LlamaIndex, AutoGen, CrewAI)를 비교 평가한 결과, **LangChain 과 LangGraph** 가 가장 적합하다는 결론에 도달했습니다. LangGraph 는 상태 기계 모델을 통해 워크플로우에 대한 **최고 수준의 제어, 결정론, 그리고 감사 가능성**을 제공합니다. 또한, 명시적인 상태 관리, 순환과 같은 복잡한 로직 지원, 그리고 다른 생태계의 최고 도구(예: RAG 를 위한 LlamaIndex)를 유연하게 통합할 수 있는 능력은 본 프로젝트의 성공적인 구현을 위한 핵심적인 장점입니다.

최종적으로, LangGraph 를 주 오케스트레이션 프레임워크로 채택하고, 그 안에서 각 태스크에 특화된 에이전트들을 노드로 구성하는 하이브리드 아키텍처를 제안합니다. 특히 개인화 단계에서는 LlamaIndex 의 강력한 RAG 기능을 도구로 활용하여, 각 프레임워크의 강점을 극대화하는 전략이 필요합니다.

EdTech 분야에서 에이전트 시스템의 미래는 단일 프레임워크에 의존하기보다는, 이처럼 복잡하고 실제적인 문제를 해결하기 위해 각 분야에서 가장 뛰어난 전문 프레임워크들을

조합하는 하이브리드 아키텍처로 나아갈 것으로 전망됩니다. 본 보고서에서 제시된 분석과 권고안이 성공적인 서비스 개발의 초석이 되기를 기대합니다.

참고 자료

1. Simulating Classroom Education with LLM-Empowered Agents - ACL Anthology, 7 월 14, 2025 에 액세스, https://aclanthology.org/2025.naacl-long.520.pdf
2. LLM Agents for Education: Advances and Applications - arXiv, 7 월 14, 2025 에 액세스, https://arxiv.org/html/2503.11733v1
3. Lesson Plan Generator - GitHub, 7 월 14, 2025 에 액세스, https://github.com/DivanshiJain2005/AI-lesson-planner
4. Building a Multi-Agent Learning Tutor with AutoGen Framework | by Venugopal Adep | AI Product Leader @ Jio | Medium, 7 월 14, 2025 에 액세스, https://medium.com/@venugopal.adep/building-a-multi-agent-learning-tutor-with-autogen-framework-f70aa076e0b9
5. LangChain Tutorial: Build 100% Reliable AI Agents with Structured Output (Code Revealed), 7 월 14, 2025 에 액세스, https://www.youtube.com/watch?v=4EwGIXaM6C8
6. Structured outputs - LangChain, 7 월 14, 2025 에 액세스, https://python.langchain.com/docs/concepts/structured_outputs/
7. Teaching LangChain Agents to Plan & Run Multi-Step, Multi-Tool Workflows - Medium, 7 월 14, 2025 에 액세스, https://medium.com/@avigoldfinger/teaching-langchain-agents-to-plan-run-multi-step-multi-tool-workflows-82ac908fd56e
8. Framework — LlamaIndex - Build Knowledge Assistants over your Enterprise Data, 7 월 14, 2025 에 액세스, https://www.llamaindex.ai/framework
9. AI Agency for Education | GPT-trainer Blog, 7 월 14, 2025 에 액세스, https://gpt-trainer.com/blog/transforming+education+with+llm+powered+ai
10. 40 LLM Projects to Upgrade Your AI Skillset in 2025 - ProjectPro, 7 월 14, 2025 에 액세스, https://www.projectpro.io/article/llm-project-ideas/881
11. Comparing Open-Source AI Agent Frameworks - Langfuse Blog, 7 월 14, 2025 에 액세스, https://langfuse.com/blog/2025-03-19-ai-agent-comparison
12. LlamaIndex vs. LangChain: Which RAG Tool is Right for You? - n8n Blog, 7 월 14, 2025 에 액세스, https://blog.n8n.io/llamaindex-vs-langchain/
13. Improved Long & Short-Term Memory for LlamaIndex Agents, 7 월 14, 2025 에 액세스, https://www.llamaindex.ai/blog/improved-long-and-short-term-memory-

for-llamaindex-agents
14. Memory - LlamaIndex, 7 월 14, 2025 에 액세스, https://docs.llamaindex.ai/en/stable/module_guides/deploying/agents/memory/
15. LangGraph memory - Overview, 7 월 14, 2025 에 액세스, https://langchain-ai.github.io/langgraph/concepts/memory/
16. Memory in LangChain: A Deep Dive into Persistent Context - Comet ML, 7 월 14, 2025 에 액세스, https://www.comet.com/site/blog/memory-in-langchain-a-deep-dive-into-persistent-context/
17. Memory for agents - LangChain Blog, 7 월 14, 2025 에 액세스, https://blog.langchain.com/memory-for-agents/
18. Memory and RAG — AutoGen - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/stable//user-guide/agentchat-user-guide/memory.html
19. Agent with memory using Mem0 | AutoGen 0.2 - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/notebooks/agentchat_memory_using_mem0/
20. CrewAI memories : multi-users environment + conversational history - Crews, 7 월 14, 2025 에 액세스, https://community.crewai.com/t/crewai-memories-multi-users-environment-conversational-history/4237
21. CrewAI: Herding LLM Cats - Tribe AI, 7 월 14, 2025 에 액세스, https://www.tribe.ai/applied-ai/crewai-herding-llm-cats
22. langchain-ai/langchain-teacher: Teach LangChain using LangChain! - GitHub, 7 월 14, 2025 에 액세스, https://github.com/langchain-ai/langchain-teacher
23. Educational content generation using multi-LLM agents - MedCrave online, 7 월 14, 2025 에 액세스, https://medcraveonline.com/IRATJ/educational-content-generation-using-multi-llm-agents.html
24. Agentic AI #3 — Top AI Agent Frameworks in 2025: LangChain, AutoGen, CrewAI & Beyond | by Aman Raghuvanshi - Medium, 7 월 14, 2025 에 액세스, https://medium.com/@iamanraghuvanshi/agentic-ai-3-top-ai-agent-frameworks-in-2025-langchain-autogen-crewai-beyond-2fc3388e7dec
25. Introduction | 🦜 LangChain, 7 월 14, 2025 에 액세스, https://python.langchain.com/docs/introduction/
26. langchain-ai/langchain: Build context-aware reasoning applications - GitHub, 7 월 14, 2025 에 액세스, https://github.com/langchain-ai/langchain
27. LangChain, 7 월 14, 2025 에 액세스, https://www.langchain.com/

28. LangGraph - LangChain, 7 월 14, 2025 에 액세스, https://www.langchain.com/langgraph
29. LangGraph - GitHub Pages, 7 월 14, 2025 에 액세스, https://python.langchain.com/docs/langgraph
30. Comparing LLM Agent Frameworks Controllability and Convergence: LangGraph vs AutoGen vs CREW AI | by ScaleX Innovation, 7 월 14, 2025 에 액세스, https://scalexi.medium.com/comparing-llm-agent-frameworks-langgraph-vs-autogen-vs-crew-ai-part-i-92234321eb6b
31. A Detailed Comparison of Top 6 AI Agent Frameworks in 2025 - Turing, 7 월 14, 2025 에 액세스, https://www.turing.com/resources/ai-agent-frameworks
32. Building Multi-Agent Systems with LangGraph: A Step-by-Step Guide | by Sushmita Nandi, 7 월 14, 2025 에 액세스, https://medium.com/@sushmita2310/building-multi-agent-systems-with-langgraph-a-step-by-step-guide-d14088e90f72
33. Llamaindex vs Langchain: What's the difference? - IBM, 7 월 14, 2025 에 액세스, https://www.ibm.com/think/topics/llamaindex-vs-langchain
34. daveebbelaar/langchain-experiments: Building Apps with LLMs - GitHub, 7 월 14, 2025 에 액세스, https://github.com/daveebbelaar/langchain-experiments
35. kyrolabs/awesome-langchain: Awesome list of tools and projects with the awesome LangChain framework - GitHub, 7 월 14, 2025 에 액세스, https://github.com/kyrolabs/awesome-langchain
36. LlamaIndex - LlamaIndex, 7 월 14, 2025 에 액세스, https://docs.llamaindex.ai/
37. www.ibm.com, 7 월 14, 2025 에 액세스, https://www.ibm.com/think/topics/llamaindex-vs-langchain#:~:text=LlamaIndex%20and%20LangChain%20are%20two,platform%20supporting%20numerous%20use%20cases.
38. LlamaIndex vs LangChain vs Haystack | by Hey Amit - Medium, 7 월 14, 2025 에 액세스, https://medium.com/@heyamit10/llamaindex-vs-langchain-vs-haystack-4fa8b15138fd
39. Advanced RAG using Llama Index - by Plaban Nayak - AI Planet, 7 월 14, 2025 에 액세스, https://medium.aiplanet.com/advanced-rag-using-llama-index-e06b00dc0ed8
40. Workflows - LlamaIndex, 7 월 14, 2025 에 액세스, https://docs.llamaindex.ai/en/stable/module_guides/workflow/
41. Agents - LlamaIndex, 7 월 14, 2025 에 액세스, https://docs.llamaindex.ai/en/stable/use_cases/agents/

42. LlamaIndex - Build Knowledge Assistants over your Enterprise Data, 7 월 14, 2025 에 액세스, https://www.llamaindex.ai/
43. AutoGen — AutoGen, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/stable//index.html
44. Getting Started | AutoGen 0.2 - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/Getting-Started/
45. Introduction to AutoGen - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/tutorial/introduction/
46. Multi-agent Conversation Framework | AutoGen 0.2, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/Use-Cases/agent_chat/
47. How to use the Microsoft Autogen framework to Build AI Agents? - ProjectPro, 7 월 14, 2025 에 액세스, https://www.projectpro.io/article/autogen/1139
48. Conversation Patterns | AutoGen 0.2 - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/tutorial/conversation-patterns/
49. Group Chat — AutoGen - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/stable//user-guide/core-user-guide/design-patterns/group-chat.html
50. autogen.GroupChat - AG2, 7 월 14, 2025 에 액세스, https://docs.ag2.ai/0.8.2/docs/api-reference/autogen/GroupChat/
51. agentchat.groupchat | AutoGen 0.2 - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/reference/agentchat/groupchat/
52. agentchat.conversable_agent | AutoGen 0.2, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/docs/reference/agentchat/conversable_agent/
53. Group Chat with Customized Speaker Selection Method | AutoGen 0.2, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/notebooks/agentchat_groupchat_customized/
54. Agents — AutoGen - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/stable//user-guide/agentchat-user-guide/tutorial/agents.html
55. Tool Use | AutoGen 0.2 - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/tutorial/tool-use/

56. 7 Autogen Projects to Build Multi-Agent Systems - ProjectPro, 7 월 14, 2025 에 액세스, https://www.projectpro.io/article/autogen-projects-and-examples/1129
57. CrewAI: Introduction, 7 월 14, 2025 에 액세스, https://docs.crewai.com/en/introduction
58. Overview - CrewAI, 7 월 14, 2025 에 액세스, https://docs.crewai.com/en/learn/overview
59. Processes - CrewAI, 7 월 14, 2025 에 액세스, https://docs.crewai.com/en/concepts/processes
60. bhancockio/crewai-updated-tutorial-hierarchical - GitHub, 7 월 14, 2025 에 액세스, https://github.com/bhancockio/crewai-updated-tutorial-hierarchical
61. feat: implement hierarchical agent delegation with allowed_agents parameter by Vardaan-Grover · Pull Request #2068 · crewAIInc/crewAI - GitHub, 7 월 14, 2025 에 액세스, https://github.com/crewAIInc/crewAI/pull/2068
62. CrewAI: Introduction, 7 월 14, 2025 에 액세스, https://docs.crewai.com/core-concepts/Memory/
63. Tools - CrewAI, 7 월 14, 2025 에 액세스, https://docs.crewai.com/en/concepts/tools
64. Please suggest a tutorial to create a custom tool in crew.ai : r/crewai - Reddit, 7 월 14, 2025 에 액세스, https://www.reddit.com/r/crewai/comments/1innni1/please_suggest_a_tutorial_to_create_a_custom_tool/
65. Writing a Custom Agent Tool Using CrewAI | by Yogendra Sisodia | Medium, 7 월 14, 2025 에 액세스, https://medium.com/@scholarly360/writing-a-custom-agent-tool-using-crewai-aafdcc70a542
66. Building Multi-Agent Systems With CrewAI - A Comprehensive Tutorial - Firecrawl, 7 월 14, 2025 에 액세스, https://www.firecrawl.dev/blog/crewai-multi-agent-systems-tutorial
67. How to Build an AI Agent with CrewAI? - ProjectPro, 7 월 14, 2025 에 액세스, https://www.projectpro.io/article/build-an-ai-agent-with-crewai/1095
68. Technical Comparison of AutoGen, CrewAI, LangGraph, and OpenAI Swarm | by Omar Santos | Artificial Intelligence in Plain English, 7 월 14, 2025 에 액세스, https://ai.plainenglish.io/technical-comparison-of-autogen-crewai-langgraph-and-openai-swarm-1e4e9571d725
69. Group Chat with Retrieval Augmented Generation | AutoGen 0.2 - Open Source at Microsoft, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/notebooks/agentchat_groupchat_R

AG/

70. AutoGen 0.4 Tutorial - Create a Team of AI Agents (+ Local LLM w/ Ollama), 7 월 14, 2025 에 액세스, https://www.gettingstarted.ai/autogen-multi-agent-workflow-tutorial/
71. Quickstart - CrewAI, 7 월 14, 2025 에 액세스, https://docs.crewai.com/en/quickstart
72. llama_index/docs/docs/DOCS_README.md at main - GitHub, 7 월 14, 2025 에 액세스, https://github.com/run-llama/llama_index/blob/main/docs/docs/DOCS_README.md
73. A Tour of Popular Open Source Frameworks for LLM-Powered Agents - Dataiku blog, 7 월 14, 2025 에 액세스, https://blog.dataiku.com/open-source-frameworks-for-llm-powered-agents
74. Group Chat | AutoGen 0.2 - Microsoft Open Source, 7 월 14, 2025 에 액세스, https://microsoft.github.io/autogen/0.2/docs/notebooks/agentchat_groupchat/