2026년형 차세대 **LangChain** 및 **LangGraph** 기반 업무 자동화 **AI** 에이전트 개발 심층 분석 및 마스터플랜

1. 서론: **Gemini 3** 시대의 에이전트 패러다임 전환

2026년 1월, 인공지능 기술 환경은 단순한 대규모 언어 모델(LLM)의 활용을 넘어, 자율적인 의사결정과 도구 사용이 가능한 '에이전틱 워크플로우(Agentic Workflow)'로 급격히 재편되었습니다. 귀하가 제시한 초기 개발 계획은 LLM을 단순한 텍스트 생성기로 간주하던 시기의 아키텍처에 기반하고 있으나, 현재 Google의 Gemini 3 생태계는 추론(Reasoning), 초고속 경량화(Flash), 그리고 심층 연구(Deep Research)라는 세 가지 축을 중심으로 혁신적인 도약을 이루었습니다.

본 보고서는 귀하의 개발 계획을 2026년 최신 기술 스택으로 전면 재설계하기 위한 심층 기술 분석서이자 실행 가이드입니다. 특히, 비용 효율성을 극대화하면서도 최상의 성능을 유지하기 위해 **Gemini 3 Flash**를 기반 엔진으로 채택하고, 복잡한 추론이 필요한 구간에 **Gemini 3 Pro Preview**를 배치하며, 장시간의 조사가 필요한 작업에는 **Gemini Deep Research**를 투입하는 '하이브리드 모델 오케스트레이션' 전략을 제안합니다. 또한, 보안과 비용 절감을 위한 로컬 LLM 전략으로 **Qwen 3** 및 **Gemma 3**의 활용 방안을 구체적으로 제시합니다.

본 문서는 개발을 담당할 AI 에이전트가 즉시 참고하여 구현에 착수할 수 있도록, 각 기술 요소의 아키텍처적 의의, 구현 세부 사항, 최적화 전략, 그리고 필수 참조 공식 문서(URL)를 망라하여 작성되었습니다.

---

2. 핵심 엔진 전략: **Google Gemini 3** 생태계의 최적화

성공적인 업무용 AI 에이전트 구축의 첫 번째 단계는 작업의 성격에 따라 가장 적합한 모델을 동적으로 선택하는 것입니다. 2026년 현재 Google의 모델 라인업은 비용과 성능의 트레이드오프를 정교하게 조절할 수 있도록 세분화되어 있습니다.

2.1 메인 프로세서: **Gemini 3 Flash (**비용 효율성의 혁명**)**

업무용 에이전트의 트래픽 중 70~80%는 단순한 정보 검색, 요약, 이메일 초안 작성, 기본적인 코드 수정과 같은 루틴한 작업입니다. 이러한 작업에 고비용의 모델을 사용하는 것은 자원 낭비입니다. **Gemini 3 Flash**는 이러한 요구를 완벽하게 충족하는 모델로, 본 프로젝트의 'Default Model'로 선정됩니다.

2.1.1 기술적 특성 및 선정 이유

2025년 12월 출시된 Gemini 3 Flash는 'Flash'라는 이름이 무색할 정도로 강력한 성능을

보여줍니다. 이전 세대 플래그십인 Gemini 2.5 Pro를 능가하는 추론 능력을 갖추었으며, 특히 코딩 벤치마크인 SWE-bench Verified에서 78.0%를 기록하여 Gemini 3 Pro(76.2%)를 상회하는 기염을 토했습니다.[1] 이는 모델의 파라미터 크기보다 아키텍처의 효율성이 특정 태스크(코딩, 도구 사용)에 더 큰 영향을 미친다는 것을 증명합니다.

- 비용 효율성**:** 입력 100만 토큰당 $0.50, 출력 100만 토큰당 $3.00으로 책정되어 있어, 대규모 문맥(Context)을 상시 유지해야 하는 에이전트 환경에서 경제적 부담을 최소화합니다.[4]
- 속도와 지연 시간**(Latency):** Gemini 2.5 Pro 대비 3배 빠른 처리 속도를 제공하여, 사용자와의 실시간 인터랙션이 중요한 채팅 인터페이스에서 탁월한 사용자 경험(UX)을 보장합니다.[5]
- 개발 가이드**:** 개발 에이전트는 LangChain의 ChatGoogleGenerativeAI 클래스 초기화 시 gemini-3-flash-preview를 기본 모델 ID로 설정해야 합니다.

특성	**Gemini 3 Flash**	**Gemini 3 Pro**	비고
모델 **ID**	gemini-3-flash-preview	gemini-3-pro-preview	API 호출 시 사용
입력 비용	**$0.50** / 1M tokens	$2.00 / 1M tokens	Flash가 4배 저렴 [4]
출력 비용	**$3.00** / 1M tokens	$12.00 / 1M tokens	Flash가 4배 저렴 [4]
SWE-bench (코딩)	**78.0%**	76.2%	Flash 우위 [2]
GPQA Diamond (추론)	90.4%	**91.9%**	Pro 소폭 우위 (박사급 추론) [2]
Context Window	1,000,000+	1,000,000+	동일

2.1.2 필수 참조 리소스

- 공식 문서**:** Gemini 3 Flash 모델 가이드
- 가격 정책**:** Gemini API Pricing

2.2 고도 추론 엔진**: Gemini 3 Pro Preview (**복잡성 해결**)**

사용자의 요청이 복잡한 논리적 추론을 요구하거나, 다단계 계획(Planning) 수립, 혹은 매우 미묘한 뉘앙스의 창의적 작문이 필요한 경우에는 **Gemini 3 Pro**로 에스컬레이션(Escalation)해야 합니다.

2.2.1 **Deep Think** 모드와 추론 능력

Gemini 3 Pro는 'Deep Think' 모드를 통해 문제 해결 과정을 시각화하고(Thought Signatures), 복잡한 지시사항을 단계별로 분해하여 처리하는 능력이 탁월합니다.[6] LangGraph의 'Planner' 노드나 최종 결과물을 검수하는 'Critic' 노드에서 이 모델을 활용함으로써 전체 시스템의 신뢰도를 높일 수 있습니다.

- 활용 전략**:** LangGraph의 라우터(Router) 노드에서 사용자 질의의 복잡도를 판단하여, 높은 추론 능력이 필요한 경우에만 Pro 모델을 호출하도록 분기 로직을 설계합니다.
- 개발 가이드**:** LangChain 설정 시 temperature 값을 상황에 맞춰 조절해야 합니다. Gemini 3.0 이상 모델에서는 temperature가 명시되지 않을 경우 기본값이 1.0으로 설정되므로, 정밀한 제어를 위해 0.7(일반) 또는 0.2(엄격한 추론) 등으로 명시적인 설정이 필요합니다.[7]

2.3 심층 연구 자동화**: Gemini Deep Research (**자율 에이전트**)**

기존 계획에 포함된 '자료 조사' 기능을 획기적으로 업그레이드하기 위해 **Gemini Deep Research**를 통합합니다. 이는 단순한 검색 쿼리 생성을 넘어, 수백 개의 웹사이트를 탐색하고, 정보를 교차 검증하며, 장문의 보고서를 작성하는 완전 관리형 에이전트 서비스입니다.

2.3.1 **LangGraph** 통합 아키텍처

Deep Research는 실행에 수 분에서 수십 분이 소요되는 장기 실행(Long-running) 작업입니다. 따라서 동기식(Synchronous) 호출 대신 비동기식(Asynchronous) 폴링 패턴을 적용해야 합니다.

1. 작업 위임**:** LangGraph의 특정 노드에서 Deep Research API를 호출하여 조사 작업을 시작합니다.
2. 비동기 대기**:** 작업 ID(interaction ID)를 받아 상태(State)에 저장하고, 워크플로우를 일시 중단하거나 다른 작업을 수행합니다.
3. 결과 수집**:** 주기적으로 작업 상태를 폴링(Polling)하다가, 완료 시 생성된 보고서를 가져와 에이전트의 컨텍스트에 통합합니다.[8]

- 필수 참조 리소스**:**(https://ai.google.dev/gemini-api/docs/deep-research)

---

3. 아키텍처 혁신**: LangGraph** 기반의 상태 주도형**(State-Driven)** 설계

LangChain의 전통적인 체인(Chain) 구조는 선형적(Linear)이어서, 에러가 발생했을 때 이전 단계로 돌아가거나, 결과가 만족스럽지 않을 때 반복(Loop)하는 유연한 대처가 어렵습니다. 본 프로젝트는 이를 극복하기 위해 **LangGraph**를 도입하여 순환형(Cyclic) 그래프 아키텍처를 구현합니다.

3.1 **LangGraph** 도입의 필연성 및 구조적 이점

LangGraph는 에이전트를 '상태 머신(State Machine)'으로 모델링합니다. 이는 에이전트가 현재

어떤 단계에 있는지, 어떤 데이터를 보유하고 있는지 명확하게 정의하고 관리할 수 있게 해줍니다.

- 순환성 **(Cycles):** 계획 수립 -> 도구 실행 -> 결과 관찰 -> (불충분 시) 계획 수정 -> 도구 재실행과 같은 반복 루프를 자연스럽게 구현할 수 있습니다.[9]
- 지속성 **(Persistence):** 대화 도중 시스템이 중단되더라도, 데이터베이스에 저장된 '체크포인트(Checkpoint)'를 통해 중단된 지점부터 작업을 재개할 수 있습니다. 이는 장기 기억(Long-term Memory) 구현의 핵심입니다.
- 인간 개입 **(Human-in-the-loop):** 중요한 의사결정(예: 이메일 발송, 코드 배포) 직전에 워크플로우를 일시 정지하고, 사용자의 승인을 받은 후 진행하는 로직을 표준화된 방식으로 구현할 수 있습니다.

3.2 에이전트 상태**(State)** 스키마 설계

개발 에이전트는 아래와 같은 Python TypedDict 구조를 기반으로 상태를 관리하도록 코드를 작성해야 합니다.

Python

```
from typing import TypedDict, Annotated, List, Union, Optional
from langchain_core.messages import BaseMessage
import operator

class AgentState(TypedDict):
    """에이전트의 전체 상태를 관리하는 스키마"""

    # 대화 기록: 사용자와 에이전트 간의 모든 메시지 (자동 append 처리)
    messages: Annotated, operator.add]

    # 현재 작업 계획: 에이전트가 수립한 단계별 계획
    plan: List[str]

    # 수집된 정보: 검색이나 도구 사용을 통해 얻은 데이터 저장소
    research_context: Annotated[dict, operator.ior] # 딕셔너리 병합 연산자 사용

    # 다음 행동 지시: Router 노드가 결정한 다음 단계
    next_step: str

    # 에러 카운터: 무한 루프 방지용
    error_count: int
```

```
    # 사용 모델: 현재 단계에서 사용할 모델 ID (Flash/Pro)
    model_selection: str
```

3.3 핵심 노드(Node) 및 엣지(Edge) 구성 상세

LangGraph는 노드(작업 수행)와 엣지(흐름 제어)로 구성됩니다. 본 프로젝트를 위한 최적의 그래프 토폴로지는 다음과 같습니다.

1. **Supervisor Node (라우터):**
    - 역할: 사용자의 입력을 분석하여 작업의 종류를 분류합니다.
    - 로직: Gemini 3 Flash를 사용하여 입력을 '일반 대화', '심층 조사', '코딩 작업', '정보 검색' 등으로 분류하고 next_step 상태를 업데이트합니다.
    - 비용 절감: 이 단계는 매우 빈번하게 실행되므로 반드시 Flash 모델을 사용합니다.
2. **Research Node (Deep Research):**
    - 역할: next_step이 '심층 조사'일 때 활성화됩니다.
    - 구현: Gemini Deep Research API를 비동기로 호출하고, 결과를 폴링하여 research_context에 병합합니다.
3. **Executor Node (도구 실행):**
    - 역할: 검색, 계산, 코드 실행 등의 도구를 실제로 수행합니다.
    - **Grounding:** 여기서 Google Search Grounding 도구가 호출됩니다.
4. **Generator Node (답변 생성):**
    - 역할: 수집된 정보를 바탕으로 최종 답변을 작성합니다.
    - 모델 선택: 단순 요약은 Flash, 창의적 작문이나 복합 추론은 Pro를 사용하도록 동적으로 설정합니다.
5. **Critique Node (검증 및 반성):**
    - 역할: 생성된 답변이 사용자 요구사항을 충족하는지, 팩트 체크가 되었는지 검증합니다.
    - 순환: 기준 미달 시 다시 Executor Node나 Generator Node로 엣지를 연결하여 재작업을 지시합니다.

3.4 필수 참조 리소스

- **LangGraph** 공식 문서:(https://langchain-ai.github.io/langgraph/)
- 구현 예제 **(ReAct Agent):**(https://ai.google.dev/gemini-api/docs/langgraph-example)
- 비동기 처리 패턴: Async Node Patterns

---

4. 정보의 정확성 확보: Google Search Grounding 및 RAG 고도화

AI 에이전트의 가장 치명적인 약점인 환각(Hallucination) 현상을 방지하기 위해, 외부 지식에 답변을 접지(Grounding)시키는 기술이 필수적입니다. 2026년 기준, 별도의 검색 API(Serper, Tavily 등)를 사용하는 것보다 Google의 네이티브 Grounding 기능을 사용하는 것이 비용과 성능

면에서 유리합니다.

4.1 Google Search Grounding 통합 전략

Gemini API는 'Google Search Grounding'을 기본 기능으로 제공합니다. 이를 활성화하면 모델이 답변을 생성할 때 실시간 웹 검색 결과를 참조하고, 답변에 출처(Citation)를 자동으로 표기합니다.

- 구현 방식: LangChain의 ChatGoogleGenerativeAI 객체 생성 시 도구 설정을 통해 활성화합니다.
- 비용 구조: Gemini 3 Flash 사용 시 월 5,000회까지 무료로 제공되며, 이후 1,000 쿼리당 $14의 비용이 발생합니다.[1] 이는 타 검색 API 대비 경쟁력 있는 가격 정책입니다.
- 기술적 이점: 모델이 검색 쿼리를 스스로 최적화하며, 최신 뉴스나 주가 정보와 같은 실시간 데이터에 대해 높은 정확도를 보장합니다.

개발 에이전트를 위한 코드 구현 가이드 **(Python)**

Python

```
from langchain_Genai import ChatGoogleGenerativeAI
from langchain_Genai import GoogleSearchRetrieval

# Google Search Grounding이 활성화된 Gemini 3 Flash 모델 초기화
llm_with_search = ChatGoogleGenerativeAI(
    model="gemini-3-flash-preview",
    temperature=0,
    google_search_retrieval=GoogleSearchRetrieval() # 네이티브 검색 기능 활성화
)

# 호출 예시: 별도의 툴 바인딩 없이도 내부적으로 검색 수행
response = llm_with_search.invoke("2026년 현재 생성형 AI 시장 점유율 1위 기업은 어디야?")
print(response.content) # 출처가 포함된 답변 출력
```

- 참조 리소스:(https://docs.langchain.com/oss/python/integrations/chat/Generative_ai)

4.2 문맥 캐싱(Context Caching)을 통한 RAG 비용 최적화

업무용 에이전트는 회사의 내규, 프로젝트 문서, 지난 회의록 등 방대한 양의 배경 지식(Context)을 참고해야 합니다. 매 요청마다 이 방대한 텍스트를 전송하는 것은 비용과 지연 시간을 초래합니다. Gemini의 **Context Caching** 기능은 이를 해결하는 핵심 기술입니다.

- 작동 원리: 자주 사용되는 긴 프롬프트(예: 전체 사규 문서)를 미리 캐싱해두고, 이후

요청에서는 캐시된 토큰을 참조합니다.
- 비용 절감: 캐시된 입력 토큰은 일반 입력 토큰 대비 1/10 수준($0.05/1M tokens)으로 매우 저렴합니다.[4]
- 구현 전략: RAG(검색 증강 생성) 시스템에서 검색된 관련 문서(Chunks)의 양이 많을 경우, 이를 캐싱하여 후속 멀티턴 대화에서의 비용을 획기적으로 낮춥니다.

---

5. 로컬 LLM 전략: 보안과 가용성을 위한 Qwen 3 및 Gemma 3 활용

모든 데이터를 클라우드로 전송할 수 없는 보안 민감도가 높은 작업이나, 인터넷 연결이 불안정한 환경을 대비하여 로컬 LLM(Local Large Language Model)을 하이브리드로 운용하는 전략이 필요합니다.

5.1 최적의 로컬 모델 선정: Qwen 3 (Thinking & Korean)

2026년 1월 기준, 오픈소스 모델 중 한국어 처리 능력과 도구 사용(Tool Use), 그리고 코딩 능력에서 가장 두각을 나타내는 모델은 Alibaba의 **Qwen 3** 시리즈입니다.

- 선정 이유:
  - 한국어 성능: 다국어 학습 데이터 비중이 높아 한국어의 문맥과 높임말 등을 자연스럽게 처리합니다.
  - **Thinking Mode:** Qwen 3는 'Thinking Process'를 내재화하여, 복잡한 문제 해결 시 내부 독백을 통해 추론의 정확도를 높일 수 있습니다. 이는 로컬 환경에서도 에이전트의 지능을 유지하는 데 결정적입니다.[11]
  - 도구 사용(Tool Call): LangChain과의 연동성이 뛰어나며, 함수 호출(Function Calling) 벤치마크에서 높은 점수를 기록했습니다.

5.2 대안 모델: Gemma 3 (Google 생태계 호환성)

Google의 오픈 모델인 **Gemma 3** 또한 강력한 후보입니다. 특히 멀티모달(이미지 인식) 기능이 필요한 경우, 별도의 비전 모델 없이 텍스트와 이미지를 동시에 처리할 수 있는 Gemma 3가 유리합니다.

- 특징: 2025년 3월 출시된 Gemma 3는 텍스트와 이미지를 네이티브하게 이해하며, Gemini 모델과 아키텍처를 공유하여 프롬프트 호환성이 높습니다.[12]

5.3 로컬 모델 구동 및 LangChain 연동 가이드

로컬 모델을 에이전트 시스템에 통합하기 위해 **Ollama**를 실행 엔진으로 사용합니다. Ollama는 REST API를 제공하여 LangChain과 손쉽게 연동됩니다.

- 설치 및 실행:
  1. Ollama 공식 사이트에서 설치.
  2. 터미널에서 ollama run qwen3:32b 명령어로 모델 다운로드 및 실행.

- **LangChain** 코드 예시:

```python
from langchain_ollama import ChatOllama

# 로컬 Qwen 3 모델 설정
local_llm = ChatOllama(
    model="qwen3:32b",
    temperature=0.1,
    keep_alive="1h" # 메모리 상주 설정으로 응답 속도 향상
)

# 민감 정보(PII) 마스킹 등 보안 작업 수행
response = local_llm.invoke("다음 텍스트에서 주민등록번호를 마스킹해줘:...")
```

---

6. 개발 에이전트를 위한 상세 구현 로드맵 (Step-by-Step Implementation)

이 섹션은 실제 코드를 작성할 에이전트가 따라야 할 구체적인 개발 순서와 체크리스트입니다.

단계 1: 환경 설정 및 기본 연결 (Week 1)

- 목표: Google AI Studio API 키 발급 및 Gemini 3 Flash, Pro 모델의 기본 입출력 테스트.
- 필수 작업:
  - Python 3.12+ 가상환경 구축.
  - langchain-google-genai, langgraph, langchain-ollama 최신 라이브러리 설치.
  - .env 파일에 GEMINI_API_KEY 설정.
  - ChatGoogleGenerativeAI를 사용하여 Flash와 Pro 모델의 응답 속도 및 비용 차이 로깅 시스템 구축.

단계 2: LangGraph 상태 머신 구축 (Week 2)

- 목표: 순환형 에이전트의 뼈대(Scaffold) 완성.
- 필수 작업:
  - AgentState TypedDict 정의 (메시지, 계획, 컨텍스트 포함).
  - StateGraph를 생성하고 START, END 노드 연결.
  - 라우터(Router) 노드 구현: 사용자 입력에 따라 경로를 분기하는 프롬프트 엔지니어링 수행.
  - 체크포인터(MemorySaver 또는 AsyncSqliteSaver)를 연결하여 대화 기억 유지 테스트.

단계 3: 도구(Tool) 통합 및 Grounding (Week 3)

- 목표: 에이전트의 실행 능력 확보.

- 필수 작업**:**
  - Google Search Grounding 도구 연동 및 검색 쿼리 최적화 테스트.
  - 파일 입출력, 이메일 초안 작성 등의 커스텀 도구 함수(@tool 데코레이터 사용) 구현.
  - Gemini Deep Research API 비동기 노드 구현 및 폴링 로직 작성.

단계 **4:** 로컬 **LLM** 하이브리드 구성 **(Week 4)**

- 목표**:** 보안 및 비용 최적화.
- 필수 작업**:**
  - Ollama 서버 구축 및 Qwen 3 모델 로딩.
  - 개인정보가 포함된 질의를 탐지하여 로컬 모델로 라우팅하는 가드레일(Guardrails) 로직 구현.
  - 인터넷 연결 단절 시 로컬 모델로 자동 절체(Failover)되는 로직 구현.

단계 **5:** 평가 및 배포 **(Week 5)**

- 목표**:** 시스템 안정성 검증.
- 필수 작업**:**
  - **LangSmith**를 연동하여 트레이스(Trace) 모니터링 및 디버깅.[14]
  - SWE-bench 스타일의 자체 테스트셋을 구축하여 코딩 및 추론 능력 정량 평가.
  - 사용자 피드백을 반영할 수 있는 'Human-in-the-loop' 인터페이스(예: Streamlit) 구축.

---

7. 결론**: 2026**년형 에이전트의 경쟁력

본 마스터플랜에 따라 개발된 AI 에이전트는 단순한 챗봇을 넘어 실질적인 업무 처리가 가능한 '디지털 동료'로 기능할 것입니다.

1. 압도적 가성비**: Gemini 3 Flash**와 **Context Caching**의 적극적인 활용으로 운영 비용을 기존 대비 70% 이상 절감합니다.
2. 검증된 정확성**: Google Search Grounding**과 **Deep Research**를 통해 최신 정보에 기반한 신뢰할 수 있는 답변을 제공합니다.
3. 유연성과 확장성**: LangGraph** 아키텍처는 업무 프로세스의 변화에 따라 에이전트의 로직을 손쉽게 수정하고 확장할 수 있는 기반을 제공합니다.
4. 보안**: **로컬 LLM(Qwen 3)**의 하이브리드 운용으로 기업의 민감 데이터를 안전하게 보호합니다.

이제 개발 에이전트는 첨부된 링크와 가이드를 바탕으로, 2026년 AI 기술의 정점을 구현하는 여정을 시작하시기 바랍니다.

[부록**]** 개발 에이전트를 위한 기술 스택 요약표

구분	기술 요소	버전/모델	필수 참조 링크	비고
LLM (Main)	Google AI API	**Gemini 3 Flash**	Pricing & Features	기본 모델
LLM (Reasoning)	Google AI API	**Gemini 3 Pro**	Model Card	복잡 추론용
Research	Google AI API	**Deep Research**	API Guide	심층 조사용
Local LLM	Ollama	**Qwen 3 (32B)**	Ollama Library	보안/백업용
Orchestration	LangGraph	Latest (0.2.x+)	(https://langchain-ai.github.io/langgraph/)	상태 관리
Search	Google Grounding	Built-in	LangChain Integration	할루시네이션 방지
Monitoring	LangSmith	Latest	(https://docs.smith.langchain.com/)	디버깅/평가

[주의사항] Gemini 3 시리즈는 Preview 단계이므로 API 엔드포인트나 파라미터가 변경될 수 있습니다. 개발 시 반드시 최신 공식 문서를 재확인하십시오.

참고 자료

1. Gemini 3 Flash API Pricing Guide: Complete Cost Breakdown (December 2025), 1월 29, 2026에 액세스, https://www.aifreeapi.com/en/posts/gemini-3-flash-api-price
2. Google's Gemini 3 Flash Just Made the Flagship Model Obsolete, even Gemini 3 Pro, and It Costs 69%..., 1월 29, 2026에 액세스, https://medium.com/@cognidownunder/googles-gemini-3-flash-just-made-the-flagship-model-obsolete-even-gemini-3-pro-and-it-costs-69-a28ee71d7471
3. Gemini 3 Flash: The "Budget" Model That's DESTROYING Premium AI (And It's Free), 1월 29, 2026에 액세스, https://www.youtube.com/watch?v=bW6GC9lqQng
4. Gemini Developer API pricing, 1월 29, 2026에 액세스, https://ai.google.dev/gemini-api/docs/pricing
5. Gemini 3.0 Flash: Google's Greatest Model Ever? Most Powerful, Cheapest, & Fastest Model! (Tested), 1월 29, 2026에 액세스,

https://www.youtube.com/watch?v=izXjYxKTI_k
6. Release notes | Gemini API - Google AI for Developers, 1월 29, 2026에 액세스, https://ai.google.dev/gemini-api/docs/changelog
7. ChatGoogleGenerativeAI - Docs by LangChain, 1월 29, 2026에 액세스, https://docs.langchain.com/oss/python/integrations/chat/Generative_ai
8. Gemini Deep Research Agent | Gemini API | Google AI for Developers, 1월 29, 2026에 액세스, https://ai.google.dev/gemini-api/docs/deep-research
9. Build a LangGraph Multi-Agent system in 20 Minutes with LaunchDarkly AI Configs, 1월 29, 2026에 액세스, https://launchdarkly.com/docs/tutorials/agents-langgraph
10. LangGraph: Multi-Agent Workflows - LangChain Blog, 1월 29, 2026에 액세스, https://www.blog.langchain.com/langgraph-multi-agent-workflows/
11. Ultimate Guide - The Best Open Source LLM For Korean In 2026 - SiliconFlow, 1월 29, 2026에 액세스, https://www.siliconflow.com/articles/en/best-open-source-llm-for-korean
12. Gemma AI Models: Google's Open Multimodal LLMs for Text, Code, and Vision (2025), 1월 29, 2026에 액세스, https://www.inferless.com/learn/the-ultimate-guide-to-gemma-models
13. Gemma (language model) - Wikipedia, 1월 29, 2026에 액세스, https://en.wikipedia.org/wiki/Gemma_(language_model)
14. Setting up Gemini with grounding_tool in prompt hub - LangChain Forum, 1월 29, 2026에 액세스, https://forum.langchain.com/t/setting-up-gemini-with-grounding-tool-in-prompt-hub/221