Qwen

반도체 IDMs의 직원 문제 해결을 위한 AI 가이드 시스템 구축 계획서

본 보고서는 대규모 사용자 환경에서 다양한 업무 도메인에 걸친 복잡한 문제를 해결하기 위해, FastAPI와 LangGraph를 활용한 프로토타입 기반의 AI 가이드 시스템을 설계하고 구축하는 체계적인 계획을 제시한다. 특히, 반도체 종합반도체제조회사(IDM) 내 10만명의 사용자를 대상으로 하며, 각기 다른 전문 영역(경영, 제조, 연구개발, SW 등)에서 발생하는 문제들을 자동화된 가이드 및 문서 생성을 통해 지원하는 시스템의 필요성을 분석하고, 이를 실현하기 위한 상세한 실행 방안을 제시하는 것을 목표로 한다.

사용자 여정 지도 (User Journey Map) 설계 및 요구사항 정의

사용자 여정 지도(User Journey Map, UJM)는 단순히 고객이나 사용자의 행동을 시각적으로 표현하는 것을 넘어, 시스템의 목표 달성을 위한 핵심 통찰력과 개선 기회를 도출하는 강력한 디자인 사고 도구이다 [3] [6]. 본 프로젝트의 성공은 UJM을 중심으로 한 철저한 사용자 이해와 요구사항 정의에서부터 시작된다. 이는 시스템이 제공할 가치를 명확히 하고, 불필요한 기능을 차단하며, 최종 사용자인 직원들의 실제 고통점(DP: Difficulty Point)을 해결하는 데 필수적이다.

우선, 본 시스템의 주된 사용자는 경영, 제조, 연구개발, SW 등 다양한 배경을 가진 10만명 규모의 반도체 IDM 직원들이다 . 이러한 다층적인 사용자 집단을 포괄하기 위해서는 단일한 퍼소나(Persona)만으로는 부족하다. 따라서, 각 도메인별로 최소한 하나 이상의 특화된 퍼소나를 구축해야 한다. 예를 들어, 'SW 개발자 유나'는 "최신 프레임워크 적용 시 연동 문제 해결 가이드", '설계 엔지니어 준호'는 "최신 공정 표준 충족을 위한 설계 변경 가이드", '생산 현장 리더 민수'는 "신규 장비 도입 시 안전 점검 항목 리스트"와 같이 특정 업무에 초점을 맞춘 요구사항을 추려낼 수 있다. 이러한 퍼소나는 UX 디자인 방법론 중 인터뷰, 관찰, 설문조사 등을 통해 얻은 데이터를 바탕으로 구축되어야 신뢰도가 높아진다 [2] [5].

UJM의 구조는 여러 버전이 존재하지만, 본 프로젝트에는 '행동-감정-문제점' 모델이 가장 적합하다 [7]. 이 모델은 사용자가 문제를 입력하고 결과물을 받는 전체 여정을 시간 순으로 나누고, 각 단계에서의 활동(Activity), 느끼는 감정(Emotion), 겪는 고통점(Pain Point)을 시각적으로 나타낸다. 아래 표는 제조 부문 엔지니어의 여정을 예시로 한 UJM의 구조이다.

단계	활동 (Activity)	감정 (Emotion)	고통점 (Pain Point)
문제 인지	생산 현장에서 예상치 못한 불량률 증가를 확인함.	당황, 불안	관련 규정 및 기술 문서 찾는 데 오래 걸림.

단계	활동 (Activity)	감정 (Emotion)	고통점 (Pain Point)
문제 설명	PC 웹 UI에 "불량률 5% 이상 증가, 패턴: 특정 패턴의 균일한 결함"과 같은 텍스트를 작성하여 제출함.	압박감, 서두름	문법 오류나 어휘 부족으로 문제의 핵심이 제대로 전달되지 않을까 하는 불안.
가이드 생성	시스템이 백엔드에서 작업을 수행하며 진행률을 실시간으로 표시함.	기대, 긴장	응답 속도가 느리거나, 중간에 연결이 끊길 경우.
결과 확인	"불량률 증가 원인 분석 및 대응 가이드"라는 제목의 PDF 파일이 다운로드 가능하게 표시됨.	해탈, 신뢰	가이드 내용이 비전문적이거나, 현장 상황에 맞지 않는 경우.

이러한 UJM은 '고통점'이라는 키워드를 통해 문제를 구체화하는 데 큰 역할을 한다. 예를 들어, "문서를 찾는 데 시간이 너무 많이 소요된다"는 일반적인 불평보다 "특정 노하우(NOA-2024-001) 문서를 찾기 위해 3개의 검색 엔진과 2명의 동료에게 물어야 했다"는 식의 구체적인 스토리텔링이 더 효과적이다 [3]. 이 고통점들은 바로 제품 요구사항 정의(PRD)로 발전할 수 있는 잠재력을 가진다. PRD는 시스템이 반드시 갖춰야 할 기능과 성능을 명시하는 문서로, UJM에서 파생된 고통점들이 중심이 된다. 예를 들어, 위의 '문제 설명' 단계에서의 고통점은 "사용자가 자연어로 문제를 입력했을 때, 그 의미를 정확히 이해하고 관련 정보를 찾아주는 RAG 시스템"이라는 핵심 기능 요구사항으로 변환될 수 있다.

마지막으로, UJM을 통해 파악된 요구사항은 스토리맵(User Story Map)으로 구성될 수 있다 [3]. 스토리맵은 작은 사용자 이야기들을 조합하여 전체 서비스의 그림을 그려내는 구조로, 애자일 개발에서 MVP(Minimum Viable Product)를 도출하는 데 널리 사용된다. 예를 들어, "SW 개발자 유나"라는 퍼소나를 중심으로 한 스토리맵은 다음과 같이 구성될 수 있다. * ** Backbone (핵심 흐름):** * 사용자가 문제를 입력한다. * 시스템이 문제를 분석하고 해결책을 제시한다. * 사용자 스토리 (작업 항목): * [SW 개발자 유나] 특정 Python 라이브러리 업그레이드 후 발생하는 버그에 대한 해결책을 검색하고 싶다. * [SW 개발자 유나] 새로운 클라우드 서비스를 도입하고 있는데, 관련 설정 가이드를 찾고 있다. * [SW 개발자 유나] 코드 품질 저하를 막기 위한 커뮤니티 BEST PRACTICE 가이드를 참고하고 싶다. * 우선순위 매김: 초기에는 Python 관련 문제에 특화된 가이드를 제공하는 것이 중요하므로, 해당 스토리들은 높은 우선순위를 가진다.

이처럼 UJM은 시스템의 개념을 명확히 하고, 요구사항을 구체화하며, 이후의 기술 선택과 프로토타입 개발 방향을 제시하는 통합적인 프레임워크 역할을 수행한다.

시스템 아키텍처 및 기술 스택 선정

시스템의 확장성, 안정성 및 유지보수 용이성을 고려할 때, 미들웨어 없는 독립적인 프로토타입 시스템을 먼저 구축하고, 성공적인 검증 후 본격적인 서비스 개발로 이어가는 단계적 접근이 바람직하다 . 이 접근법은 리스크를 분산시키고, 초기 자원 투입을 최소화하면서도 핵심 기능의 타당성을 입증하는 데 유리하다.

시스템 아키텍처

프로토타입의 총괄 아키텍처는 크게 세 가지 계층으로 구성된다. 첫째, 클라이언트(Client) 계층은 사용자가 직접 상호작용하는 Frontend이다. 사용자의 요청을 서버로 전송하고, 서버로부터 받아온 응답을 화면에 표시하는 역할을 한다. 둘째, 애플리케이션(Application) 계층은 서버 로직을 처리하는 핵심 영역으로, FastAPI를 통해 RESTful API를 제공한다. 이 계층은 클라이언트의 요청을 받아 처리하고, 필요한 경우 백그라운드 작업을 트리거하며, 결과를 반환한다. 셋째, 인프라(Infrasrtucture) 계층은 백그라운드에서 실행되는 작업들을 관리하는 역할을 수행한다. 여기에는 LangGraph를 이용한 복잡한 에이전트 워크플로우와, LLM(LLM)과 벡터 DB(Vector DB)를 포함한 RAG 시스템이 위치한다. 이 계층들은 비동기적으로 실행되며, 완료되면 결과를 저장소에 기록한다. FastAPI는 이러한 완료 알림을 WebSocket을 통해 실시간으로 클라이언트에 전달하거나, 또는 사용자가 다시 요청할 때 결과를 조회할 수 있도록 한다.

이러한 아키텍처는 클라이언트와 서버 간의 상태 비저장(Stateless) 관계를 유지하면서도, Long-Running Task를 효율적으로 처리할 수 있는 장점을 가진다. 특히, UJM에서 언급된 '가이드 생성' 단계와 같이 시간이 오래 걸리는 작업을 처리하는 데 매우 효과적이다.

기술 스택 선정

기술 스택은 사용자의 요구사항과 시스템 아키텍처를 기반으로 철저히 검토되어야 한다.

계층	기술 스택	선택 이유 및 구체적 사용 방안	관련 근거
Frontend	Vanilla JS	요구사항에 따른 정확한 구현을 위해 최소한의 라이브러리로 구축. WebSocket API를 사용해 FastAPI로부터 실시간 스트리밍 응답을 받는다.	
Backend	FastAPI	ASGI 서버를 기반으로 비동기 처리 성능이 우수하여 머신러닝 모델 연동 및 동시 요청 처리에 적합하다. LangGraph와의 통합이 용이하다.	[24] [25]
Workflow Engine	LangGraph	복잡한 대화형 에이전트 워크플로우를 그래프 형태로 시각화하고 관리하여 재사용성과 확장성을 높임. 메모리 상태 관리를 위해 `MemorySaver` 사용.	[24] [27]
LLM (Large Language Model)	Groq llama-3.3-70b-versatile / OpenAI GPT-4o-mini	Groq는 빠른 응답 속도를 제공하여 실시간 스트리밍에 유리하고, OpenAI는 풍부한 생태계와 강력한 능력을 제공하여 두 모델을 A/B 테스트 및 비교 평가 대상으로 설정.	[24]

계층	기술 스택	선택 이유 및 구체적 사용 방안	관련 근거
RAG (Retrieval-Augmented Generation)	Chroma DB + LangChain	검색된 문서(context)를 LLM에 제공하여 사실 기반의 정확한 답변 생성. LangGraph의 `retrieve` 노드에서 사용. Chroma는 Lightweight하고 Python 환경에서 쉽게 설치/실행 가능.	[27]
Vector Store	Chroma DB	문서를 임베딩 벡터로 변환하여 저장하고, 사용자 질문의 임베딩 벡터와 유사도를 비교하여 관련 문서를 빠르게 검색.	[27]
DB (Database)	PostgreSQL	Structured data (사용자 정보, 요청 로그 등)를 관리하기 위한 관계형 데이터베이스. FastAPI와 잘 호환됨.	[25]
Cache	Redis	자주 요청되는 결과를 캐싱하여 서버 부하를 줄이고 응답 속도를 개선. LangGraph의 체크포인트에서도 사용될 수 있음.	[24]
Ingestion	LangChain, Unstructured Libraries	HTML, PDF 등 다양한 형식의 기존 문서를 LangChain 파이프라인을 통해 chunking, 임베딩, Chroma DB에 저장하는 과정을 자동화.	[27]

이러한 기술 스택은 각각의 역할을 명확히 분리하고, 각 부분을 독립적으로 업데이트하고 확장할 수 있는 유연성을 제공한다. 예를 들어, RAG 시스템의 검색 정확도를 높이기 위해 벡터 DB를 Pinecone이나 Milvus로 교체하거나, LLM을 Mistral이나 Qwen으로 바꾸는 것도 상대적으로 용이하다. 이러한 설계는 시스템이 미래 기술 변화에 유연하게 대응할 수 있는 기반을 마련해준다.

에이전트 워크플로우 설계 및 RAG 시스템 구현

FastAPI와 LangGraph를 결합하면 복잡한 비즈니스 로직을 모듈화된 노드와 그래프 형태의 워크플로우로 설계할 수 있어, 시스템의 논리적 일관성과 유지보수성을 획기적으로 향상시킨다 [24]. 본 섹션에서는 '문제 해결 가이드 생성'이라는 핵심 기능을 수행하기 위한 에이전트 워크플로우와, 그 핵심 기술인 RAG 시스템의 구체적인 설계를 제시한다.

에이전트 워크플로우 설계

LangGraph의 StateGraph를 사용하여 '문제 해결 가이드 생성' 워크플로우를 설계한다. 이 워크플로우의 state는 `{"question": str, "answer": str, "sources": list}`

와 같은 형태로 정의된다. 이는 사용자의 질문(question), 생성된 답변(answer), 그리고 답변의 근거가 되는 소스(sources)를 관리하는 역할을 한다.

워크플로우는 다음과 같은 5단계로 구성될 수 있다:

1. Validation Node (검증 노드): 사용자로부터 전달받은 질문이 유효한지 확인하는 단계. 만약 질문이 비어있거나 의도하지 않은 내용이라면, 에이전트는 사용자에게 명확한 피드백을 보내고 워크플로우를 종료한다. 이는 시스템의 안정성을 높이는 중요한 보안 장치 역할을 한다.
2. Routing Node (루팅 노드): 사용자의 질문을 분석하여 어떤 종류의 가이드를 생성해야 하는지 판단한다. 예를 들어, "SW 개발자 유나"가 "Python 3.12에서 가비지 컬렉션 관련 메모리 누수 문제 해결법"이라고 물었을 경우, 이 질문을 분류하여 'SW_001_MemoryLeak_Guide'와 같은 특정 템플릿을 호출하도록 라우팅한다. 이 과정은 LLM을 활용하여 질의 분류(Classification) 작업을 수행할 수 있다.
3. RAG Retrieval Node (RAG 검색 노드): 분류된 템플릿에 맞춰 관련 Knowledge Base(KB)에서 최신 정보를 검색한다. 이는 LangGraph의 `StatefullyRunnable(retrieve)` 노드를 통해 구현되며, 이 노드는 Chroma DB에 저장된 임베딩 벡터를 사용하여 사용자의 질문과 유사한 문서들을 검색한다[27]. 이 노드는 워크플로우의 state에 `sources`를 추가한다.
4. Generation Node (생성 노드): 검색된 소스와 사용자의 질문을 함께 LLM에 전달하여 최종 답변을 생성한다. 이 과정은 LangGraph의 `StatefullyRunnable(generate)` 노드로 구현되며, 이 노드는 RAG 노드에서 가져온 소스를 context로 활용하여 답변을 생성하고, 생성된 답변을 state의 `answer` 필드에 업데이트한다[27].
5. Finalization Node (최종화 노드): 답변 생성이 완료되면, 최종적인 가이드 문서를 PDF나 Markdown 형식으로 포맷팅하고, 결과를 사용자에게 제공할 준비를 마친다. 이 단계에서는 사용자에게 다운로드 링크를 제공하거나, 웹 페이지에 출력하는 등의 작업을 수행할 수 있다.

이러한 워크플로우는 LangGraph의 `add_edge`, `add_conditional_edges` 등을 사용하여 노드 간의 흐름을 명시적으로 연결한다. 예를 들어, 루팅 노드의 결과에 따라 다음 실행될 노드를 동적으로 결정할 수 있다. 이 구조는 각 에이전트가 단일 책임을 가지도록 모듈화하여, 향후 특정 에이전트의 알고리즘을 수정하거나 새로운 에이전트를 추가하는 것이 매우 용이하다.

RAG 시스템 구현

RAG 시스템은 LLM의 사실 오류(factual hallucination) 문제를 해결하고, 특정 도메인에 특화된 정확한 정보를 제공하는 데 필수적이다. 본 시스템의 RAG 구현은 '문맥(Context) 기반 질문 응답'이라는 핵심 원칙을 따를 것이다.

1. 데이터 수집 및 전처리(Ingestion): 시스템의 Knowledge Base는 다양한 출처에서 올 수 있다. 반도체 IDM 내부의 정책 문서, 기술 매뉴얼, 실험 보고서, 그리고 외부의 공개된 연구 자료 등이 포함될 수 있다. LangChain의 `WebBaseLoader`와 `PyPDFLoader` 등을 사용하여 URL이나 로컬 파일에서 데이터를 수집하고, `RecursiveCharacterTextSplitter`와 같은 유틸리티를 사용하여 데이터를 적

절한 크기의 chunk로 분할한다 [27]. 이때 `chunk_size=500`과 같이 설정하여 문맥의 연속성을 유지하는 것이 중요하다 [27].
2. 임베딩 및 벡터화(Vectorization): 분할된 chunk들을 임베딩(embedding) 모델을 사용하여 수치 벡터로 변환한다. 이 벡터들은 각 chunk의 의미적(mnipanic) 의미를 담고 있으며, 이 벡터들을 Chroma DB와 같은 벡터 데이터베이스에 저장한다 [27]. 이 과정에서 Chroma DB는 자체적으로 임베딩 모델을 내장하고 있으므로, 추가적인 모델 설치 없이도 쉽게 사용할 수 있다.
3. 검색 및 생성 파이프라인(Retrieve-and-Generate Pipeline): 사용자가 질문을 입력하면, 시스템은 먼저 사용자의 질문을 same embedding model을 사용하여 벡터화한다. 그 후, Chroma DB에서 이 벡터와 유사도(similarity)가 높은 상위 N개의 chunk 벡터를 검색한다. 검색된 chunk들은 'context'로 불리며, 이 context와 원래 사용자의 질문을 함께 LLM의 프롬프트에 포함시켜 최종 답변을 생성하도록 요청한다. 이 과정은 LangChain의 `create_stuff_documents_chain` 또는 `create_retrieval_chain` 유틸리티를 통해 쉽게 구현할 수 있다.
4. LLM 응답 생성(Generation): 생성된 프롬프트는 사용자 질문과 관련된 최신이고 정확한 정보를 담고 있다. 이 프롬프트를 Groq나 OpenAI의 LLM에 전달하면, LLM은 이 정보를 바탕으로 사실에 근거한 답변을 생성한다. LangChain의 `create_history_aware_embeddings` 및 `create_retrieval_chain`을 사용하면, 대화의 히스토리까지 고려하여 더욱 자연스러운 대화형 경험을 제공할 수도 있다.

이러한 RAG 시스템은 단순히 검색을 넘어, 정보의 '정확성'과 '최신성'을 보장하는 핵심 기술로, 특히 반도체와 같은 정교하고 빠르게 변화하는 기술 분야에서 시스템의 신뢰도를 높이는 데 결정적인 역할을 할 것이다. LangGraph를 통해 이 RAG 파이프라인이 에이전트 워크플로우의 일부로 통합되면, 사용자가 요청한 문제의 유형에 따라 자동으로 최적의 정보를 검색하고 가이드를 생성하는 지능형 시스템이 구현된다.

프로토타입 구축 계획 및 진행 방향

프로젝트의 복잡성을 감안하여, 프로토타입 구축은 4단계로 나누어 단계적으로 진행하는 것이 가장 효과적이다. 이 단계적 접근은 각 단계에서 명확한 목표를 설정하고, 그 목표를 달성했을 때마다 피드백을 받으며 시스템을 진화시키는 데 기여한다.

Phase 1: Core Functionality Prototype (Weeks 1-4)

이 단계의 목표는 시스템의 핵심 기능을 최소한의 노력으로 검증하는 것이다. 즉, '사용자가 질문을 입력하면, 무엇인가를 생성하여 돌려주는' 가장 기본적인 동작을 구현하는 것이다. * 기술 스택: FastAPI, LangGraph, OpenAI GPT-4o-mini, In-memory Dictionary for State Management. * 구현 내용: 1. Backend API: `/chat` 엔드포인트를 POST 메서드로 구현. 이 엔드포인트는 사용자로부터 JSON 형식의 질문을 받는다. 2. Simple Agent Workflow: LangGraph를 사용하여 단일 노드 `generate`를 만든다. 이 노드는 받은 질문을 LLM에 그대로 전달하고, 그 응답을 그대로 클라이언트에게 반환한다. 3. Frontend: 사용자에게 질문을 입력할 `<input>` 요소와, 결과를 보여줄 `<div>` 요소를 가진 단순 HTML 페이지를 구현한다. Vanilla JS를 사용하여 버튼 클릭 시 FastAPI 엔드포인트에 요청을 보내고, 응답을 화면에 표시한다. * 성공 기준: 사용자가 질문을 입력하면, LLM이 답변을 생성하여 화면에

표시되는지 확인한다. 이 단계는 기술 선정의 타당성과 기본적인 통신 흐름을 검증하는 데 초점을 맞춘다.

Phase 2: RAG Integration Prototype (Weeks 5-8)

이 단계에서는 앞서 검증한 핵심 기능에 RAG 시스템을 통합하여, 사실 기반의 답변 생성 능력을 추가한다. * 기술 스택: Phase 1의 스택에 Chroma DB, LangChain 추가. * 구현 내용: 1. Knowledge Base 준비: 샘플 기술 문서 1~2개를 준비하고, LangChain 파이프라인을 통해 이 문서들을 Chunking, Embedding, Chroma DB에 저장한다 [27]. 2. Enhanced Agent Workflow: LangGraph 워크플로우에 `retrieve` 노드를 추가한다. 이 노드는 사용자의 질문을 받아 Chroma DB에서 관련 문서를 검색한다. 3. Context-Aware Generation: `retrieve` 노드에서 검색된 문서를 `generate` 노드의 프롬프트에 'context'로 포함시켜 LLM에 전달한다. 이로써 LLM은 주어진 문서에 기반한 답변을 생성하게 된다. 4. Real-time Streaming (Optional but Recommended): LangGraph의 `astream()` 메서드와 SSE(EventSource)를 활용하여, LLM이 청크(chunk) 단위로 답변을 생성하는 것을 실시간으로 웹페이지에 스트리밍할 수 있도록 구현한다 [24] [27]. 이는 사용자에게 더 나은 상호작용 경험을 제공한다. * 성공 기준: RAG 시스템이 작동하여, 제공된 샘플 문서에 기반한 질문에 대해 정확하고 사실에 근거한 답변을 생성하는지 확인한다. 스트리밍 기능이 원활하게 작동하는지 또한 점검한다.

Phase 3: Full-featured Prototype with User Journey Simulation (Weeks 9-12)

이 단계에서는 프로토타입을 실제 사용자의 여정을 시뮬레이션할 수 있는 형태로 완성한다. * 기술 스택: Phase 2의 스택에 PostgreSQL, Redis 추가. * 구현 내용: 1. Persistent State & Checkpointing: LangGraph의 `MemorySaver`를 사용하여 에이전트 워크플로우의 상태를 PostgreSQL에 저장한다. 이렇게 하면 사용자의 대화 흐름을 지속적으로 유지할 수 있다 [24]. 2. Advanced Routing: 사용자의 질문을 더 정교하게 분류하여, 서로 다른 템플릿에 따라 다른 워크플로우를 실행하도록 루팅 로직을 개선한다. 3. Mock Data Integration: 실제 IDM 내부의 구조와 데이터를 참조하여, 몇 개의 테스트용 사용자 퍼소나와 관련 문서를 준비한다. 이 데이터를 기반으로 UJM의 각 단계를 시뮬레이션한다. 4. Result Formatting: 생성된 답변을 PDF나 Markdown 형식으로 포맷팅하는 기능을 추가하여, 사용자가 다운로드할 수 있게 한다. * 성공 기준: UJM의 각 단계를 충실히 시뮬레이션하며, 사용자가 자신의 문제를 해결하는 것처럼 느낄 수 있는 완성도 높은 프로토타입을 완성한다.

Phase 4: Scalability and Performance Benchmarking (After Week 12)

프로토타입의 성공적인 검증 후, 실제 서비스를 운영하기 위한 준비를 시작한다. * 기술 스택: Phase 3의 스택에 AWS/GCP/Azure 등 클라우드 플랫폼 추가. * 구현 내용: 1. 서비스 배포: Uvicorn과 같은 ASGI 서버를 사용하여 FastAPI 애플리케이션을 배포한다. Heroku, AWS Elastic Beanstalk, Google Cloud Run 등을 고려할 수 있다 [24]. 2. 스케일링 테스트: 여러 사용자가 동시에 접속했을 때 시스템의 성능을 테스트하고, 필요하다면 Worker Queue(예: Celery)를 도입하여 Long-Running Task를 비동기로 처리하는 방안을 모색한다. 3. 비용 분석: 클라우드 서비스 사용량에 따른 비용을 분석하고, 예산 범위 내에서 최적의 인프라를 설계한다. 4. 보안 검토: 사용자 인증, 데이터 암호화, DDoS 공격 방어 등 보안 관련 사항을 검토하고 대응 방안을 마련한다.

이러한 단계적인 구축 계획은 프로젝트의 목표를 명확히 하고, 각 단계에서의 성공 가능성을 높이며, 최종적으로 성공적인 프로토타입을 통해 후속 프로젝트에 대한 투자와 신뢰를 확보하는 데 기여할 것이다.

시스템 운영 및 유지보수 전략

프로토타입의 성공적인 검증과 서비스화 이후, 시스템의 안정적인 운영과 꾸준한 성능 향상을 위해서는 체계적인 운영(Ops) 및 유지보수(Maintenance) 전략이 필수적이다. 이는 단순히 서버를 켜두는 것을 넘어, 시스템의 건강 상태를 지속적으로 점검하고, 사용자 피드백을 수집하여 지능형 시스템의 '지능'을 꾸준히 강화하는 지속적인 과정을 포함한다.

모니터링 및 로깅

시스템의 안정성을 확보하기 위해 실시간 모니터링과 체계적인 로깅 시스템을 구축해야 한다. FastAPI는 `loguru`와 같은 라이브러리를 사용하여 애플리케이션 로그를 생성할 수 있다 [25]. 이 로그는 각 요청의 경로(path), 메서드(method), 응답 시간(response time), 상태 코드(status code), 그리고 에러 발생 시 스택 트레이스(stack trace)를 포함해야 한다. 이러한 로그는 로그 관리 도구(Splunk, ELK Stack 등)로 집중화하여 시스템의 전반적인 건강 상태를 파악하고, 문제가 발생했을 때 원인을 신속하게 진단하는 데 활용될 수 있다.

또한, Prometheus와 같은 도구를 사용하여 시스템의 메트릭(metrics)을 수집해야 한다. CPU 사용률, 메모리 사용량, 디스크 I/O, 데이터베이스 연결 수, 그리고 LLM API 호출 지연 시간 등 중요한 지표들을 지속적으로 모니터링하여, 시스템이 정상 범위를 벗어났을 때 알림을 보내는 Alerting 규칙을 설정해야 한다. 특히, LLM API 호출 지연 시간은 시스템의 응답성에 직접적인 영향을 미치므로 가장 중요한 모니터링 대상 중 하나이다.

사용자 피드백 및 지능 강화

AI 가이드 시스템의 가치는 시간이 지남에 따라 사용자의 피드백을 통해 '지능'이 강화되는 데 있다. 따라서 사용자로부터 직접적인 피드백을 받을 수 있는 메커니즘을 설계해야 한다. Frontend에는 '이 가이드가 도움이 되었나요? / '와 같은 간단한 평가 버튼을 추가하여 사용자의 만족도를 즉시 수집할 수 있다. 이 정보는 백엔드에서 집계되어 시스템의 성능을 정량적으로 평가하는 지표로 사용될 수 있다.

더 나아가, 사용자로부터 직접적인 수정 제안을 받는 기능을 도입할 수 있다. 예를 들어, "이 가이드의 3번 항목이 잘못되었어요"와 같은 피드백을 받았을 때, 이 피드백을 관리자에게 알리는 알림 시스템을 구축해야 한다. 이 피드백은 시스템의 지식 베이스를 업데이트하고, LLM의 답변을 개선하는 데 중요한 자료가 된다. 또한, 사용자의 질문과 그에 대한 최종적인 답변을 무작위로 추출하여 human-in-the-loop 방식으로 품질 검사를 수행하고, 그 결과를 LLM의 fine-tuning에 활용하는 방안도 고려할 수 있다.

지식 베이스(Knowledge Base) 관리

RAG 시스템의 핵심인 지식 베이스는 반도체와 같은 빠르게 변화하는 산업에서 매우 중요하다. 시스템이 제공하는 가이드의 정확성과 신뢰성은 언제나 최신의 정보에 기반해야 한다. 따라서 지식 베이스를 관리하는 시스템적인 프로세스가 필요하다. * 자동화된 데이

터 수집(Ingestion): HR 시스템에서 신규 정책 문서가 등록될 때 자동으로 Webhook을 통해 데이터 수집 파이프라인을 트리거하는 등의 방안을 고려할 수 있다. * 수동 업데이트 권한: 특정 분야의 전문가가 자신의 영역에 해당하는 문서를 직접 업데이트할 수 있는 관리자 인터페이스를 제공해야 한다. * 정보의 신뢰도 등급화: 내부 정책 문서와 외부 블로그 글을 구분하여, 각 정보의 신뢰도를 다르게 평가하여 RAG 시스템의 검색 결과에 반영할 수 있다. 예를 들어, 내부 문서는 외부 문서보다 더 높은 우선순위를 가질 수 있다.

보안 및 데이터 관리

대규모 사용자 시스템을 운영함에 있어 보안은 최우선 과제이다. 사용자의 질문은 회사 내부 정보나 개인 정보를 포함할 수 있으므로, 모든 데이터는 안전하게 처리되어야 한다. * 데이터 암호화: 전송 중(SSL/TLS) 및 저장 중(At Rest) 데이터 모두 암호화해야 한다. * 사용자 인증 및 권한 관리: 시스템의 특정 기능에 접근할 수 있는 사용자 그룹을 구분하는 Role-Based Access Control(RBAC)을 구현해야 한다. 예를 들어, 모든 직원은 가이드를 조회할 수 있지만, 관리자는 지식 베이스를 수정할 수 있어야 한다. * 정책 준수: GDPR, CCPA 등 관련 개인정보 보호법규를 준수해야 한다.

이러한 운영 및 유지보수 전략은 단순히 기술적인 문제를 해결하는 것을 넘어, 시스템이 조직 내에서 안정적으로 자리 잡고, 사용자로부터 신뢰를 얻으며, 지속적으로 가치를 창출하는 길을 보장하는 데 결정적인 역할을 할 것이다.

프로젝트 성공을 위한 최종 제언

본 보고서에서 제시한 방향성은 반도체 IDM의 10만명 사용자 문제 해결을 위한 AI 가이드 시스템 구축이라는 복잡하고 도전적인 프로젝트를 성공적으로 이끌기 위한 체계적인 로드맵이다. 성공적인 구현을 위해 몇 가지 핵심 제언을 마지막으로 정리하고자 한다.

첫째, 초기 목표의 낮추기(Beginner's Mindset). 사용자의 요구사항은 매우 거창하지만, 성공적인 프로토타입 구축을 위해서는 처음부터 모든 것을 완벽하게 구현하려는 시도를 버려야 한다. 앞서 제시한 4단계 프로토타입 구축 계획을 명확히 이해하고, 각 단계의 목표를 명확히 설정하며, 그 목표를 달성했을 때마다 성공을 축하하고 피드백을 반영하는 'Iterative Development' 방식을 채택해야 한다. 이는 시스템의 복잡성을 관리하고, 프로젝트의 진행 상황을 명확히 보여줌으로써 팀의 자신감을 높이는 데 기여할 것이다.

둘째, 디자인 사고를 통한 사용자 중심적 접근(User-Centric Design). 시스템의 핵심은 사용자이다. 사용자 여정 지도(UJM)를 만들고 요구사항을 정의하는 과정은 단순히 문서화 작업이 아니다. 이 과정을 통해 얻는 통찰력은 기술 선택부터 UI/UX 설계, 마케팅 전략에 이르기까지 프로젝트의 모든 단계에 영향을 미칠 것이다. 특히, 사용자의 '고통점'을 명확히 파악하고, 그것들을 해결하는 데 집중하는 것이 프로젝트의 성공을 좌우하는 핵심 변수가 될 것이다.

셋째, 테스트와 평가(T&E)의 중요성 강조. 프로토타입을 구축한 후에는 그저 배포하는 것이 아니라, 실제 사용자나 퍼소나를 대상으로 체계적인 테스트를 수행해야 한다. 사용자의 문제 해결 효율성, 가이드의 정확성, 시스템의 사용 용이성 등 다양한 지표를 측정하여 시스템의 성능을客觀地评估. 이 평가 결과는 후속 프로젝트에 대한 투자 결정을 내릴 때 가장 중요한 근거가 될 것이다.

넷째, 팀의 전문성 강화 및 협업 촉진. 본 프로젝트는 SW, 제조, 연구개발 등 다양한 전문 분야를 아우르므로, 각 분야의 전문가가 팀에 참여하여 지식을 공유하고 협업하는 문화가 중요하다. 특히, 기술 팀은 비전문적인 언어로 된 사용자의 문제를 어떻게 정확히 이해하고 처리할 수 있을지를, 비기술 팀은 기술 팀의 기술적 한계를 이해하고 현실적인 기대치를 설정하는 데 협력해야 한다.

마지막으로, 미래를 위한 설계(Future-Proof Design). 기술은 빠르게 변화하고 있다. LangGraph나 LLM의 새로운 기능이 출시될 때마다, 시스템을 쉽게 업데이트하고 확장할 수 있는 유연한 아키텍처를 유지해야 한다. 각 기술 스택이 수행하는 역할을 명확히 분리하고, 독립적으로 교체할 수 있는 설계를 함으로써, 미래의 기술 변화에 유연하게 대응하고 시스템의 수명을 연장할 수 있다.

이러한 제언들을 바탕으로, 본 프로젝트는 단순한 기술 시스템을 넘어, 10만명의 직원을 지원하고 조직의 지식을 지능적으로 관리하는 핵심적인 인프라로 자리매김할 수 있을 것이다. 성공적인 프로젝트 추진을 기원한다.

참고문헌

1. 고객 여정 지도 작성 방법 https://advertising.amazon.com/ko-kr/library/guides/customer-journey-map
2. 사용자 여정 지도 (User Journey Map) - JX Design - 티스토리 https://jxdesign.tistory.com/26
3. [기초/이론] 사용자 여정 지도, 테스크 플로우, 스토리 맵 https://2suyeon.tistory.com/13
4. #7. 사용자 여정 지도 (User Journey Map) : 네이버 블로그 https://m.blog.naver.com/khj90733/221688794034
5. 10화 Journey Map을 어떻게 만드는 것일까? - 브런치 https://brunch.co.kr/@uxuxlove/241
6. 고객 여정 지도(Customer Journey Map) 정리 - velog https://velog.io/@kwon0koang/%EA%B3%A0%EA%B0%9D-%EC%97%AC%EC%A0%95-%EC%A7%80%EB%8F%84Customer-Journey-Map%EB%A1%9C-%EC%84%9C%EB%B9%84%EC%8A%A4-%EC%84%B1%EA%B3%B5%EC%9D%84-%EC%9D%B4%EB%81%84%EB%8A%94-%EB%B0%A9%EB%B2%95
7. 저니맵(Journeymap)은 왜 필요하고 어떻게 써야하나요? - 아람말싸미 https://aram5.tistory.com/14
8. 여정 맵 시작하기 (Getting Started with Journey Maps) https://kr.docs.enterprise.appier.com/docs/getting-started-with-journey-maps
9. [1주차] 1-3. 모바일 카카오톡 앱의 '선물 보내기' 에 대한 사용자 여정 ... https://m.blog.naver.com/tabpink123/223095829621
10. 반도체 후공정 테스트 하우스 관련주 - 테스나, 네패스아크, 엘비세미콘 ... https://jublary.tistory.com/46

11. 반도체 후공정 산업공부① (웨이퍼테스트,패키징테스트) - 블로그 https://m.blog.naver.com/258863/222934379784
12. 반도체 후공정 정리 (패키징) https://engineering-ladder.tistory.com/139
13. 반도체 공정의 이해 - 홍길동 라이프 - 티스토리 https://gildong-life.tistory.com/entry/%EB%B0%98%EB%8F%84%EC%B2%B4-%EA%B3%B5%EC%A0%95%EC%9D%98-%EC%9D%B4%ED%95%B4
14. 반도체 공정 - 나무위키 https://namu.wiki/w/%EB%B0%98%EB%8F%84%EC%B2%B4%20%EA%B3%B5%EC%A0%95
15. 반도체 8대 공정, 10분만에 이해하기 - 브런치 https://brunch.co.kr/@wyz/62
16. 반도체 제조 공정 단계별 정리 (웨이퍼 제조, 프론트엔드, 백엔드 공정) https://computing-jhson.tistory.com/144
17. [반도체 후공정 1편] 반도체 테스트의 이해 (1/11) - SK하이닉스 뉴스룸 https://news.skhynix.co.kr/seominsuk-column-test/
18. 웨이퍼 패키징 및 테스트 요약 - shenminghu456 - 티스토리 https://shenminghu456.tistory.com/1191
19. 반도체 후공정 (Assembly 및 패키지) Rev.1 : 네이버 블로그 https://blog.naver.com/stoxxer/222976744039
20. 반도체 8대 공정 9탄. 외부환경으로부터 반도체를 보호하는 패키징 ... https://news.samsungsemiconductor.com/kr/%EB%B0%98%EB%8F%84%EC%B2%B4-8%EB%8C%80-%EA%B3%B5%EC%A0%95-9%ED%83%84-%EC%99%B8%EB%B6%80%ED%99%98%EA%B2%BD%EC%9C%BC%EB%A1%9C%EB%B6%80%E
%EB%B0%98%EB%8F%84%EC%B2%B4%EB%A5%BC-%EB%B3%B4%ED%98%B8-2/
21. <테스트(Test) 공정>반도체 제조 공정의 발전 방향과 투자 기회(2부) https://m.blog.naver.com/chcmg2022/222876165501
22. 8탄, 완벽한 반도체로 태어나기 위한 첫 번째 테스트 'EDS공정' https://semiconductor.samsung.com/kr/support/tools-resources/fabrication-process/eight-essential-semiconductor-fabrication-processes-part-8-eds-electrical-die-sorting-for-the-perfect-chips/
23. 반도체 패키징 - FOWLP 공정 // FOWLP 공정의 개요, TSV기술 ... https://hongya-world.tistory.com/entry/%EB%B0%98%EB%8F%84%EC%B2%B4-%ED%8C%A8%ED%82%A4%EC%A7%95-FOWLP-%EA%B3%B5%EC%A0%95-FOWLP-%EA%B3%B5%EC%A0%95%EC%9D%98-%EA%B0%9C%EC%9A%94-TSV%EA%B8%B0%EC%88%A0-FOWLP-%EA%B3%B5%EC%A0%95-%EC%88%9C%EC%84%9C
24. Deploying LangGraph with FastAPI: A Step-by-Step Tutorial - Medium https://medium.com/@sajith_k/deploying-langgraph-with-fastapi-a-step-by-step-tutorial-b5b7cdc91385
25. FastAPI로 머신 러닝 마이크로서비스 구현하기 : 네이버 블로그 https://blog.naver.com/uclick2016/222863169111?viewType=pc

26. 머신 비전 및 딥러닝을 사용해 반도체 제조에서 경쟁 우위 확보 - Cognex https://www.cognex.com/ko-kr/blogs/deep-learning/how-to-gain-a-competitive-edge-in-semiconductor-manufacturing-with-machine-vision-and-deep-learning
27. [langgraph] FastAPI와 LangGraph로 기본 RAG를 Streaming으로 ... https://rudaks.tistory.com/entry/langgraph-FastAPI%EC%99%80-LangGraph%EB%A1%9C-%EA%B8%B0%EB%B3%B8-RAG%EB%A5%BC-Streaming%EC%9C%BC%EB%A1%9C-%EC%B6%9C%EB%A0%A5%ED%95%98%EA%B8%B0
28. 반도체 이야기 #21 : 종합 반도체 회사, IDM - 네이버 블로그 https://m.blog.naver.com/kasagave/223150491026
29. 반도체 기업 형태(IDM, IP기업, 팹리스, 디자인하우스, 파운드리, OSAT) https://miniva.tistory.com/15
30. 반도체산업 쉽게 이해하기 4편 : 반도체 업체의 유형 상세히 살펴보기 https://blog.naver.com/hodolry/221720463900
31. 반도체 회사에도 종류가 있다? (IDM, 팹리스, 파운드리, OSAT) https://spark104.com/81
32. 파운드리? 팹리스? 반도체 생태계 한눈에 보기! https://news.samsungsemiconductor.com/kr/%ED%8C%8C%EC%9A%B4%EB%93%9C%EB%A6%AC-%ED%8C%B9%EB%A6%AC%EC%8A%A4-%EB%B0%98%EB%8F%84%EC%B2%B4-%EC%83%9D%ED%83%9C%EA%B3%84-%ED%95%9C%EB%88%88%EC%97%90-%EB%B3%B4%EA%B8%B0/
33. 반도체 기업 유형 - 까망 하르방 - 티스토리 https://zoosso.tistory.com/722
34. 반도체 산업 비즈니스 모델 (IDM, 팹리스, 파운드리, 패키징) https://venture-capital.tistory.com/80
35. 반도체 생태계 _반도체 기업 형태, 파운더리, 메모리 정리 https://dh2note.tistory.com/158
36. 전 세계 반도체 회사 정리 : 종합반도체기업(IDM), 칩리스, 팹리스 ... https://m.blog.naver.com/kim1222ys/223022000099.
37. IDM과 파운드리 모델: 반도체 제조의 두 가지 접근법 비교 - 마켓소호 https://marketsoho.tistory.com/entry/IDM%EA%B3%BC-%ED%8C%8C%EC%9A%B4%EB%93%9C%EB%A6%AC-%EB%AA%A8%EB%8D%B8-%EB%B0%98%EB%8F%84%EC%B2%B4-%EC%A0%9C%EC%A1%B0%EC%9D%98-%EB%91%90-%EA%B0%80%EC%A7%80-%EC%A0%91%EA%B7%BC%EB%B2%95-%EB%B9%84%EA%B5%90
38. [반도체 공부] 자소서에 사용할 반도체 기업 종류를 알아보자 ! (IDM ... https://estj-egnr.tistory.com/1
39. 반도체 생산방식에 따른 반도체 기업 분류 - 공정 단계를 기준으로 https://donmyoung.tistory.com/entry/%EB%B0%98%EB%8F%84%EC%B2%B4-%EC%83%9D%EC%82%B0%EB%B0%A9%EC%8B%9D%EC%97%90-%EB%94%B0%EB%A5%B8-%EB%B0%98%EB%8F%84%EC%B2%B4-

%EA%B8%B0%EC%97%85-%EB%B6%84%EB%A5%98-%EA%B3%B5%EC%A0%95-%EB%8B%A8%EA%B3%84%EB%A5%BC-%EA%B8%B0%EC%A4%80%EC%9C%BC%EB%A1%9C

40. 반도체란 무엇인가? IC란 무엇인가? 한 장의 그림으로 이해하는 ... https://www.lorric.com/kr/Industries/semiconductor-industry

41. 반도체 기업 분류 (팹리스,DSP,파운드리,IDM,OSAT) - 바람부는대로 https://windlov2.tistory.com/entry/%EB%B0%98%EB%8F%84%EC%B2%B4-%EA%B8%B0%EC%97%85-%EB%B6%84%EB%A5%98-%ED%8C%B9%EB%A6%AC%EC%8A%A4-%ED%8C%8C%EC%9A%B4%EB%93%9C%EB%A6%AC-IDM-OSAT-DSP

42. 01. Cleaning 공정 2024년 최신 Ver (feat. 물리적, 화학적, RCA 등) https://hongya-world.tistory.com/entry/Cleaning-%EA%B3%B5%EC%A0%95-2024%EB%85%84-%EC%B5%9C%EC%8B%A0-Ver-feat-%EB%AC%BC%EB%A6%AC%EC%A0%81-%ED%99%94%ED%95%99%EC%A0%81-RCA-%EB%93%B1

43. 반도체 공정에서의 Wafer Map Image 분석 방법론 - 대한산업공학회지 https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE06358343

44. [반도체 공정] 반도체? 이 정도는 알고 가야지:(8) Wafer test ... https://www.skcareersjournal.com/1203

45. [반도체 후공정 8편] 웨이퍼 레벨 패키지 공정 (8/11) - SK하이닉스 뉴스룸 https://news.skhynix.co.kr/seominsuk-column-wafer-level-package-2/

46. 웨이퍼 테스트(Wafer Test): 보이지 않는 반도체 속 결함을 가려내는 기술 https://hypersever.tistory.com/entry/%EC%9B%A8%EC%9D%B4%ED%8D%BC-%ED%85%8C%EC%8A%A4%ED%8A%B8Wafer-Test-%EB%B3%B4%EC%9D%B4%EC%A7%80-%EC%95%8A%EB%8A%94-%EB%B0%98%EB%8F%84%EC%B2%B4-%EC%86%8D-%EA%B2%B0%ED%95%A8%EC%9D%84-%EA%B0%80%EB%A0%A4%EB%82%B4%EB%8A%94-%EA%B8%B0%EC%88%A0