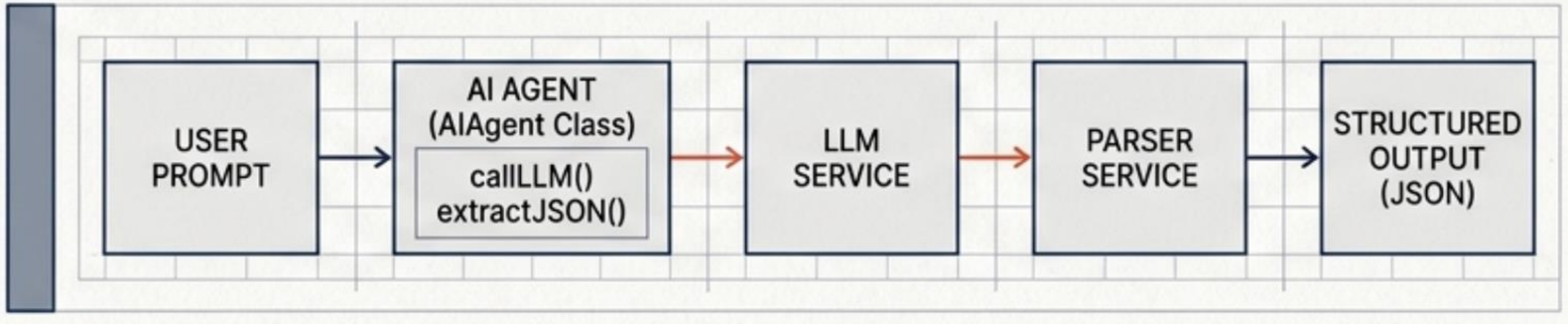
AI Agent
개발 실습 강의

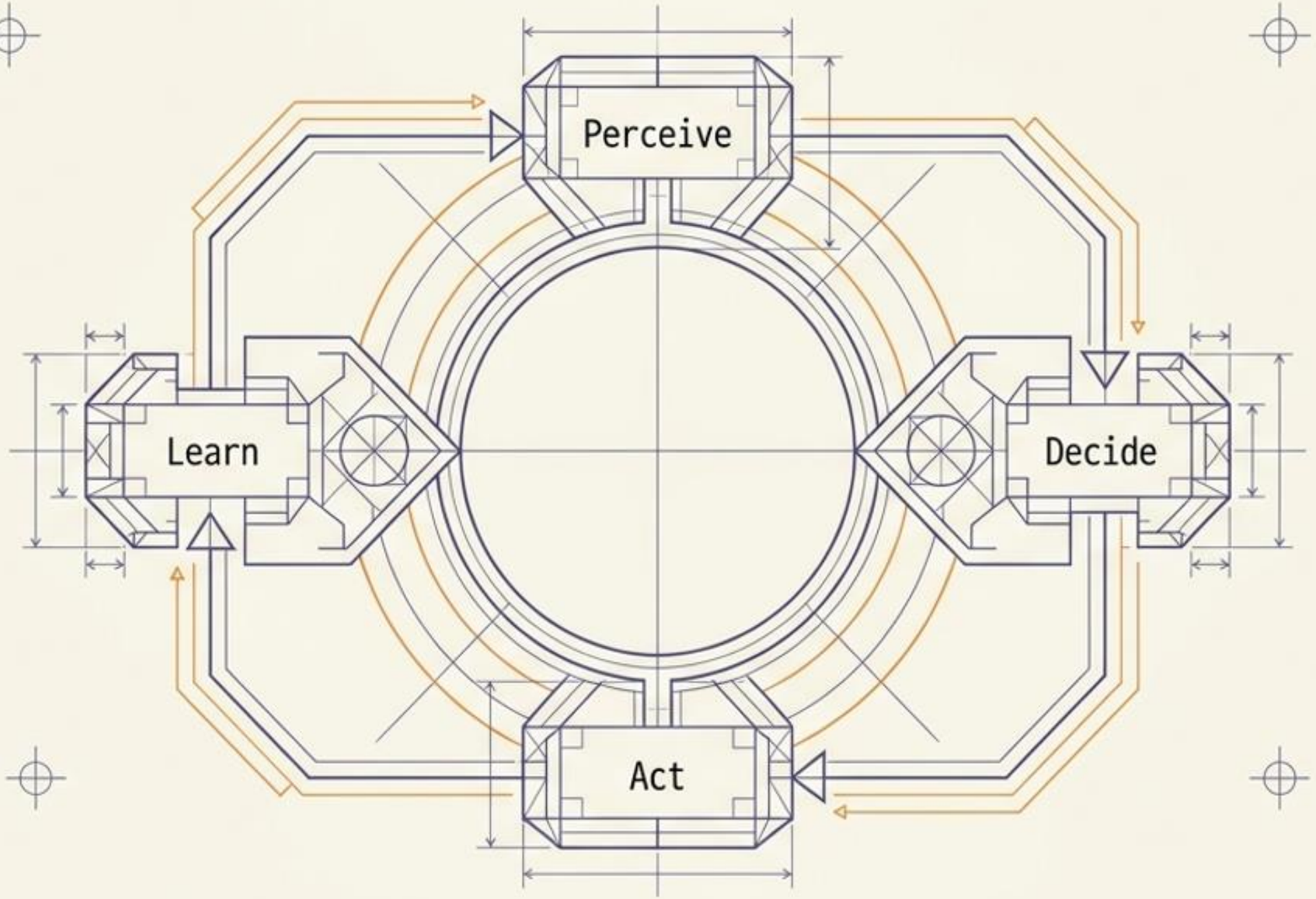
강사 : 전한철 위원(SKAX)

```
// AI 에이전트 코어 기능 정의
import { LLMService, ParserService } from './services';

interface AgentConfig {
  model: string;
  temperature: number;
}

class AIAgent {
  private llm: LLMService;
  private parser: ParserService;

  constructor(config: AgentConfig) {
    this.llm = new LLMService(config.model,
      config.temperature);
    this.parser = new ParserService();
    console.log('AI Agent 초기화 완료: ', config.model);
  }

  public async executeTask(prompt: string):
  Promise<any> {
    //
    console.log('작업 실행: ', prompt);
    const llmResponse = await this.callLLM(prompt);

    // JSON 데이터 추출 및 파싱
    const result = this.extractJSON(llmResponse);
    return result;
  }

  // LLM 호출 함수
  private async callLLM(input: string): Promise<string> {
    //
    const response = await this.llm.generate(input);
    return response;
  }

  // JSON 구조 데이터 추출 함수
  private extractJSON(text: string): any {
    const parsedData = this.parser.parse(text);
    if (!parsedData) {
      throw new Error('JSON 추출 실패');
    }
    return parsedData;
  }
}
```



HANS17 ACADEMY: 프로덕션 레벨 AI Agent 실전 배포 마스터플랜

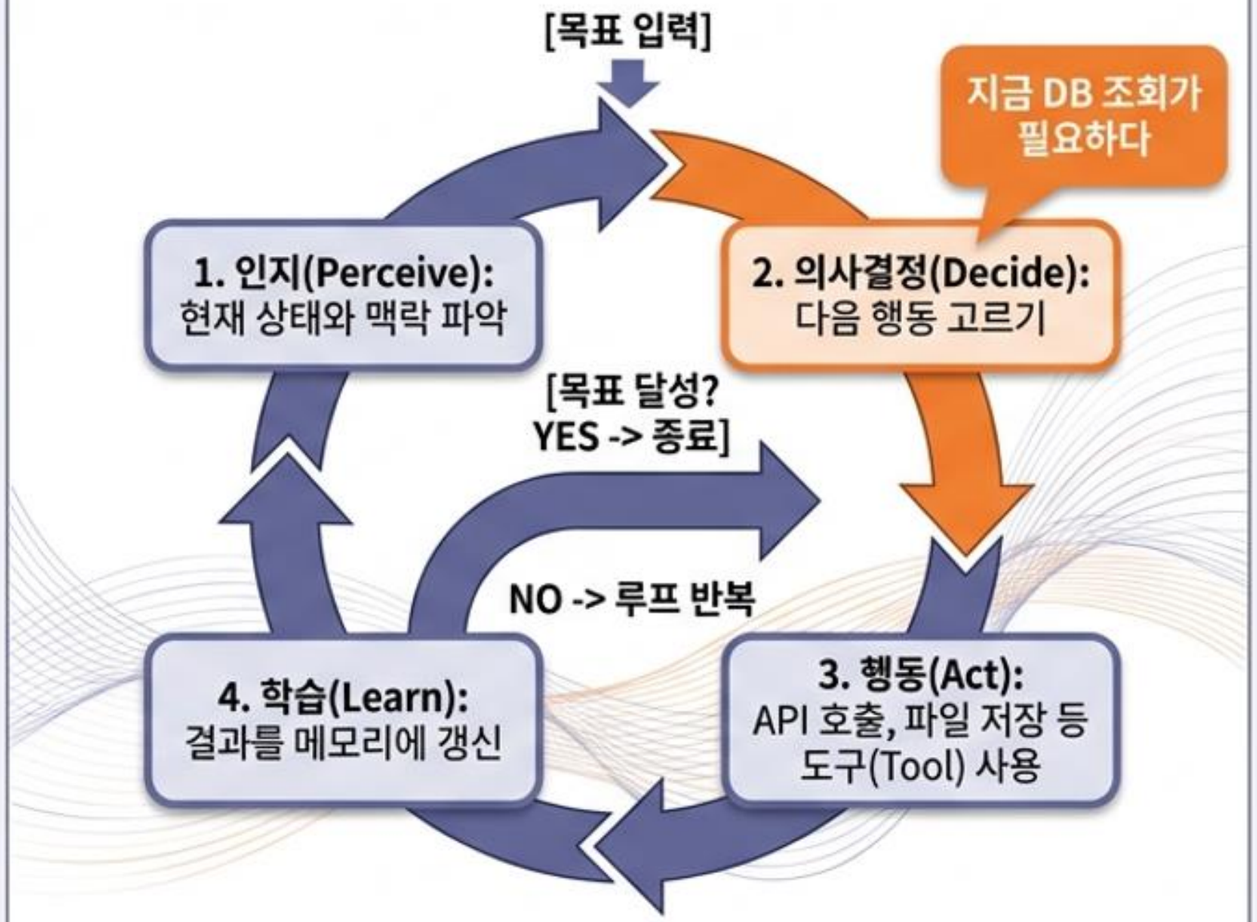
프레임워크 없는(No-framework) 결정적 통제와 실전 아키텍처 설계
이론 및 실습 완벽 통합 가이드



패러다임 시프트: 챗봇에서 에이전트로

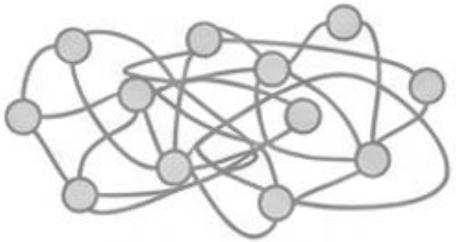
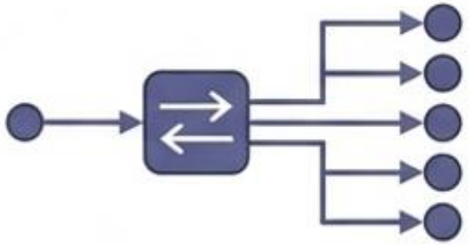
단순 챗봇 - 선형 구조

AI 에이전트 - 무한 루프 구조



Agent는 대화하는 봇이 아니라 목표를 위해 스스로 도구를 쓰는 시스템입니다.

아키텍처 철학: 프레임워크의 함정과 Dispatcher 패턴

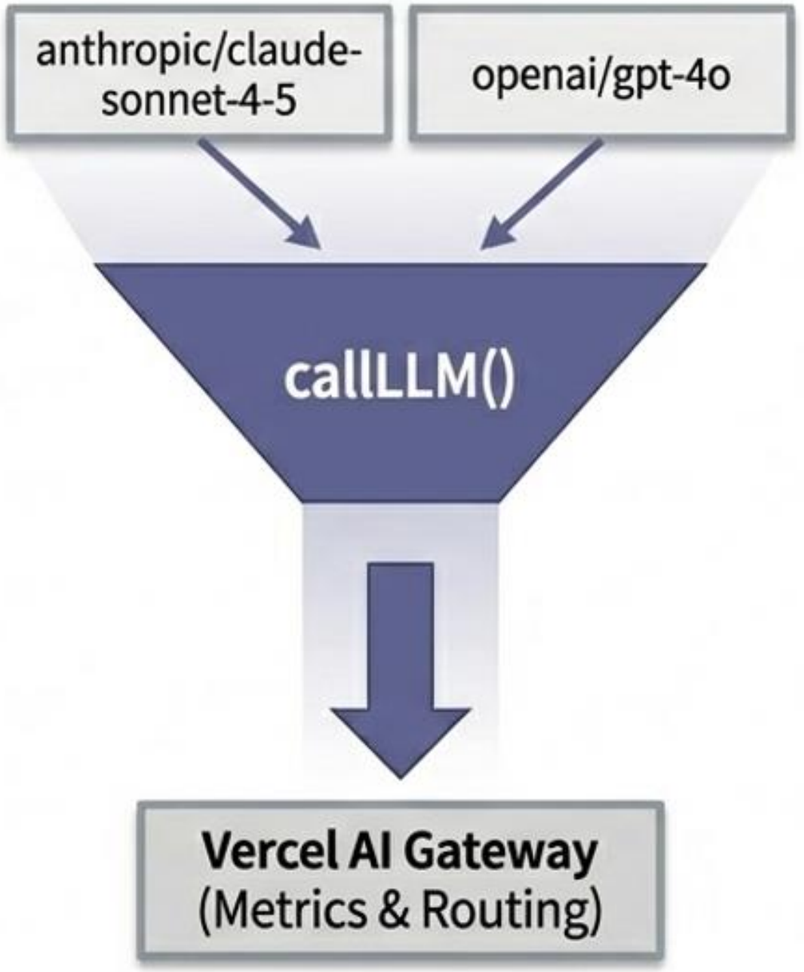
블랙박스 프레임워크 (LangGraph / AutoGen)	명시적 라우팅 (No-framework)
복잡한 분기, 비결정적 흐름, 비용 예측 불가, 디버깅 어려움.	완전한 제어권, 하드코딩 분기, 결정성 100%, 보안 격리.
	

프레임워크가 알아서 해주는 블랙박스를 버려야 프로덕션에서 버틸 수 있습니다.

에이전트의 두뇌: 토큰 경제학과 callLLM() 추상화

한국어 토큰 소모의 현실

한국어 토큰 소모
영어 대비 약 2.5배



extractJSON() 강건 파서 로직

```
raw.replace(/^```
  `(?:json)?\n'?/i, '')
```

- Prompt Caching: 반복 컨텍스트 비용 최대 90% 절감
- JSON 강제: jsonMode와 스키마 주입으로 결정론적 파싱

단기/장기 기억: 지식 단절과 환각을 끊어내는 RAG 파이프라인

임베딩
(text-embedding-3-small, 1536d)

→

코사인 유사도 검색
(Threshold 0.15~0.2 적용)

→

프롬프트 주입
(근거 밖 답변 금지)

→

답변 생성
(출처 인용 강제)

환각 차단 4원칙 (Anti-Hallucination Shields)

- **Score 임계값:** 매치 점수 0.15 미만은 과감히 버림
- **출처 인용 강제:** [출처: {source}] 표기 없이는 생성 불가
- **청크 사이즈 조정:** 300~600 토큰 단위로 문맥 조율
- **모른다의 명시화:** 문서에 없으면 억지로 지어내지 않음

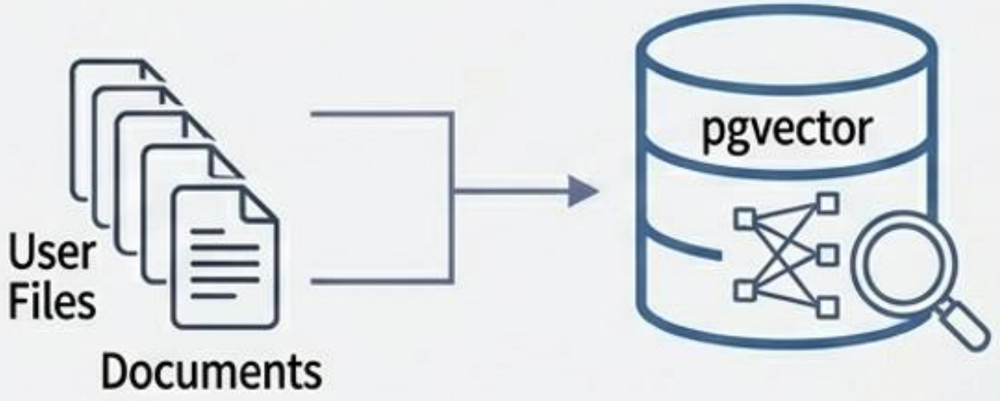
RAG 스케일링 결단: 규모가 방식을 결정한다

[In-Memory Cosine (강사 Agent 적용)]

- **Scale:** 1만 청크 미만, 고정 데이터
- **Tech:** 별도 DB 없음. content/embeddings.json 로드 후 수학 공식 직접 계산.
- **Pros:** 인프라 비용 0, Vercel 최적화 (200ms 응답).

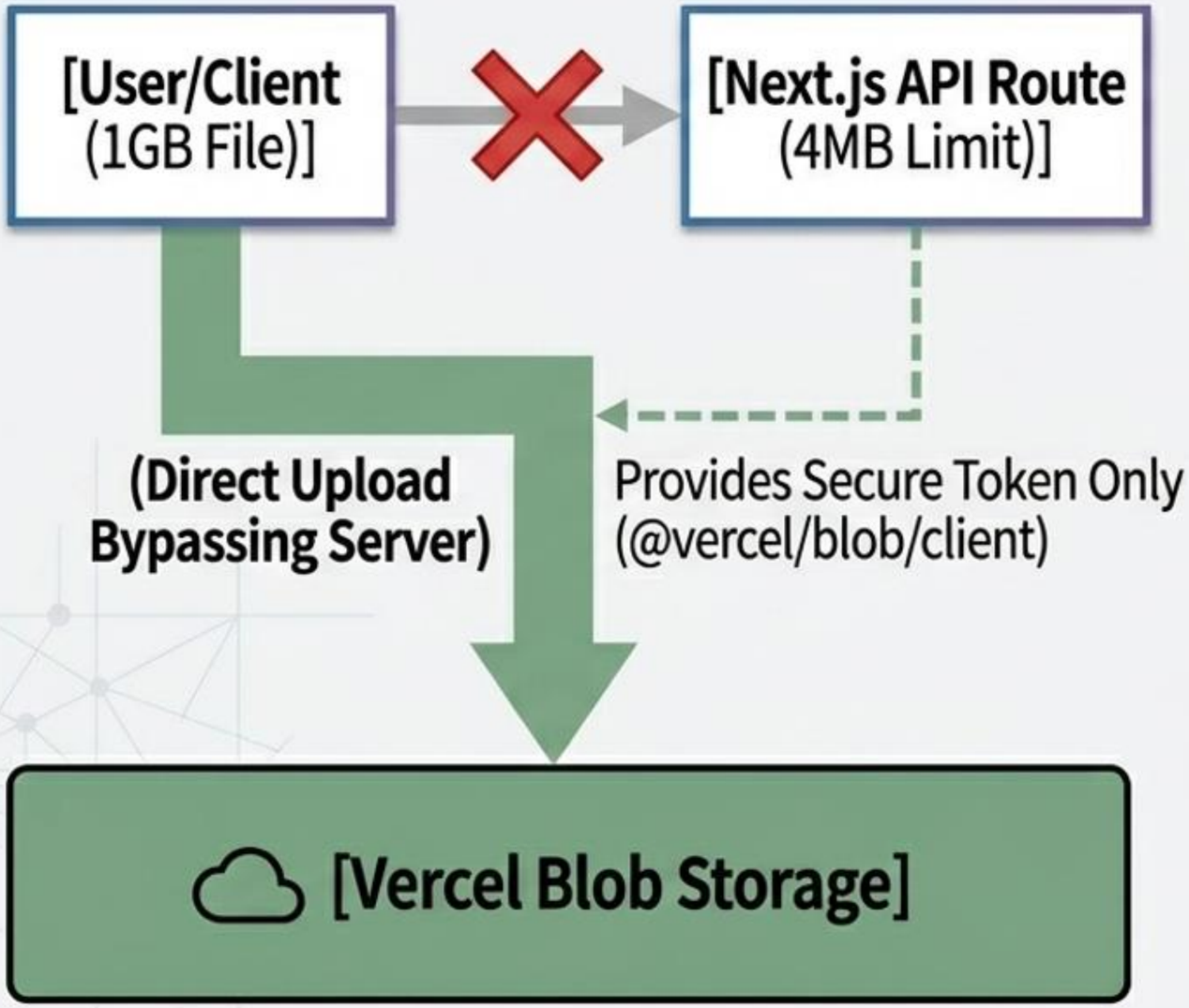
[Vector DB (RAG Lab 적용)]

- **Scale:** 수만 ~ 700만 자 이상, 사용자 가변 업로드
- **Tech:** Supabase pgvector 확장 + HNSW 인덱스 + match_documents RPC.
- **Pros:** 영속성 보장, 메타데이터 필터링 (카테고리/태그) 가능.



청크 스키마({content, embedding, metadata})만 동일하게 유지하면
두 방식 간 마이그레이션은 거의 복붙(Copy-Paste)입니다.

상태 유지와 대용량 저장소: Vercel Blob KV & Direct Upload

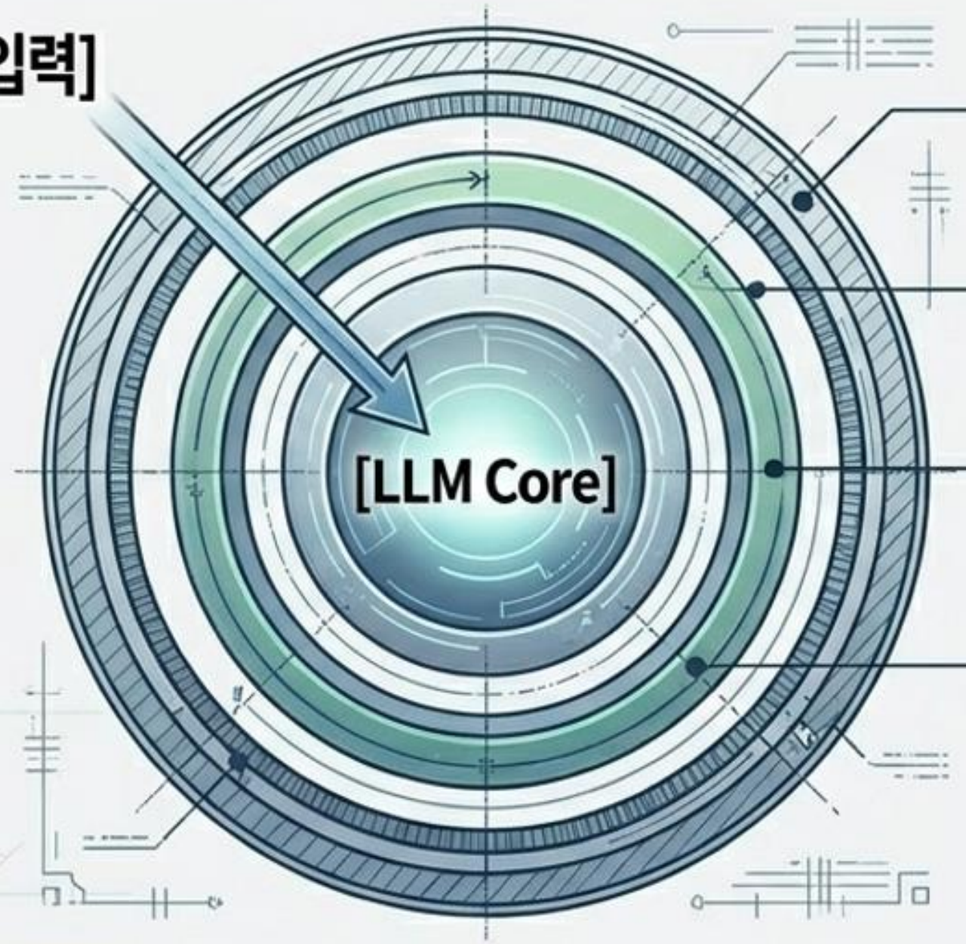


blob-kv.ts

- **kvPutText()** -> JSON 직렬화 저장
  (addRandomSuffix: false 덮어쓰기)
- **kvGetJson()** -> 상태 복원
- **kvDelete()** -> 사용 후 원본 즉시 삭제
  (개인정보 보호)

Prefix 경로(projects/<id>/...)를 이용한 테넌트 격리.

보안 설계: 프롬프트 인젝션 심층 방어(Defense in Depth) 레이어



Ring 1 (Outer): System 분리: System/User 역할을 절대 문자열로 합치지 않음

Ring 2: 길이 Cap: 사용자 입력을 서버 사이드에서 최대 2,000자로 강제 절사

Ring 3: 키워드 필터: ignore previous instructions 등 알려진 패턴 정규식 차단

Ring 4 (Inner): 구조적 래핑: 사용자 입력을 **XML 태그 <user-input>**로 감싸고 태그 밖 명령 무시 지시

Tool Use 시 도구의 args는 반드시 서버 사이드에서 Zod 스키마로 타입 검증해야 API 오염을 막을 수 있습니다.

응용 1: 결정론적 조립 - 보고서 생성 에이전트

Phase 1: 의미 추출
(LLM - 창의성 통제)

[혼돈의 원문 회의록]

LLM
(claude-sonnet,
temp: 0.2,
JSON Schema 강제)

Phase 2: 문서 조립
(순수 TypeScript - 100% 결정적)

[구조화된 JSON 데이터
(Highlights, Schedule, Flow)]

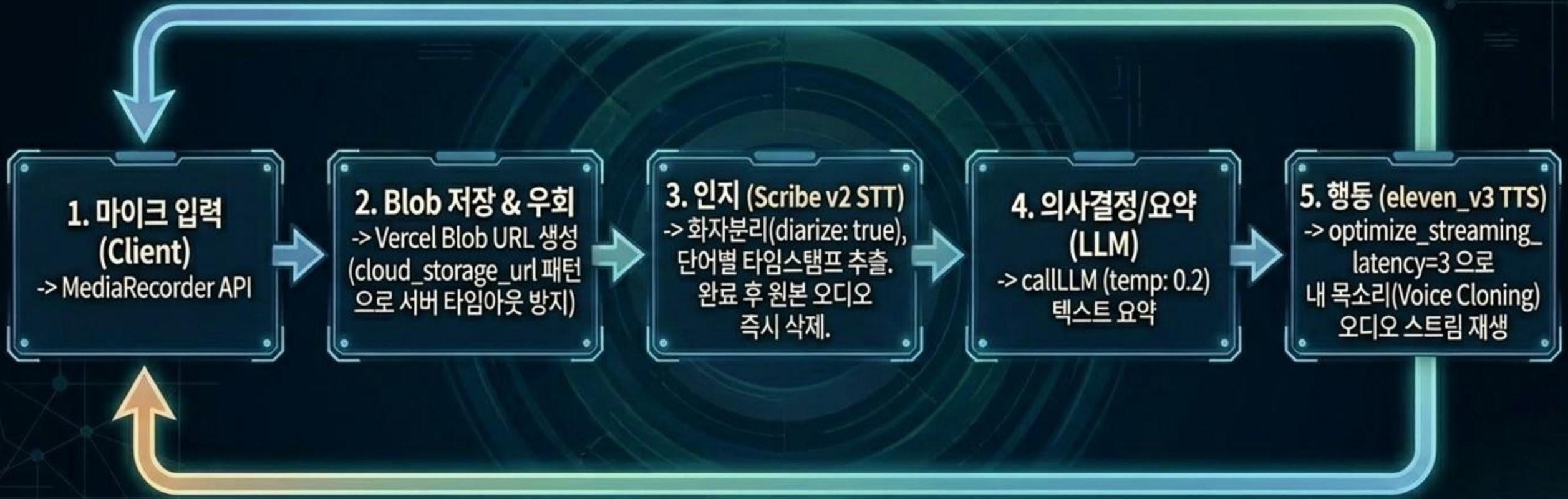
buildMarkdownReport()
(순수 함수, 하드코딩된
마크다운 템플릿)

완벽히 일관된 6단 마크다운 보고서

[완벽히 일관된 6단 구조
마크다운 보고서]

5회 반복 일관성 테스트 ✓
✓ Flow 분산: **1** 이내
✓ Highlights 개수 분산: **0**

응용 2: 풀-루프 음성 파이프라인 (ElevenLabs 통합)



주의: HTMLAudioElement.play()는 반드시 브라우저 정책상 정책상 사용자 제스처(클릭) 이후 실행되어야 합니다.

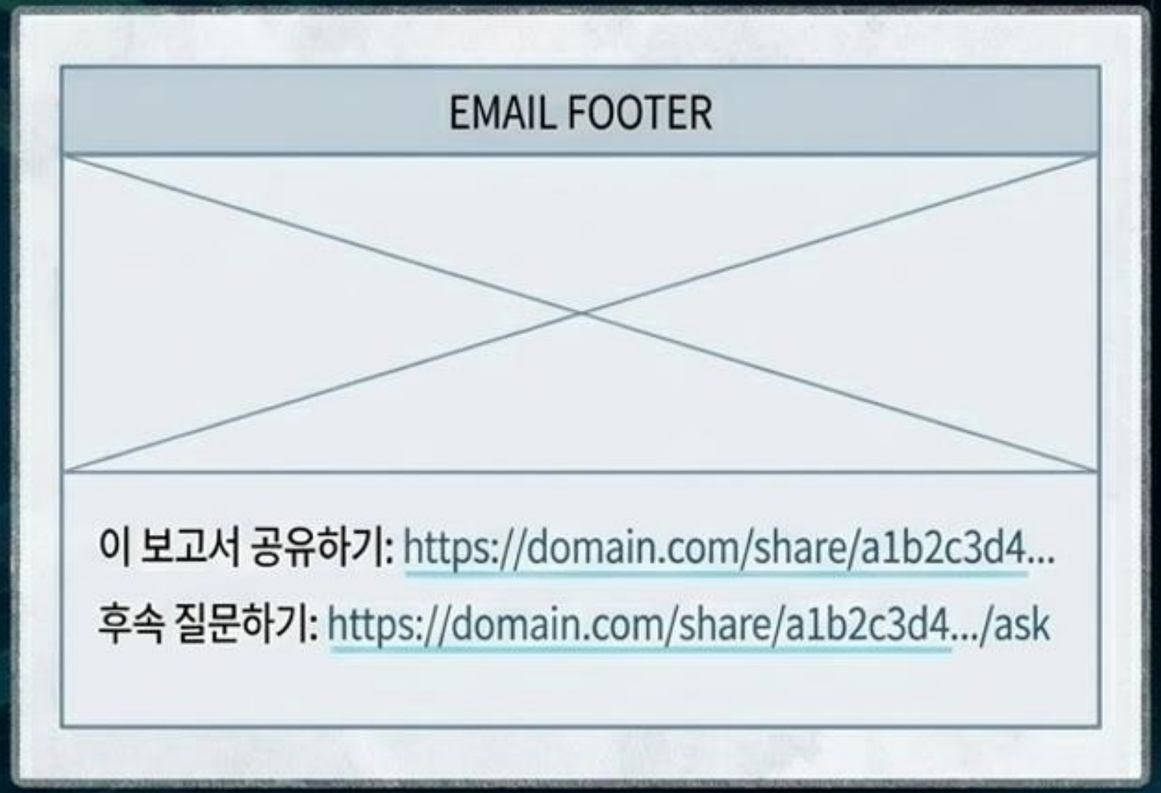
확산 및 상호작용: 메일 발송과 비로그인 Q&A 공유망

발송: Markdown 보고서를 HTML로 변환하여 Resend SDK로 전송

토큰화: crypto.getRandomValues(16) 기반의 예측 불가능한 128-bit 공유 토큰 생성 후 Blob KV 저장

공유 페이지 (/share/[token]): 수신자가 로그인 없이 안전하게 URL 접근

후속 질의 (Ask): 공유된 원본 보고서를 컨텍스트로 삼아 Claude에 직접 스트리밍 질의

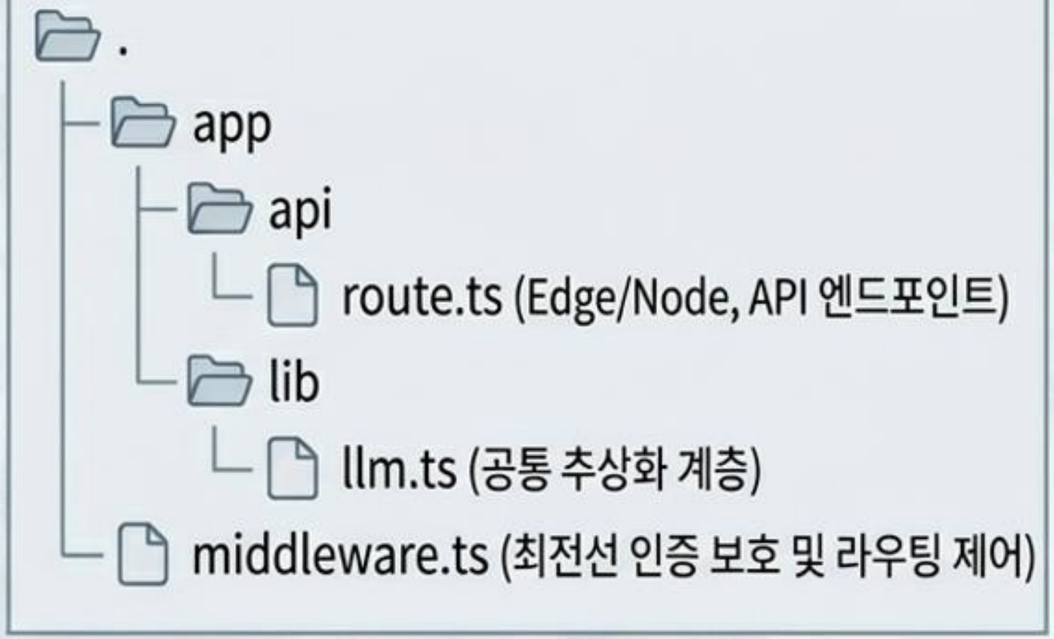


개발 워크플로우: Claude Code와 Next.js 융합

.claude/settings.json

- **MCP** (Model Context Protocol): **Vercel, Supabase, ElevenLabs** 커넥터 활성화 (인프라 직접 조작)
- **Skills**: /deploy-check 같은 반복 명령어 박제 (SKILL.md)
- **Hooks**:
  -> 파일 수정 후 자동 포맷팅 스크립트 연결

Architecture Layout



손코딩 대신 프롬프트로 지시하며, 9개의 서비스 API 키를 .env.local 단일 파일로 결합합니다.

프로덕션 배포 및 시스템 관측 (Observability)

Deployment CLI

Step 1

vercel link
(프로젝트 연결)

Step 2

vercel env pull
(환경변수 동기화)

Step 3

vercel --prod
(라이브 배포)

AI Gateway Metrics

Model Latency (ms) Token Costs ($)

Fallback Routing

```
"fallback": ["openai/gpt-4o"]
```

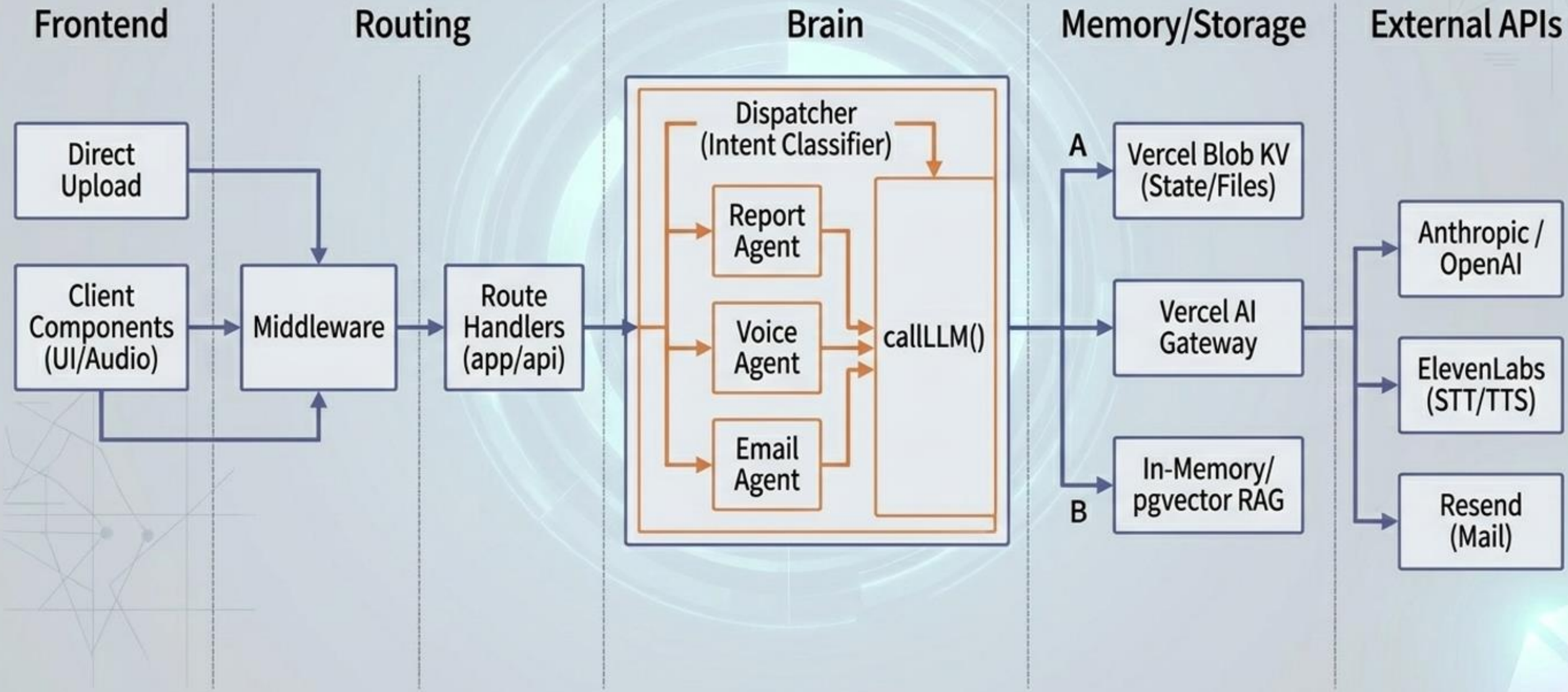
→ Claude 429 에러 시 자동 우회

Log Drains

Vercel Logs

- **Sentry** (에러 추적)
- **Datadog** (장기 메트릭)

종합: Hans17 AI Agent 마스터 아키텍처 조감도



통제권을 잃지 말고 확장하라.
규모가 방식을 결정한다.

- 프레임워크의 블랙박스에서 벗어나 결정적 파이프라인을 구축하십시오.
- LLM의 창의성은 통제하고, 시스템의 일관성은 코드로 보장하십시오.
- 논리와 감성, 과학과 사운드의 경계를 넘나드는 당신만의 에이전트를 설계하십시오.