Assignment03 요약

1. API 구성

- **엔드포인트**: `POST /chat` (FastAPI, `assignment03/app/main.py`)
- **요청 스키마**: `{ "message": str, "prompt_version": str, "model": Optional[str] }`
- **응답 스키마**: `{ "reply", "model", "prompt_version", "latency_ms", "total_tokens", "timestamp" }`
- **LLM 백엔드**: `OPENAI_API_KEY`가 있으면 OpenAI Chat Completions, 없으면 deterministic 스텁. 실습에서는 스텁을 사용했습니다.

2. 실험 설정

- **프롬프트 버전**
  - `v1`: 간단한 Q&A 요약에 집중
  - `v2`: 운영 팁과 강조(굵은 글씨)를 포함해 설명 길이를 늘림
- **모델 이름**: `gpt-4o-mini-stub` (스텁이므로 접미사 `-stub`)
- **호출 횟수**: 버전당 7회 (총 14회) – `scripts/send_requests.py` 사용

3. 로그 기반 통계

- **로그 위치**: `assignment03/logs/llm_responses.csv`

prompt_version	calls	평균 latency_ms	평균 total_tokens
v1	7	123.71	91.71
v2	7	152.29	106.86

- 모델 단일이므로 `gpt-4o-mini-stub` 14회 평균 지연 138.00ms 기록

4. 인사이트

1. v2 템플릿은 평균 토큰 수가 약 16% 많고 지연시간도 23%가량 길어져, 운영 팁을 추가하는 대가로 처리 시간이 늘어났습니다.
2. 롱폼 답변이 꼭 필요하지 않은 경우 v1에 핵심 문장만 덧붙이는 편이 토큰·지연 비용 면에서 더 유리합니다.
3. latency, tokens, provider를 모두 기록해 둔 덕분에 단일 모델이라도 호출 패턴을 쉽게 비교할 수 있고 Langfuse 등 외부 관제 도구로의 이전도 간단합니다.