

DAY 2 / CLASS 1

Industrial LLM & Prompt Engineering

데이터의 바다에서 '언어'로 길을 찾다



TIME

10:00 ~ 11:00 (60min)



TOPICS

Hallucination, RAG, Function Calling



INDUSTRIAL AI EDU PROGRAM 2025

Recap: "어제 만든 우리의 공장, 무엇이 부족할까요?"

DAY 1: NEURAL NETWORK



말 못하는 공장

신경망(IoT)은 연결되었지만
단순한 신호만 보낼 뿐입니다.

- ⚡ 고장 나면 빨간불만 깜빡임
- 🔇 "어디가 아픈지" 설명 불가



Evolution

DAY 2: LLM INTERFACE

"3호기 베어
링 점검 필
요!"

대화하는 공장

인공지능이 데이터를 해석하고
사람의 언어로 소통합니다.

- 🗣 문제의 원인과 조치법 설명
- 👤 누구나 쉽게 서비스 제어 가능



Language Interface

오늘 우리는 복잡한 기계어와 사람 사이를 연결하는 '통역사(LLM)'를 고용할 것입니다.

왜 제조 현장에 LLM인가?

| 데이터의 민주화: 전문가만 아는 언어에서 누구나 쓰는 언어로

PROBLEM



데이터의 장벽

현장의 데이터는 SQL, Python, PLC Code 등
전문가 언어로 굳게 잡겨 있습니다.

신입 사원이나 경영진은 접근조차 어렵습니다.

- ✖ 데이터 사일로(Silo) 발생
- ✖ 의사결정 지연 (IT팀 의존)

LLM

통역사

→

SOLUTION



자연어 인터페이스

LLM은 복잡한 기계어와 인간의 말 사이를
실시간으로 이어주는 **통역사(Interface)**입니다.
"말만 하면" 데이터가 보입니다.

- ✓ 누구나 데이터 접근 가능
- ✓ 즉각적인 현장 대응

기존 방식

{ SELECT * FROM vibration_logs WHERE val > 50;

→

{ "오늘 진동 심한 설비 다 찾아줘."

LLM 방식

범용 LLM VS 산업용 LLM

"시를 잘 쓰는 AI"와 "나사 규격을 아는 AI"는 다릅니다.

GENERAL PURPOSE



범용 LLM

ChatGPT, Claude, Gemini

SPECIALIZED



산업용 LLM

Manufacturing Specific AI

VS

▶ 강점: 유창한 언어 능력

시, 소설, 코딩, 요약 등 일반적인 작업에 탁월함.

⚠️ 약점: 도메인 지식 부재

우리 공장의 고유 설비명, 부품 코드, 은어를 모름.

🛡️ 보안: 데이터 유출 우려

외부 서버로 데이터 전송 시 보안 리스크 존재.

⚙️ 핵심: 특화 지식 (Fine-tuning)

제조/화학/반도체 등 특정 분야 데이터로 추가 학습.

📘 방식: RAG (검색 증강)

사내 매뉴얼, 도면, 작업 표준서를 실시간 참조.

🛡️ 보안: On-Premise / Private

폐쇄망 환경 구축으로 데이터 주권 확보.

💡 전략적 결론

현장 적용을 위해서는 단순히 똑똑한 AI가 아니라, **우리 공장의 문맥(Context)을 이해하는 산업용 AI 전략(RAG + Fine-tuning)**이 필수입니다.

Domain Adaptation

적용 사례 1: 베테랑의 지식 복제

Log Analysis & Knowledge Distillation: 은퇴자의 노하우를 자산화하다

Input: 비정형 데이터



30년치 수리 일지, 알람 로그, 작업자 메모
(김반장의 머릿속 암묵지)



Process: RAG & 학습



패턴 추출 및 벡터화(Embedding)
"소음 A = 윤활유 부족" 규칙 학습



Output: 실시간 조언



이상 징후 발생 시 베테랑의 판단 기준으로
초기 대응 가이드 제시

김반장 (30년차, 은퇴 예정)



[2018-05-12] 모터 3호기 웅웅거리는 소리 남. 베어링 쪽에 구리스
쳤더니 조용해짐.

"이 기계는 날씨 흐리면 꼭 관절이 쑤신다고 하소연을 해. 그럴 땐 구리
스를 듬뿍 줘야 돼."

LLM Knowledge Distillation

AI Assistant (신입사원 보조)



⚠ 이상 감지: 저주파 소음 발생

"과거 수리 이력(2018, 2021년) 분석 결과, **습도가 높을 때 베어링 윤
활 부족일 확률이 92%**입니다."

👉 조치 권장: 구리스 급유 및 점검"



Core Value: 은퇴 리스크 완화

숙련공의 경험을 데이터 자산으로 남겨, 사람이 바뀌어도 **일정한 품질과 대응 능력을** 유지합니다.

Use Case 2 Operator Co-Pilot

| 작업자 보조: 복잡한 HMI를 대체하는 대화형 인터페이스



Natural Language Query (현장 질문)

"A라인 3호기, 지금 멈춰도 문제없어?"

복잡한 메뉴 검색 대신 작업자가 자연어로 질문



Multi-Agent Reasoning (판단)

생산 스케줄 + 납기 + 공정 병목 + 품질 리스크

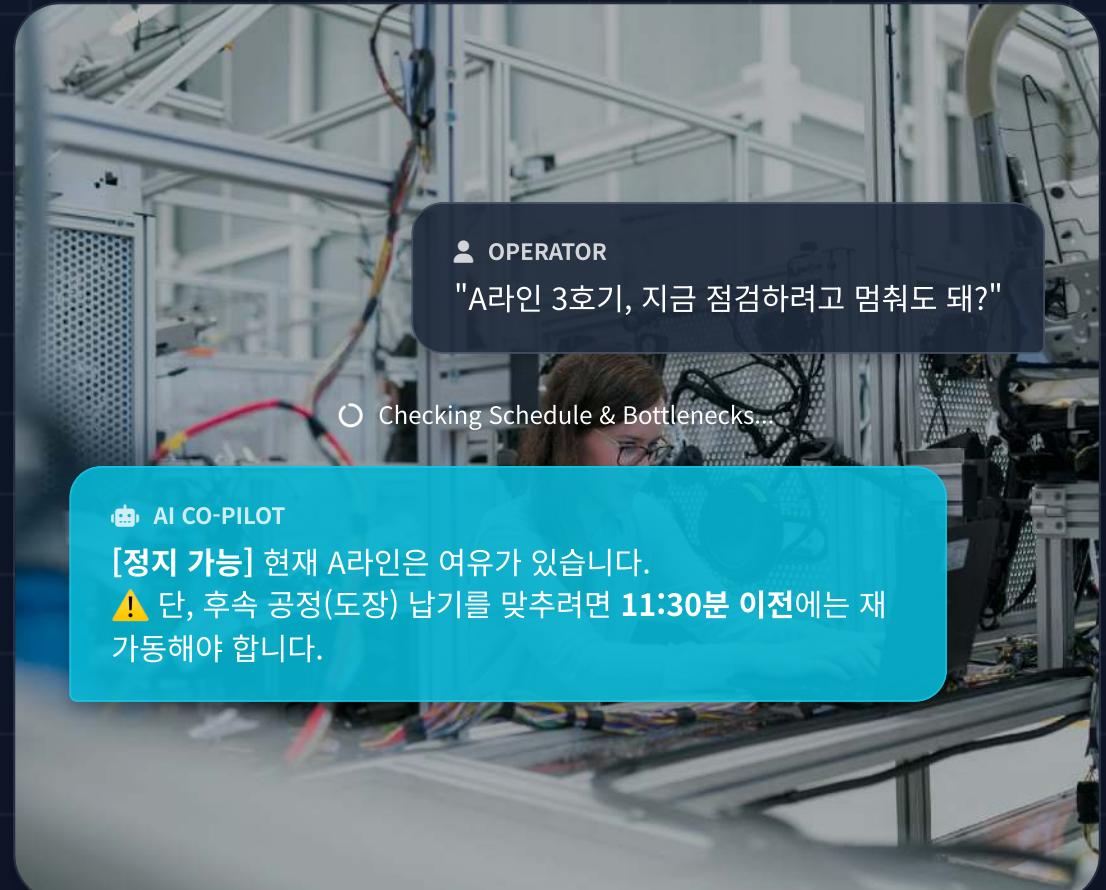
여러 시스템의 데이터를 실시간으로 조회하고 종합 판단



Actionable Insight (답변)

"네, 정지 가능합니다. 단, 11:30분 전 재가동 필수"

단순 정보 전달을 넘어 구체적인 가이드라인 제시



Key Benefits

↓ HMI 복잡도 감소: 수십 개의 모니터링 화면 불필요

↑ 의사결정 속도 향상: 데이터 취합 시간 단축

치명적 과제: 환각 (Hallucination)

⚠️ AI는 자신감 넘치는 거짓말쟁이다



THE PROBLEM

Probabilistic Generation

- ❖ **확률적 생성기:** LLM은 팩트를 말하는 게 아니라, 그럴듯한 다음 단어를 예측 할 뿐입니다.
- ❖ **치명적 리스크:** 공장에서의 거짓말은 설비 파손이나 인명 사고로 직결됩니다.



THE GUARDRAILS

Verification Mechanisms

AI를 믿지 말고, AI가 '검증된 근거'를 찾도록 강제해야 합니다.



근거 강제
RAG



논리 검증
CoT



실제 데이터 조회
Function Calling

해결책 1: CoT (Chain of Thought)

"답만 말하지 말고, 풀이 과정을 보여줘" - 논리의 사슬로 환각을 끊다

Zero-Shot (Bad)



User Prompt

"이 설비 고장 원인이 뭐야?"



"모터 과열입니다."

(근거 없이 가장 그럴듯한 답변 생성)

⚠ Hallucination Risk: High

Chain of Thought (Good)



CoT Prompt

"단계별로(step-by-step) 분석해줘.

- 1) 진동 수치 기준 초과 여부 확인
- 2) 베어링 주파수와 비교
- 3) 최종 결론 도출"

1 "진동 센서 A값이 7.5mm/s로 기준치(5.0)를 초과했습니다."

2 "주파수 분석 결과 120Hz 대역에서 피크가 발생했습니다."

✓ "따라서, 베어링 외潤 결함일 가능성이 높습니다."



Logical Constraint (논리 강제)

단계별 추론을 강제하여 비약적인 거짓말 방지



Auditability (감사 가능성)

AI가 왜 그런 결론을 내렸는지 과정 확인 가능

해결책 2: ReAct (Reasoning + Acting)

| 🧩 생각(Thought)과 행동(Action)을 분리하여 투명성을 확보하다



1. Question

사용자 질문 입력
(Input)



2. Thought

필요 정보 파악
& 도구 선정



3. Action

API/DB 조회
(Function Call)



4. Observation

실행 결과 확인
& 데이터 수신



5. Answer

최종 답변 생성
(Output)

QUESTION "현재 3호기 온도가 정상인가요?"

ANSWER 온도 데이터를 확인해야 한다. [get_temperature] 함수를 사용하자.

ACTION `get_temperature(machine_id=3)`

OBS Observation: 96.8°C (Threshold: 90.0°C)

ANSWER 기준치(90도)를 초과했다. 비정상 상태라고 답변해야 한다.

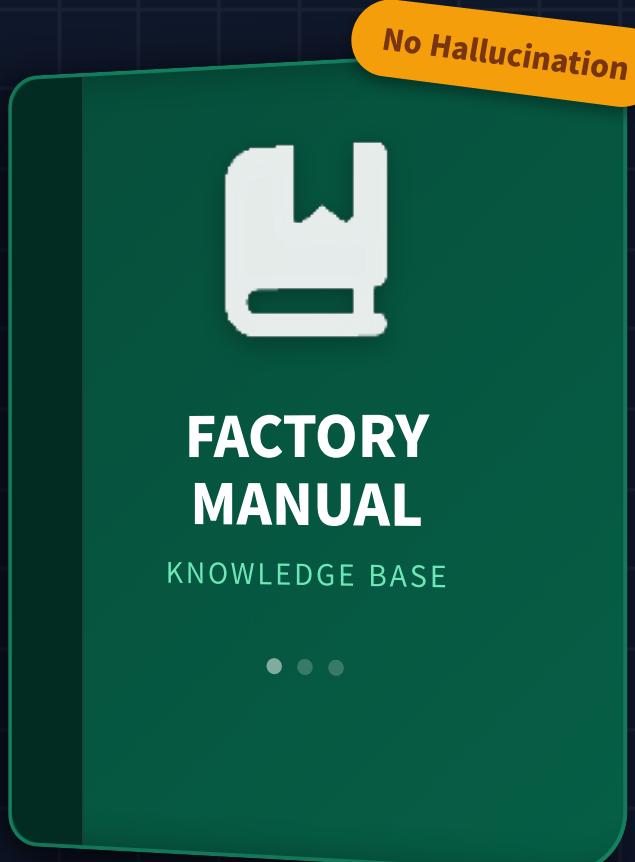
ANSWER "아니요, 현재 96.8°C로 기준치(90°C)를 초과하여 **비정상(과열)** 상태입니다."

REACT TRACE (임시 테스트)

Trace ID: #8291-A

RAG (Retrieval-Augmented Generation)

AI에게 '오픈북 테스트(Open-book Test)'를 허용하다



Fact Check

매뉴얼에 있는 내용만 근거로 답변하므로 **환각(거짓말)**을 원천 차단합니다.



Freshness

AI 재학습 없이, PDF 파일만 교체하면 즉시 **최신 정보**가 반영됩니다.



Security

외부로 데이터를 보내지 않고, **사내 벡터 DB** 안에서만 지식을 검색합니다.

"시험 볼 때 교과서를 펴놓고 답을 쓰면, 절대 없는 말을 지어내지 않습니다."

PROMPT: "다음 [검색된 문서]를 참고하여 질문에 답하시오."

Function Calling: LLM에게 손발을 주다

★ 고립된 언어 모델을 현실 세계와 연결하는 핵심 기술



THE PROBLEM

현실과 단절된 뇌

LLM은 인터넷 없는 방 안의 천재와 같습니다. 엄청난 지식은 있지만, 실제 공장 불을 끄거나 센서 온도를 측정할 물리적 수단(손발)이 없습니다.



THE SOLUTION

도구(Tool) 제공

LLM에게 사용 가능한 함수 설명서(Schema)를 미리 학습시킵니다.

get_temperature()

stop_machine()

send_alert()



THE ROLE

실행자가 아닌 '판단자'

LLM이 직접 코드를 실행하지 않습니다. "지금은 온도를 재야해", "지금은 멈춰야 해"라고 의사결정(Reasoning)하고 요청만 보냅니다.



SECURITY

안전한 통제 (Guardrail)

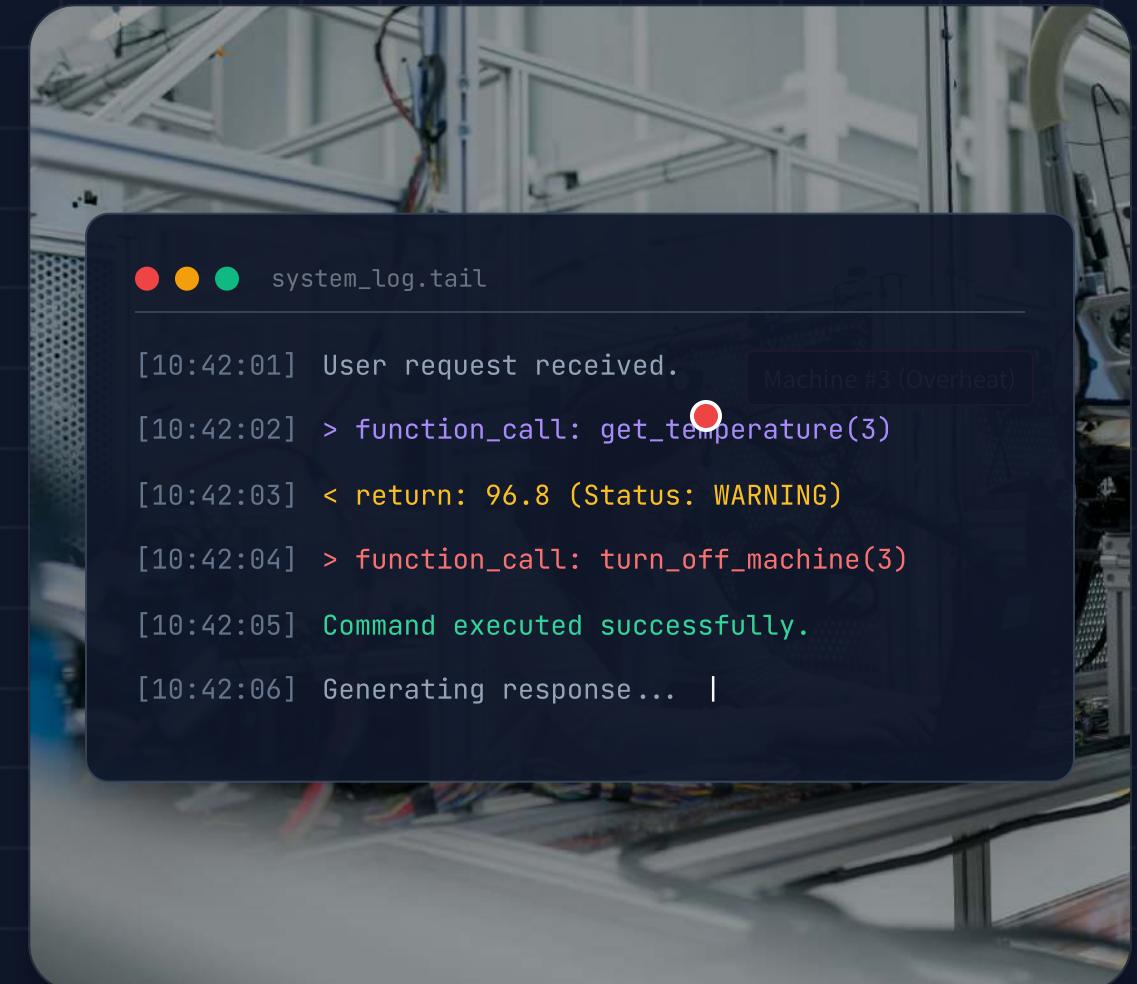
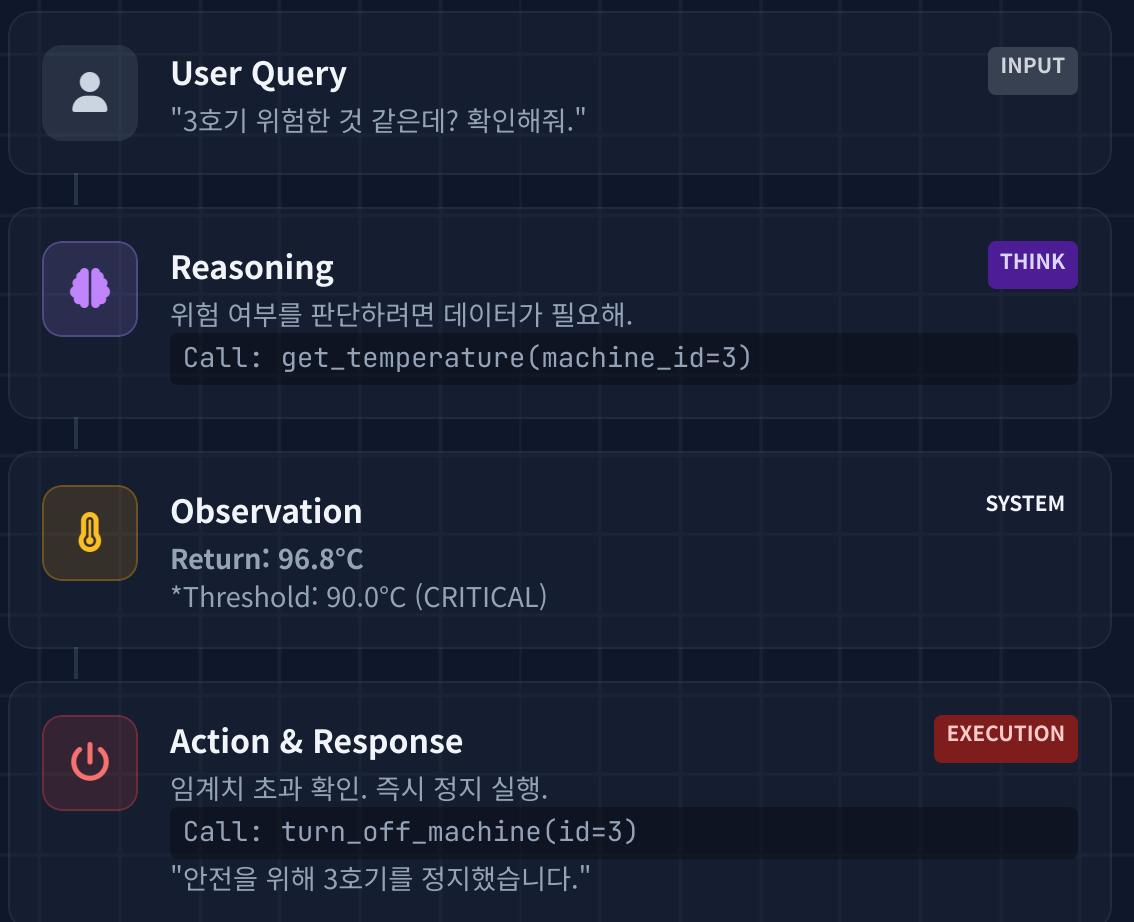
아무나 공장을 멈추면 안 됩니다. **누가, 언제, 어떤 조건**에서 함수를 실행할 수 있는지 엄격한 권한 관리와 승인 절차(Human-in-the-loop)가 필수입니다.

Insight: 우리는 코딩을 하는 게 아니라, "언제 이 함수를 쓸지" AI에게 가르치는 것입니다.

</> JSON Schema

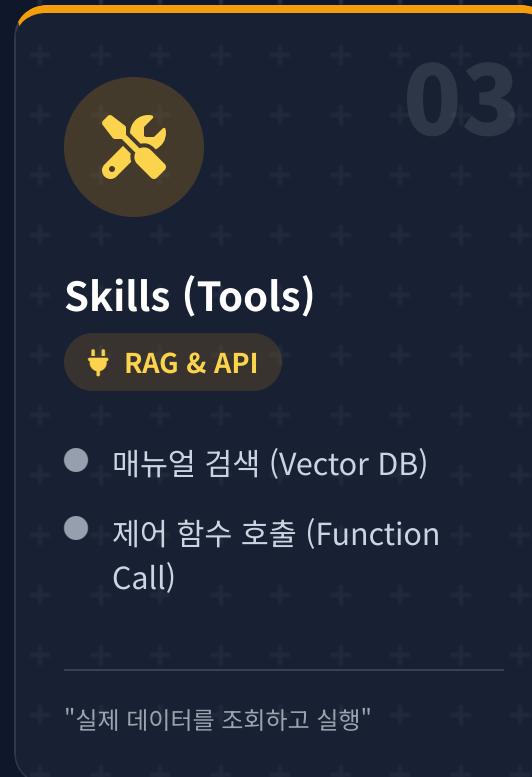
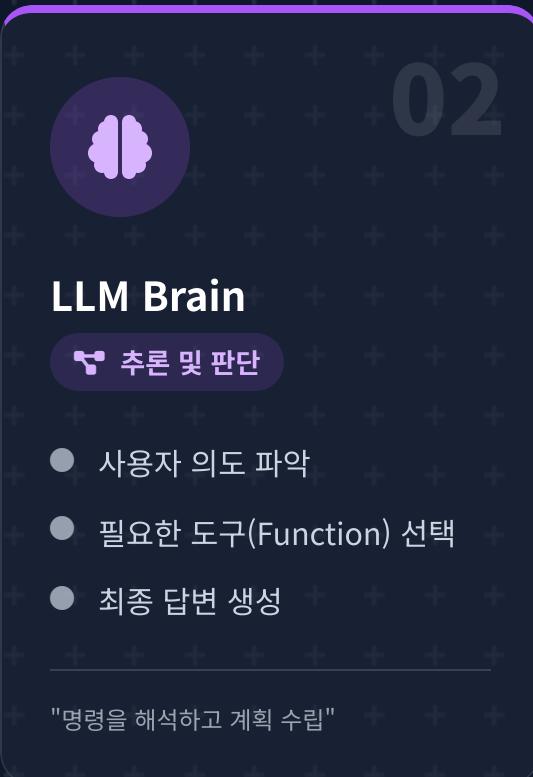
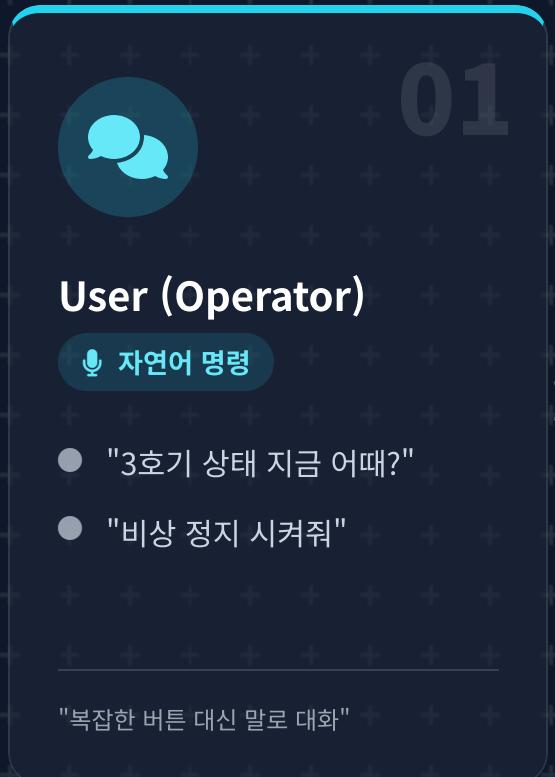
Function Calling 시나리오

LLM의 생각(Reasoning)이 실제 행동(Action)으로 이어지는 과정



Digital Twin × LLM Operator Architecture

자연어(Natural Language)로 공장을 제어하는 차세대 운영 시스템 흐름도



오늘 오후 실습 목표: 이 전체 파이프라인의 **MVP (최소 기능 제품)** 를 직접 코드로 구현합니다.

Python Streamlit

오늘의 목표 (Hands-on Preview)

▣ 이론을 넘어 실전으로: 안전하고 똑똑한 AI 오퍼레이터 만들기



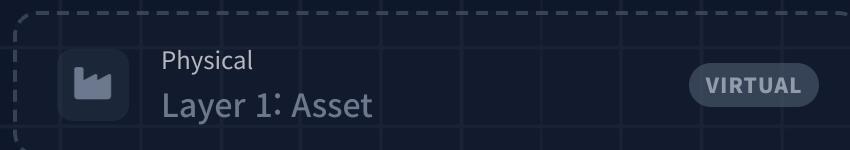
FOCUS



FOCUS



VIRTUAL



VIRTUAL

☰ PM Session Checklist



Prompt Engineering 실습

CoT & ReAct 기법을 코드로 구현하여 환각을 줄이고 논리적인 추론을 유도합니다.



RAG System 구축

사내 매뉴얼 PDF를 벡터화하여 질문에 대해 근거 기반(Evidence-based)의 답변을 생성합니다.



Function Calling 설계

LLM이 상황을 판단하여 제어 함수를 호출하는 안전한 제어 워크플로를 설계합니다.



Safety Guardrails 점검

실제 현장 적용을 위해 필요한 권한, 로그, 승인 절차 등 운영 가드레일을 체크합니다.

Q & A

💬 오늘 학습한 LLM과 디지털 트윈에 대해 질문해주세요.

SUGGESTED DISCUSSION TOPICS



우리 공장 데이터로
RAG 구축하려면?



Function Calling
안전 가이드 (권한/승인)



프롬프트 템플릿
Best Practice?

Next Session: **Hands-on Lab (11:10~)**

Break Time: 10 mins