

DAY 2 / CLASS 2

LLM Integration & Prompt Engineering

OpenAI & Hugging Face 연동 환경 구축

+ 제조 특화 프롬프트 설계 실습



TIME

11:10 ~ 12:10 (60min)



TOPIC

API Connect, Persona, CoT

오늘의 학습 목표 (Today's Goals)

LLM 연동부터 프롬프트 엔지니어링 실습까지

 Application
Prompt Engineering

FOCUS

 Platform
OpenAI & Hugging Face

FOCUS

 Network
API Connection

VIRTUAL

 Environment
Google Colab

VIRTUAL

❖ Key Objectives

환경 구축 (Setup)

Colab에서 OpenAI API Key 및 Hugging Face Token을 안전하게 연동합니다.

기본 연동 (Hello LLM)

API를 통해 LLM 모델(GPT-3.5)을 호출하고 첫 응답을 받아 **연동 상태**를 확인합니다.

프롬프트 설계 (Engineering)

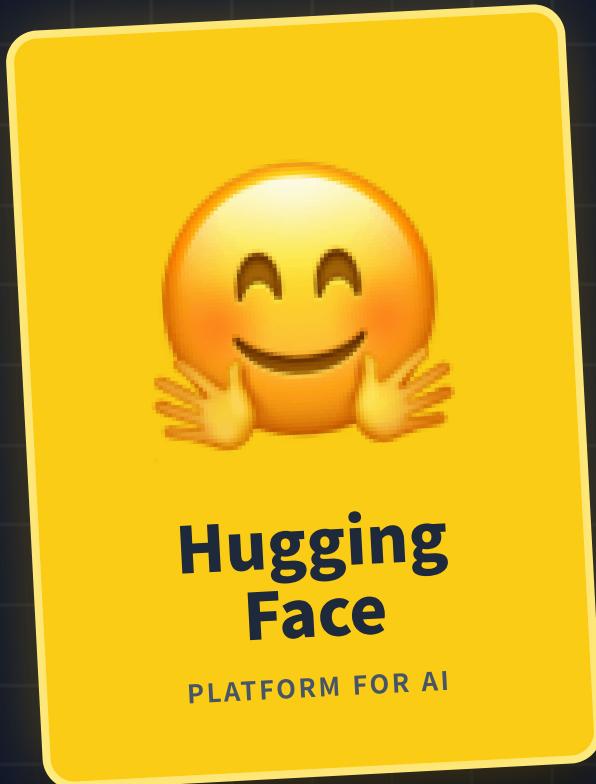
단순 질문과 페르소나(Persona)를 부여한 질문의 답변 품질 차이를 비교합니다.

심화 추론 (Advanced)

CoT(Chain of Thought) 기법을 활용하여 복잡한 제조 설비 문제를 단계적으로 해결합니다.

Hugging Face

😊 The AI Community Building the Future



Models + Datasets = AI Hub



Concept: AI계의 GitHub

전 세계 개발자들이 만든 오픈소스 모델과 코드
가 공유되는 저장소.
"AI 모델을 찾으려면 가장 먼저 가야 할 곳"



Role: Model & Data Hub

100만 개 이상의 모델(LLM 포함)과
수십만 개의 데이터셋을 무료로 다운로드 가능.



Usage: RAG & Embedding

추후 실습할 RAG(검색 증강) 시스템 구축 시,
한국어 특화 임베딩 모델을 여기서 가져옵니다.

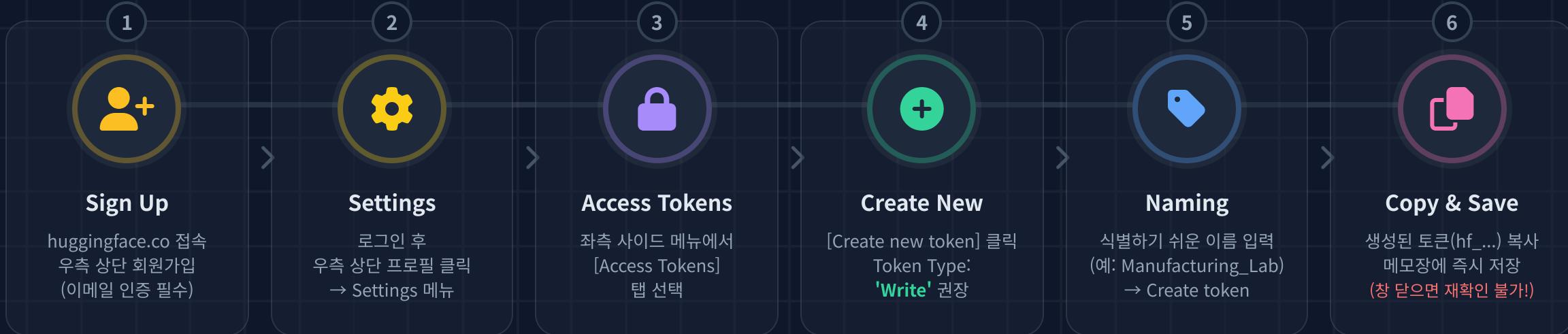


Effect: 빠른 실험과 협업

Spaces 기능을 통해 데모 앱을 쉽게 배포하고,
팀원들과 모델 버전을 관리하며 협업 가능.

Hugging Face 토큰 발급 절차

🔑 Step-by-Step Guide: 오픈소스 모델 접근 권한 획득하기



Warning: 토큰 관리 주의사항



생성된 토큰은 비밀번호와 같습니다. 깃허브 등 공개된 장소에 절대 업로드하지 마세요. 창을 닫으면 토큰 값을 다시 볼 수 없으니 반드시 지금 안전한 곳에 복사해주세요.

Copy Now

>_ 실습 환경 구축 (Environment Setup)

OpenAI 및 LangChain 라이브러리 설치와 보안형(Secret) 키 입력 실습



setup_env.ipynb

1. 필수 라이브러리 설치

```
!pip install -q openai langchain langchain-community huggingface_hub
```

```
import os
import getpass
from huggingface_hub import login
```

2. OpenAI API Key 설정 (보안 입력)

```
print("== OpenAI API Key 입력 ==")
os.environ["OPENAI_API_KEY"] = getpass.getpass("OpenAI Key (sk-...): ")
```

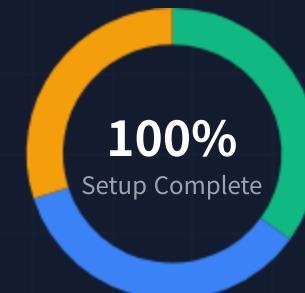
3. Hugging Face 로그인

```
print("\n== Hugging Face Token 입력 ==")
hf_token = getpass.getpass("HF Token (hf-...): ")
login(token=hf_token)
```

```
print("\n✓ 모든 인증이 완료되었습니다!")
```

System Status

Ready



✓ OpenAI SDK	Installed
✓ LangChain Core	Installed
✓ HuggingFace Hub	Logged In



Security Tip

getpass를 사용하면 입력한 키가 화면에 노출되지 않아(Masking) 보안 사고를 예방할 수 있습니다.

🤖 Hello LLM (기본 연동 테스트)

가장 기본적인 대화를 통해 OpenAI API 연결이 성공했는지 확인합니다.



hello_world.py

```
from openai import OpenAI

# 1. 클라이언트 생성 (API Key 자동 로드)
client = OpenAI()

# 2. 간단한 질문 던지기 (Chat Completion)
response = client.chat.completions.create(
    model="gpt-3.5-turbo", # 비용 절감용 모델
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "안녕? 제조 현장에서 디지털 트윈이 왜 중요한지 한 문장으로 설명해줘."}
    ]
)

# 3. 응답 출력
print(f"🤖 AI 응답: {response.choices[0].message.content}")
```

Execution Result

Status: 200 OK



GPT-3.5-Turbo

"디지털 트윈은 물리적 설비를 가상 공간에 실시간으로 복제하여 고장을 미리 예측하고 생산 효율을 최적화할 수 있기 때문에 중요합니다."



Key Point

첫 연결 성공! 이제 이 '기본 대화' 구조에 프롬프트(역할, 상황)만 잘 설계해서 넣으면 어떤 전문가도 만들어낼 수 있습니다.

프롬프트 엔지니어링이란?

Definition of Prompt Engineering



학술적 정의 (Technical)

Optimization Technology

LLM(Large Language Model)이 사용자의 의도에 가장 부합하는 결과를 생성하도록 **입력값(Prompt)을 최적화**하는 기술 및 프로세스.

- ✓ NLP(자연어 처리) 기반의 인터페이스 설계
- ✓ 모델의 잠재 능력(Latent Capabilities) 활성화
- ✓ Few-shot Learning 등 인컨텍스트 러닝 기법 활용



실무적 정의 (Practical)

Designing Work Instructions

AI에게 일을 시키기 위한 완벽한 '**업무 지시서**'를 작성하는 것. 맥락, 역할, 제약을 명확히 하여 원하는 산출물을 얻어내는 과정.

- ✓ **Role & Context:** "누가(Who)", "어떤 상황에서"
- ✓ **Goal:** "무엇을(What)" 달성해야 하는가
- ✓ **Constraints:** "어떤 형식(How)"으로 출력하는가

핵심 가치

5대 핵심 요소 (INPUT)
역할 · 상황 · 목표 · 형식 ·
제약



기대 효과 (OUTPUT)
일관성 ↑ · 오류(환각) ↓ ·
재현성 ↑

"어떻게 질문하느냐가 AI의 지능을 결정합니다."

Prompt Battle: Bad vs Good

화학물질 누출 사고 시나리오: 질문의 차이가 결과의 차이를 만든다



Bad Prompt

맥락과 형식이 없는 단순 질문

USER INPUT

"공장에 화학 물질이 쌓어. 어떻게 해?"

✗ 문제점 1: 모호한 상황

어떤 물질인지, 누출 규모는 어느 정도인지 알 수 없음.

✗ 문제점 2: 일반적 답변

"환기하세요", "신고하세요" 같은 교과서적인 답변만 출력.



RESULT

실제 위급 상황에서 도움이 되지 않는 **추상적 조언**



Good Prompt

Persona + Context + Format

SYSTEM & USER INPUT

"당신은 반도체 공장 CSO입니다. OSHA 규격에 따라..."

"에칭 라인에서 불산(HF) 누출. 작업자 패닉."

VS

✓ 특징 1: 명확한 역할(Persona)

최고 안전 책임자(CSO)로서 전문적이고 단호한 태도.

✓ 특징 2: 구체적 형식(Format)

위험등급 → 대피절차(3단계) → 신고체계 순으로 구조화.



RESULT

즉시 실행 가능한 **Actionable Plan** 및 매뉴얼 준수

Persona Technique

| 역할 부여(Role-Playing)로 AI의 답변 품질을 전문가 수준으로 격상



Why Persona? (필요성)

일반적인 AI는 "평균적인 답변"만 내놓습니다.

전문가 역할을 부여하면 도메인 지식과 특수 용어를 사용하기 시작합니다.



Key Components (구성 요소)

Role (안전 책임자) + **Domain** (반도체) + **Norms** (ISO규격)

+ **Output Format** (보고서 형식, 3단계 절차 등 구체적 지시)



Practical Tips (실무 팁)

금지 사항("~하지 마시오")을 명시하고, 타겟 독자(초보 작업자용)를 지정하세요.

"단계 수는 5개 이하로 제한", "불확실하면 답하지 말 것"

Example: Manufacturing Safety Expert

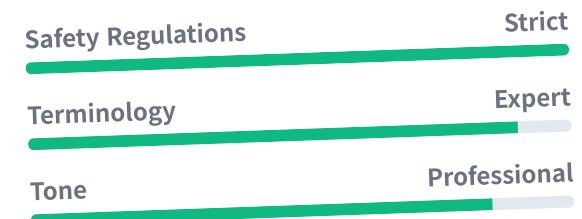
"당신은 10년차 CSO입니다. 모든 답변은 OSHA 규정에 근거해야 합니다."



SYSTEM PROMPT

Chief Safety Officer

SEMICONDUCTOR FACTORY



“ Priority: Worker Protection

Base: ISO 45001 / OSHA

Format: Step-by-step Action Plan

👤 Persona 실습 (Role-Playing)

안전 관리자(CSO) 역할을 부여하여 규정에 입각한 전문적인 답변 유도하기



👤 persona_lab.py

1. 시스템 페르소나 정의 (CSO 역할 부여)

```
system_persona = """당신은 반도체 제조 공장의 '최고 안전 책임자(CSO)'입니다. 긴급 상황 시 작업자의 생명 보호를 최우선으로 하며, OSHA/ISO 국제 안전 규격에 의거하여 답변해야 합니다. 답변은 반드시 다음 형식으로 작성하세요: 1. 🚨 위험 등급 판단 (Critical/Warning/Info) 2. 🚨 즉각적인 대피 및 차단 절차 (3단계) 3. 📞 신고 및 보고 체계 """
```

2. 사용자 상황 입력 (긴급 상황)

```
user_query = "에칭(Etching) 공정 라인에서 '불산(HF)' 누출 경보가 울리고 있어! 작업자들이 패닉 상태야."
```

3. LLM 호출

```
response = client.chat.completions.create(  
model="gpt-3.5-turbo",  
messages=[  
{"role": "system", "content": system_persona},  
{"role": "user", "content": user_query}  
])
```

Response Quality

● Basic ● Persona



Chain of Thought (CoT)

| 생각의 사슬: 복잡한 문제를 단계별로 추론하여 해결하는 기법



Purpose (목적)

AI에게 "바로 답하지 말고 생각 좀 해봐"라고 지시하는 것.
복잡한 진단/추론 문제에서 논리적 비약을 방지합니다.



Steps Design (단계 설계)

- 증상 분석 (현상 파악)
- 가설 수립 (원인 추정) → 3. 검증 방법 → 4. 최종 결론



Anti-Hallucination (환각 방지)

중간 추론 과정(Reasoning)을 거치면서
근거 없는 거짓 답변(Hallucination)을 획기적으로 줄여줍니다.

Example: Fault Diagnosis

"진동은 정상인데 소음이 심하다? → 베어링 파손은 아님 → 윤활 불량이나 이물질 유입 가능성 검토"

1

OBSERVATION

증상 분석 (Data Check)

2

HYPOTHESIS

가설 수립 (Possibilities)

3

VERIFICATION

검증 방법 (Checklist)



CONCLUSION

최종 결론 (Action Item)

☞ Logical Flow

ⓐ CoT (Chain of Thought) 실습

모터 이상 진단을 위한 4단계 사고 프로세스 설계 및 추론



cot_diagnosis.py

1. 복잡한 상황 제시 (모순적인 증상)

```
user_query = """  
3호기 컨베이어 모터에서 '끼이익' 하는 고주파 소음이 들리고,  
표면 온도가 평소보다 15도 높습니다.  
하지만 진동 센서 수치는 정상 범위입니다.  
원인이 무엇이고 어떻게 조치해야 합니까?  
"""
```

2. CoT 시스템 프롬프트 (사고 과정 유도)

```
cot_system = """  
당신은 설비 진단 전문가입니다.  
바로 결론을 내리지 말고, 다음 단계로 생각한 뒤 답변하세요:  
"""
```

[생각의 단계]

- 증상 분석: 소음, 온도, 진동 상태를 개별 분석
- 가설 수립: 진동은 정상이지만 소음/발열이 있는 원인 추론
- 검증 방법: 현장에서 확인해야 할 체크포인트
- 최종 결론: 가장 유력한 원인과 조치 사항

"""

3. 추론 실행

Logic Flow

Thinking Process

ANALYSIS

- 1 소음(O), 발열(O), 진동(X)
→ 기계적 불균형보다는 마찰 문제 의심

HYPOTHESIS

- 2 베어링 윤활 부족(Dry) 또는
모터 내부 권선 절연 파괴 가능성

VERIFICATION

- 3 구리스 주입구 확인 및
절연 저항(Megger) 측정 필요

CONCLUSION

- 4 "베어링 윤활 고갈"이 가장 유력.
→ 즉시 윤활유 보충 및 모니터링

SOP 주입을 통한 도메인 지식 강화

System Prompt에 제조 현장 SOP(표준운영절차)를 주입하여 "우리 공장 맞춤형 전문가" 만들기



sop_injection.py

1. 우리 공장의 가상 SOP (표준 운영 절차) 정의

```
my_factory_sop = """ [사출 성형기 에러 대응 매뉴얼 Ver 2.0] 1. 에러 코드: ERR-707 (노
즐 온도 편차 과다) - 원인: 히터 밴드 단선 또는 SSR 고장. - 조치 순서: 1) 제어 패널에
서 히터 전원 OFF. 2) 멀티미터로 히터 밴드 저항 측정 (정상 범위: 20~30옴). 3) 저항 정
상이면 제어반 내 SSR 교체 후 오토 튜닝 실시. """
```

2. System Prompt 설계 (SOP 주입)

```
system_prompt = f""" 당신은 사출 공정의 '설비 보전 전문가'입니다. 아래 [대응 매뉴얼]
을 숙지하고, 이에 근거하여 정확한 조치 방법을 안내하세요. {my_factory_sop} """
```

3. 사용자 질문 (상황 발생)

```
user_query = "3호기 ERR-707 떴어. 히터 저항 25옴인데 어떻게 해?"
```

4. LLM 호출 (Temperature=0: 매뉴얼 준수)

```
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[{"role": "system", "content": system_prompt},
              {"role": "user", "content": user_query}],
    temperature=0
)
```

Knowledge Accuracy

● General LLM ● With SOP



Output Preview

[AI 보전 전문가 답변]

매뉴얼에 따르면 ERR-707 에러에서 히터 저항이 25옴(정상 범위 20~30옴)이라면 히터는 정상입니다. 따라서 원인은 **SSR 고장**일 가능성이 높습니다.

조치 방법:

- 제어반 내 **SSR**을 **교체**하십시오.
- 교체 후 반드시 **오토 튜닝(Auto Tuning)**을 실시하십시오.

▣ 비정형 로그 정형화 (Data Extraction)

지저분한 현장 텍스트 로그를 파싱하여 분석 가능한 JSON 데이터로 변환



log_parser.py

1. 현장의 비정형(Unstructured) 작업 일지

```
raw_logs = """ [김반장 작업일지] - 10시 반쯤 A라인 컨베이어 벨트가 찢어져서 교체함. -  
오후 2시 15분, 2호기 유압 펌프에서 기름 샘. 오링 교체 완료. """
```

2. 정보 추출(Extraction) 프롬프트 설계

```
extraction_prompt = """ 당신은 '데이터 엔지니어'입니다. 작업 일지를 JSON으로 변환하세요. [추출 규칙] - 시간: YYYY-MM-DD HH:MM (오늘은 2026-01-10) - 설비: 문제가 발생한 설비명 - 문제: 발생한 증상 - 조치: 수행한 조치 내용 - 부품교체: Y/N """
```

3. LLM 호출

```
response = client.chat.completions.create(  
model="gpt-3.5-turbo",  
messages=[  
{"role": "system", "content": extraction_prompt},  
{"role": "user", "content": raw_logs}  
],  
temperature=0 # 포맷 준수 위해 0 설정  
)
```

```
print(response.choices[0].message.content)
```

Data Transformation Flow

Unstructured → Structured

INPUT (RAW TEXT)

UNSTRUCTURED

| "...A라인 컨베이어 벨트가 찢어져서 교체함..."



OUTPUT (JSON)

STRUCTURED

```
{  
    "시간": "2026-01-10 10:30",  
    "설비": "A라인 컨베이어 벨트",  
    "문제": "벨트 찢어짐",  
    "조치": "벨트 교체",  
    "부품교체": "Y"  
}
```



Value Creation

버려지던 텍스트 로그가 통계 분석 가능한 엑셀 데이터로 재탄생합니다.

자동 요약 및 리포트 생성 (Auto Reporting)

일일 작업 로그 요약 및 교대 근무 인계 리포트 자동 생성 실습



shift_report.py

1. 요약(Summarization) 프롬프트 설계

```
summary_prompt = """당신은 공장의 '교대 근무 팀장'입니다. 작업 내역을 바탕으로 다음  
근무자에게 전달할 '일일 설비 이슈 리포트'를 작성하세요. [작성 가이드] 1. 오늘 발생한  
총 건수와 부품 교체 건수를 상단에 요약할 것. 2. 가장 심각해 보이는 이슈 1가지를 선정  
하여 '중점 관리 항목'으로 강조할 것. 3. 톤앤매너: 간결하고 명확한 보고서체 (~함, ~임)  
사용. """
```

2. 앞서 추출한 데이터 활용 (Chaining)

```
extracted_content = response.choices[0].message.content
```

3. LLM 호출 (GPT-4o 권장 for Reasoning)

```
summary_response = client.chat.completions.create(  
model="gpt-4o",  
messages=[  
{"role": "system", "content": summary_prompt},  
{"role": "user", "content": f"작업 내역 JSON:\n{extracted_content}"  
]  
)
```

```
print(summary_response.choices[0].message.content)
```

Generated Report

Shift Handover

[일일 설비 이슈 리포트 - 2026.01.10]

1. 작업 개요

- 금일 발생 이슈: 총 3건
- 부품 교체 작업: 2건 (컨베이어 벨트, 유압 펌프 오링)

2. 중점 관리 항목

- 대상: A라인 컨베이어 벨트
- 내용: 벨트 파손으로 라인 정지(30분). 교체 완료했으나 내일 오전 장력
재점검 요망.

3. 특이사항

- 포장기 센서는 단순 먼지 이슈임. 반복 시 교체 검토.

이상 전달함.

Total Issues

3 cases

Part Replacement

2 EA

Efficiency Gain

"퇴근 전 30분 걸리던 일보 작성이 3초 만에 초안 생성 완료. 작업자는 내용
확인만 하면 됩니다."

다음 스텝 (Next Steps)

▶ LangChain & RAG: 더 똑똑한 제조 AI 만들기

Service
RAG Chatbot

ADVANCED

Framework
LangChain

ADVANCED

Today
LLM API Connect

DONE

Today
Python Env

DONE

🚀 Advanced Curriculum

→ LangChain 체인 구성

LLM과 프롬프트, 도구를 엮어 연속적인 작업을 수행하는 **Chain**을 설계합니다.

→ RAG & Embeddings

Hugging Face의 임베딩 모델을 활용해 문서를 벡터화하고 **검색 증강 생성**을 구현합니다.

→ 제조 매뉴얼 챗봇

수백 페이지의 설비 매뉴얼을 AI가 학습하고 질문에 답변하는 **대화형 앱**을 구축합니다.

→ 보안 및 비용 관리

API Key 관리, 토큰 사용량 모니터링, 캐싱 전략 등 **실무 운영 전략**을 다룹니다.

Q & A



실습 과정에서 궁금했던 점을 자유롭게 질문해주세요.

SUGGESTED DISCUSSION TOPICS



API 키 관리 및 비용 최적화



GPT-3.5 vs GPT-4o 선택 기준



제조 안전 시나리오 설계



Hugging Face 모델 선택

Next Session: **LangChain & RAG Practice**

Lunch Break: 12:00 ~ 13:00