LLM 모델

1. LLM 모델 개념 및 종류

LLM 모델 개념 정리 & 배경

- LangChain은 여러 LLM을 추상화된 인터페이스로 연결할 수 있게 해주는 프레임워크다.
- LangChain 문서에서는 다양한 LLM 제공자를 “integrations / llms” 페이지에서 지원 목록을 소개한다.
- 로컬에서 구동 가능한 오픈소스 LLM들도 llama.cpp, gpt4all, Ollama 등을 통해 LangChain과 함께 사용할 수 있다.
- 또한, OpenLLM 같은 툴을 통해 오픈소스 모델을 OpenAI 호환 API처럼 쓰게 감싸주는 방식도 지원된다.

유료 / 상업용 LLM 모델

구분	모델명	제공사	특징
OpenAI 계열	GPT-3.5 Turbo, GPT-4, GPT-4o / 4o-mini	OpenAI API Key 필요, LangChain `ChatOpenAI`로 연동	
Anthropic Claude	Claude 1–3, Claude 3 Opus/Sonnet/Haiku	LangChain `ChatAnthropic` 지원	
Google Vertex AI / Gemini	Gemini 1.5 Pro / Flash / Nano	`ChatVertexAI` 또는 `ChatGoogleGenerativeAI` 모듈로 연결	
Azure OpenAI Service	GPT-4, GPT-3.5	기업용 Azure 계정 기반, `AzureChatOpenAI`로 연동	
Cohere	Command / Command-R / R+	텍스트 생성·RAG 검색 특화, `ChatCohere` 사용	

유료 / 상업용 LLM 모델

Mistral Small / Medium / Large (API)	Mistral AI 공식 API	로컬 무료판과 달리 상업용은 유료 엔드포인트 제공
Anthropic Bedrock 통합형	Claude on AWS Bedrock	AWS Bedrock LLM Provider로 사용
AI21 Labs Jurassic-2	Large / Grande / Jumbo	글쓰기, 요약, 질의응답 중심
Perplexity Sonar, Together API 등	상용 RAG · API 형태	OpenAI 호환 엔드포인트로 LangChain 연동 가능

무료 / 오픈소스 LLM 모델

구분	모델명	주요 특징
Ollama 계열	Llama 2, Llama 3, Mistral, Phi-3, Gemma 2, Yi 1.5, Falcon 등	로컬 환경에서 `Ollama` 명령어로 실행 가능. LangChain `Ollama` 인터페이스와 바로 연동됨
GPT4All 계열	gpt4all-falcon, gpt4all-mpt, gpt4all-orca 등	PC용 로컬 GUI 및 API 서버 제공, 인터넷 연결 없이 사용 가능
llama.cpp 계열	Llama 2, Mistral, Phi 3 Mini 등	C/C++ 기반 초경량 LLM 엔진, LangChain에서 `LlamaCpp`로 연동
Hugging Face Hub 모델들	BLOOM, Falcon-7B/40B, OPT, GPT-NeoX, MPT 등	`HuggingFaceHub` 클래스 사용으로 API 없이 무료 모델 접근 가능 (토큰 제한 있음)
Google Gemma (로컬)	Gemma 2 7B / 2B	오픈소스 허가 모델, Ollama·HuggingFace 통해 LangChain 연동 가능

무료 / 오픈소스 LLM 모델

Mistral 7B / Mixtral 8×7B	성능 높은 공개 LLM, 상업 이용도 일부 가능	`Ollama` 또는 `transformers` 경유 연동
Phi-3 Mini / Phi-3 Small	Microsoft 공개 초경량 모델	로컬·클라우드 겸용, Python transformers 연동
OpenHermes / Nous Hermes / Orca-Mini 등	커뮤니티 튜닝 모델	Ollama·GPT4All 통해 LangChain과 통합 가능

성능 비교표

모델 / 계열	벤치마크 항목	점수 / 성능 지표	비고 / 해석
Claude 3 Opus	MMLU (학부 수준 지식)	86.8 % (5-shot)	GPT-4의 86.4 %보다 약간 우세함 promptlayer.com +2
	Graduate Reasoning	50.4 % (0-shot CoT)	GPT-4은 35.7 % 수준임 → 복잡한 추론에서 Claude 쪽 강점 promptlayer.com +2
	초등 수준 수학 (GSM8K)	95.0 %	GPT-4의 92.0 %보다 약간 더 높음 promptlayer.com +2
	일반 수학 문제 해결 (MATH)	60.1 %	GPT-4은 52.9 % 수준임 promptlayer.com +1
	다국어 수학 (MGSM)	90.7 %	GPT-4은 74.5 % 수준으로 차이 큼 promptlayer.com +2
	코드 생성 (HumanEval)	84.9 %	GPT-4은 67.0 % 수준임 — 코드 성능에서 Claude 쪽 우위 promptlayer.com +4

GPT-4 (비교 기준)	MMLU	86.4 %	위 벤치마크 비교점 기준치 PromptLayer +2
	GSM8K	92.0 %	기준치 중 하나 PromptLayer +2
	MATH	52.9 %	기준치 중 하나 PromptLayer +2
	코드 생성	67.0 %	기준치 중 하나 promptlayer.com +2
Llama 3 70B	MMLU	~79.5 %	여러 벤치마크에서 이 정도 수준 보고됨 PromptLayer +3
	GSM8K	~79.6 %	기준 중 하나로 자주 인용됨 PromptLayer
	MATH (수학)	~50.4 %	기준치로 제시됨 PromptLayer
	코드 생성	~62.2 %	GPT-4보다 낮지만 경쟁력 있는 수준 PromptLayer
	기타 성능 / 비용 / 속도 비교	- 저비용 / 빠른 응답 가능성 강점 - 복잡한 논리·수학·공통지식에서는 GPT-4 대비 열위	예: "Llama 3 70B는 GPT-4 대비 10배 빠르다" 등의 주장도 있음 Vellum AI +2

기타 비교 지표	문맥 길이 (context window)	Claude 3 Opus: ~200,000 tokens	GPT-4는 일반적으로 최대 128,000 토큰까지 지원하는 버전 존재 Vellum AI +3
	요약 / 긴 문서 처리 능력	Claude 쪽이 긴 문서 요약에 강점 있음	일부 벤치마크 / 글에서 요약, 긴 문맥 처리에서 Claude 유리 언급됨 PromptLayer +3
	비용 대비 성능	Llama 3 쪽이 낮은 비용에 유리	오픈소스이므로 인프라 비용만 부담하면 됨. 일부 평가에서는 "GPT-4 대비 50배 저렴하고 10배 빠름" 주장 있음 Vellum AI

✅ 해석 & 유의점 요약

- Claude 3 Opus는 여러 벤치마크 항목에서 GPT-4보다 우세하거나 대등한 결과를 보이는 경우가 많다. 특히 코드 생성, 다국어 수학, 복잡한 추론 등에서 강점이 자주 언급됨.
- 그러나 "벤치마크 점수"는 동일한 조건(프롬프트, 데이터셋, 샷 수, 평가 방식 등)이 아니면 직접 비교하기 어렵다.
- Llama 3 계열은 공개모델임에도 불구하고 상당히 경쟁력 있는 성능을 보이며, 비용·유연성 측면에서 장점이 크다.
- 문맥 길이, 요약 능력, 긴 문서 처리 등 실무 응용에서는 점수 이외의 요소들도 중요하다.

요약 비교

구분	실행 방식	대표 모델	주요 사용 예
무료/로컬	CPU 또는 GPU 기반 로컬 실행	Llama 3, Mistral, Phi-3, Gemma	오프라인 실험, 교육용 LLM 프로젝트, 개인 연구
유료/API 기반	클라우드 API 호출	GPT-4, Claude 3, Gemini 1.5	상용 RAG 서비스, 멀티 에이전트 시스템, 기업용 서비스 구축

2. 무료 LLM 모델: Ollama

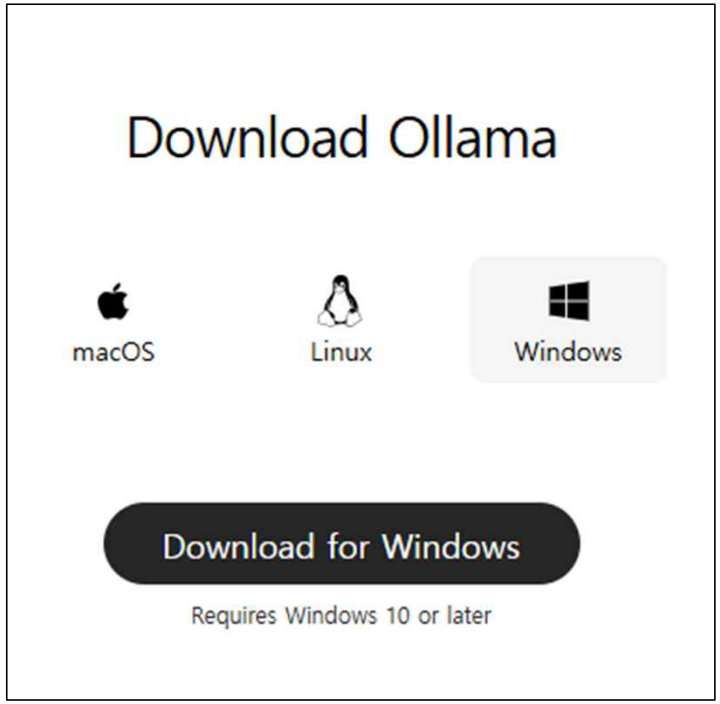
🧠 Ollama 개요

항목	설명
정의	Ollama는 로컬 환경(PC, 서버 등)에서 LLM(Large Language Model)을 손쉽게 실행하고 관리할 수 있게 해주는 **오픈소스 LLM 런처 플랫폼**이다.
주요 목적	클라우드 API 없이도 ChatGPT, Llama, Mistral 등 다양한 모델을 **로컬에서 직접 실행**할 수 있게 함.
개발사 / 배포 형태	Ollama Inc. — macOS, Windows, Linux에서 실행 가능 (`ollama` CLI 및 API 제공)
LangChain 연동	LangChain에서는 `from langchain_community.chat_models import ChatOllama` 로 불러와 사용 가능.

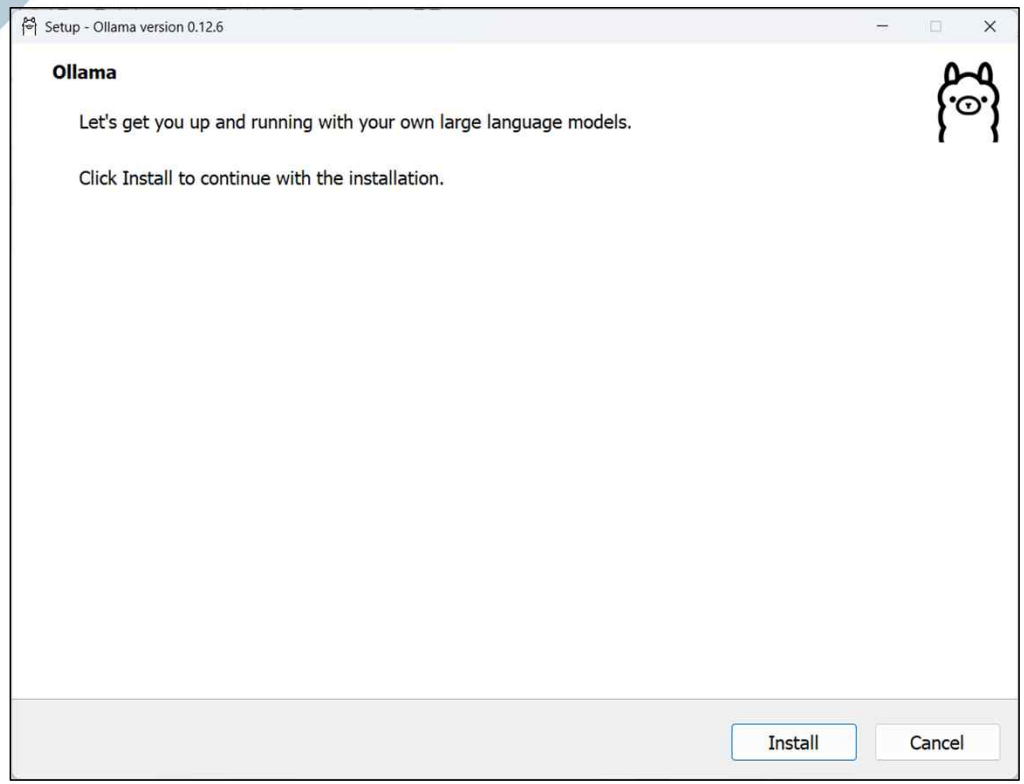
⚙️ Ollama의 구조 및 동작 방식

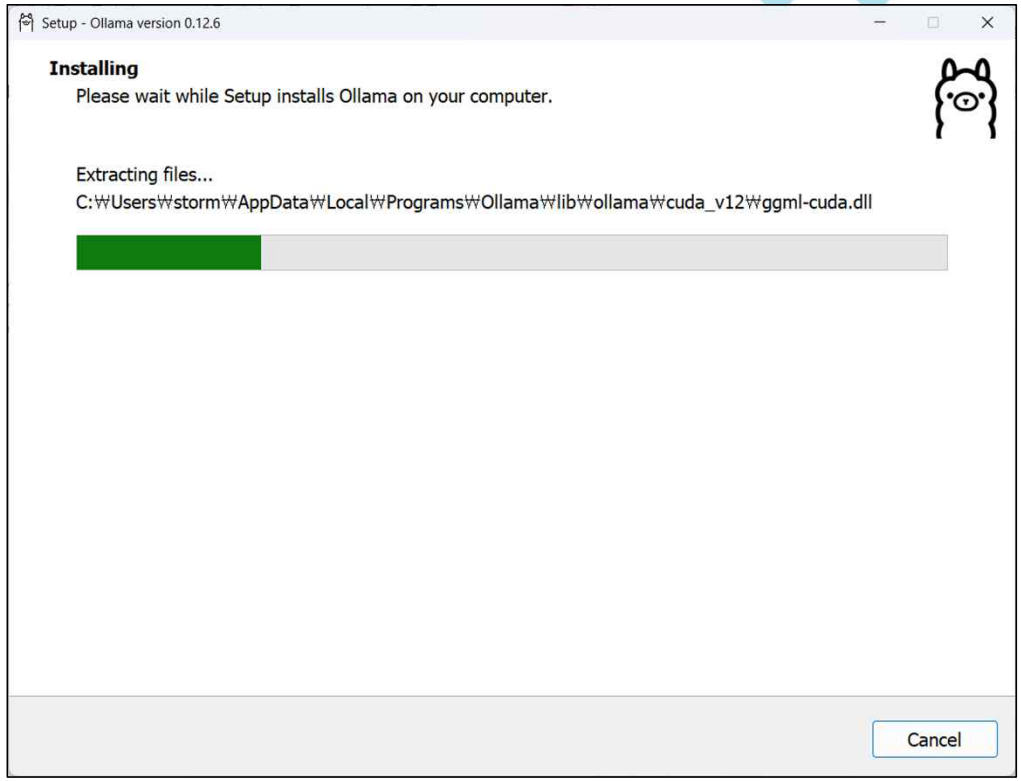
구성 요소	설명
모델 저장소 (Model Library)	`ollama.com/library`에 다양한 모델(Llama 2/3, Mistral, Gemma, Phi, Yi 등)이 등록되어 있으며, 명령 한 줄로 다운로드 가능 (`ollama pull mistral`)
로컬 서버	설치 후 `ollama serve`를 실행하면 로컬 포트(기본 11434)에서 API 서버로 동작
CLI 명령어 예시	`ollama run llama3`, `ollama list`, `ollama pull gemma`, `ollama rm mistral`
모델 파일 형식	`.bin` 또는 `.gguf` 포맷 (llama.cpp 기반 구조)
API 연동 방식	REST API를 제공하여 LangChain, Python, Node.js 등과 통합 가능

https://ollama.com/ 에서 OllamaSetup.exe 파일 다운로드 (1.12GB)

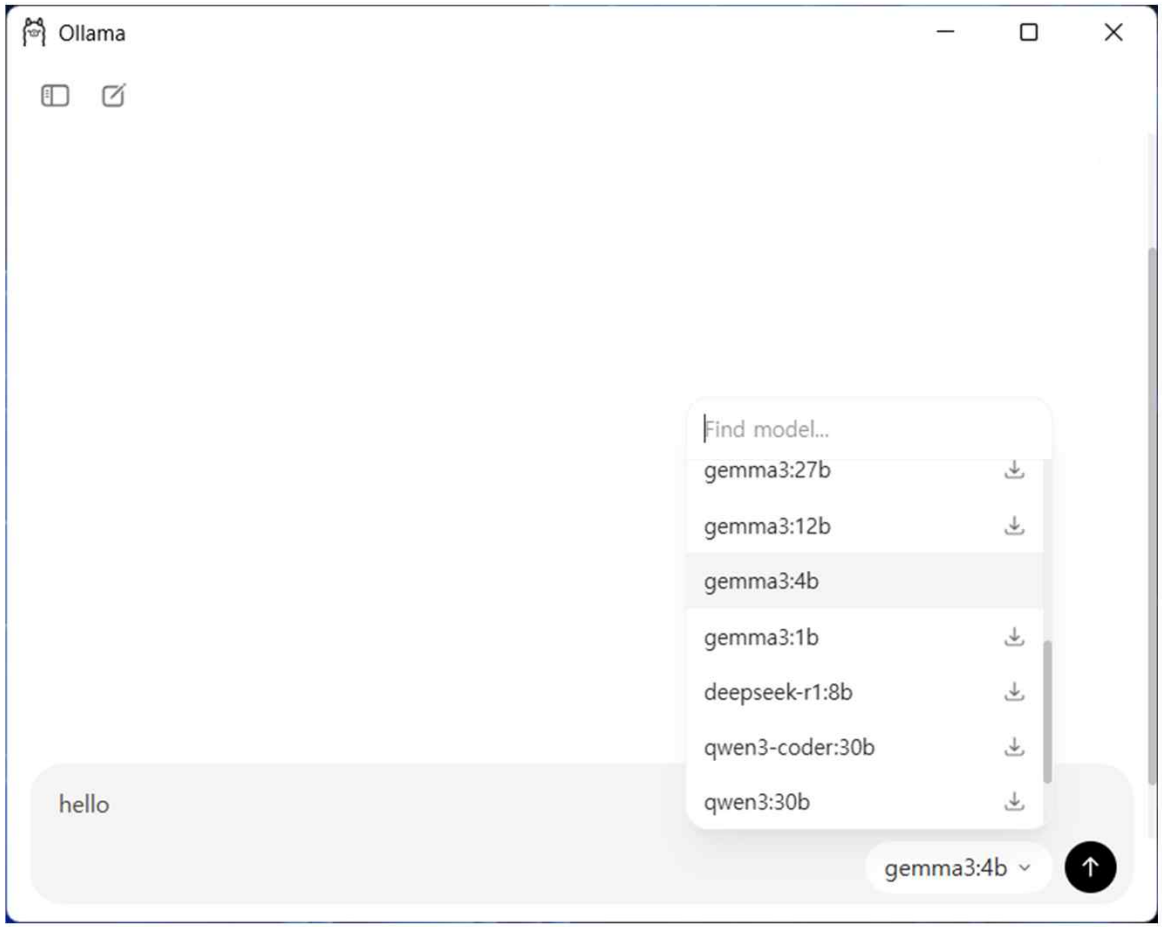


[Install] 버튼을 눌러 설치를 진행한다

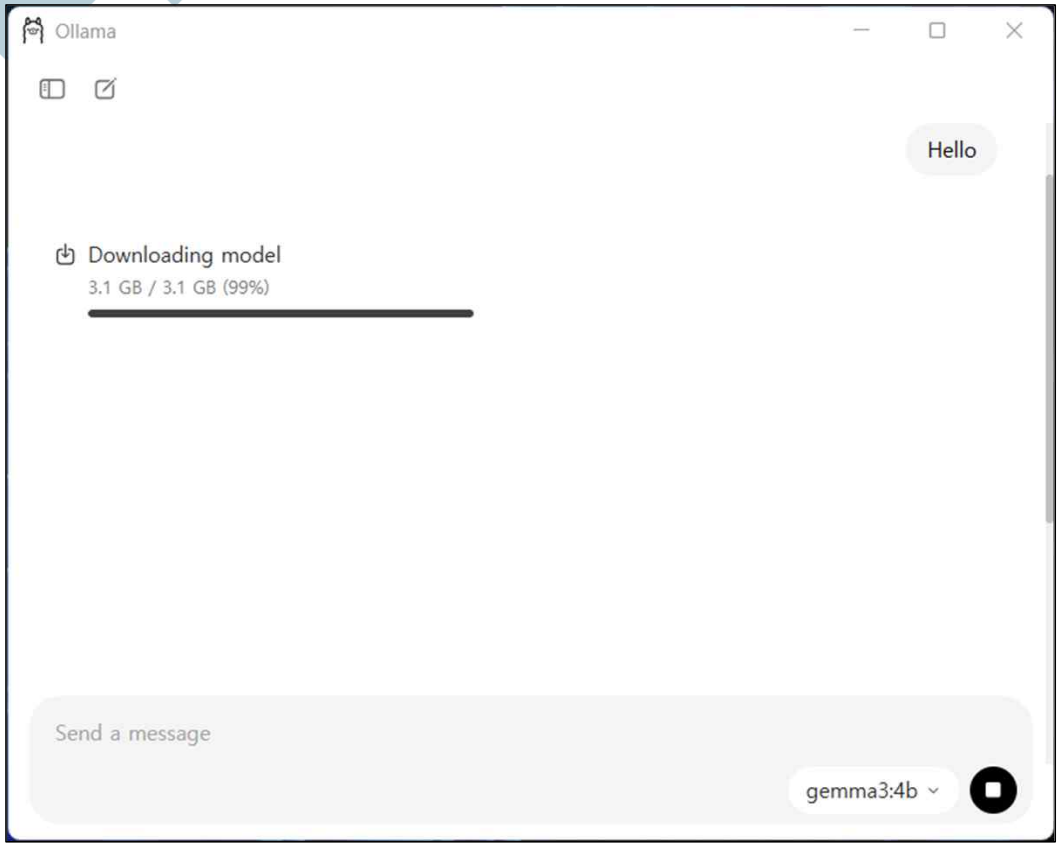


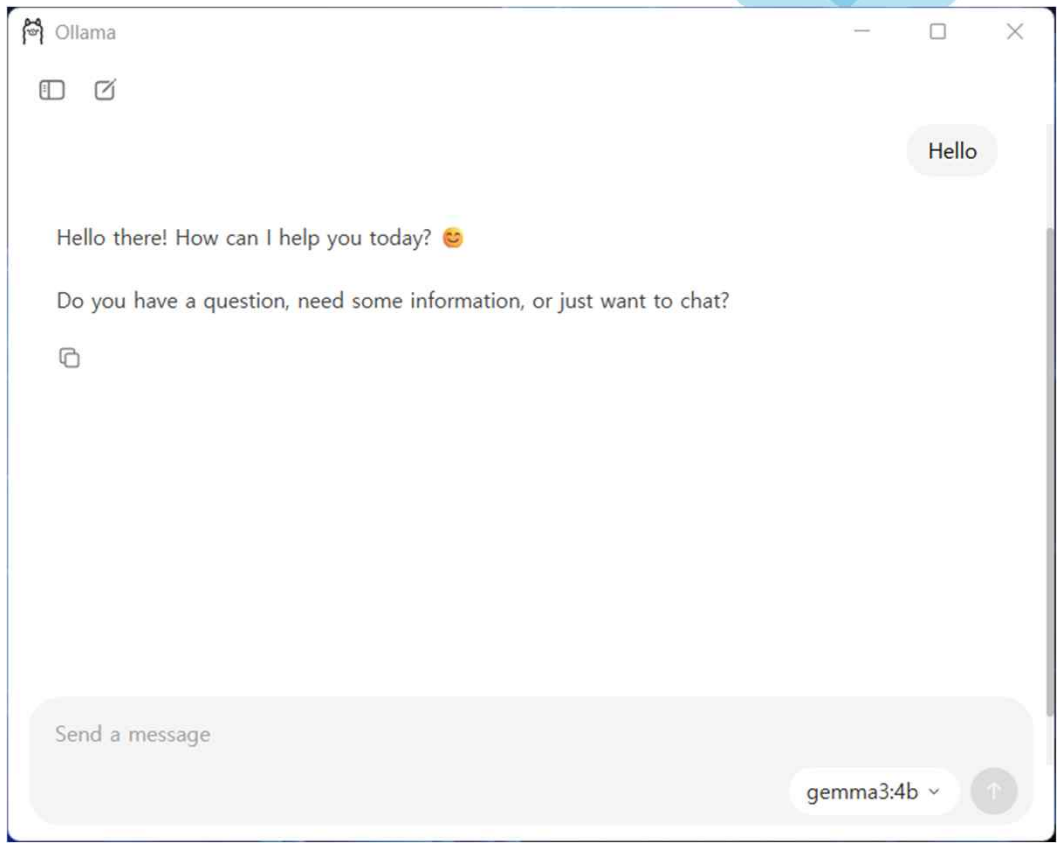


채팅창에 hello 입력 후 "gemma3:4b"모델 선택 후 보내기(↑) 버튼을 클릭

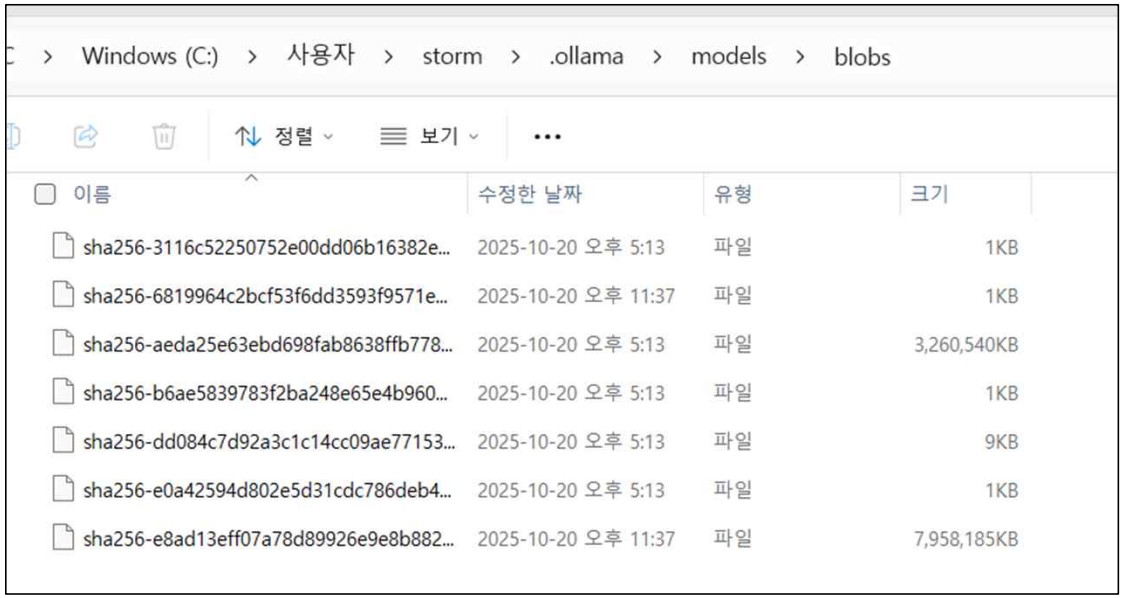


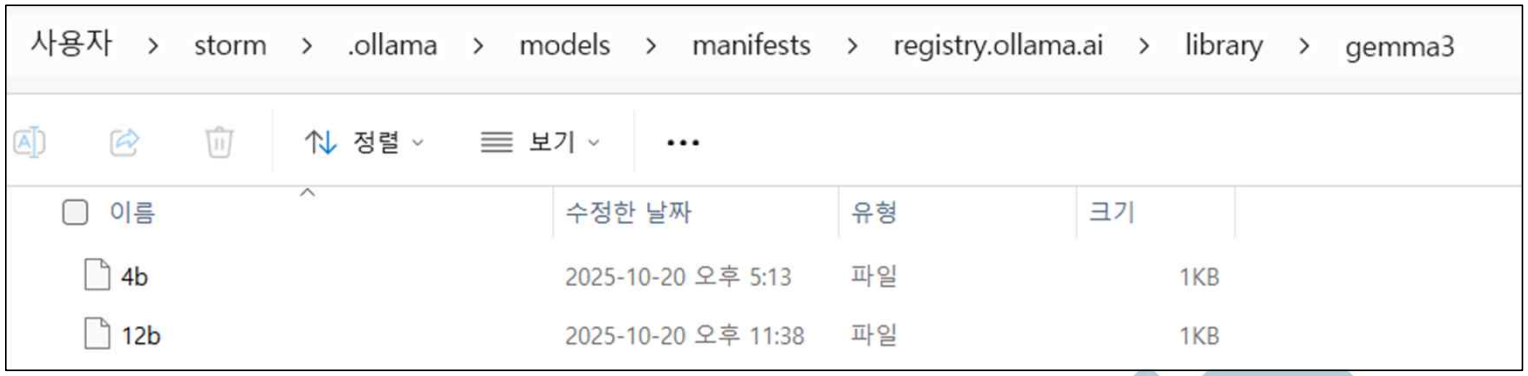
gemma3:4b (약 3.1 GB 크기 파라미터) — 비교적 리소스 부담이 적음, 다운로드 장시간 소요



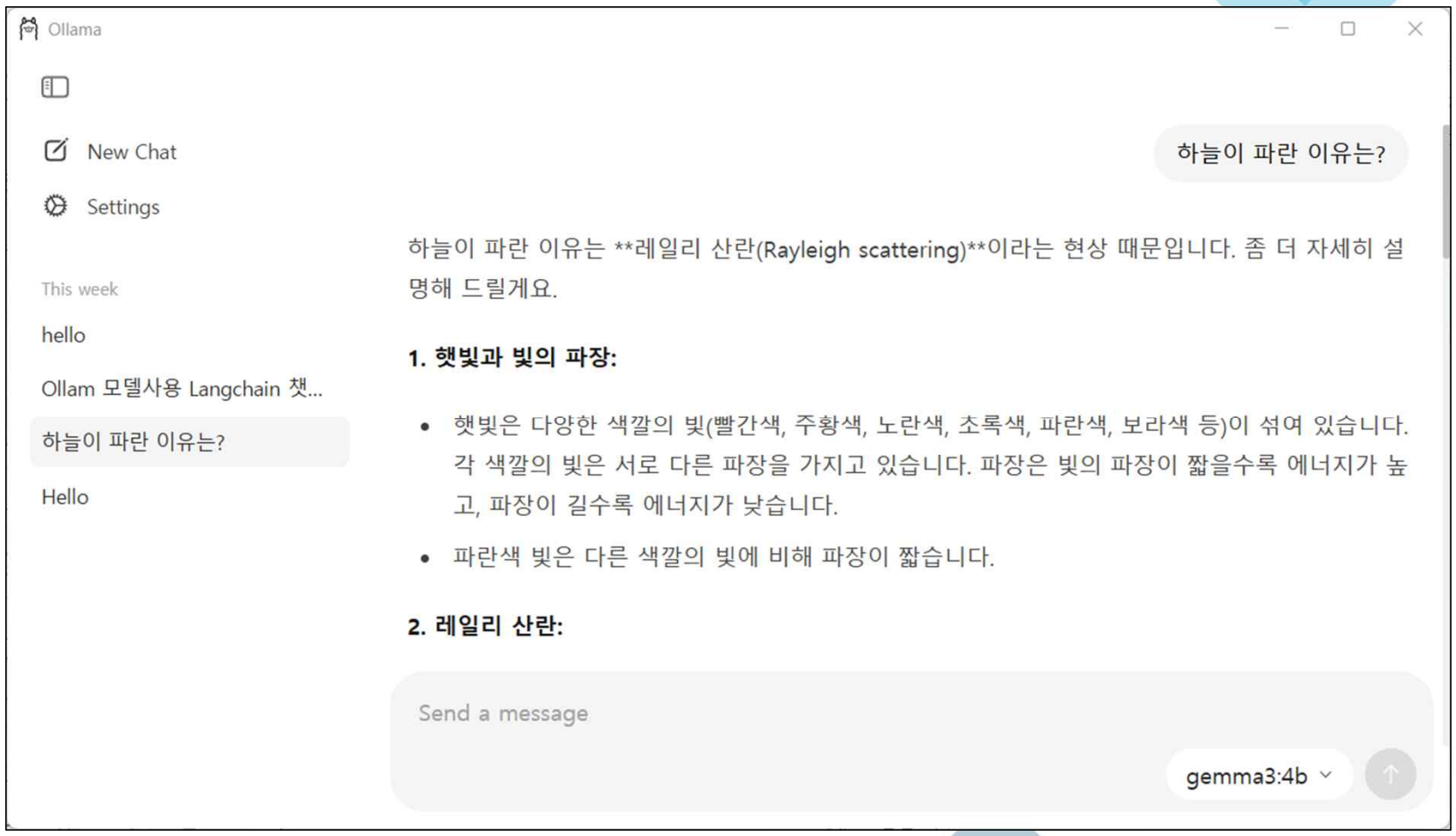


모델 다운로드 경로 : C:₩Users₩storm₩.ollama₩models₩

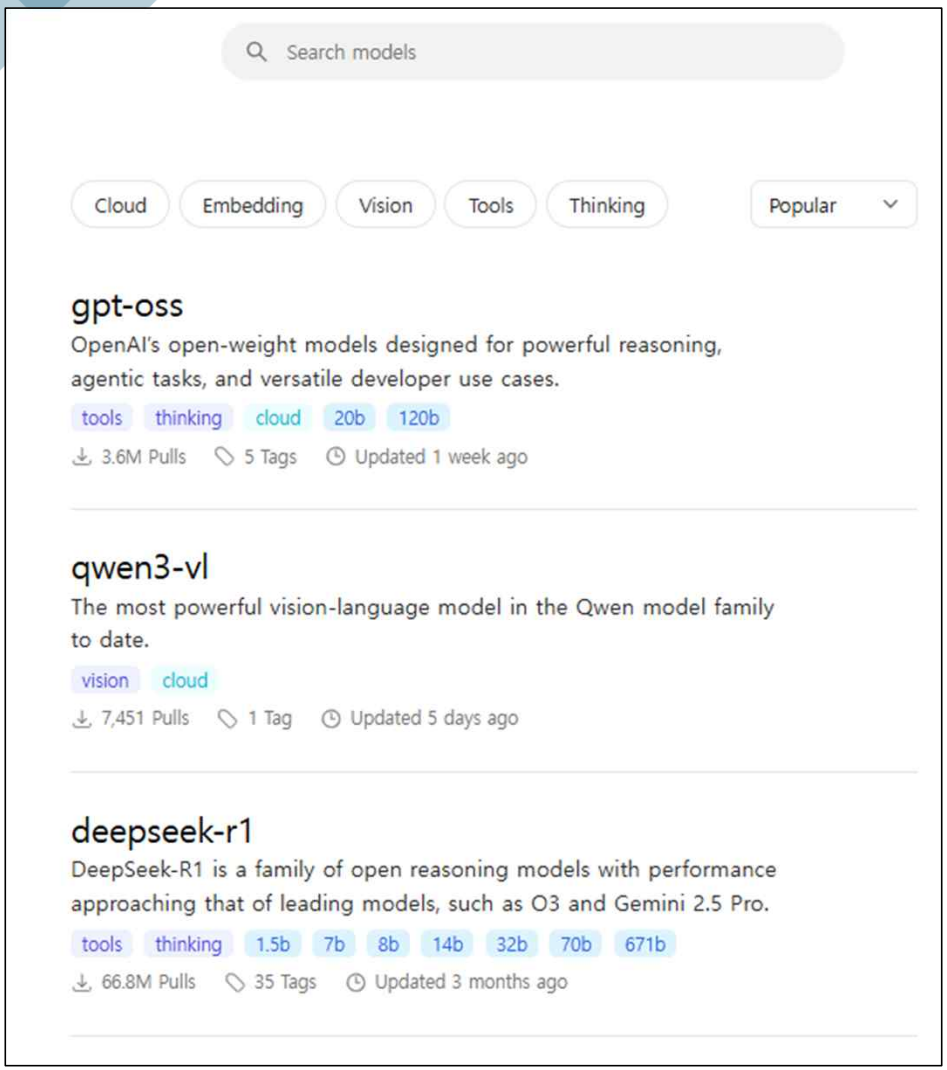


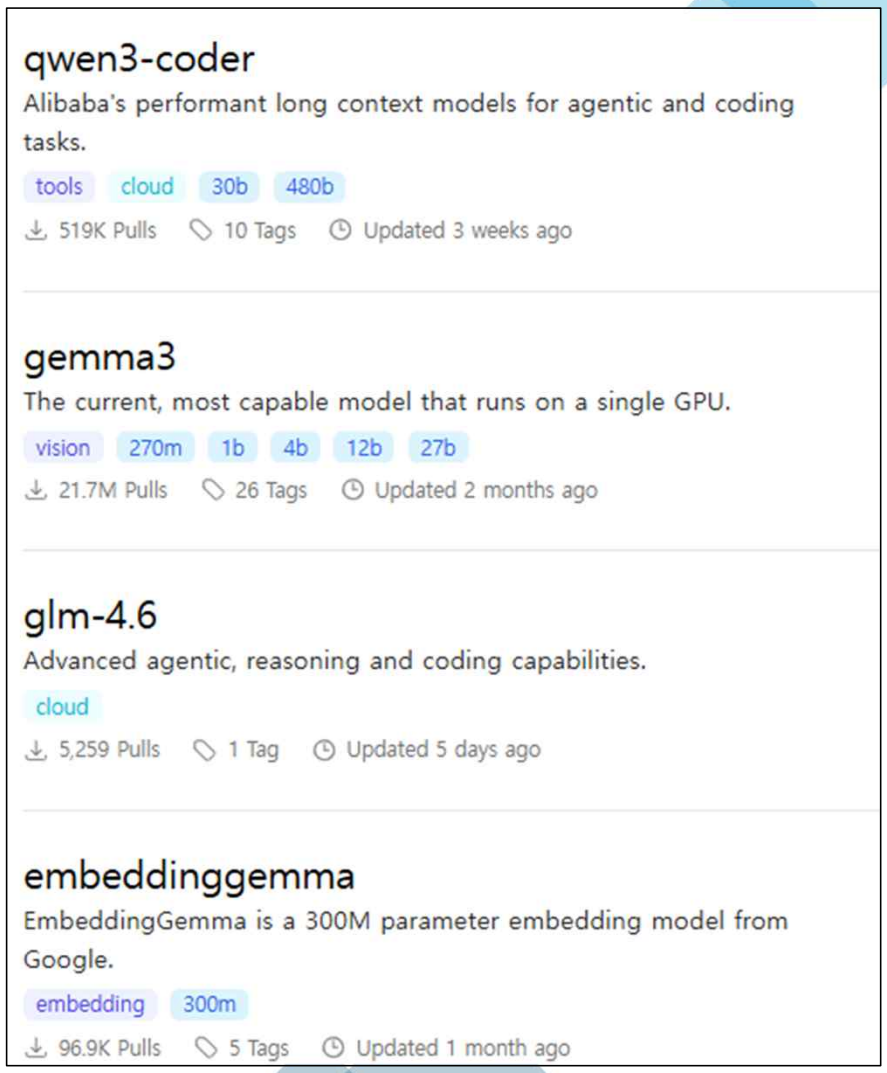


Windows 실행 메뉴에서 Ollama 직접 실행 가능(오프라인 사용 가능)



모델 종류 https://ollama.com/search





qwen3

Qwen3 is the latest generation of large language models in Qwen series, offering a comprehensive suite of dense and mixture-of-experts (MoE) models.

tools thinking 0.6b 1.7b 4b 8b 14b 30b 32b 235b

10.4M Pulls 58 Tags Updated 1 week ago

deepseek-v3.1

DeepSeek-V3.1-Terminus is a hybrid model that supports both thinking mode and non-thinking mode.

tools thinking cloud 671b

107.2K Pulls 8 Tags Updated 3 weeks ago

llama3.1

Llama 3.1 is a new state-of-the-art model from Meta available in 8B, 70B and 405B parameter sizes.

tools 8b 70b 405b

104.5M Pulls 93 Tags Updated 10 months ago

nomic-embed-text

A high-performing open embedding model with a large token context window.

embedding

43.6M Pulls 3 Tags Updated 1 year ago

llama3.2

Meta's Llama 3.2 goes small with 1B and 3B models.

tools 1b 3b

41M Pulls 63 Tags Updated 1 year ago

mistral

The 7B model released by Mistral AI, updated to version 0.3.

tools 7b

20.9M Pulls 84 Tags Updated 3 months ago

qwen2.5

Qwen2.5 models are pretrained on Alibaba's latest large-scale dataset, encompassing up to 18 trillion tokens. The model supports up to 128K tokens and has multilingual support.

tools 0.5b 1.5b 3b 7b 14b 32b 72b

15.3M Pulls 133 Tags Updated 1 year ago

phi3

Phi-3 is a family of lightweight 3B (Mini) and 14B (Medium) state-of-the-art open models by Microsoft.

3.8b 14b

11.9M Pulls 72 Tags Updated 1 year ago

llama3

Meta Llama 3: The most capable openly available LLM to date

8b 70b

11.4M Pulls 68 Tags Updated 1 year ago

llava

LLaVA is a novel end-to-end trained large multimodal model that combines a vision encoder and Vicuna for general-purpose visual and language understanding. Updated to version 1.6.

vision 7b 13b 34b

10.7M Pulls 98 Tags Updated 1 year ago

gemma2

Google Gemma 2 is a high-performing and efficient model available in three sizes: 2B, 9B, and 27B.

2b 9b 27b

8.3M Pulls 94 Tags Updated 1 year ago

qwen2.5-coder

The latest series of Code-Specific Qwen models, with significant improvements in code generation, code reasoning, and code fixing.

tools 0.5b 1.5b 3b 7b 14b 32b

7.7M Pulls 199 Tags Updated 4 months ago

phi4

Phi-4 is a 14B parameter, state-of-the-art open model from Microsoft.

14b

5.6M Pulls 5 Tags Updated 9 months ago

gemma

Gemma is a family of lightweight, state-of-the-art open models built by Google DeepMind. Updated to version 1.1

2b 7b

5.4M Pulls 102 Tags Updated 1 year ago

mxbai-embed-large

State-of-the-art large embedding model from mixedbread.ai

embedding 335m

5.1M Pulls 4 Tags Updated 1 year ago

qwen

Qwen 1.5 is a series of large language models by Alibaba Cloud spanning from 0.5B to 110B parameters

0.5b 1.8b 4b 7b 14b 32b 72b 110b

5M Pulls 379 Tags Updated 1 year ago

"gemma3:4b" 모델 사용 예제(오프라인)

Ollama

https://ollama.com/

```
! pip install langchain-ollama
```

```
from langchain_ollama import ChatOllama

# Ollama 모델을 불러옵니다.
llm = ChatOllama(model="gemma3:4b")

response= llm.invoke("하늘은 왜 파란가요?")
print(response.content)

```

"gemma3:12b" 모델 사용 예제(오프라인)

```python
from langchain_ollama import ChatOllama

# Ollama 모델을 불러옵니다.
llm = ChatOllama(model="gemma3:12b")

response= llm.invoke("하늘은 왜 파란가요?")
print(response.content)
# 응답 오래 걸림
```

```
하늘이 파란 이유는 **레일리 산란(Rayleigh scattering)**이라는 현상 때문입니다.

좀 더 자세히 설명해 드릴게요.

1.  **햇빛의 구성:** 햇빛은 여러 가지 색깔의 빛(빨강, 주황, 노랑, 초록, 파랑, 남색, 보라)이 섞여 있
2.  **대기와 빛의 상호작용:** 햇빛이 지구 대기권으로 들어오면 공기 중의 질소, 산소 분자와 충돌하면서
3.  **레일리 산란:** 레일리 산란은 빛의 파장이 짧을수록 더 잘 일어나는 현상입니다. 파장이 짧은 빛은
4.  **하늘이 파랗게 보이는 이유:** 따라서 햇빛이 대기를 통과할 때 파장이 짧은 파란색 빛이 다른 색깔의
랗게 보이는 것입니다.
```

Ollama에서 사용 모델 성능 비교

- **Llama 3** → 🔹 **성능 최고**, 논리·코딩·대화 품질이 가장 우수하지만 모델이 커서 **속도는 다소 느림**.
- **Mistral 7B** → ⚖️ **성능과 속도의 균형형**, 중간 크기 모델로 **가볍고 빠르며 추론력도 양호**.
- **Gemma 2 / 3** → ⚡ **속도 가장 빠름**, 작은 환경에 적합하지만 **이해력·정확도는 낮음**.

👉 **정리:** 고성능이 목표라면 *Llama 3*, 속도·효율을 원하면 *Mistral*, 저사양 환경이면 *Gemma*가 가장 적합합니다.

3. 무료 LLM 모델: HuggingFacePipeline

`HuggingFacePipeline`은 **LangChain에서 Hugging Face의 Transformer 모델을 직접 호출할 수 있도록 연결해주는 클래스**이다. 즉, OpenAI API처럼 외부 LLM을 쓰지 않고 **로컬 또는 Hugging Face Hub 모델**을 바로 사용할 수 있게 해준다.

🔹 기본 개념

`HuggingFacePipeline`은 내부적으로 `transformers` **라이브러리의** `pipeline()` 함수를 감싼 래퍼(wrapper)이다.

LangChain의 `LLM` 인터페이스와 호환되도록 만들어져 있어서, LangChain의 **Chain**, **Agent**, **Retriever** 등과 함께 사용할 수 있다.

주요 특징

항목	설명
모델 소스	Hugging Face Hub 또는 로컬 모델
지원 Task	text-generation, text2text-generation, summarization, translation, etc.
장점	로컬 환경에서 LLM 사용 가능 (API 비용 없음)
단점	모델 파일이 커서 다운로드가 오래 걸릴 수 있음, GPU 없으면 느림

사용 예시

python

```python
from transformers import pipeline
from langchain_huggingface import HuggingFacePipeline

# 1. Hugging Face 파이프라인 생성
pipe = pipeline(
    "text-generation",
    model="gpt2",              # 또는 "meta-llama/Llama-2-7b-chat-hf" 등
    max_new_tokens=100,
    temperature=0.7
)

# 2. LangChain용 래퍼로 감싸기
llm = HuggingFacePipeline(pipeline=pipe)

# 3. 모델 호출
response = llm.invoke("Explain why the sky is blue.")
print(response)
```

출력 예시:

csharp

```
The sky is blue because light from the sun is scattered in the atmosphere...
```

🧠 1. 모델 크기 및 구조

- 모델 크기: **약 12.5 GB (FP16 기준)**
- 파라미터 수: **약 7 billion (7 억 × 10)**
- 로딩 시 메모리 점유: **약 14 ~ 16 GB RAM 필요**
- 사용 라이브러리: `transformers` + `torch`
- HuggingFacePipeline은 내부적으로 `AutoModelForCausalLM`과 `AutoTokenizer`를 불러와 CPU inference를 수행

🕒 2. CPU 환경에서의 실제 예상 시간

환경	CPU 스펙 예시	예상 응답 시간 (512토큰 기준)	비고
✅ 고성능 워크스테이션 (AMD Ryzen 9, i9 등)	16코어, 64GB RAM	**10 ~ 20분**	매우 느림
💻 일반 데스크탑 (i5, 16GB RAM)	6~8코어	**20 ~ 40분**	토큰 생성 속도 초당 0.5~1token
⚠️ 저사양 노트북 (4코어, 8GB RAM)		**사실상 멈춘 것처럼 보임 (1시간 이상)**	CPU 연산이 병목됨

⚡ 3. 개선 방법

(1) GPU 사용 (강력 권장)

- **CUDA 지원 GPU (예: RTX 3060 이상)**

  → `device_map="auto"` 또는 `device=0` 설정으로 GPU로 전환

  → 512토큰 기준 **5 ~ 10초 이내 완료**

```python
hf = HuggingFacePipeline.from_model_id(
    model_id="beomi/llama-2-ko-7b",
    task="text-generation",
    model_kwargs={"device_map": "auto"},  # GPU 자동 할당
    pipeline_kwargs={"max_new_tokens": 256},
)
```

🧠 왜 Hugging Face 파이프라인 대부분이 GPU 중심인가?

1️⃣ Transformers 모델의 기본 구조

- 대부분의 Hugging Face 모델(`AutoModelForCausalLM`, `AutoModelForSeq2SeqLM` 등)은
  수십억(1B~70B) 개의 파라미터를 갖는 **딥러닝 신경망(Transformer)** 입니다.
- 이 계산량은 CPU로는 너무 방대해서,
  한 토큰 생성에도 수초~수분이 걸립니다.
- 그래서 대부분의 모델은 GPU(CUDA, ROCm)나 TPU에서의 병렬 연산을 **기본 전제**로 설계되었습니다.

2️⃣ PyTorch와 TensorFlow의 연산 특성

- CPU는 벡터 연산에 약하고,
  GPU는 병렬 행렬 연산에 최적화되어 있습니다.
- 예를 들어 7B 모델 기준으로:

환경	응답 속도 (512토큰 기준)
CPU (i7-9700)	20~40분
GPU (RTX 3060)	5~10초
GPU (RTX 4090)	2~3초

Ollama와 HuggingFace Pipeline 모델 비교

구분	Ollama	HuggingFace Pipeline
설치 구조	Ollama 앱 설치 후 명령어 기반 모델 관리 (`ollama pull gemma:2b`)	Python 패키지(`transformers`, `torch`) 기반
모델 위치	로컬 캐시(`/Users/.../.ollama/models`)	HuggingFace Hub에서 다운로드 (`~/.cache/huggingface/`)
LLM 호출 인터페이스	LangChain용 `ChatOllama` 클래스 지원	`HuggingFacePipeline` 또는 `HuggingFaceEmbeddings`
GPU 지원	자동 감지 (CUDA 지원 시 속도↑)	torch.cuda 사용 직접 설정 필요
추론 속도	모델 크기에 따라 다르나 최적화되어 빠름 (특히 M-series / CUDA 환경)	일반적으로 느림, CPU 기반 시 더 느림
모델 종류	Gemma, Llama3, Mistral, Phi3, Qwen 등 다양 (양자화 버전 포함)	수천 개 모델 지원 (특히 Text2Text, Translation, Summarization 등)
LangChain 연동성	대화형 LLM 중심 (`ChatOllama`, `OllamaEmbeddings`)	텍스트 생성 / 분류 / 번역 등 task별 파이프라인 중심
환경 설정 난이도	쉽다 (한 줄로 설치 및 실행 가능)	상대적으로 복잡 (모델, 토크나이저, 프레임워크 직접 설정)
무료 사용 범위	완전 무료, 로컬 실행	완전 무료 (단, 모델 다운로드 용량 큼)
오프라인 동작	가능 (모델 로컬 캐시 사용)	가능 (모델 캐시 존재 시)

4. 무료 LLM 모델: GPT4All

1. 개요

GPT4All은 *Nomic AI*가 개발한 **오픈소스 로컬 LLM(Local Large Language Model) 프레임워크**이다.
이 모델은 인터넷 연결 없이 개인 PC(Windows, macOS, Linux)에서 실행할 수 있으며, OpenAI API처럼 동작하지만 **GPU 없이도 CPU 기반으로 추론 가능**하다는 점이 특징이다.

---

2. 주요 특징

구분	설명
오프라인 실행	클라우드나 인터넷 없이 로컬 환경에서 모델을 실행 가능
무료 모델 사용	다양한 오픈소스 모델(Llama 2, Mistral, Falcon 등)을 다운로드해 무료 사용 가능
프라이버시 보장	모든 데이터와 대화가 로컬 PC에서 처리되어 외부 전송 없음
GPU 불필요	CPU에서도 작동하며, GPU가 있을 경우 속도 향상 가능
다양한 인터페이스	CLI, GUI(챗 인터페이스), Python API, LangChain 통합 지원

3. 지원 모델 종류

GPT4All은 자체 모델뿐 아니라 다양한 오픈소스 LLM을 지원한다.
아래는 대표적인 예시이다:

모델명	파라미터 수	기반 모델	용도 및 특징
GPT4All Falcon	7B	Falcon	일반 대화용, 속도 빠름
GPT4All Mistral	7B	Mistral	품질 우수, 영어 중심
GPT4All Llama2	7B~13B	Meta Llama 2	다목적, 안정적
GPT4All Orca	7B	Orca	수학/논리 응답 개선
GPT4All Nous-Hermes	13B	Llama 기반	고품질 대화, 코드 생성

4. 설치 및 실행

(1) 데스크톱 앱 방식

- 공식 사이트: https://gpt4all.io ↗
- Windows / macOS / Linux용 설치 파일 제공
- 설치 후 GUI 인터페이스에서 모델 다운로드 및 채팅 가능

(2) Python 패키지 방식

bash

```bash
pip install gpt4all
```

python

```python
from gpt4all import GPT4All

model = GPT4All("mistral-7b-instruct")
with model.chat_session() as session:
    output = model.generate("하늘은 왜 파란가요?")
    print(output)
```

5. LangChain 연동 예제

LangChain에서도 GPT4All 모델을 간단히 사용할 수 있다:

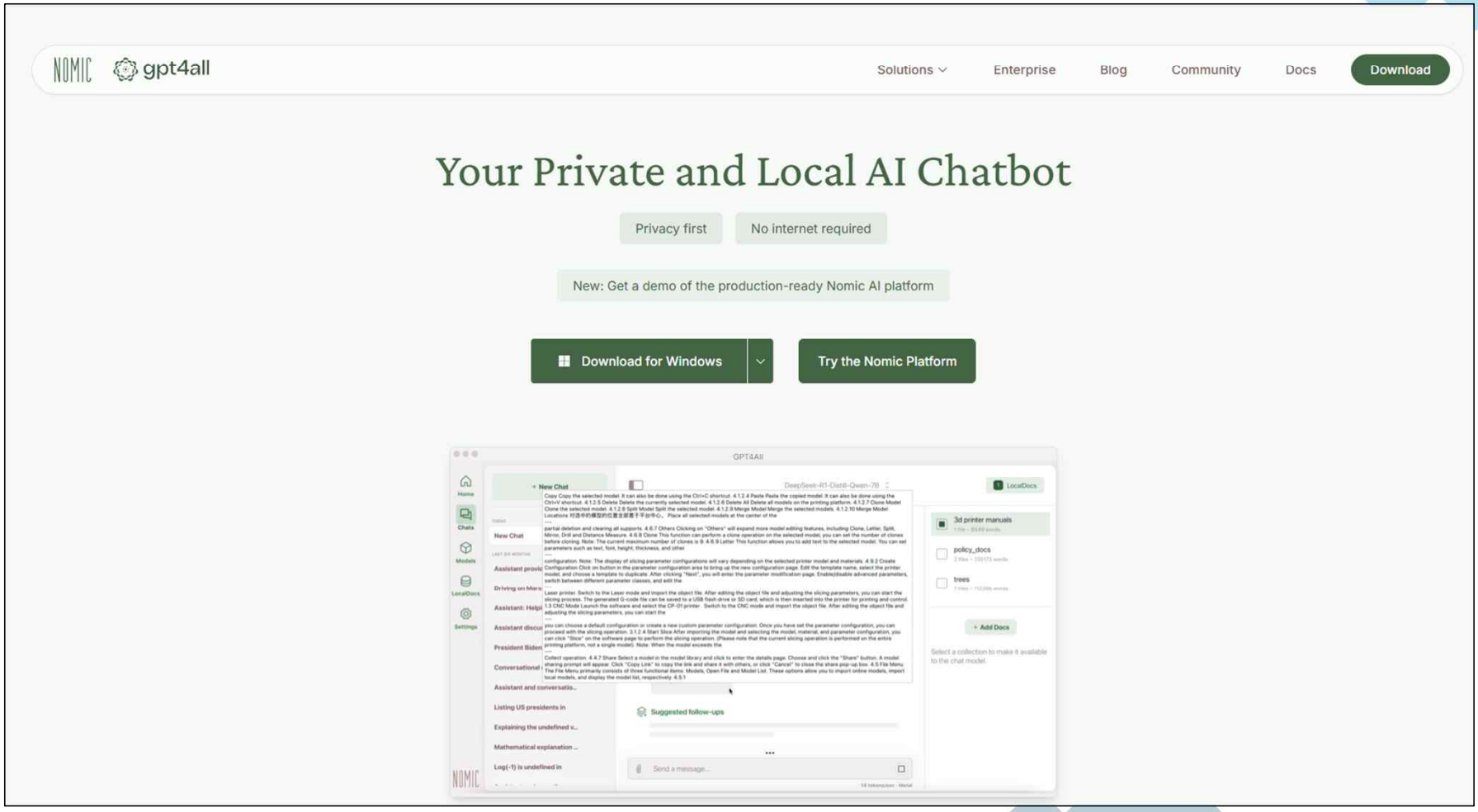
```python
from langchain_community.llms import GPT4All

llm = GPT4All(model="mistral-7b-instruct")
response = llm.invoke("LangChain이란 무엇인가요?")
print(response)
```

---

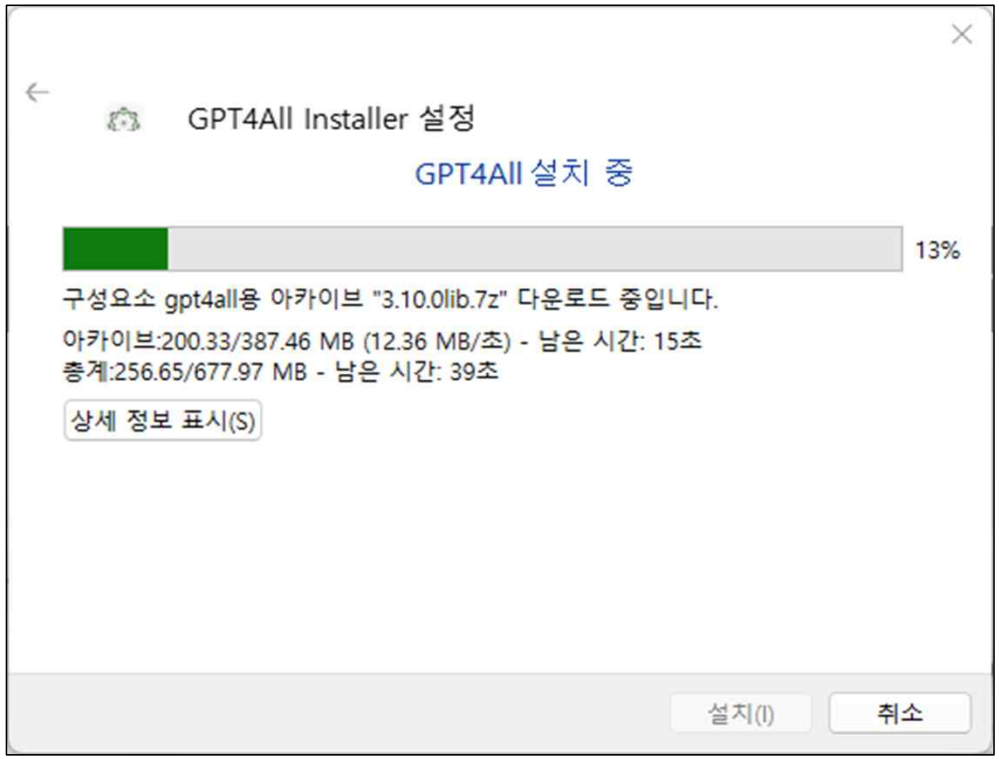
6. 장점과 한계

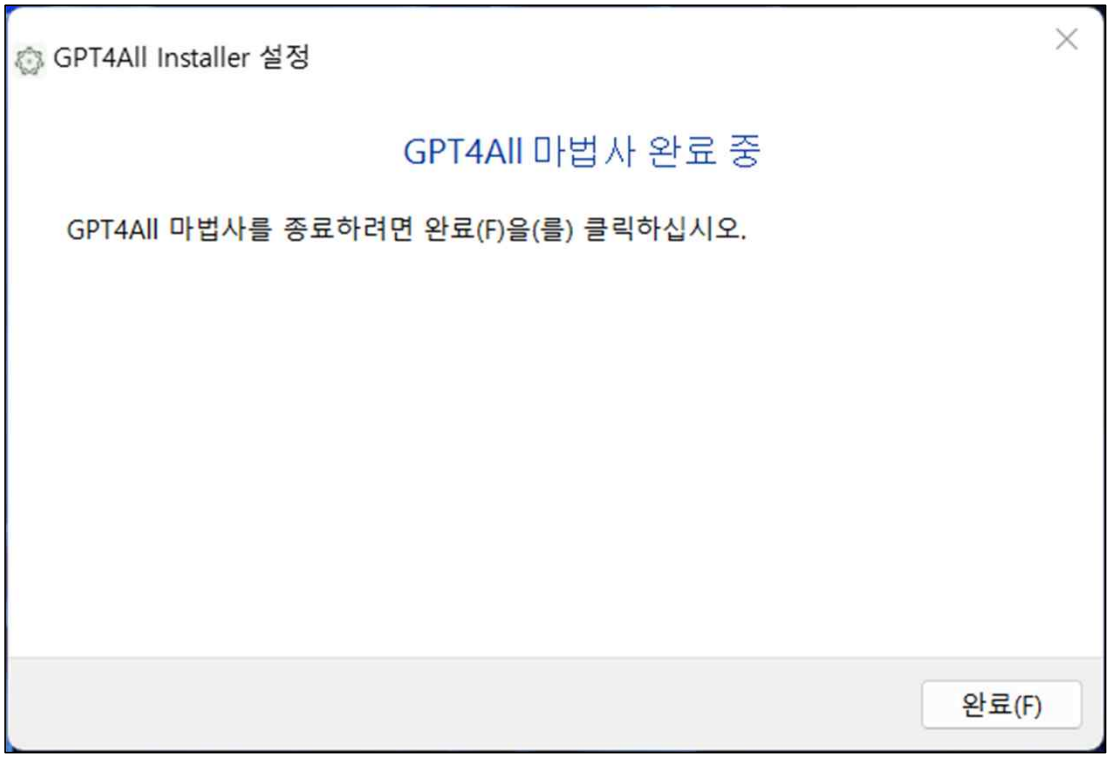
장점	한계
오프라인 환경에서도 사용 가능	성능이 상용 GPT-4보다 낮음
개인 데이터 보안 강화	한국어 모델 품질은 아직 제한적
무료 모델 다양	모델 파일 크기가 큼(4~13GB)
LangChain, Ollama 등과 통합 용이	추론 속도는 CPU 성능에 따라 다름

https://www.nomic.ai/gpt4all 사이트에서 [Download for Windows] 버튼을 클릭하여 gpt4all-installer-win64.exe 파일을 다운 받아 설치한다

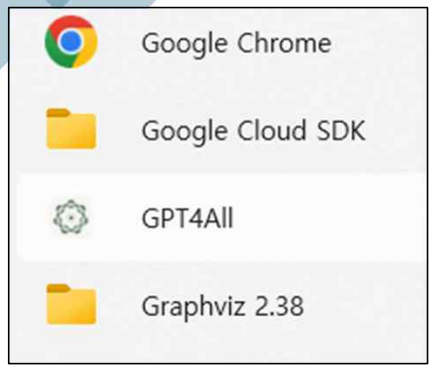


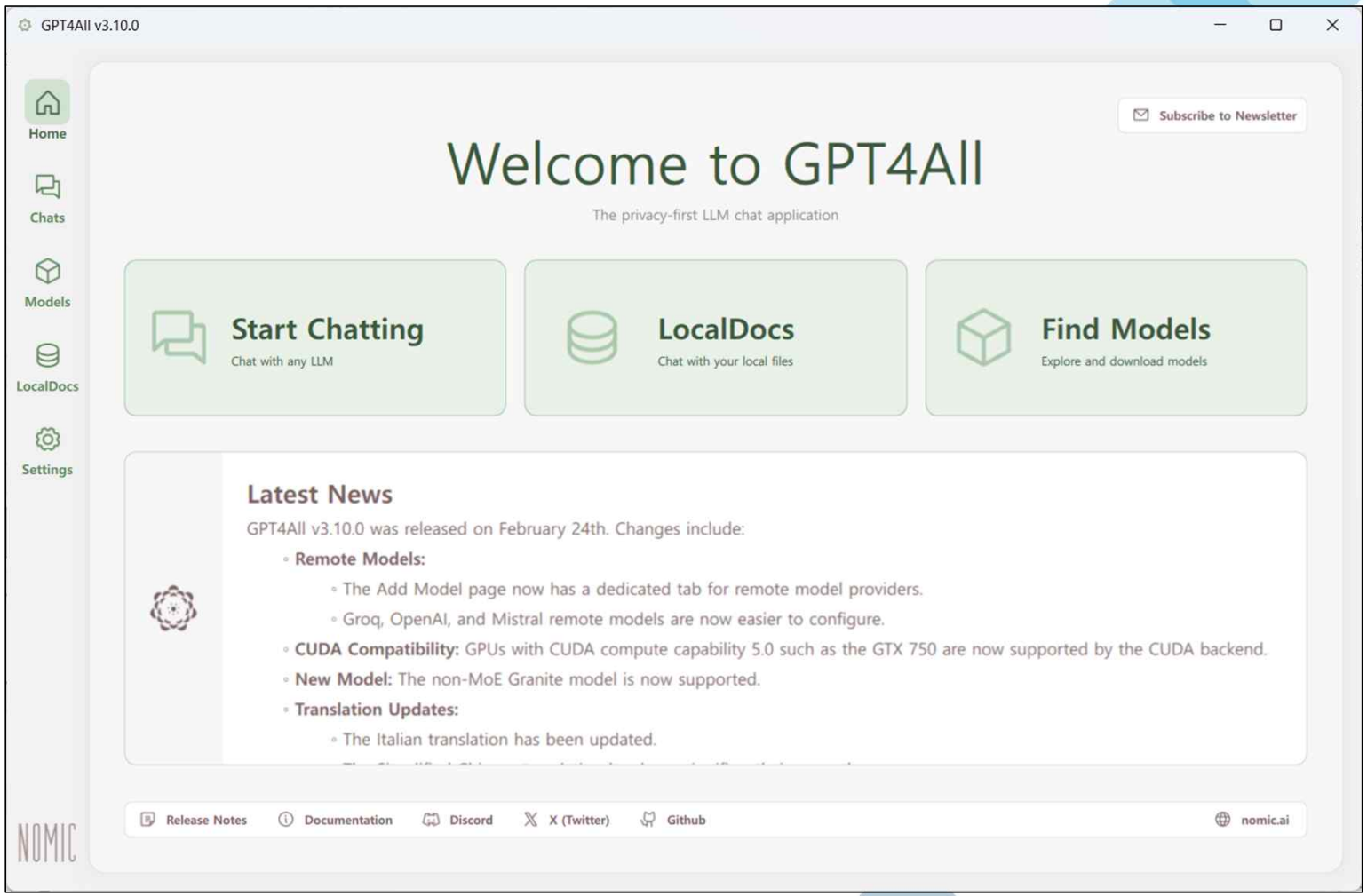
설치가 끝나면 [완료]버튼을 눌러 설치를 마친다



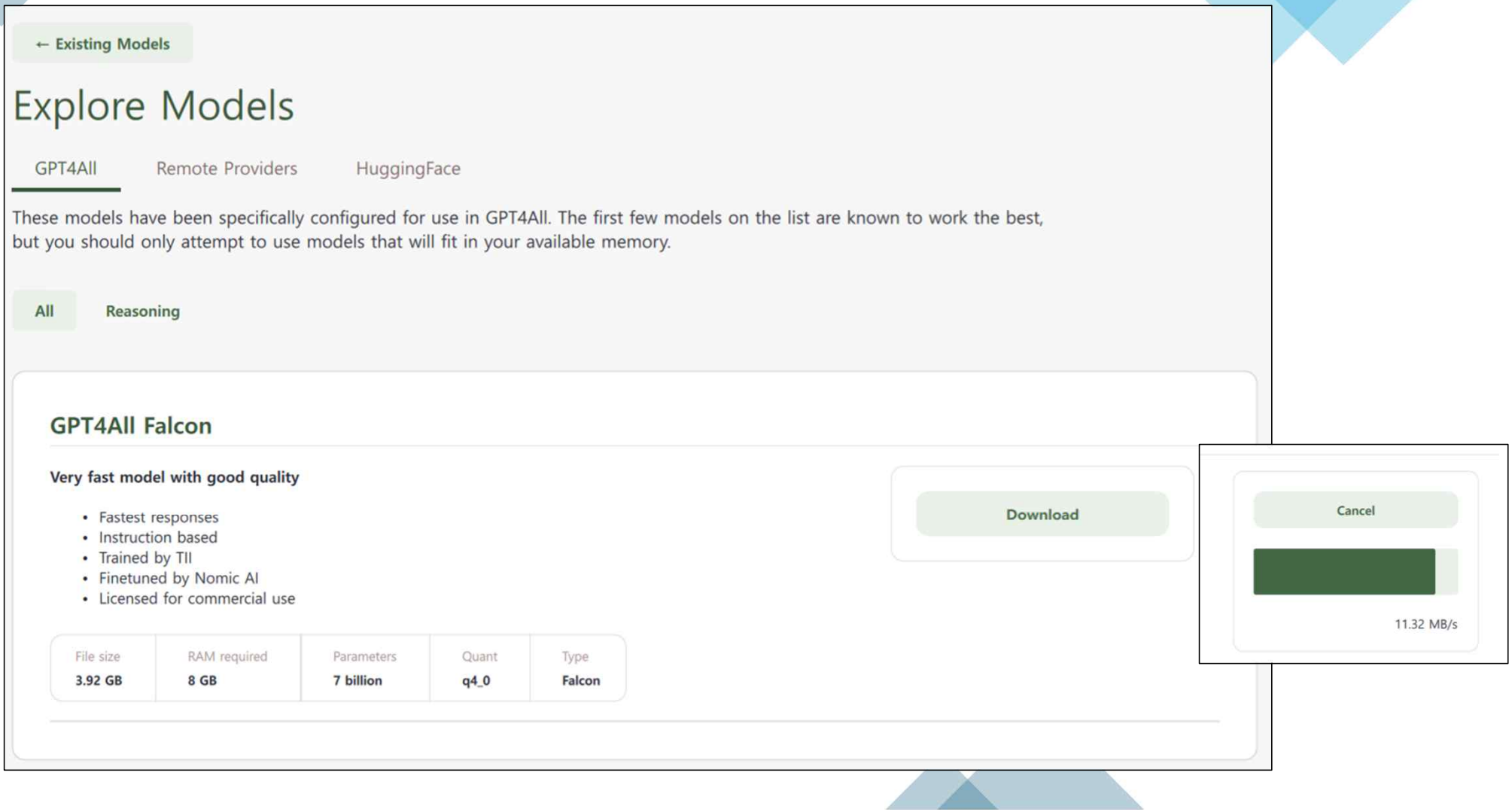


Windows 메뉴에서 GPT4AIII 프로그램을 실행한다





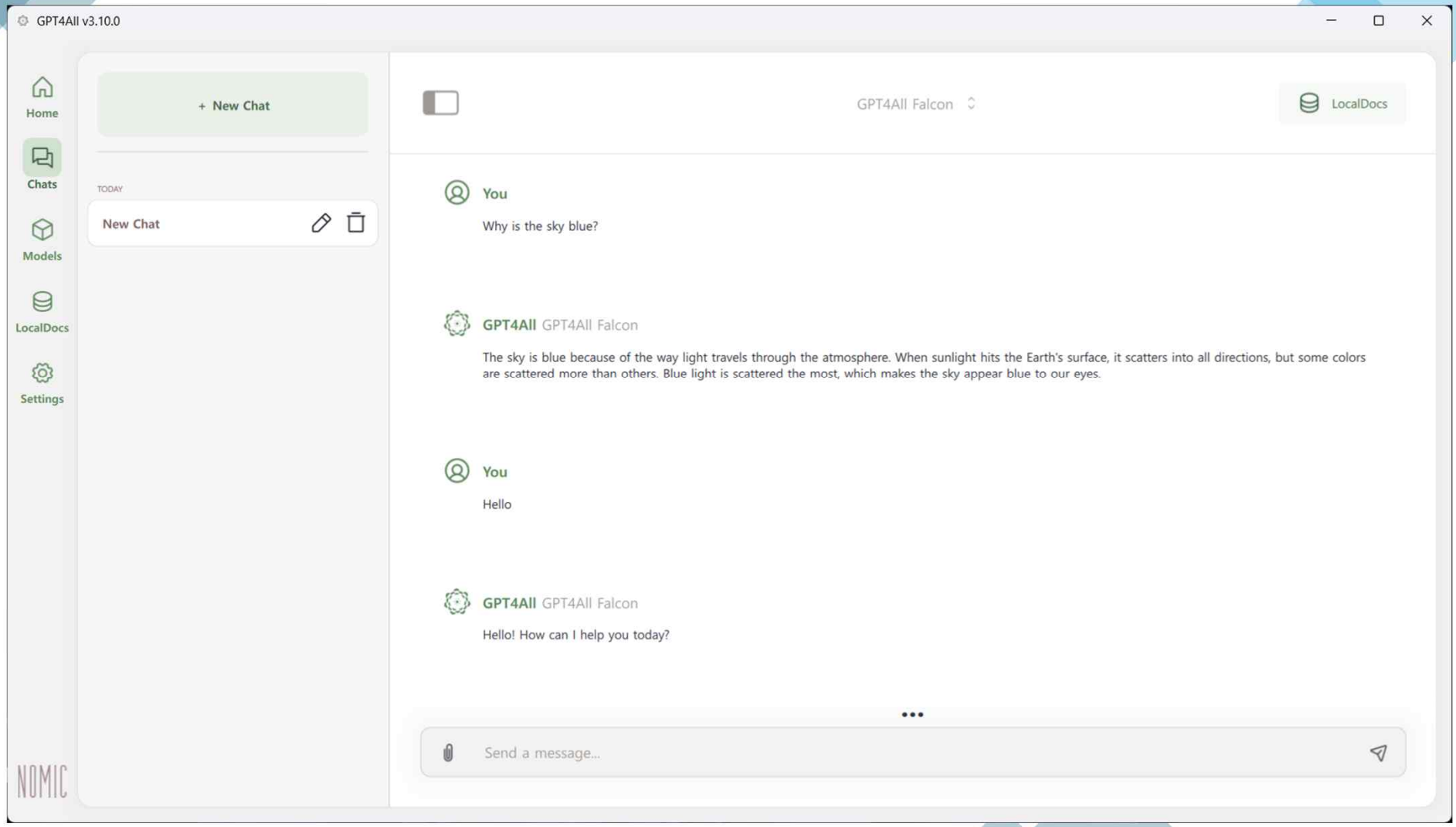
홈화면에서 [Find Model]을 클릭하고 "**GTP4All Falcon**" 모델을 찾아 다운로드한다



C:₩Users₩storm₩AppData₩Local₩nomic.ai₩GPT4All₩ 에서 다운로드 된 파일을 확인할 수 있다

이름	수정한 날짜	유형	크기
cache	2025-10-22 오후 6:37	파일 폴더	
gpt4all-falcon-newbpe-q4_0.gguf	2025-10-23 오후 2:54	GGUF 파일	4,112,299KB
localdocs_v3.db	2025-10-22 오후 6:37	Data Base File	76KB
log.txt	2025-10-23 오후 2:54	텍스트 문서	2KB
test_write.txt	2025-10-22 오후 6:37	텍스트 문서	0KB

채팅 질의 결과 화면 (한국어는 지원 안됨)



파이썬 코드 예제

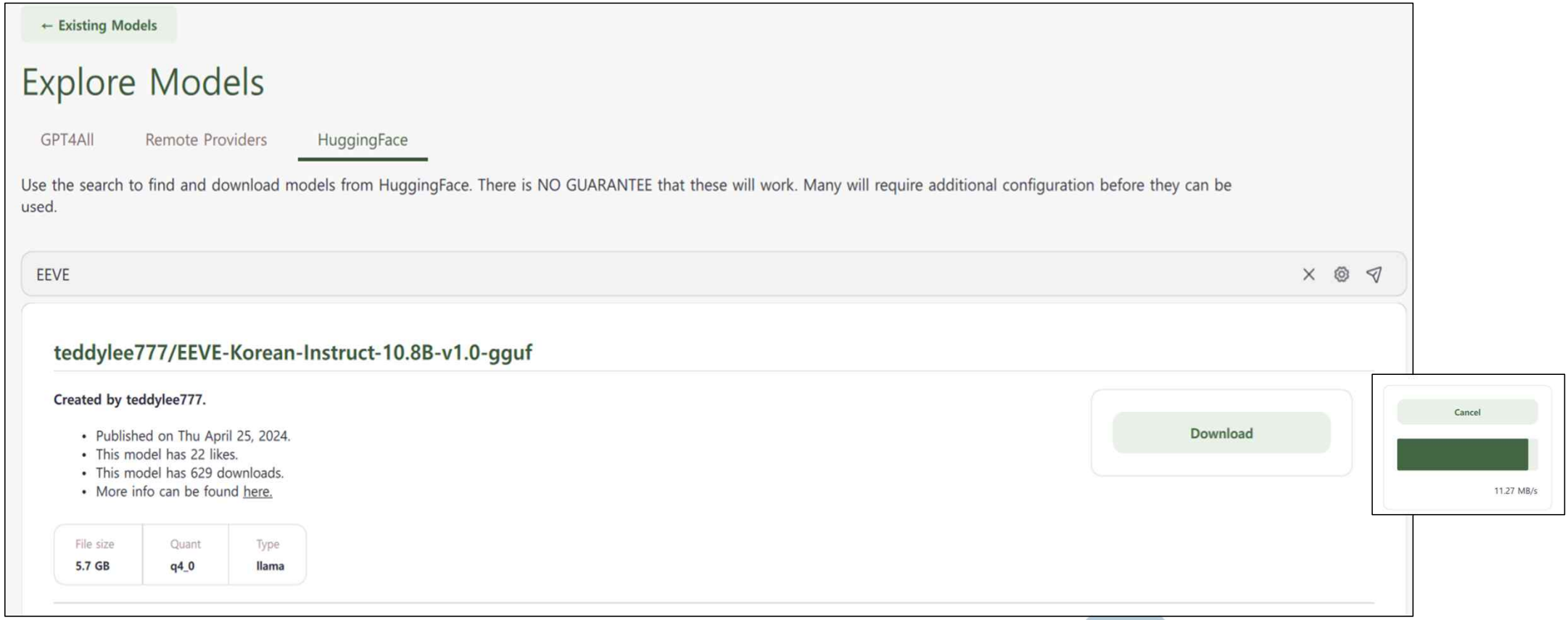
```
# ! pip install  gpt4all
```

```
# GTP4All 프로그램에서 모델(3.9GB)을 미리 다운로드해 놓은 다음 실행
# 모델 다운로드 경로:  C:\Users\storm\AppData\Local\nomic.ai\GPT4All\gpt4all-falcon-newbpe-q4_0.gguf

from langchain_community.llms import GPT4All

model_path = "C:/Users/storm/AppData/Local/nomic.ai/GPT4All/gpt4all-falcon-newbpe-q4_0.gguf"

llm = GPT4All(model=model_path)
response = llm.invoke("Why is the sky blue?") # 한국어는 지원 안됨
print(response)
```

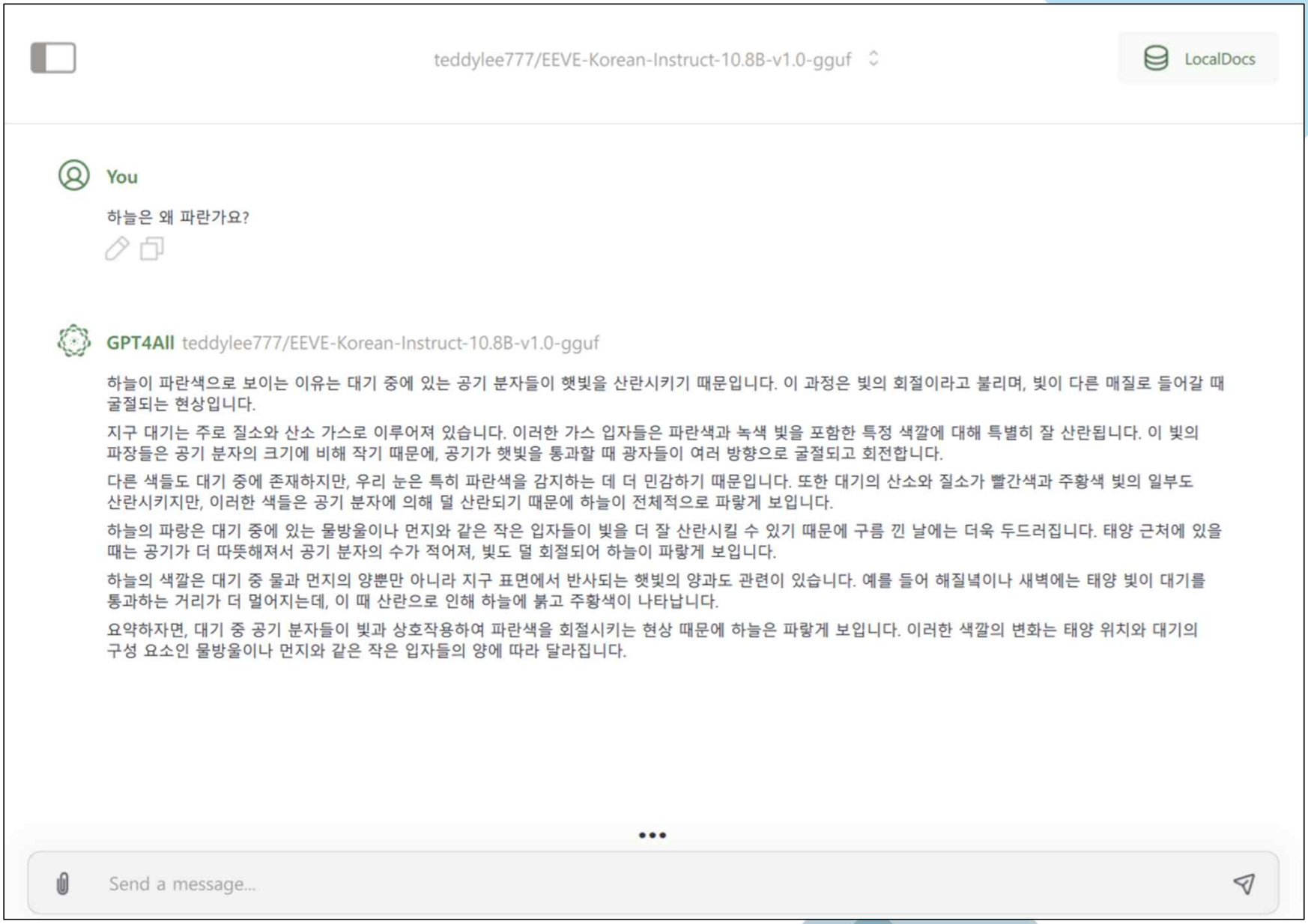
```
The sky appears blue because of a process called Rayleigh scattering. When sunlight passes through the Ea
rth's atmosphere, it scatters in all directions, including towards the blue end of the visible spectrum.
This scattering causes the blue light to be scattered more than other colors, making the sky appear blue.
```

한국어 지원 모델 다운로드

앱 실행 후 Explore Models → HuggingFace 탭을 선택하고검색창에 EEVE 입력
“teddylee/EEVE-Korean-Instruct-10.8B-v1.0.gguf” 선택 후 Download 클릭 (5.7GB)



한국어 지원 모델 질의 결과



파이썬 코드 예제

```python
#  모델 초기화
# model는 GPT4All 모델 파일의 경로를 지정
model_path = "C:/Users/storm/AppData/Local/nomic.ai/GPT4All/EEVE-Korean-Instruct-10.8B-v1.0-Q4_0.gguf"

llm = GPT4All(
    model=model_path,
    # backend="gpu",  # GPU 설정
    streaming=True,   # 스트리밍 설정
    callbacks=[StreamingStdOutCallbackHandler()],  # 콜백 설정
)

# 체인 생성
chain = prompt | llm | StrOutputParser()

# 실행
response = chain.invoke({"question": "하늘은 왜 파란가요?"})   # CPU에서 동작 응답 느림
print(response)
```

```
하늘이 파란색으로 보이는 이유는 대기 중 산소와 질소가 포함된 가스 분자들이 태양광을 흡수하고 재방출하기 때문입니다. 이 과정은 빛의 산란이라고 알려져 있습니다.

태양에서 나오는 햇빛은 주로 파랑과 보라색 같은 단파장 색깔로 구성되어 있으며, 이는 공기 입자에 의해 더 잘 산란됩니다. 대기 중 가스 분자들은 태양광을 흡수하고 재방출하는데, 이 과정에서 일부 빛을 다시 지구 표면으로 되돌려 보내게 됩니다.
```

감사합니다