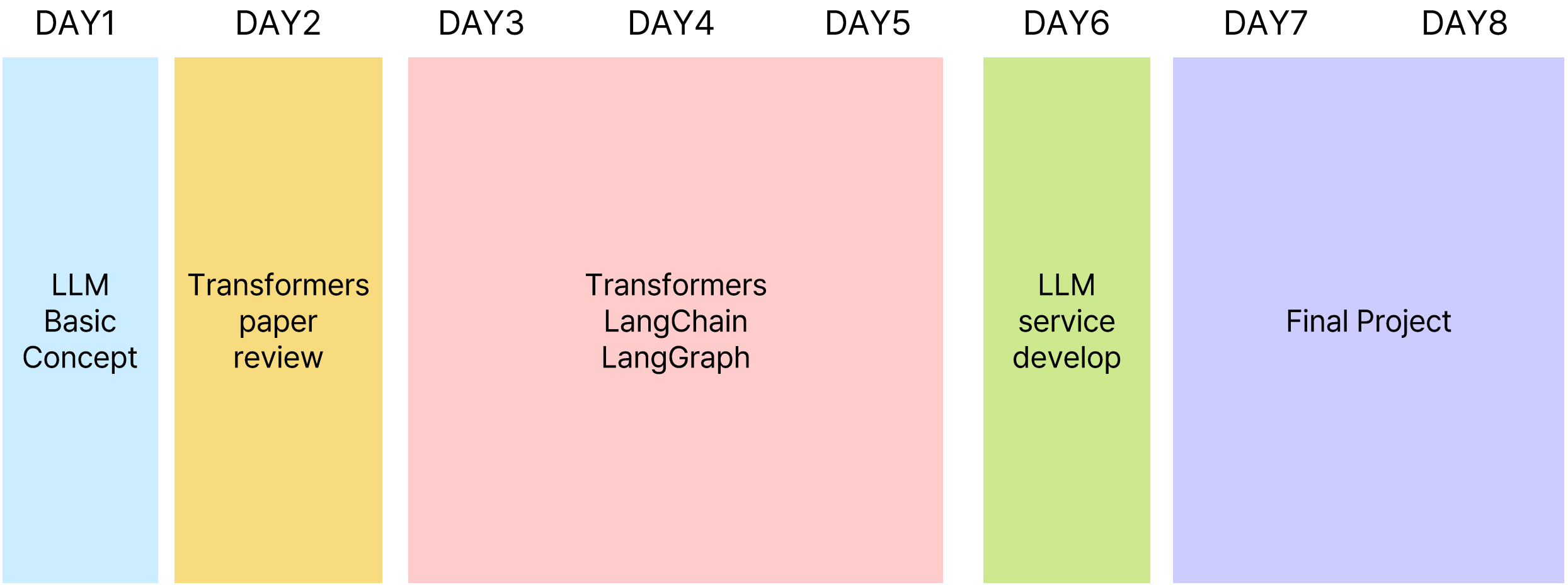
GEN AI 인텐시브 과정

강사 장철원

Section 0

코스소개



Section 0

목차

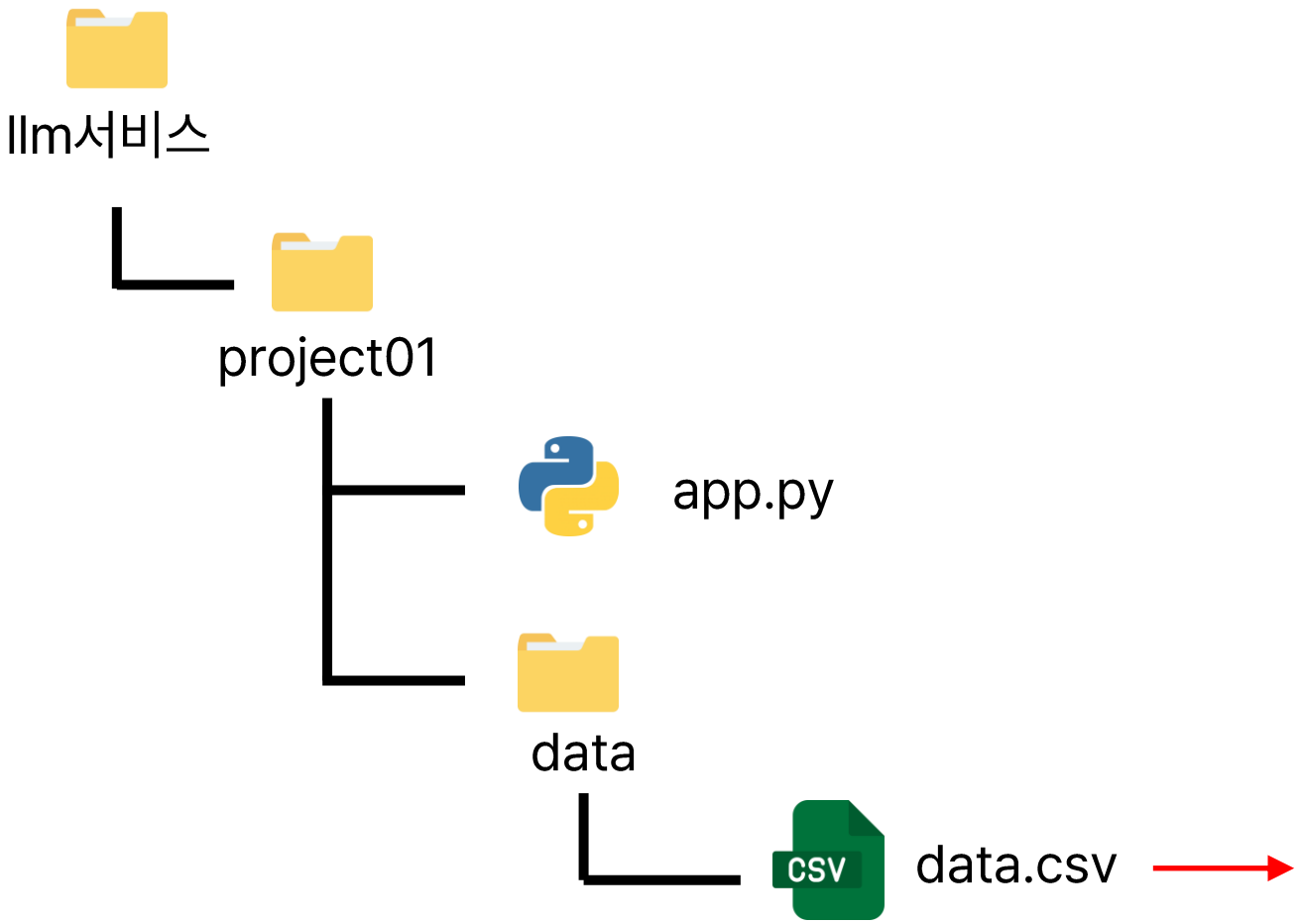
- ❑ 트랜스포머 기반 서비스 실습
- ❑ LangChain 기반 서비스 실습
- ❑ Multiturn 기반 서비스 실습
- ❑ LangGraph 기반 서비스 실습

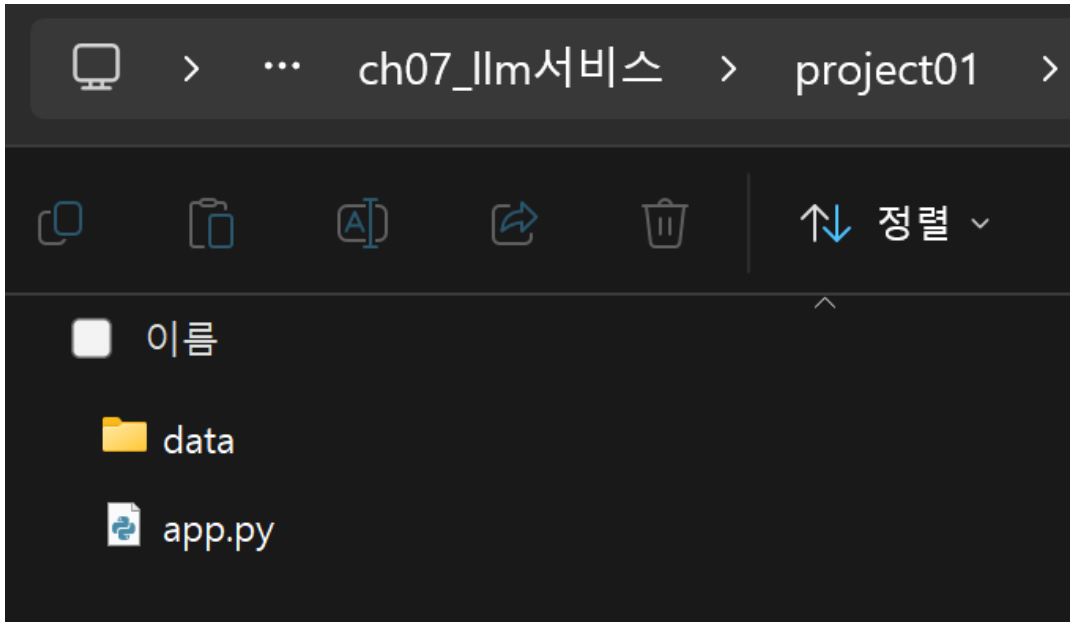
GEN AI 인텐시브 과정

Section 1. LLM 기반 챗봇 만들기

Section 1-1. 트랜스포머 기반 서비스 실습

폴더 구성





	A	B	C	D	E	F
1	text					
2	Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.					
3	PM 장비의 필터는 주 1회 정기적으로 교체해야 합니다.					
4	웨이퍼 투입 전 챔버 내부의 온도 안정화가 필요합니다.					
5	불량률이 2%를 초과할 경우 원인 분석 보고서를 제출해야 합니다.					
6	클린룸 입장 전에는 반드시 정전기 방지복을 착용해야 합니다.					
7	포토 공정 시 PR 코팅 두께는 1.5μm 이상 유지해야 합니다.					

app.py

project01 > app.py > load_qa_model

```python
import torch
import chromadb
import pandas as pd
import streamlit as st
from sentence_transformers import SentenceTransformer
from transformers import AutoTokenizer, AutoModelForQuestionAnswering

# ChromaDB 설정(로컬 테스트)
client = chromadb.PersistentClient(path="./chromaDB")
collection = client.get_or_create_collection("data01")

```

실습에 필요한 라이브러리

로컬 테스트 사용

app.py

```python
19  ##########################################
20  # Streamlit UI 탭 설정
21  ##########################################
22  st.set_page_config(page_title="RAG 기반 QA 시스템", layout='wide')
23  tab1, tab2 = st.tabs(["데이터 저장", "무엇이든 물어보세요"])
24
25
26  # 문장 임베딩
27  @st.cache_resource
28  def load_ebd_model():
29      model = SentenceTransformer("sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
30      return model
31
32  embedding_model = load_ebd_model()
33
34  # QA 모델 (질문/답변)
35  @st.cache_resource
36  def load_qa_model():
37      model_name = "monologg/koelectra-base-v3-finetuned-korquad"
38      tokenizer = AutoTokenizer.from_pretrained(model_name)
39      model = AutoModelForQuestionAnswering.from_pretrained(model_name)
40      return tokenizer, model
41
42  qa_tokenizer, qa_model = load_qa_model()
```

화면 탭은 2개로 구성
"데이터 저장" 탭: csv 파일을 업로드 하고 chromaDB 저장
"무엇이든 물어보세요": 사용자 질문에 대한 답변 생성

? 다음 페이지에서 설명

문장 임베딩 모델

QA 모델

@st.cache_resource

- streamlit에서 리소스를 캐싱하기 위해 사용하는 데코레이터
- 임베딩 모델 같은 무거운 객체를 처음 한번만 로드하고 이후로는 다시 로딩하지 않음
- 처음 실행했을 때 모델을 메모리에 로드하고, 이후 재실행 할 때는 캐시된 모델을 재사용함

- 사용 예

```
@st.cache_resource
def load_ebd_model():
  model = SentenceTransformer("sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
  return model

embedding_model = load_ebd_model()
```

```python
#-----------------------------------------
# 탭 1: 데이터 저장
#-----------------------------------------
with tab1:
  st.header("CSV 파일 업로드 및 저장")

  # CSV 파일 업로드
  upload_file = st.file_uploader("CSV 파일을 업로드 하세요", type=["csv"])

  if upload_file is not None:
    df = pd.read_csv(upload_file, encoding='utf8')

    if "sentence"not in df.columns:
      st.error("csv 파일에 sentence 컬럼이 없습니다.")
    else:
      sc_list = df['sentence'].tolist()

      # 데이터 저장 버튼
      if st.button("데이터 저장"):
        ids_list = []
        ebd_list = []
        txt_list = []

        for ith, sc in enumerate(sc_list):
          idx = f"doc{ith+1}"
          ids_list.append(idx)
          txt_list.append({"text": sc})
          embedding = embedding_model.encode(sc).tolist()
          ebd_list.append(embedding)

        # ChromaDB에 데이터 추가
        collection.add(
          ids=ids_list,
          embeddings=ebd_list,
          metadatas=txt_list
        )

        st.success("데이터가 ChromaDB에 저장되었습니다.")
```

탭 1: CSV 파일 업로드 및 ChromaDB 저장

파일 업로드

업로드 된 파일 읽기

sentence에 저장된 문장들 임베딩
&
ChromaDB에 저장하게끔 다듬기

ChromaDB에 저장

```python
#-----------------------------------------
# 탭 2: 질문 답변 (QA 시스템)
#-----------------------------------------
with tab2:
  st.header("무엇이든 물어보세요")

  # 사용자 질문 입력
  question = st.text_input("질문을 입력하세요:")

  if st.button("분석 시작"):
    if not question.strip():
      st.warning("질문을 입력해주세요")

    else:
      # 1. 입력된 질문을 벡터로 변환
      query_ebd = [embedding_model.encode(question).tolist()]

      # 2. ChromaDB에서 유사한 문장 검색
      result = collection.query(query_embeddings=query_ebd, n_results=1)

      if result['metadatas']:
        best_context = result['metadatas'][0][0]['text']

        # 3. Transformer 모델을 사용하여 답변 생성
        inputs = qa_tokenizer(question, best_context, return_tensors="pt")

        with torch.no_grad():
          outputs = qa_model(**inputs)

        start_idx = torch.argmax(outputs.start_logits)
        end_idx = torch.argmax(outputs.end_logits) + 1
        answer = qa_tokenizer.decode(inputs["input_ids"][0][start_idx:end_idx], skip_special_tokens=True)

        # 4. 결과 출력
        st.subheader("검색된 문서 내용")
        st.write(f"**{best_context}**")

        st.subheader("답변")
        st.write(f"{answer}")

      else:
        st.warning("유사한 문장을 찾을 수 없습니다.")
```

탭 2: 질문 답변 시스템

질문 입력

질문 임베딩

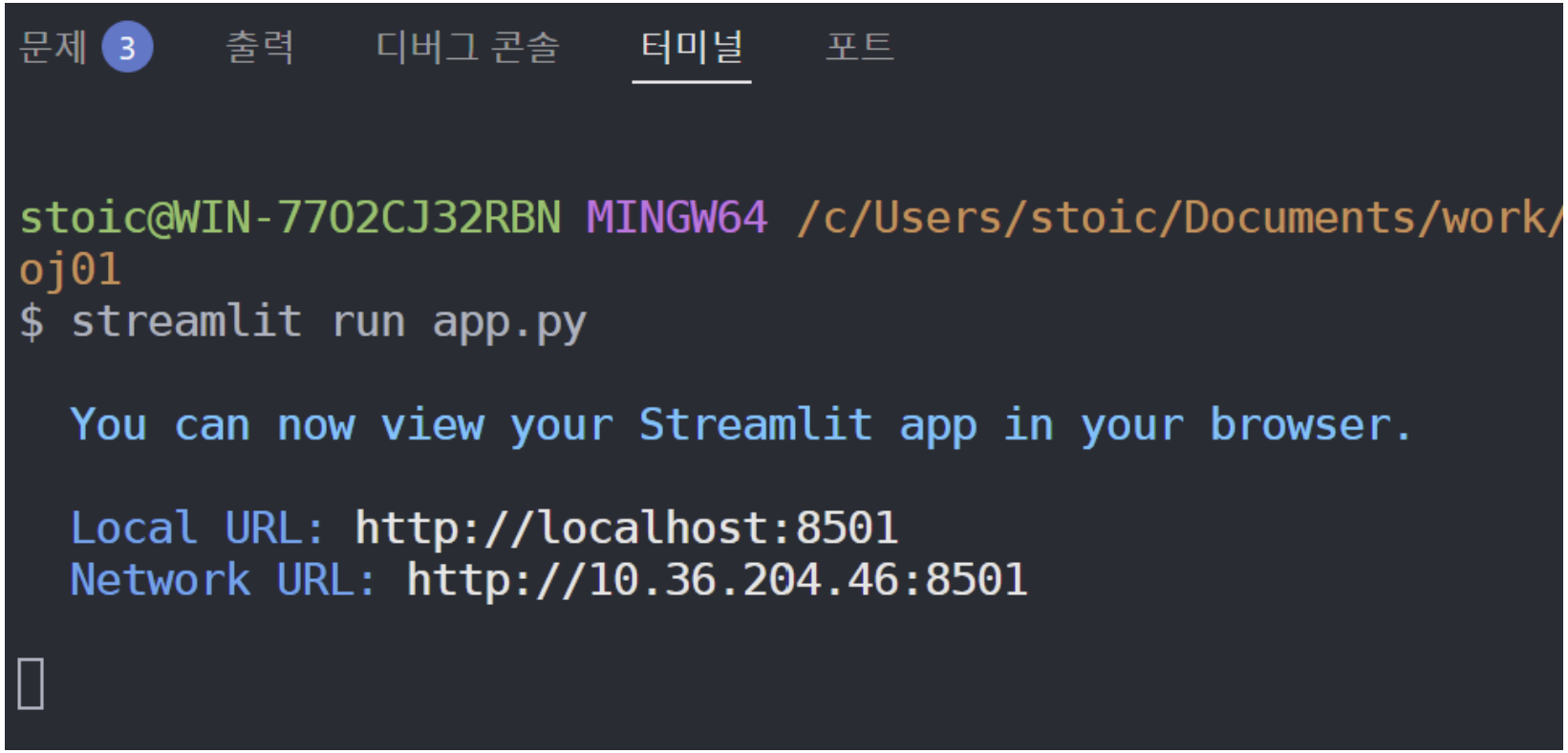
ChromaDB에서 질문과 가장 유사한 문장 검색

질문과 가장 유사한 문장

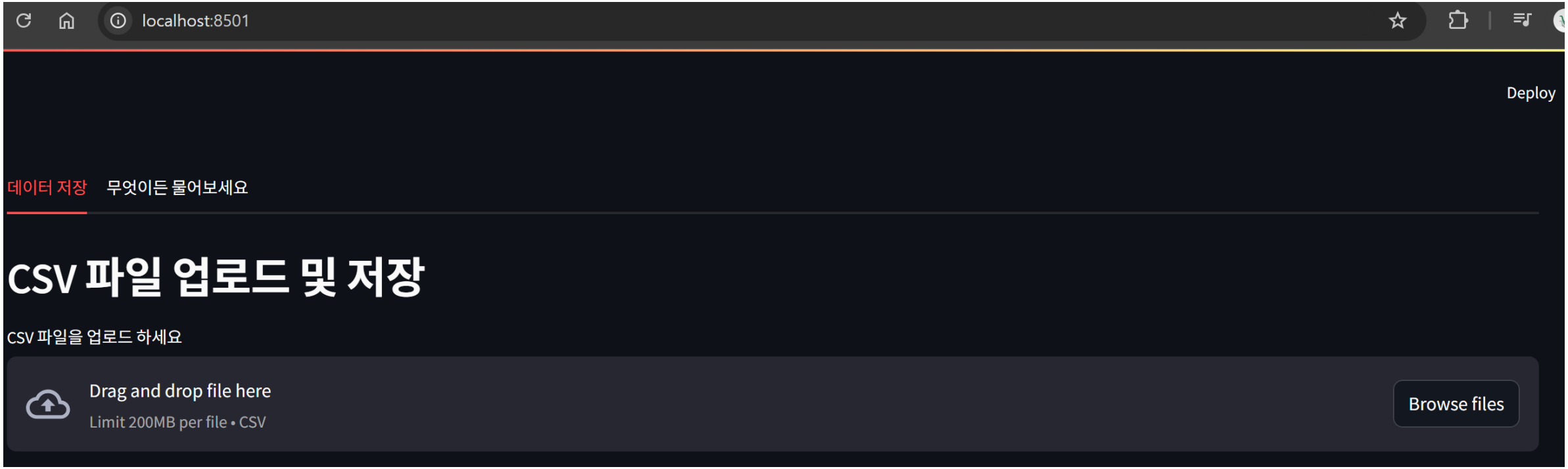
질문에 대한 답변 생성

결과 출력

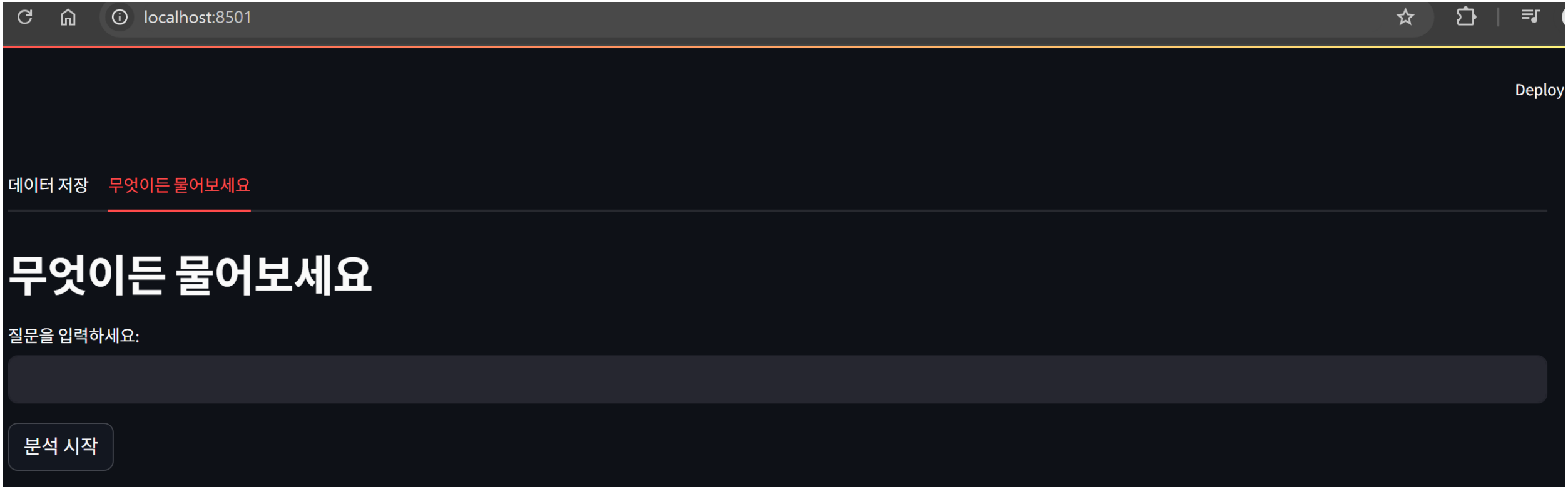
코드 실행


문제 3
출력
디버그 콘솔
터미널
포트
stoic@WIN-77O2CJ32RBN MINGW64 /c/Users/stoic/Documents/work/
oj01
$ streamlit run app.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://10.36.204.46:8501

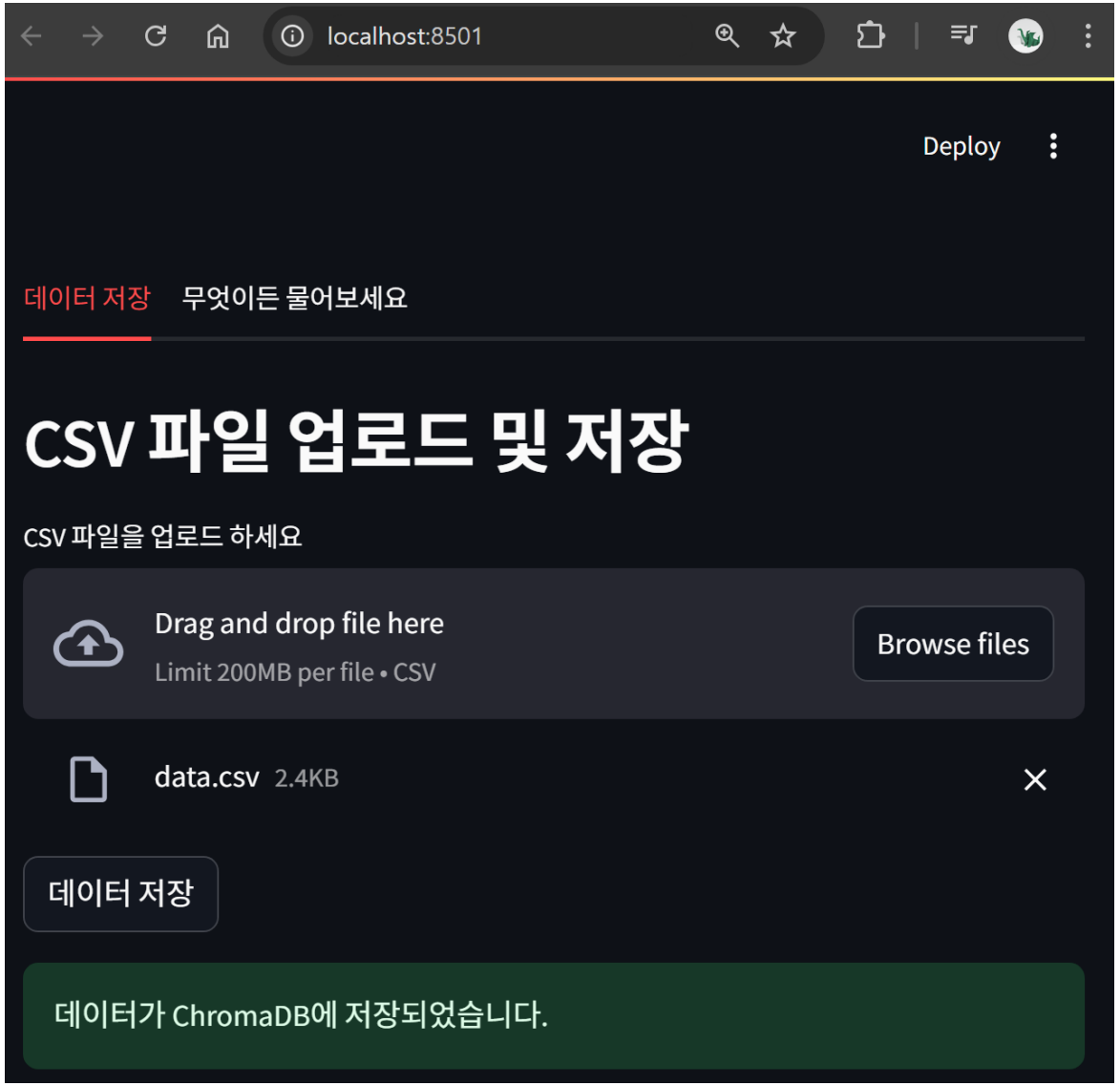
데이터 저장 탭 확인

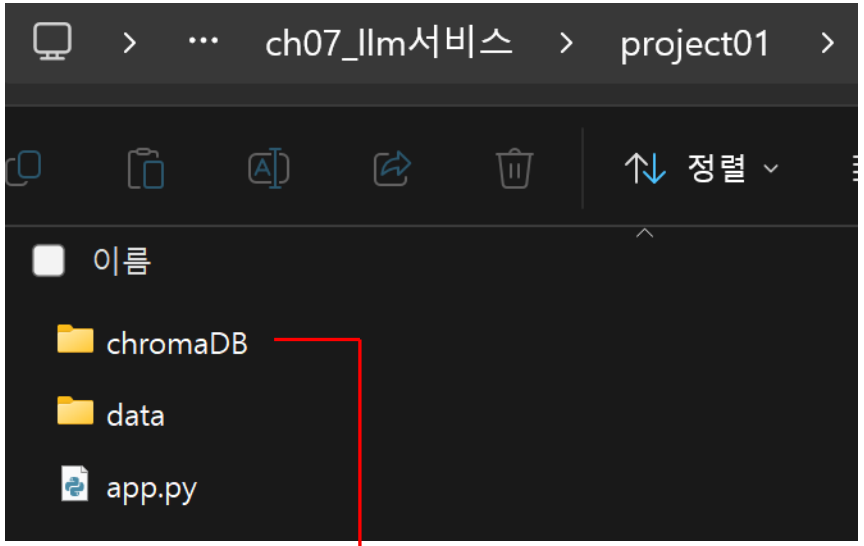

localhost:8501
Deploy
데이터 저장
무엇이든 물어보세요
CSV 파일 업로드 및 저장
CSV 파일을 업로드 하세요
Drag and drop file here
Limit 200MB per file • CSV
Browse files

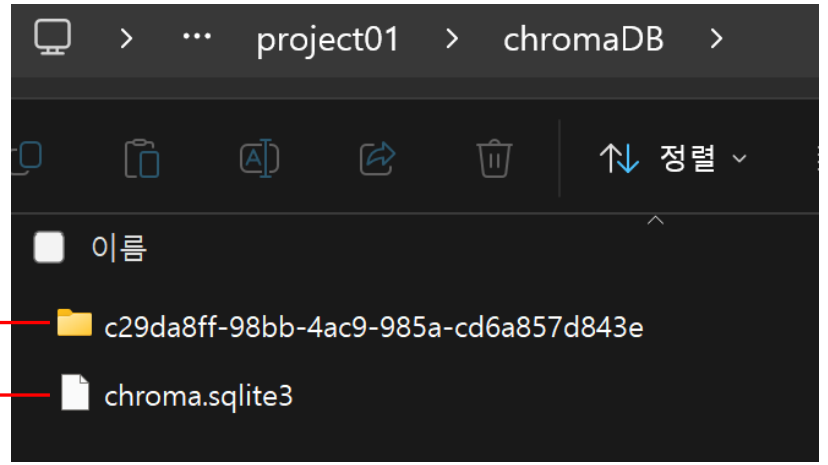
'무엇이든 물어보세요' 탭 확인



csv 파일 업로드 및 저장

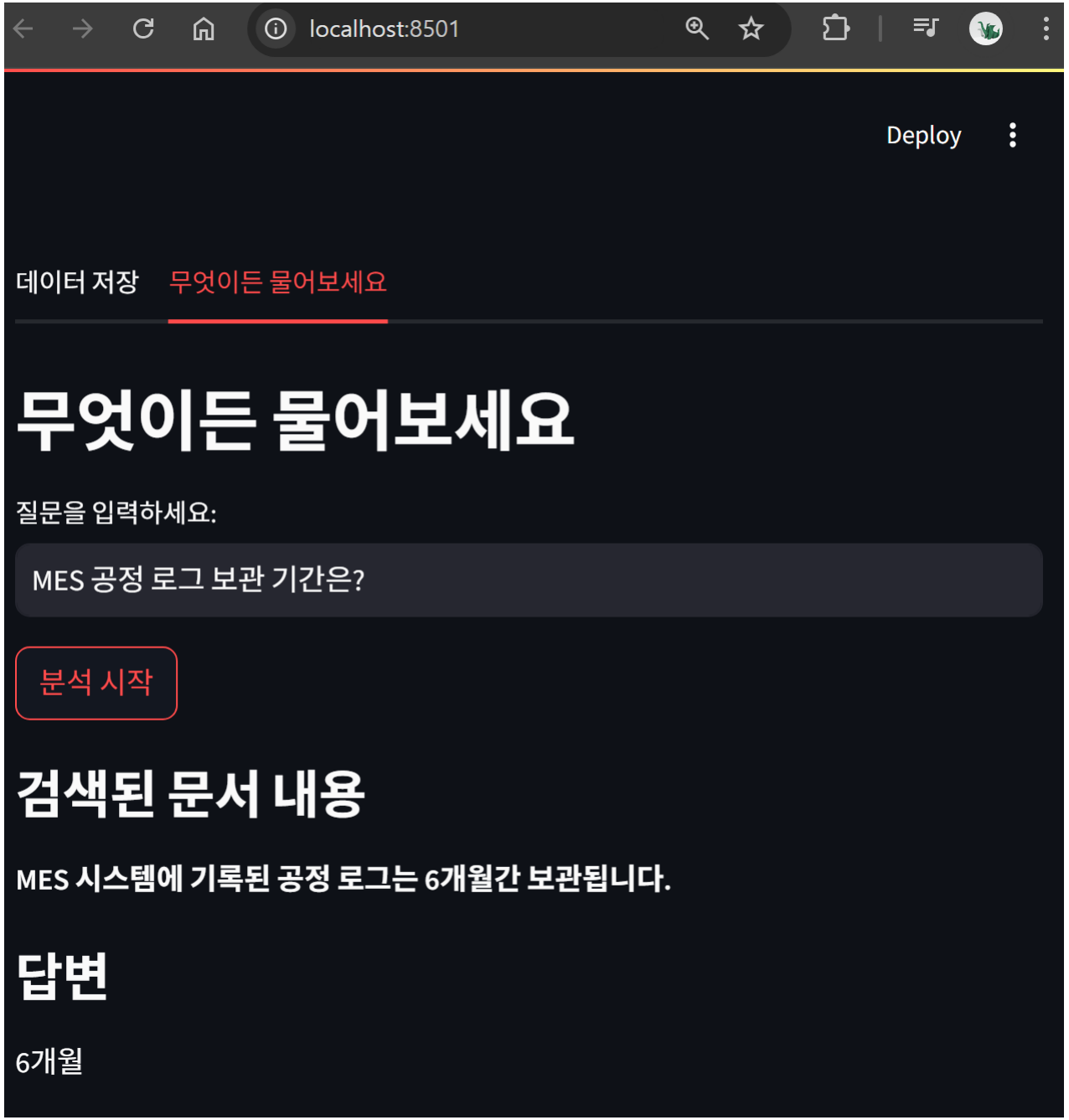






Section
트랜스포머 기반 서비스 실습

‘무엇이든 물어보세요’ 탭 확인

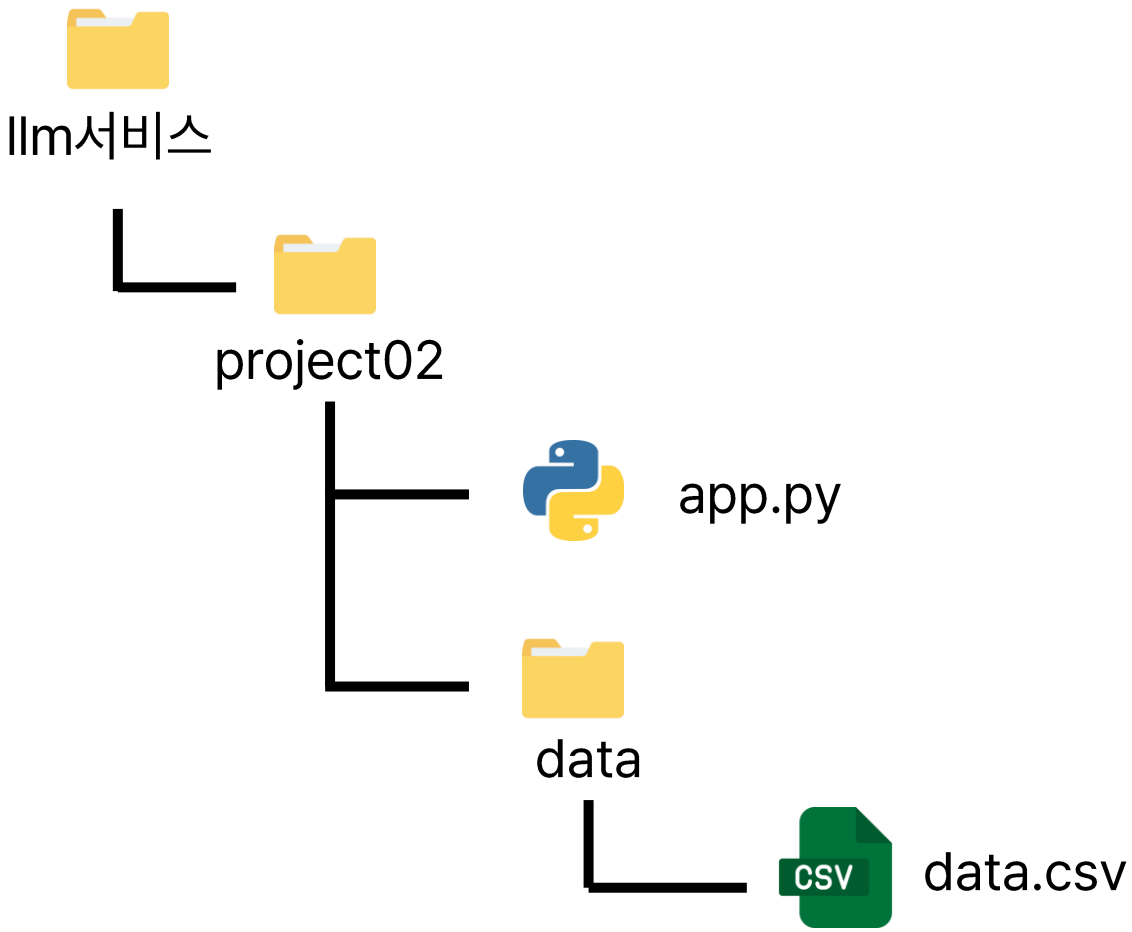


GEN AI 인텐시브 과정

Section 1. LLM 기반 챗봇 만들기

Section 1-2. 랭체인 기반 서비스 실습

폴더 구성



	A	B	C	D	E	F
1	text					
2	Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.					
3	PM 장비의 필터는 주 1회 정기적으로 교체해야 합니다.					
4	웨이퍼 투입 전 챔버 내부의 온도 안정화가 필요합니다.					
5	불량률이 2%를 초과할 경우 원인 분석 보고서를 제출해야 합니다.					
6	클린룸 입장 전에는 반드시 정전기 방지복을 착용해야 합니다.					
7	포토 공정 시 PR 코팅 두께는 1.5μm 이상 유지해야 합니다.					

소스 코드

app.py

project02 > app.py > load_models

```python
import streamlit as st
import pandas as pd
from langchain_chroma import Chroma
from langchain.schema import Document
from langchain_huggingface import HuggingFaceEmbeddings
from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline

# Streamlit UI 설정
st.set_page_config(page_title="RAG QA 시스템", layout='wide')
tab1, tab2 = st.tabs(["데이터 저장", "질문하기"])
```

소스 코드

project02 > app.py > ...

```python
# 1. 모델 및 임베딩 로딩
@st.cache_resource
def load_models():
    embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")

    model_id = "monologg/koelectra-base-v3-finetuned-korquad"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    model = AutoModelForQuestionAnswering.from_pretrained(model_id)

    qa_pipeline = pipeline(
        "question-answering",
        model=model,
        tokenizer=tokenizer,
        device=-1,
        max_length=512,
        do_sample=False,
        temperature=0.1,
        truncation=True
    )
    return embedding_model, tokenizer, qa_pipeline

embedding_model, qa_tokenizer, text_gen = load_models()
```

소스 코드

```
36  # 2. 탭 1: 데이터 저장
37  with tab1:
38      st.header("CSV 문서 업로드 및 ChromaDB 저장")
39
40      uploaded_file = st.file_uploader("CSV 파일 업로드", type="csv")
41
42      if uploaded_file:
43          df = pd.read_csv(uploaded_file, encoding='utf-8')
44
45          if "text" not in df.columns:
46              st.error("'text' 컬럼이 포함되어야 합니다.")
47          else:
48              docs = [Document(page_content=text) for text in df["text"].tolist()]
49              if st.button("Chroma에 저장"):
50                  Chroma.from_documents(
51                      documents=docs,
52                      embedding=embedding_model,
53                      persist_directory="./chromaDB1"
54                  )
55                  st.success("문서가 ChromaDB에 저장되었습니다.")
```

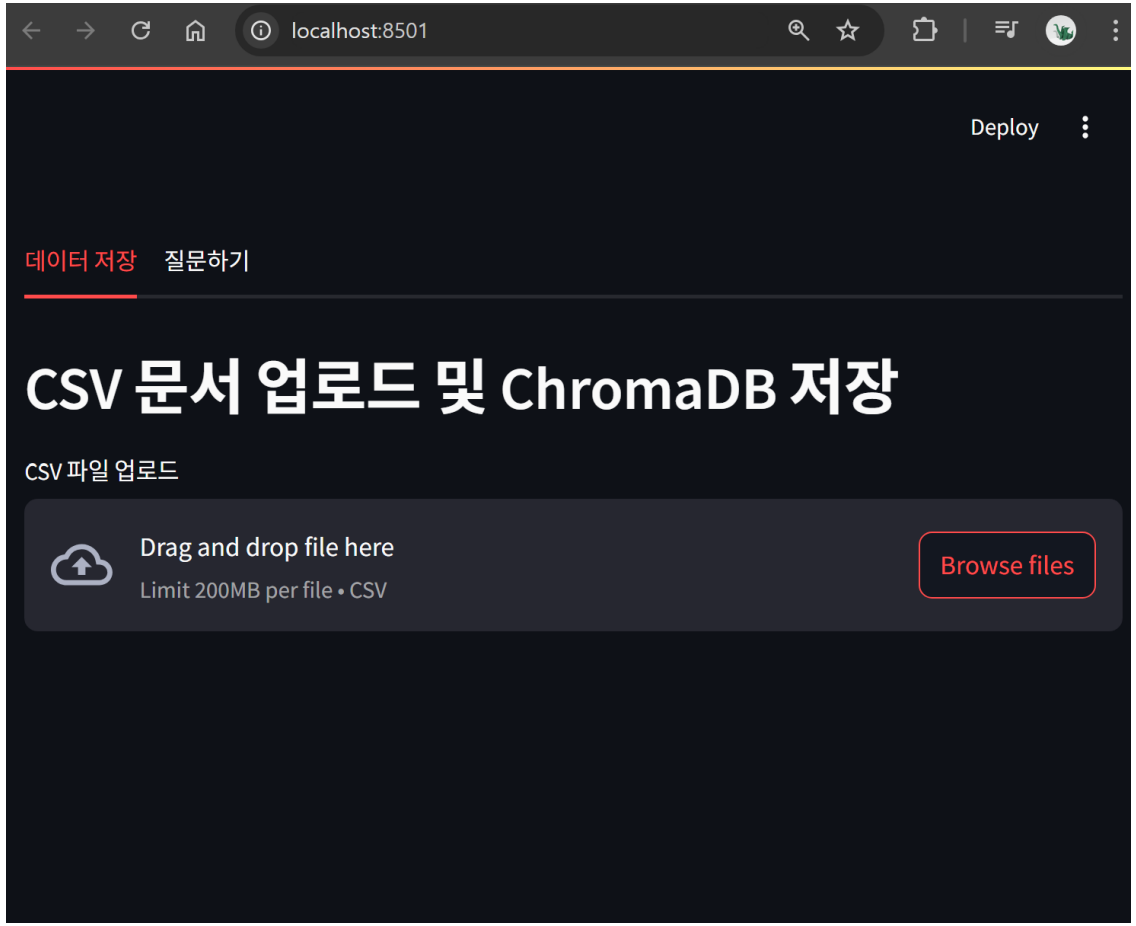
Section
랭체인기반서비스실습

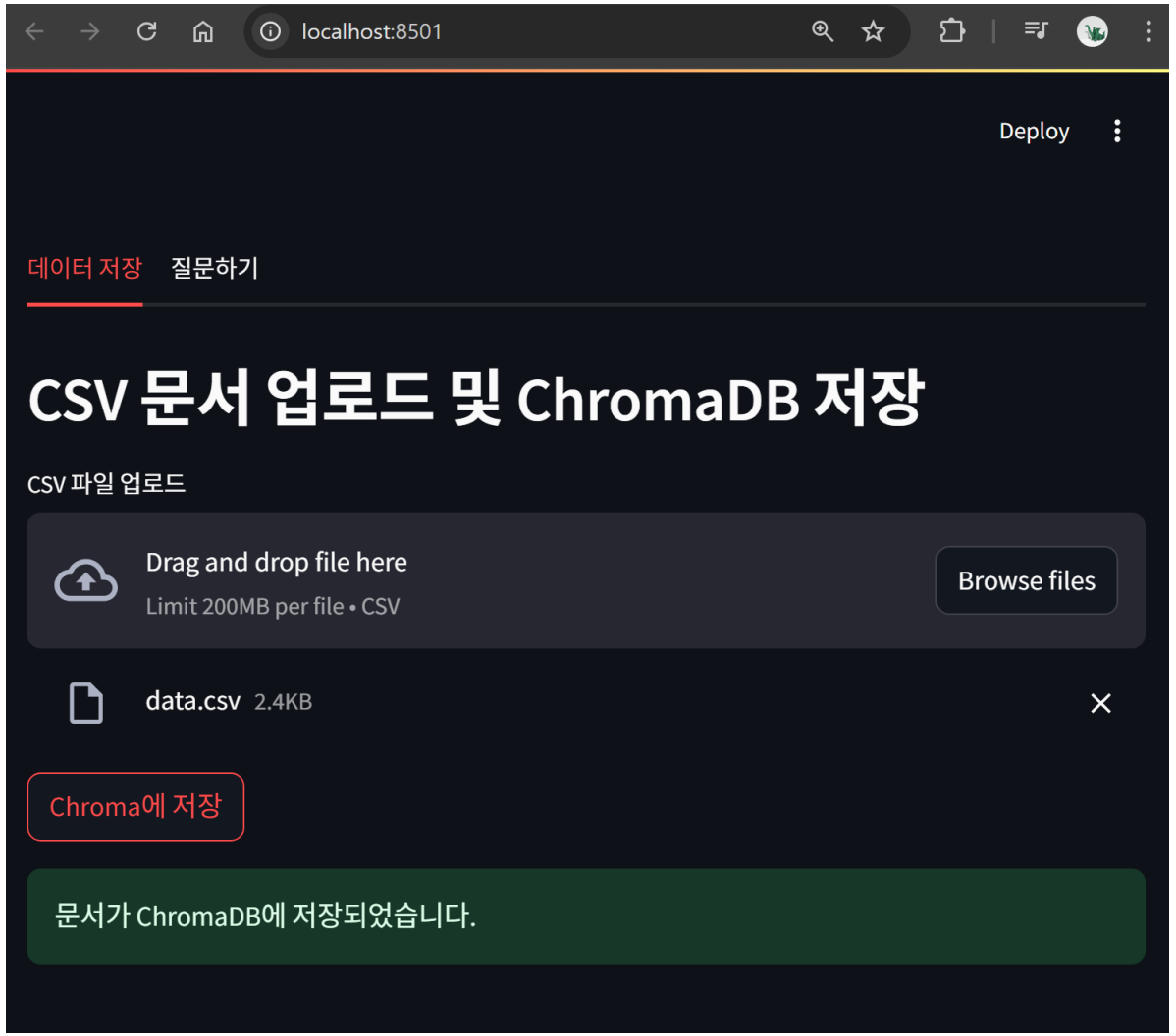
소스 코드

project02 > app.py > ...

```python
# 3. 탭 2: 질문 응답
with tab2:
    st.header("질문을 입력하고 응답을 확인하세요.")
    question = st.text_input("질문을 입력하세요:")

    if st.button("답변 생성"):

        if not question:
            st.warning("질문을 입력하세요.")
        else:
            retriever = Chroma(
                persist_directory="./chromaDB1",
                embedding_function=embedding_model
            ).as_retriever(search_kwargs={"k": 3})
            docs = retriever.invoke(question)

            if not docs:
                st.error("유사 문서를 찾지 못했습니다.")
            else:
                best_context = docs[0].page_content
                result = text_gen(question=question, context=best_context)

                st.subheader("정답")
                st.write(result["answer"])

                st.subheader("참고 문서 (Top 3)")
                for i, doc in enumerate(docs):
                    st.markdown(f"**{i+1}.** {doc.page_content}")
```

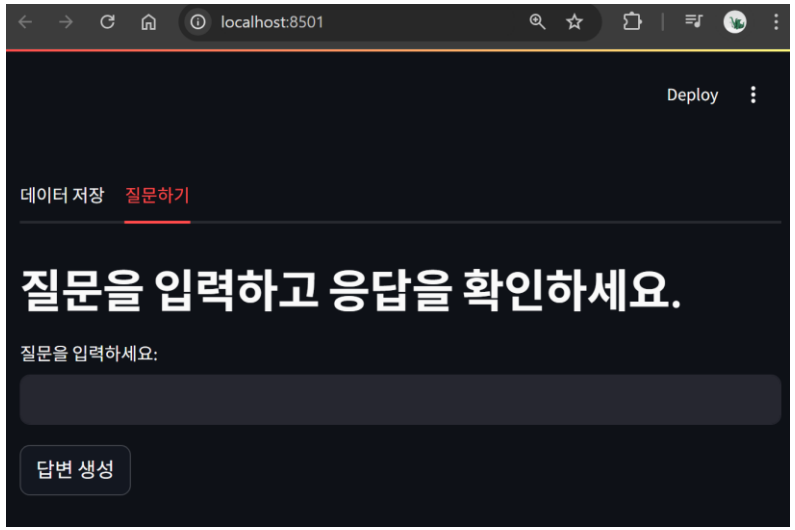
결과 확인

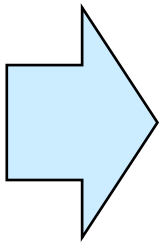

localhost:8501
Deploy
데이터 저장
질문하기
CSV 문서 업로드 및 ChromaDB 저장
CSV 파일 업로드
Drag and drop file here
Limit 200MB per file • CSV
Browse files

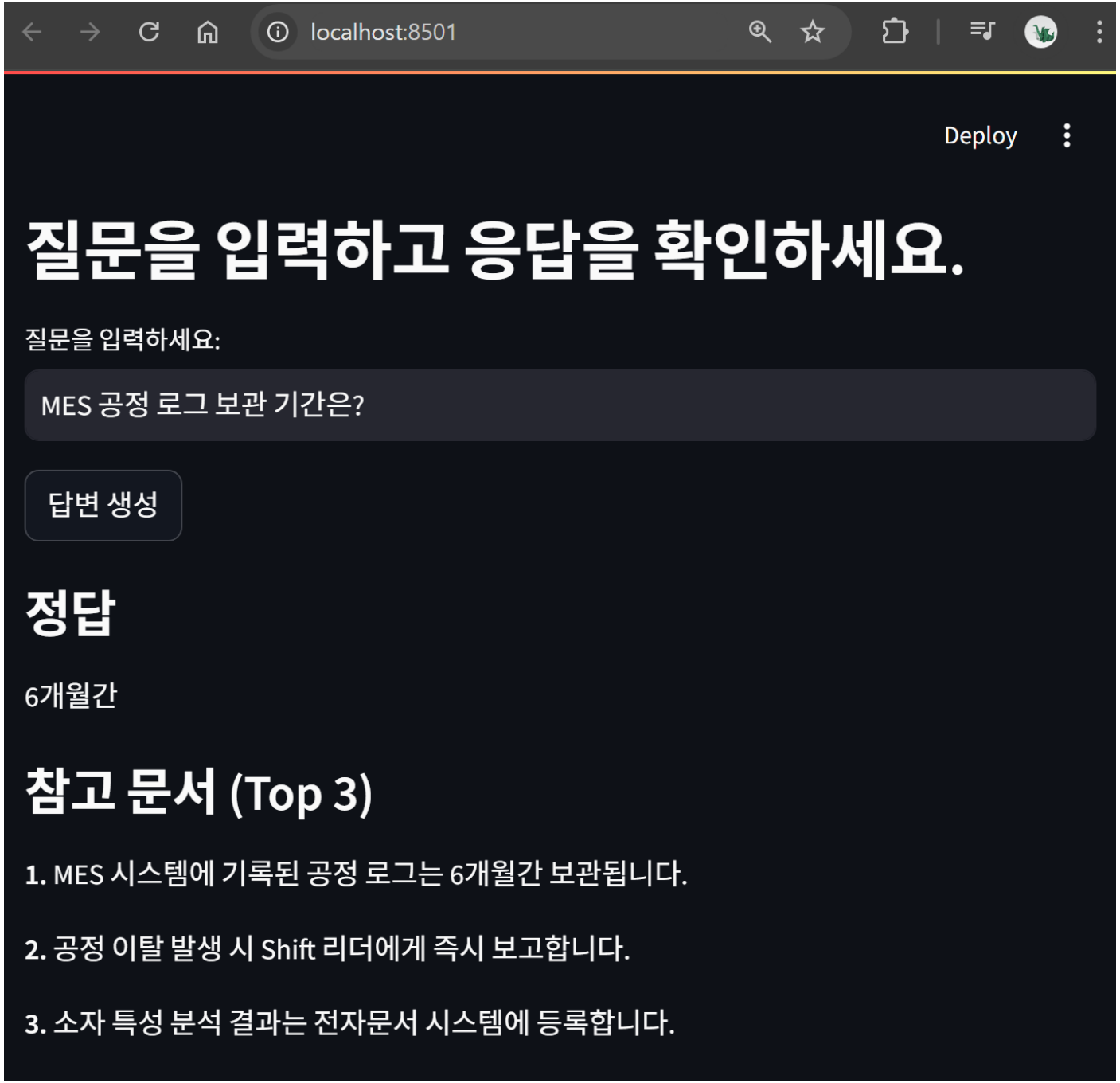

localhost:8501
Deploy
데이터 저장
질문하기
CSV 문서 업로드 및 ChromaDB 저장
CSV 파일 업로드
Drag and drop file here
Limit 200MB per file • CSV
Browse files
data.csv 2.4KB
Chroma에 저장
문서가 ChromaDB에 저장되었습니다.

Section
랭체인 기반 서비스 실습

결과 확인





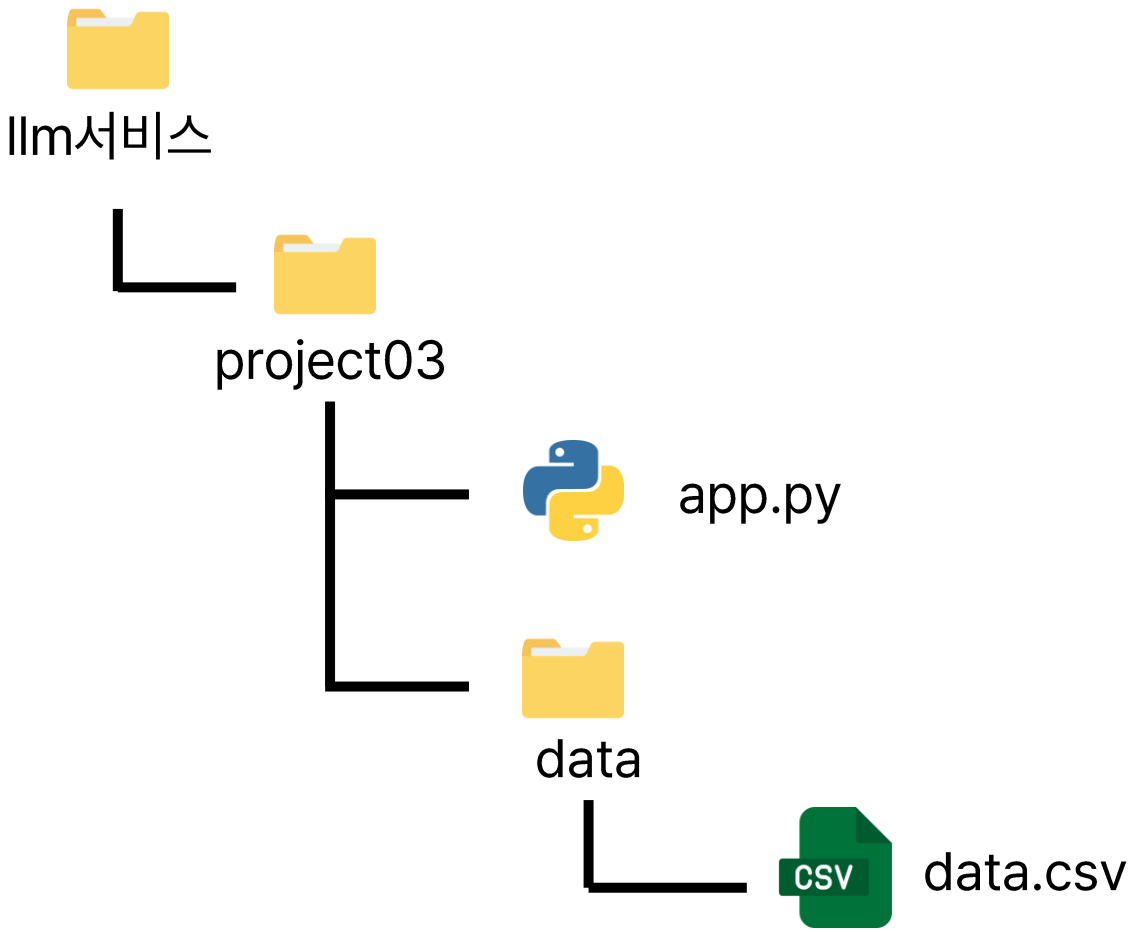


GEN AI 인텐시브 과정

Section 1. LLM 기반 챗봇 만들기

Section 1-3. 멀티턴 기반 서비스 실습

폴더 구성



	A	B	C	D	E	F
1	text					
2	Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.					
3	PM 장비의 필터는 주 1회 정기적으로 교체해야 합니다.					
4	웨이퍼 투입 전 챔버 내부의 온도 안정화가 필요합니다.					
5	불량률이 2%를 초과할 경우 원인 분석 보고서를 제출해야 합니다.					
6	클린룸 입장 전에는 반드시 정전기 방지복을 착용해야 합니다.					
7	포토 공정 시 PR 코팅 두께는 1.5μm 이상 유지해야 합니다.					

실습 내용

- 이번 실습에서는 streamlit 라이브러리를 활용한 멀티턴 대화 구현

- streamlit은 사용자 입력과 출력을 반복적으로 받는데 최적화되어 있어, 굳이 LangGraph의 상태 기반 워크플로우를 쓰지 않아도 대화 흐름 구현 가능

소스 코드

project03 > app.py > ...

```python
import pandas as pd
import streamlit as st
from langchain_chroma import Chroma
from langchain.schema import Document
from langchain_huggingface import HuggingFaceEmbeddings
from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline

# 1. 페이지 설정
st.set_page_config(page_title="멀티턴 RAG QA 시스템", layout="wide")
st.sidebar.header("사이드바 메뉴")
mode = st.sidebar.radio("모드를 선택하세요", ["데이터 저장", "질문하기"])

```

소스 코드

```python
# 2. 임베딩 및 QA 모델 준비
@st.cache_resource
def load_embedding_model():
    return HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")

@st.cache_resource
def load_qa_model():
    model_id = "monologg/koelectra-base-v3-finetuned-korquad"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    model = AutoModelForQuestionAnswering.from_pretrained(model_id)
    pipe = pipeline(
        "question-answering",
        model=model,
        tokenizer=tokenizer,
        device=-1,
        max_length=512,
        do_sample=False,
        temperature=0.1,
        truncation=True
    )
    return pipe

embedding_model = load_embedding_model()
text_gen = load_qa_model()
```

소스 코드

```python
# 3. 탭 1: 데이터 저장
if mode=="데이터 저장":
    st.header("CSV 문서 업로드 및 ChromaDB 저장")
    uploaded_file = st.file_uploader("CSV 파일 업로드", type="csv")

    if uploaded_file:
        df = pd.read_csv(uploaded_file, encoding='utf-8')
        if "text" not in df.columns:
            st.error("'text' 컬럼이 포함되어야 합니다.")
        else:
            docs = [Document(page_content=text) for text in df["text"].tolist()]
            if st.button("Chroma에 저장"):
                Chroma.from_documents(
                    documents=docs,
                    embedding=embedding_model,
                    persist_directory="./chromaDB"
                )
                st.success("문서가 ChromaDB에 저장되었습니다.")
```

소스 코드

```python
# 4. 탭 2: 질문 응답
elif mode=="질문하기":
    st.header("질문을 입력하고 대화를 이어가보세요!")

    # 벡터 DB 불러오기
    @st.cache_resource
    def get_retriever():
        res = Chroma( persist_directory="./chromaDB",
                      embedding_function=embedding_model
        ).as_retriever(search_kwargs={"k": 3})
        return res

    # 세션 상태 초기화
    if "messages" not in st.session_state:
        st.session_state.messages = []
    if "last_contexts" not in st.session_state:
        st.session_state.last_contexts = []

    # 기존 대화 내용 출력
    for msg in st.session_state.messages:
        with st.chat_message(msg["role"]):
            st.markdown(msg["content"])

    # 채팅 입력창
    user_input = st.chat_input("질문을 입력하세요:")

    if user_input:
        # 사용자 메시지 기록
        st.session_state.messages.append({"role": "user", "content": user_input})
        with st.chat_message("user"):
            st.markdown(user_input)
```

소스 코드

```python
    if user_input:
        # 사용자 메시지 기록
        st.session_state.messages.append({"role": "user", "content": user_input})
        with st.chat_message("user"):
            st.markdown(user_input)

        # 1. 참고 문서 요청 여부
        if "참고문서" in user_input or "참고 문서" in user_input:
            if st.session_state.last_contexts:
                ref_response = "### 참고 문서\n" + "\n".join(
                    f"{i+1}. {ctx}" for i, ctx in enumerate(st.session_state.last_contexts)
                )
            else:
                ref_response = "참고 문서가 없습니다. 먼저 질문을 해주세요."

            st.session_state.messages.append({"role": "assistant", "content": ref_response})
            with st.chat_message("assistant"):
                st.markdown(ref_response)
```

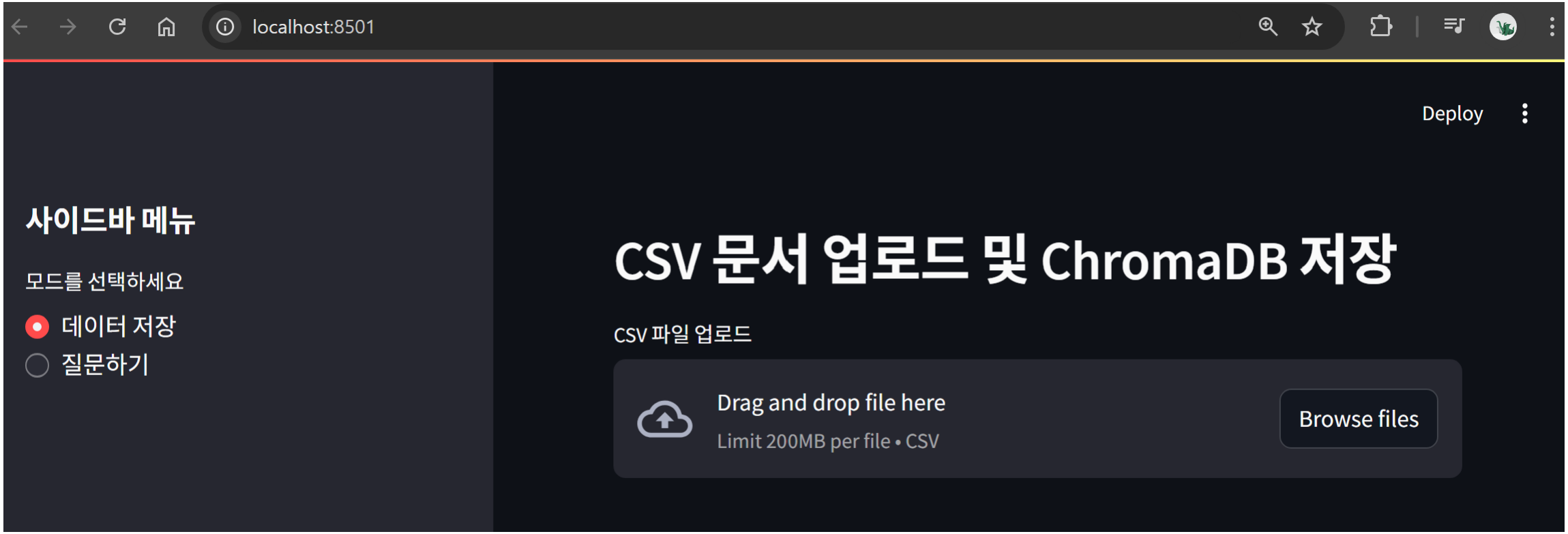
소스 코드

```
102
103         # 2. 일반 질문 처리
104         else:
105             # 문서 검색
106             retriever = get_retriever()
107             docs = retriever.invoke(user_input)
108             top_docs = docs[:3]
109             best_contexts = [doc.page_content for doc in top_docs]
110             st.session_state.last_contexts = best_contexts
111             combined_context = "\n".join(best_contexts)
112
113             # QA 실행
114             result = text_gen(question=user_input, context=combined_context)
115             answer = result["answer"]
116
117             # 챗봇 응답 기록 및 출력
118             bot_response = f"답변: {answer}"
119             st.session_state.messages.append({"role": "assistant", "content": bot_response})
120             with st.chat_message("assistant"):
121                 st.markdown(bot_response)
```

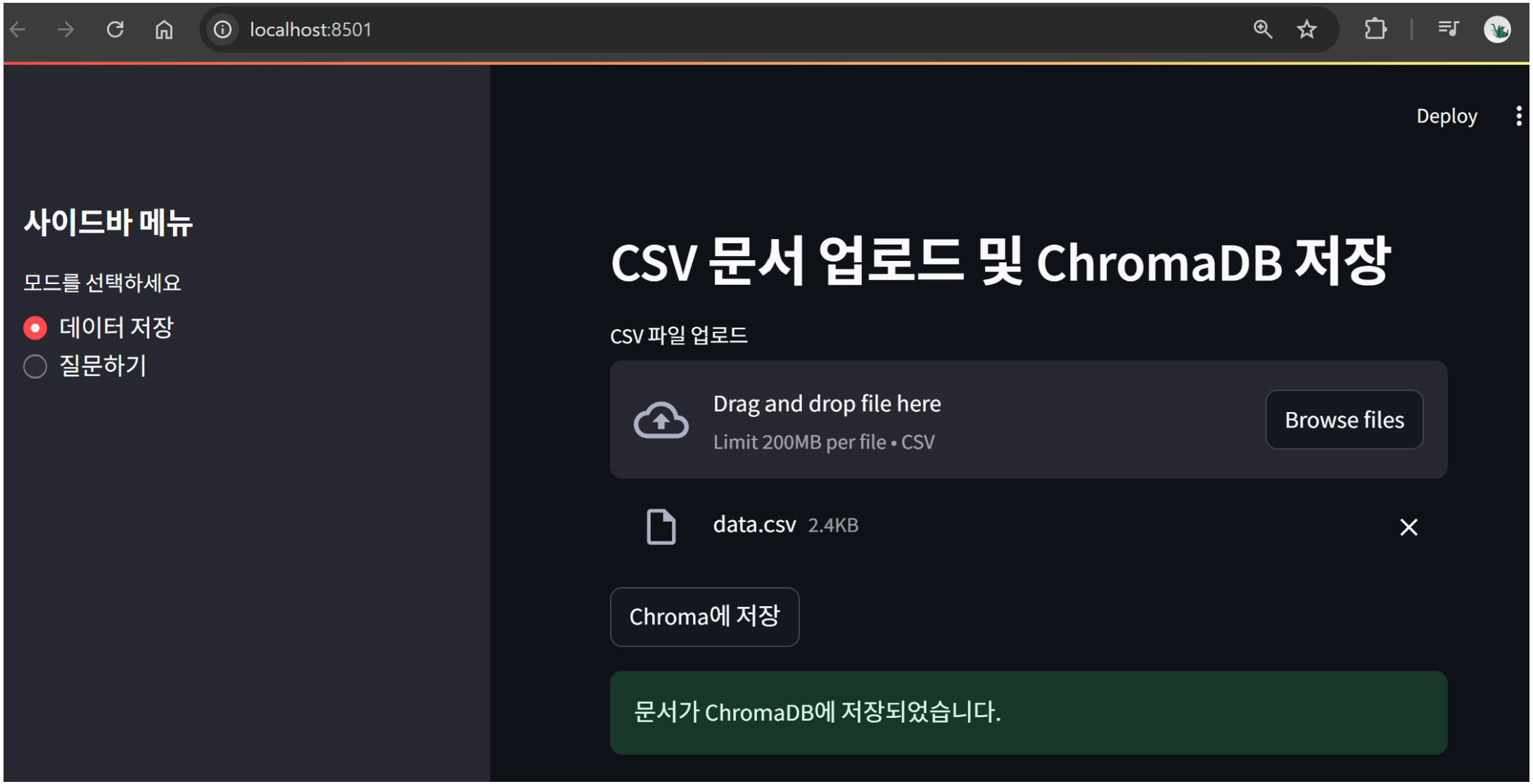
실행

```
stoic@WIN-77O2CJ32RBN:/c/Users/stoic/Documents/work/4_
 streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://10.36.204.41:8501
```

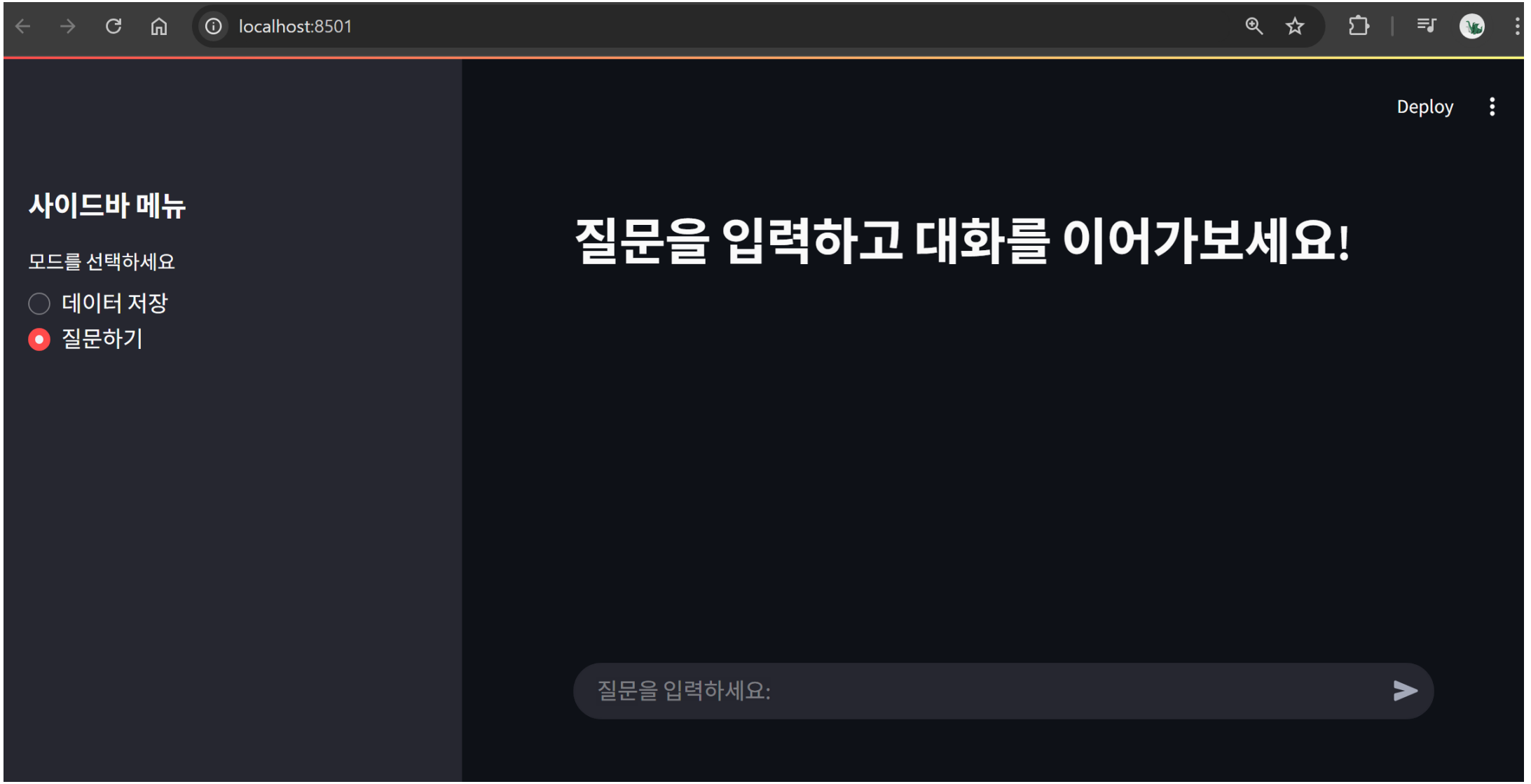
데이터 저장


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
CSV 문서 업로드 및 ChromaDB 저장
CSV 파일 업로드
Drag and drop file here
Limit 200MB per file • CSV
Browse files

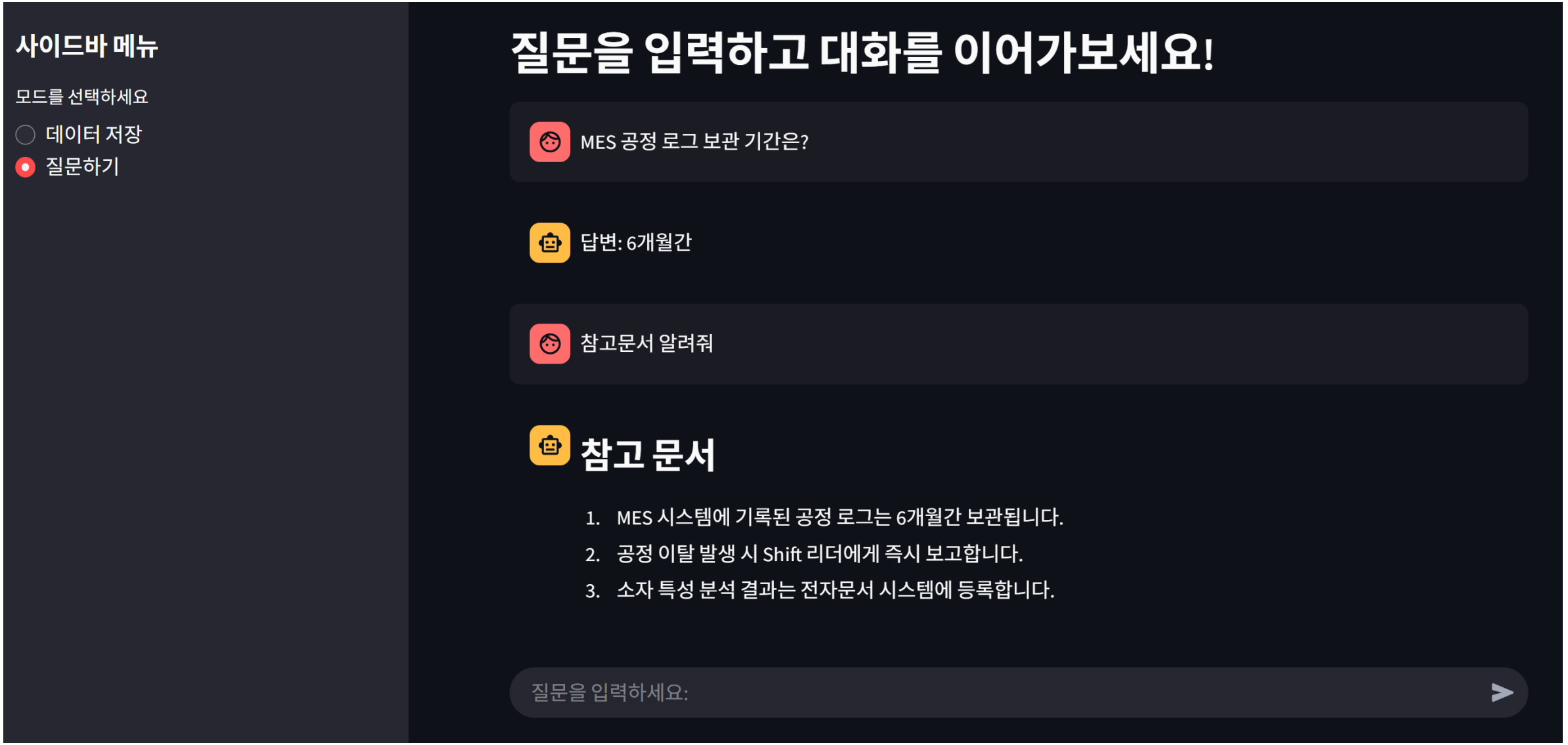
데이터 저장


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
CSV 문서 업로드 및 ChromaDB 저장
CSV 파일 업로드
Drag and drop file here
Limit 200MB per file • CSV
Browse files
data.csv 2.4KB
Chroma에 저장
문서가 ChromaDB에 저장되었습니다.

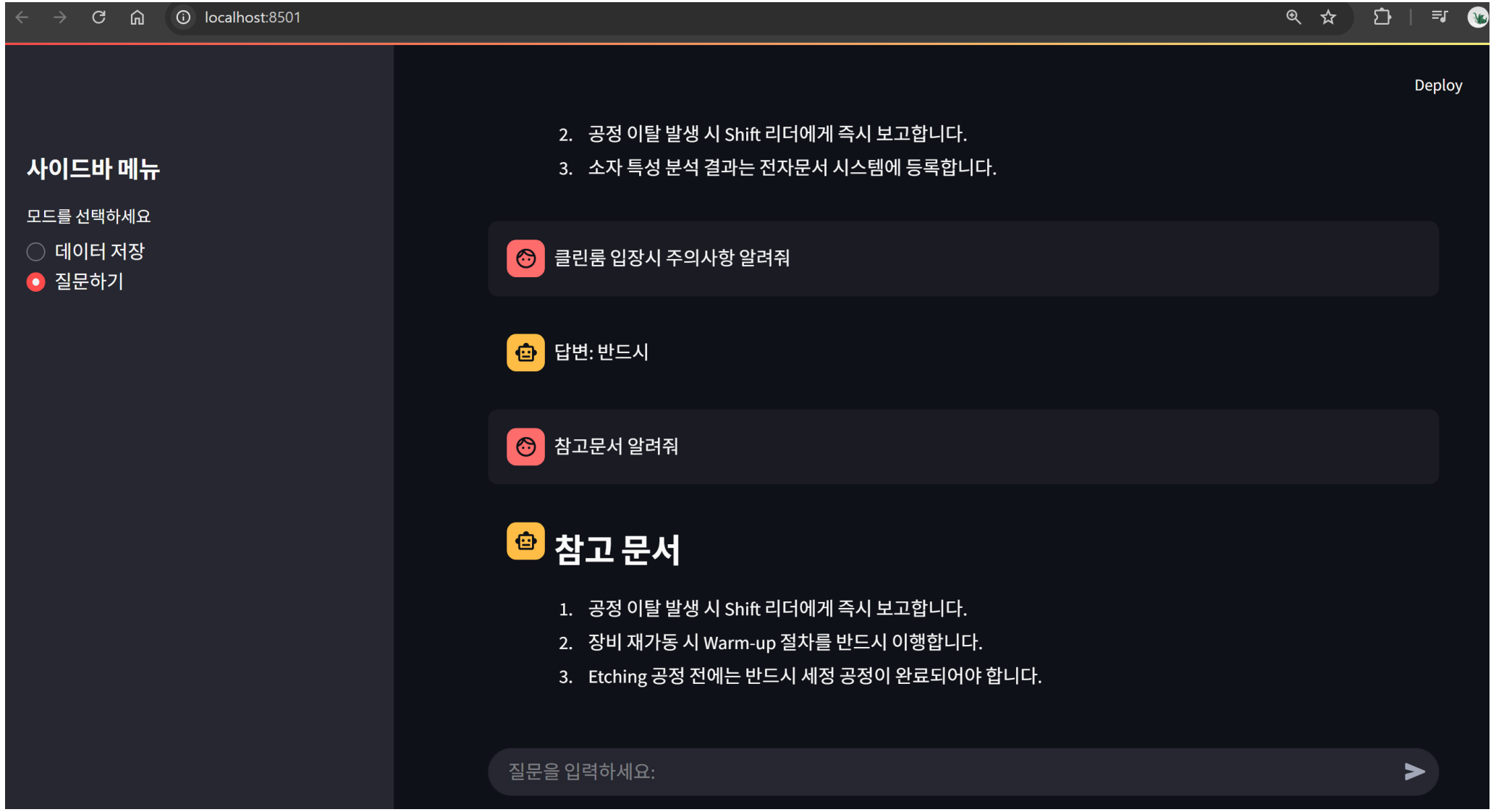
질문하기


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
질문을 입력하고 대화를 이어가보세요!
질문을 입력하세요:

질문하기


사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
질문을 입력하고 대화를 이어가보세요!
MES 공정 로그 보관 기간은?
답변: 6개월간
참고문서 알려줘
참고 문서
1. MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.
2. 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
3. 소자 특성 분석 결과는 전자문서 시스템에 등록합니다.
질문을 입력하세요:

질문하기

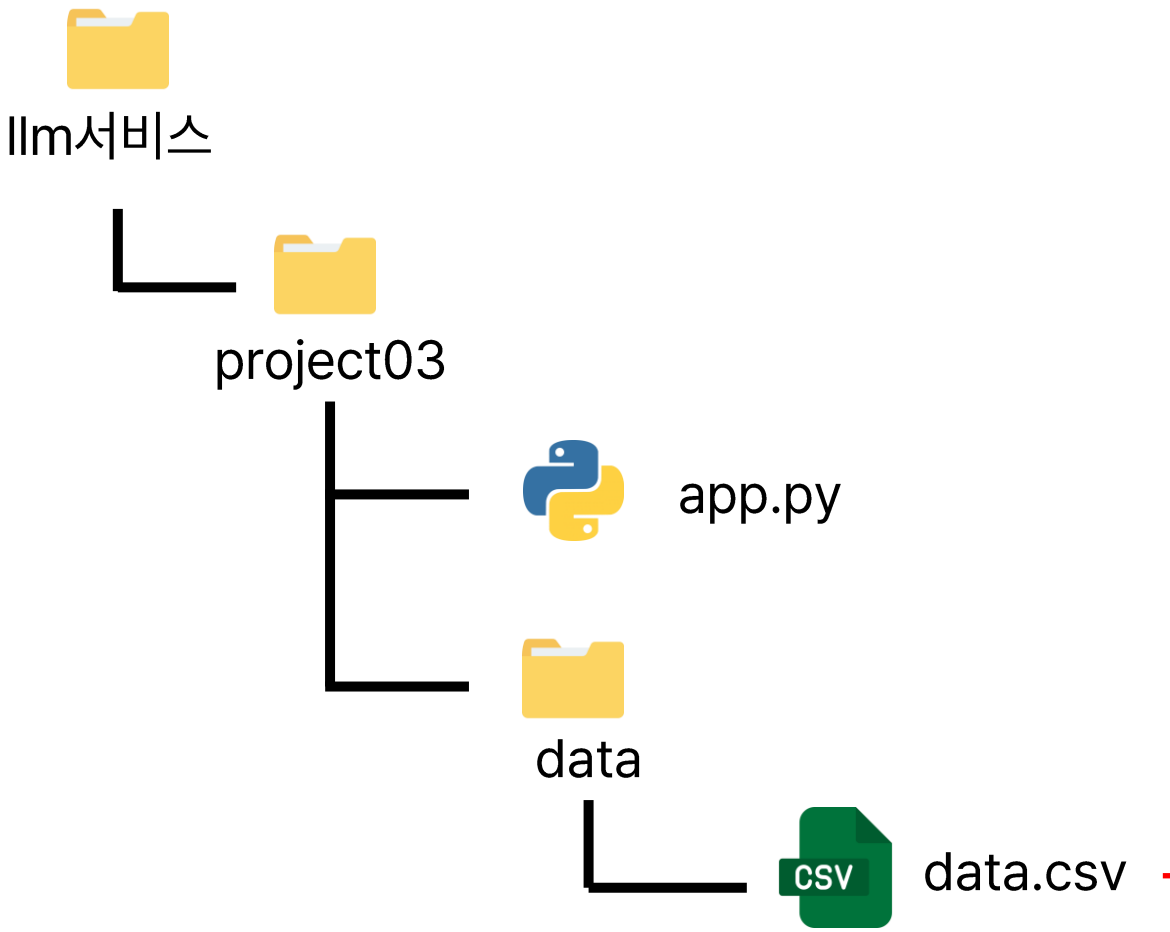

localhost:8501
Deploy
2. 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
3. 소자 특성 분석 결과는 전자문서 시스템에 등록합니다.
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
클린룸 입장시 주의사항 알려줘
답변: 반드시
참고문서 알려줘
참고 문서
1. 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
2. 장비 재가동 시 Warm-up 절차를 반드시 이행합니다.
3. Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.
질문을 입력하세요:

GEN AI 인텐시브 과정

Section 1. LLM 기반 챗봇 만들기

Section 1-4. 랭그래프 기반 서비스 실습

폴더 구성



	A	B	C	D	E	F
1	text					
2	Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.					
3	PM 장비의 필터는 주 1회 정기적으로 교체해야 합니다.					
4	웨이퍼 투입 전 챔버 내부의 온도 안정화가 필요합니다.					
5	불량률이 2%를 초과할 경우 원인 분석 보고서를 제출해야 합니다.					
6	클린룸 입장 전에는 반드시 정전기 방지복을 착용해야 합니다.					
7	포토 공정 시 PR 코팅 두께는 1.5μm 이상 유지해야 합니다.					

소스 코드

```python
import pandas as pd
import streamlit as st
from langchain.schema import Document
from langchain_chroma import Chroma
from langchain_huggingface import HuggingFaceEmbeddings
from langgraph.graph import StateGraph, END
from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline

# 페이지 설정
st.set_page_config(page_title="LangGraph를 활용한 멀티턴 RAG QA 시스템", layout="wide")
st.sidebar.header("사이드바 메뉴")
mode = st.sidebar.radio("모드를 선택하세요", ["데이터 저장", "질문하기"])

# 모델 로딩
@st.cache_resource
def load_embedding_model():
    return HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
```

소스 코드

```
@st.cache_resource
def load_qa_model():
    model_id = "monologg/koelectra-base-v3-finetuned-korquad"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    model = AutoModelForQuestionAnswering.from_pretrained(model_id)
    pipe = pipeline(
        "question-answering",
        model=model,
        tokenizer=tokenizer,
        device=-1,
        max_length=512,
        do_sample=False,
        temperature=0.1,
        truncation=True
    )
    return pipe

embedding_model = load_embedding_model()
text_gen = load_qa_model()
```

Section
랭그래프기반서비스실습

소스 코드

랭그래프 노드 작성 시에는 @st.cache_resource 사용 안함

왜냐하면 랭그래프에서는 입력이 바뀌면 새롭게 실행되어야 하므로

streamlit 환경에서는 사용자에게 되묻기 어려우므로 단발성 처리

```python
def build_graph(retriever):
    def ask_question(state):
        question = state["question"]
        docs = retriever.invoke(question)
        top_docs = docs[:3]
        best_contexts = [doc.page_content for doc in top_docs]
        combined_context = "\n".join(best_contexts)
        result = text_gen(question=question, context=combined_context)
        return {
            **state,
            "answer": result["answer"],
            "context": best_contexts
        }

    def get_answer(state):
        return state

    def ask_reference(state):
        state["show_reference"] = "참고문서" in state.get("question", "")
        return state

    def get_reference(state):
        return state

    def ask_continue(state):
        state["continue"] = False
        return state

    def should_show_reference(state):
        return "get_reference" if state.get("show_reference") else "ask_continue"

    def should_continue(state):
        return "ask_question" if state.get("continue") else END

    graph = StateGraph(dict)
    graph.add_node("ask_question", ask_question)
    graph.add_node("get_answer", get_answer)
    graph.add_node("ask_reference", ask_reference)
```

Section
랭그래프기반서비스실습

소스 코드

```
66      def ask_continue(state):
67          state["continue"] = False  # 단발성 처리
68          return state
69
70      def should_show_reference(state):
71          return "get_reference" if state.get("show_reference") else "ask_continue"
72
73      def should_continue(state):
74          return "ask_question" if state.get("continue") else END
75
76      graph = StateGraph(dict)
77      graph.add_node("ask_question", ask_question)
78      graph.add_node("get_answer", get_answer)
79      graph.add_node("ask_reference", ask_reference)
80      graph.add_node("get_reference", get_reference)
81      graph.add_node("ask_continue", ask_continue)
82
83      graph.set_entry_point("ask_question")
84      graph.add_edge("ask_question", "get_answer")
85      graph.add_edge("get_answer", "ask_reference")
86      graph.add_conditional_edges("ask_reference", should_show_reference, {
87          "get_reference": "get_reference",
88          "ask_continue": "ask_continue"
89      })
90      graph.add_edge("get_reference", "ask_continue")
91      graph.add_conditional_edges("ask_continue", should_continue, {
92          "ask_question": "ask_question",
93          END: END
94      })
95
96      return graph.compile()
```

소스 코드

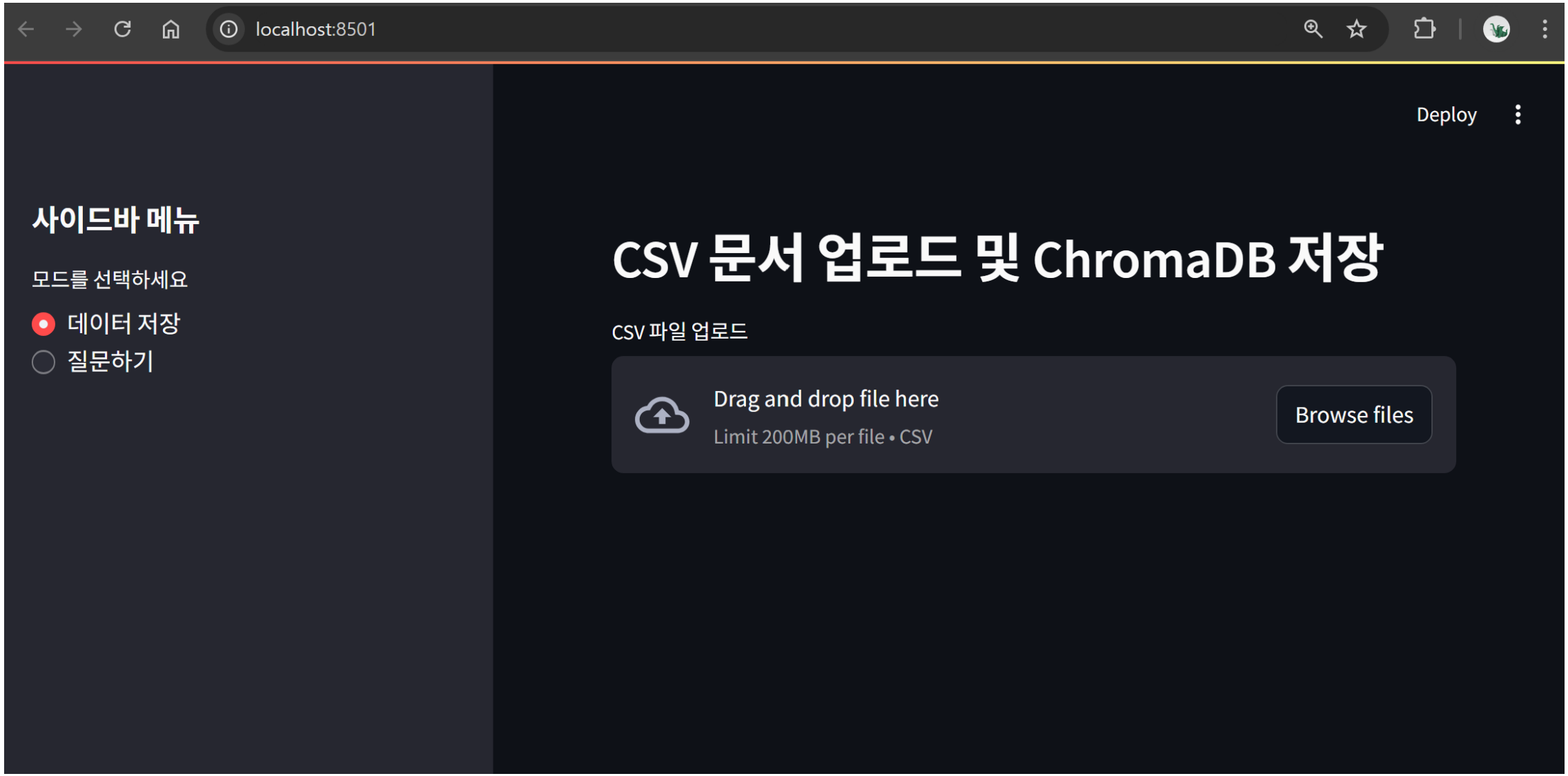
```python
# 탭 1: 데이터 저장
if mode == "데이터 저장":
    st.header("CSV 문서 업로드 및 ChromaDB 저장")
    uploaded_file = st.file_uploader("CSV 파일 업로드", type="csv")

    if uploaded_file:
        df = pd.read_csv(uploaded_file, encoding='utf-8')
        if "text" not in df.columns:
            st.error("'text' 컬럼이 포함되어야 합니다.")
        else:
            docs = [Document(page_content=text) for text in df["text"].tolist()]
            if st.button("Chroma에 저장"):
                Chroma.from_documents(
                    documents=docs,
                    embedding=embedding_model,
                    persist_directory="./chromaDB"
                )
                st.success("문서가 ChromaDB에 저장되었습니다.")
```

Section
랭그래프기반서비스실습

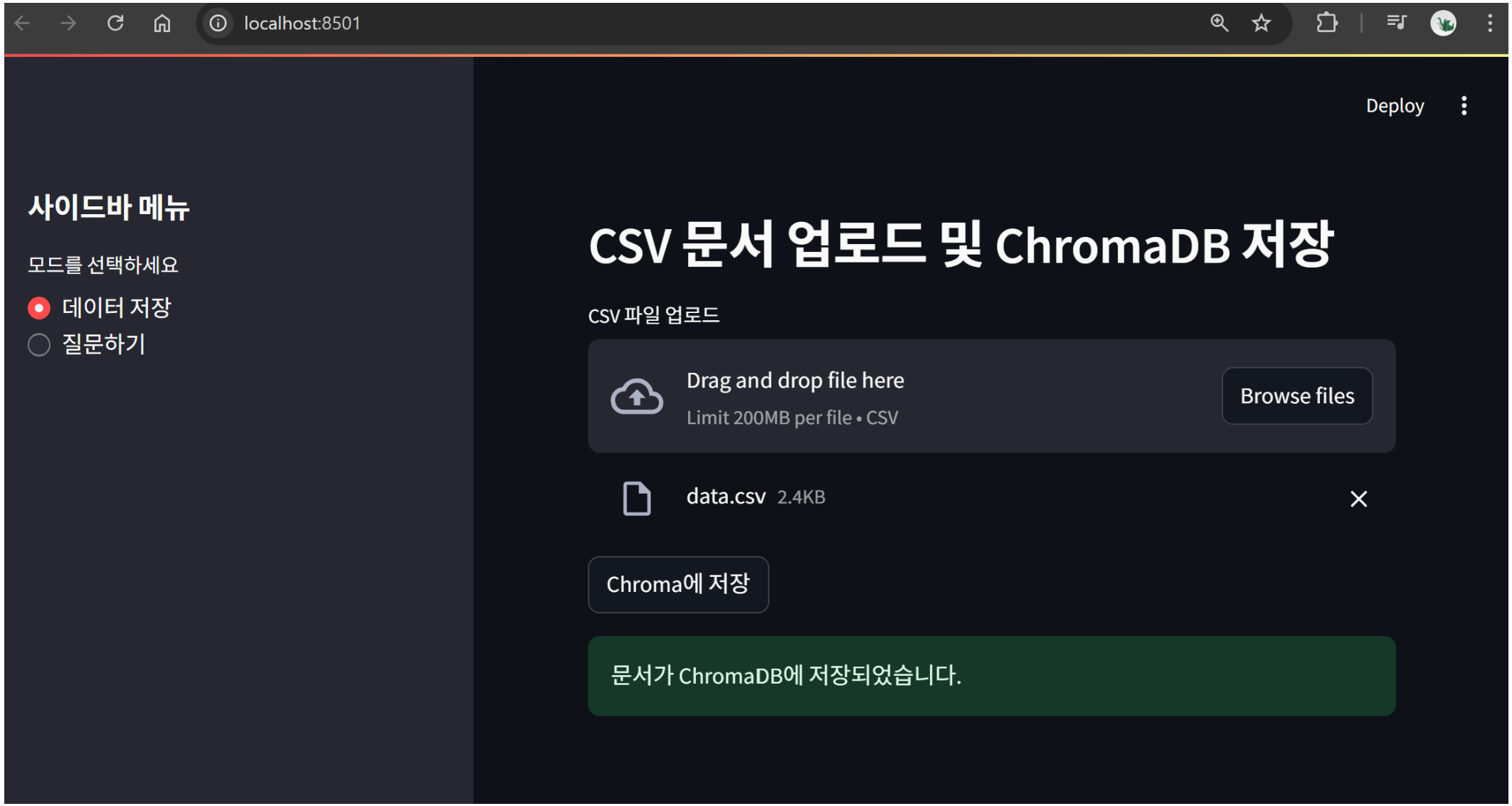
소스 코드

```python
# 탭 2: 질의 응답
elif mode == "질문하기":
    st.title("LangGraph 기반 RAG QA 시스템")

    @st.cache_resource
    def get_retriever():
        return Chroma(
            persist_directory="./chromaDB",
            embedding_function=embedding_model
        ).as_retriever(search_kwargs={"k": 3})

    retriever = get_retriever()
    app = build_graph(retriever)

    if "messages" not in st.session_state:
        st.session_state.messages = []

    for msg in st.session_state.messages:
        with st.chat_message(msg["role"]):
            st.markdown(msg["content"])

    user_input = st.chat_input("질문을 입력하세요:")

    if user_input:
        st.session_state.messages.append({"role": "user", "content": user_input})
        with st.chat_message("user"):
            st.markdown(user_input)

        result = app.invoke({"question": user_input})

        response = f"**답변:** {result['answer']}"
        st.session_state.messages.append({"role": "assistant", "content": response})

        with st.chat_message("assistant"):
            st.markdown(response)
            with st.expander("참고 문서 보기"):
                for i, ctx in enumerate(result["context"], 1):
                    st.markdown(f"{i}. {ctx}")
```

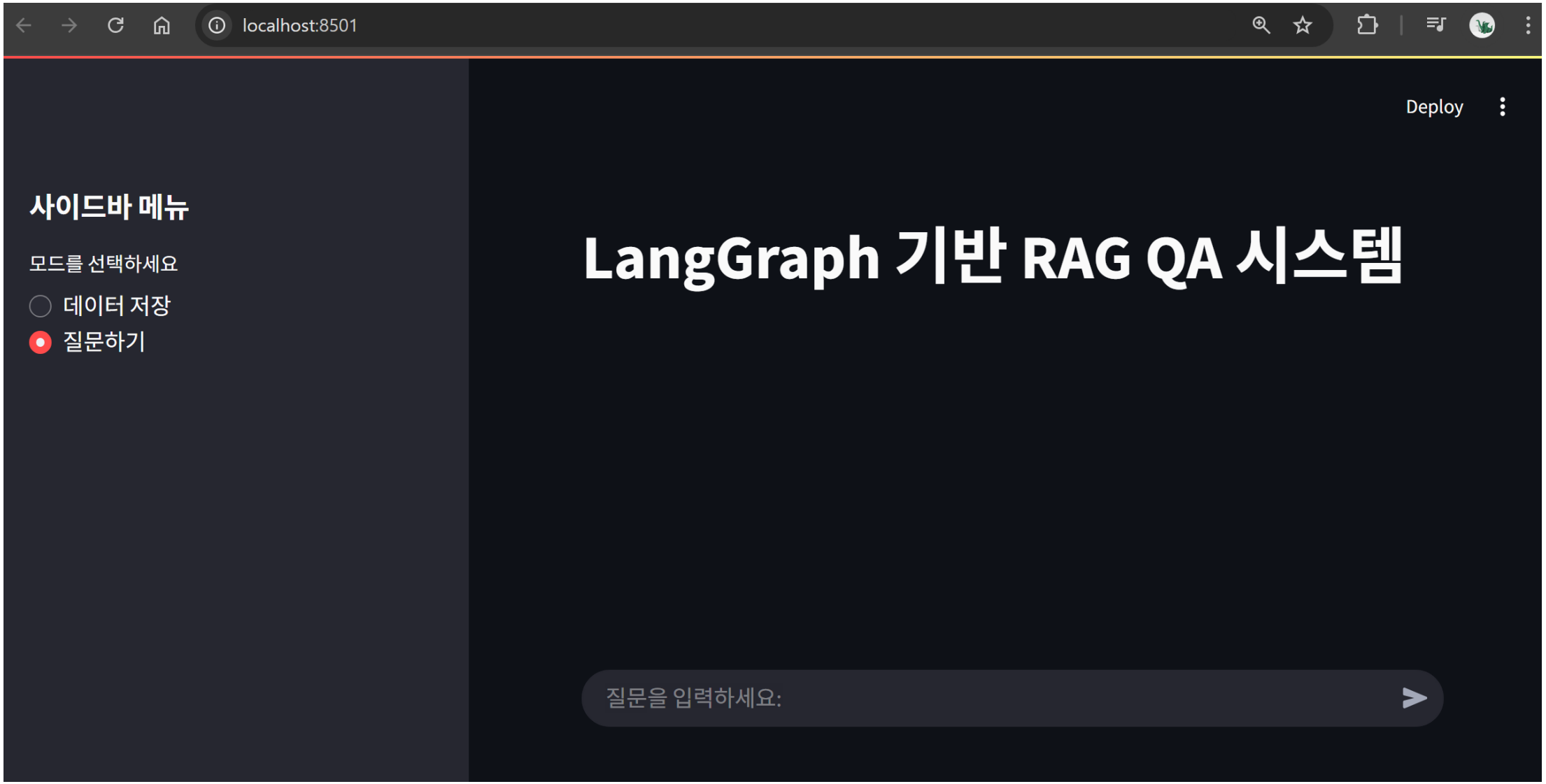
결과 화면


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
CSV 문서 업로드 및 ChromaDB 저장
CSV 파일 업로드
Drag and drop file here
Limit 200MB per file • CSV
Browse files

결과 화면


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
CSV 문서 업로드 및 ChromaDB 저장
CSV 파일 업로드
Drag and drop file here
Limit 200MB per file • CSV
Browse files
data.csv 2.4KB
Chroma에 저장
문서가 ChromaDB에 저장되었습니다.

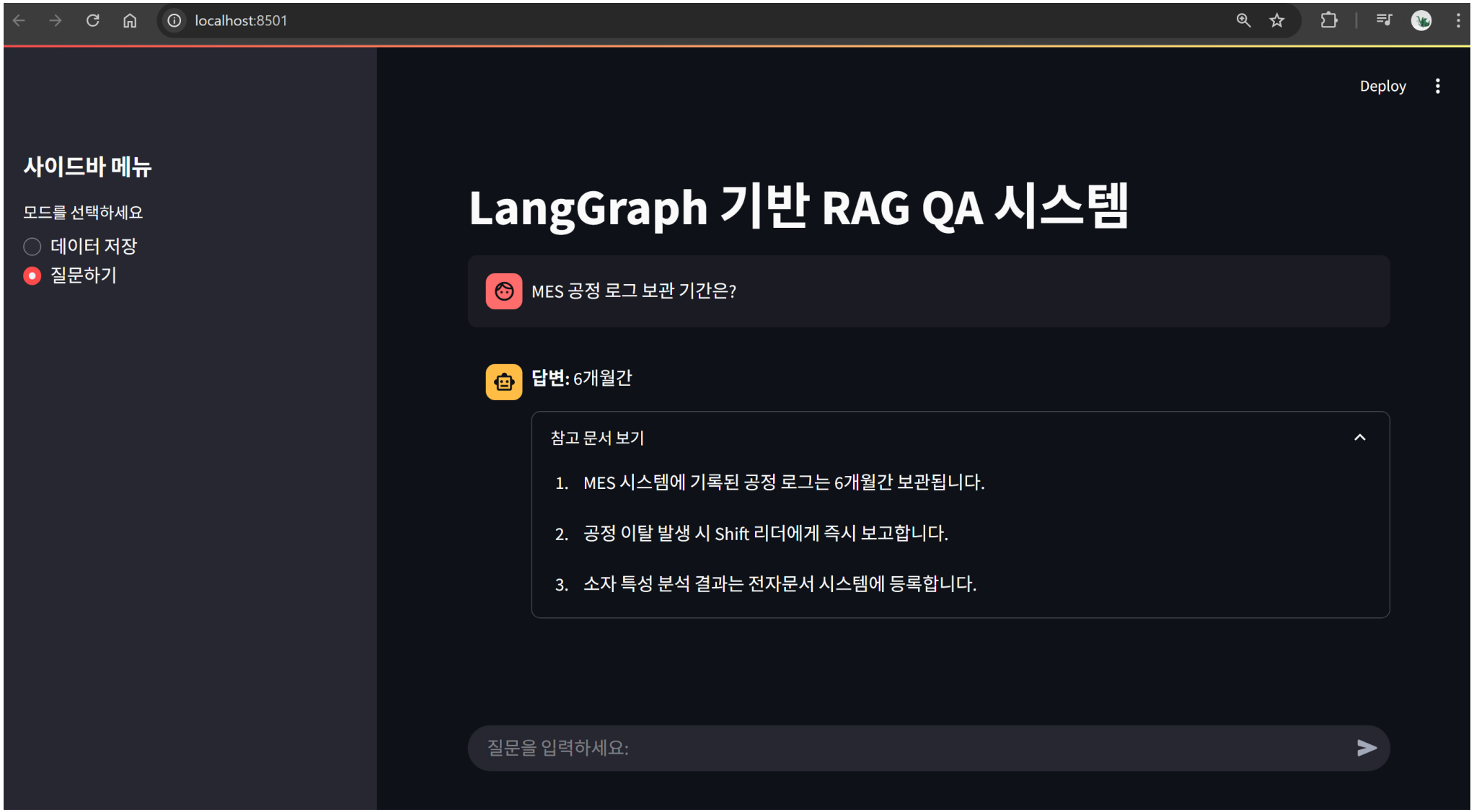
결과 화면


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
LangGraph 기반 RAG QA 시스템
질문을 입력하세요:

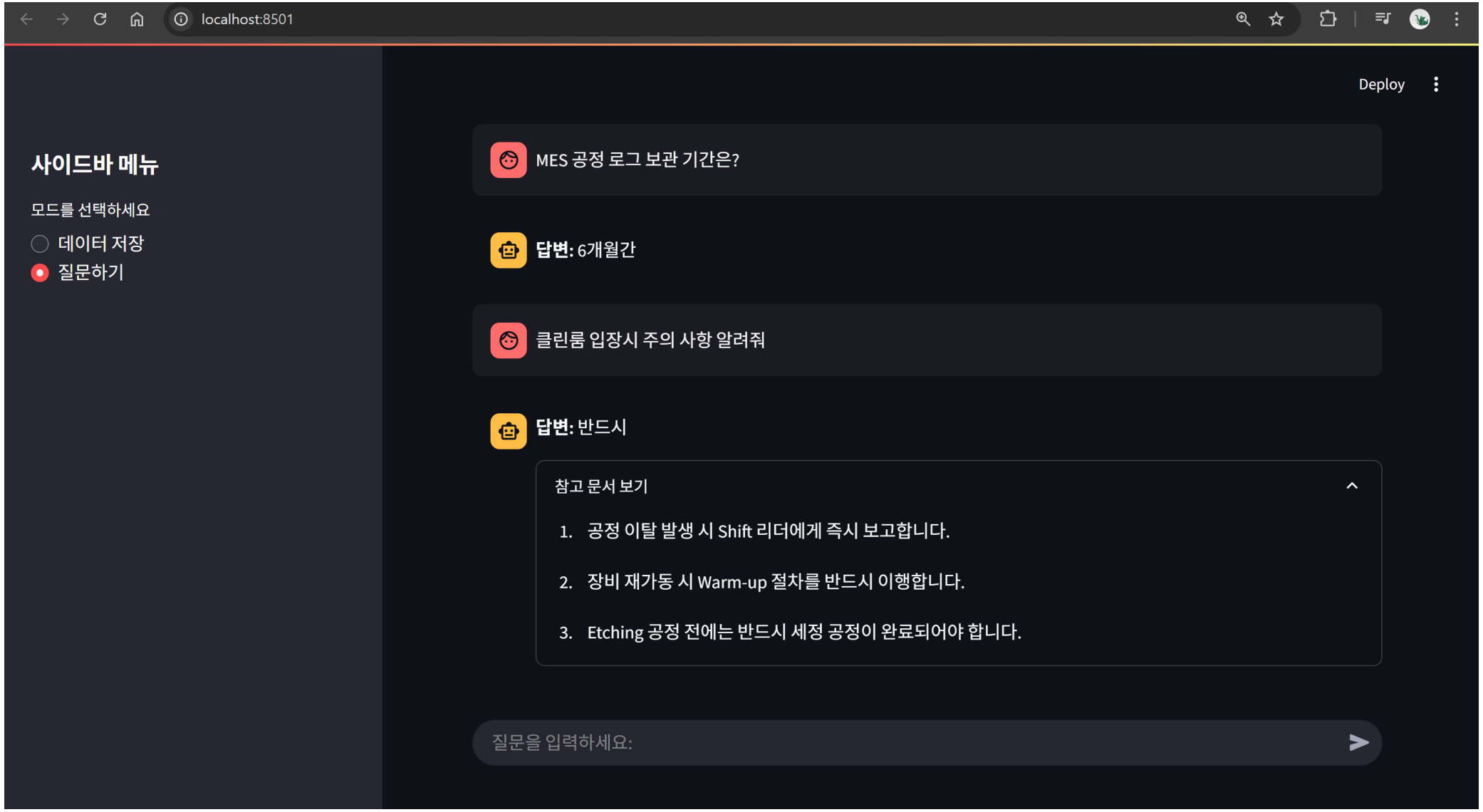
결과 화면


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
LangGraph 기반 RAG QA 시스템
MES 공정 로그 보관 기간은?
답변: 6개월간
참고 문서 보기

결과 화면


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
LangGraph 기반 RAG QA 시스템
MES 공정 로그 보관 기간은?
답변: 6개월간
참고 문서 보기
1. MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.
2. 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
3. 소자 특성 분석 결과는 전자문서 시스템에 등록합니다.
질문을 입력하세요:

결과 화면


localhost:8501
Deploy
사이드바 메뉴
모드를 선택하세요
데이터 저장
질문하기
MES 공정 로그 보관 기간은?
답변: 6개월간
클린룸 입장시 주의 사항 알려줘
답변: 반드시
참고 문서 보기
1. 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
2. 장비 재가동 시 Warm-up 절차를 반드시 이행합니다.
3. Etching 공정 전에는 반드시 세정 공정이 완료되어야 합니다.
질문을 입력하세요:

감사합니다.

Q & A