

GEN AI 인텐시브 과정

강사장철원

Section 0

코스소개

DAY1

LLM
Basic
Concept

DAY2

Transformers
paper
review

DAY3

Transformers
LangChain
LangGraph

DAY4

DAY5

DAY6

LLM
service
develop

DAY7

Final Project

DAY8

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

GEN AI 인텐시브 과정

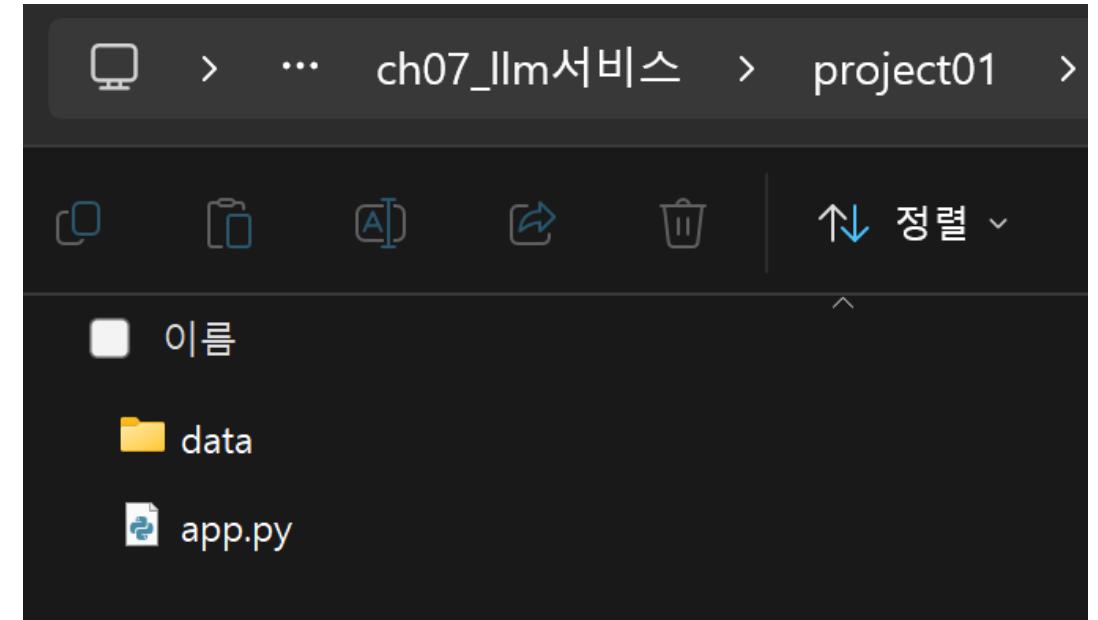
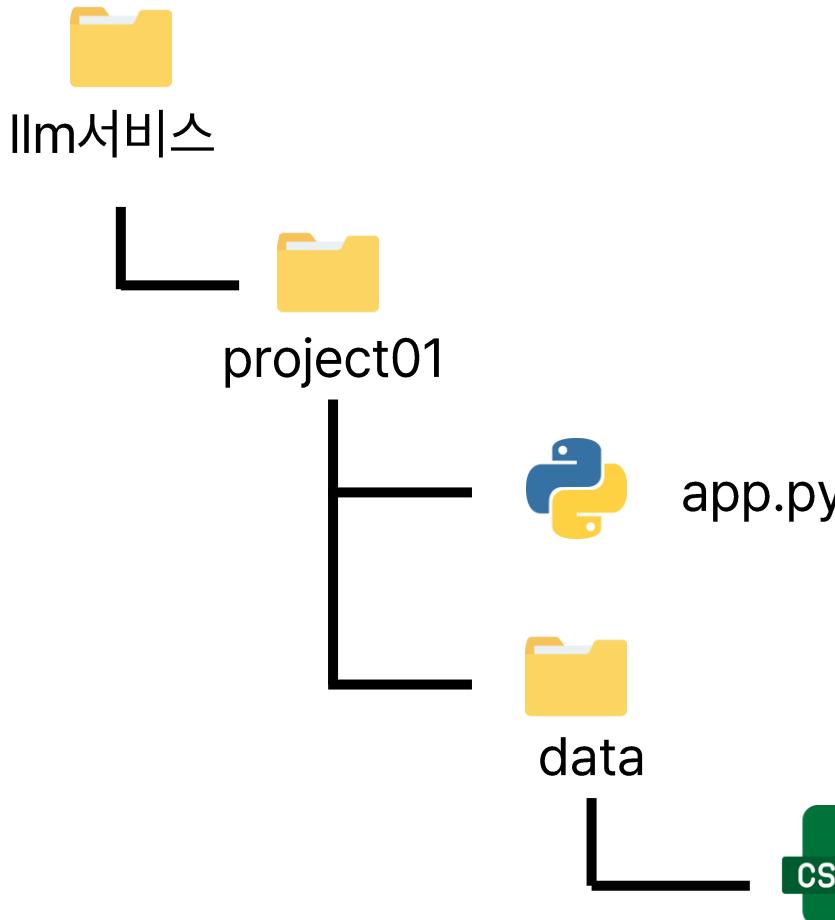
Section 1. LLM 기반 챗봇 만들기

Section 1-1. 트랜스포머 기반 서비스 실습

Section

트랜스포머기반서비스실습

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.					

Section

트랜스포머기반서비스실습

app.py

```
project01 > ⚙️ app.py > ⚭ load_qa_model
1 import torch
2 import chromadb
3 import pandas as pd
4 import streamlit as st
5 from sentence_transformers import SentenceTransformer
6 from transformers import AutoTokenizer, AutoModelForQuestionAnswering
7
8 # ChromaDB 설정(로컬 테스트)
9 client = chromadb.PersistentClient(path=".chromaDB")
10 collection = client.get_or_create_collection("data01")
11
```

} 실습에 필요한 라이브러리

} 로컬 테스트 사용

Section

트랜스포머기반서비스실습

app.py

```
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 → ? 다음 페이지에서 설명  
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 업로드하고 chromaDB 저장
"무엇이든 물어보세요": 사용자 질문에 대한 답변 생성

문장 임베딩 모델

QA 모델

Section

트랜스포머기반서비스실습

@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()
```

```

#-----#
# 템 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

```
#-----  
# 텁 2: 질문 답변 (QA 시스템)  
#-----  
with tab2:  
    st.header("무엇이든 물어보세요")  
  
    # 사용자 질문 입력  
    question = st.text_input("질문을 입력하세요:") → 질문 입력  
  
    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()] → 질문 임베딩  
  
            # 2. ChromaDB에서 유사한 문장 검색  
            result = collection.query(query_embeddings=query_ebd, n_results=1) → ChromaDB에서 질문과 가장 유사한 문장 검색  
  
            if result['metadatas']:  
                best_context = result['metadatas'][0][0]['text'] → 질문과 가장 유사한 문장  
  
                # 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) } 질문에 대한 답변 생성  
  
                # 4. 결과 출력  
                st.subheader("검색된 문서 내용")  
                st.write(f"**{best_context}**")  
  
                st.subheader("답변")  
                st.write(f"**{answer}**") } 결과 출력  
  
            else:  
                st.warning("유사한 문장을 찾을 수 없습니다.")
```

코드 실행

문제 3 출력 디버그 콘솔 터미널 포트

```
stoic@WIN-7702CJ32RBN 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>



Section

트랜스포머기반서비스실습

데이터 저장 탭 확인

The screenshot shows a web browser window with the URL `localhost:8501` in the address bar. The page has a dark theme with a navigation bar at the top featuring icons for back, forward, and search. On the right side of the header is a 'Deploy' button. Below the header, there is a horizontal menu with the first item '데이터 저장' highlighted in red, followed by '무엇이든 물어보세요'. The main content area is titled 'CSV 파일 업로드 및 저장' and contains a dark blue input field with the placeholder 'CSV 파일을 업로드 하세요'. To the left of the input field is a cloud icon with an upward arrow and the text 'Drag and drop file here'. Below this, it says 'Limit 200MB per file • CSV'. To the right of the input field is a 'Browse files' button.

Section

트랜스포머기반서비스실습

'무엇이든 물어보세요' 탭 확인

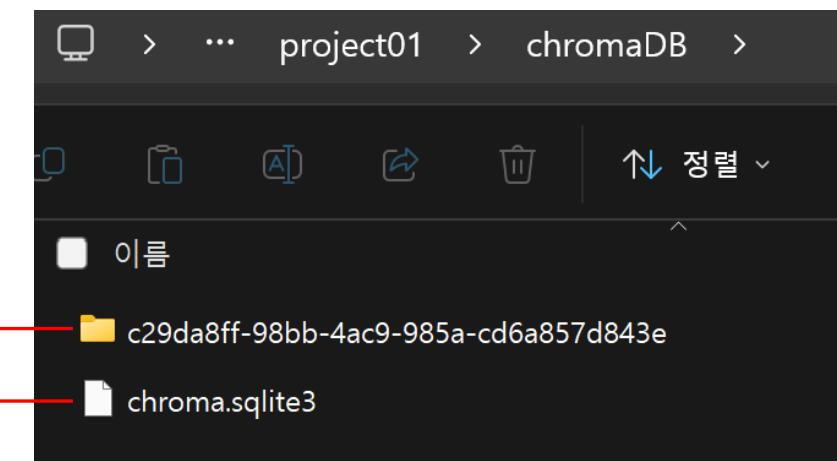
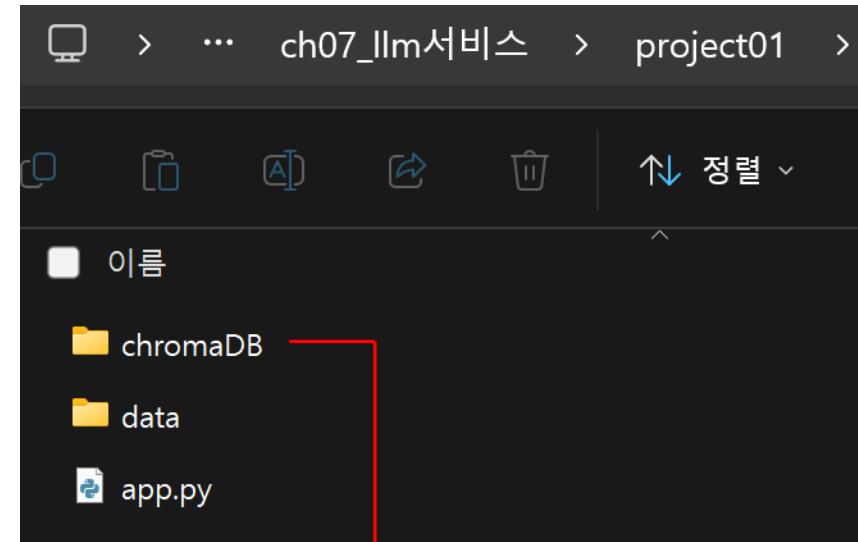
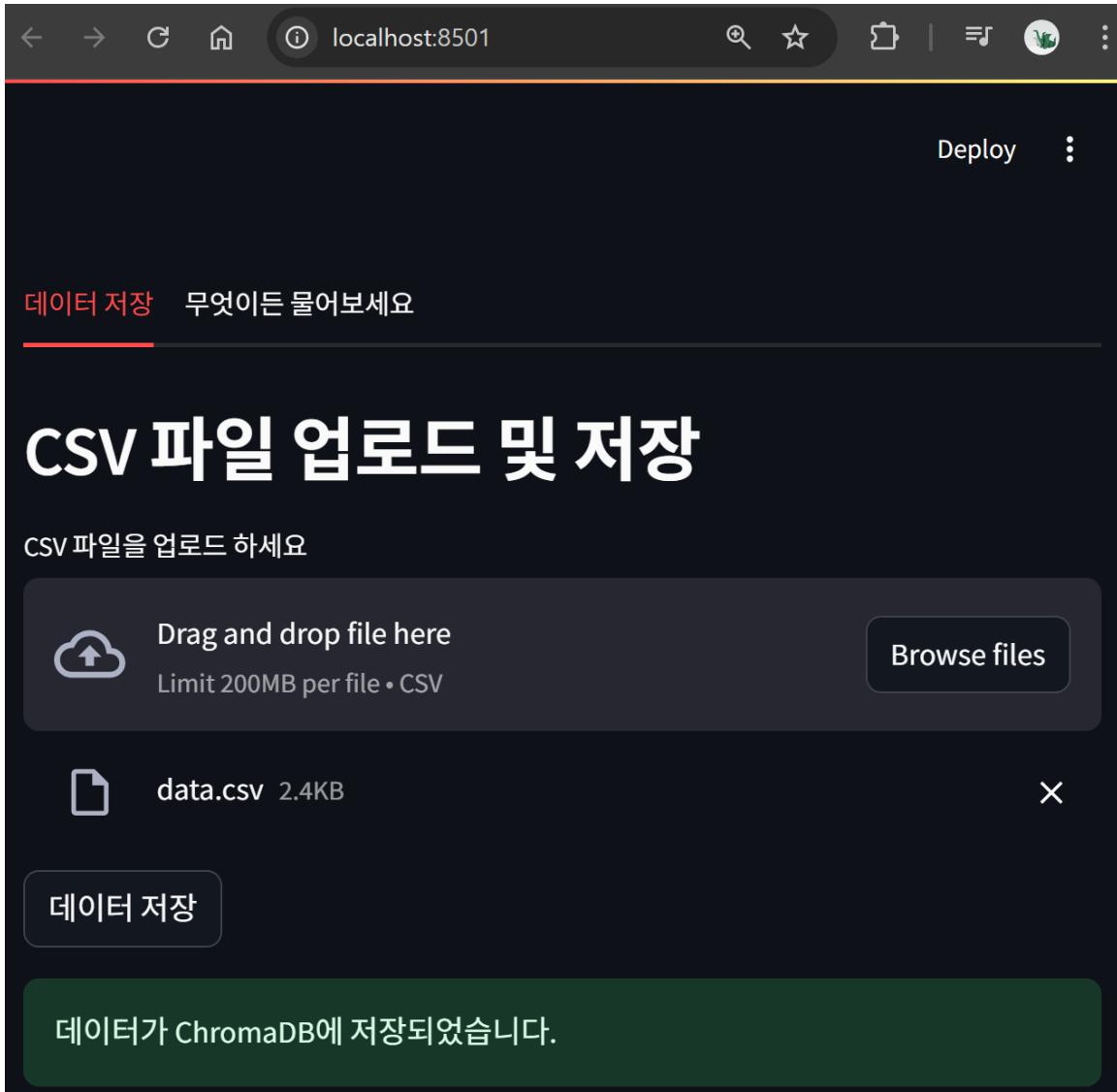
The screenshot shows a web browser window with the URL `localhost:8501` in the address bar. The page has a dark theme with a light gray header. In the header, there are icons for refresh, home, and search, followed by the URL. On the right side of the header are a star icon, a square icon, and a gear icon. Below the header, the text "Deploy" is visible.

On the left side of the main content area, there are two tabs: "데이터 저장" and "무엇이든 물어보세요". The "무엇이든 물어보세요" tab is active, indicated by a red underline. The main content area has a large title "무엇이든 물어보세요" and a text input field with the placeholder "질문을 입력하세요:". At the bottom left of the input field is a button labeled "분석 시작".

Section

트랜스포머기반서비스실습

csv 파일 업로드 및 저장



'무엇이든 물어보세요' 탭 확인

The screenshot shows a web browser window with the URL `localhost:8501` in the address bar. The page has a dark theme with red highlights. At the top, there are tabs for '데이터 저장' and '무엇이든 물어보세요', with the latter being active. On the right side of the header, there are buttons for 'Deploy' and three vertical dots. The main content area features a large heading '무엇이든 물어보세요' and a sub-instruction '질문을 입력하세요:'. Below this is a text input field containing the question 'MES 공정 로그 보관 기간은?'. A button labeled '분석 시작' is positioned below the input field. Further down, the section '검색된 문서 내용' contains the text 'MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.' Underneath this, the section '답변' contains the text '6개월'.

데이터 저장 무엇이든 물어보세요

무엇이든 물어보세요

질문을 입력하세요:

MES 공정 로그 보관 기간은?

분석 시작

검색된 문서 내용

MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.

답변

6개월

GEN AI 인텐시브 과정

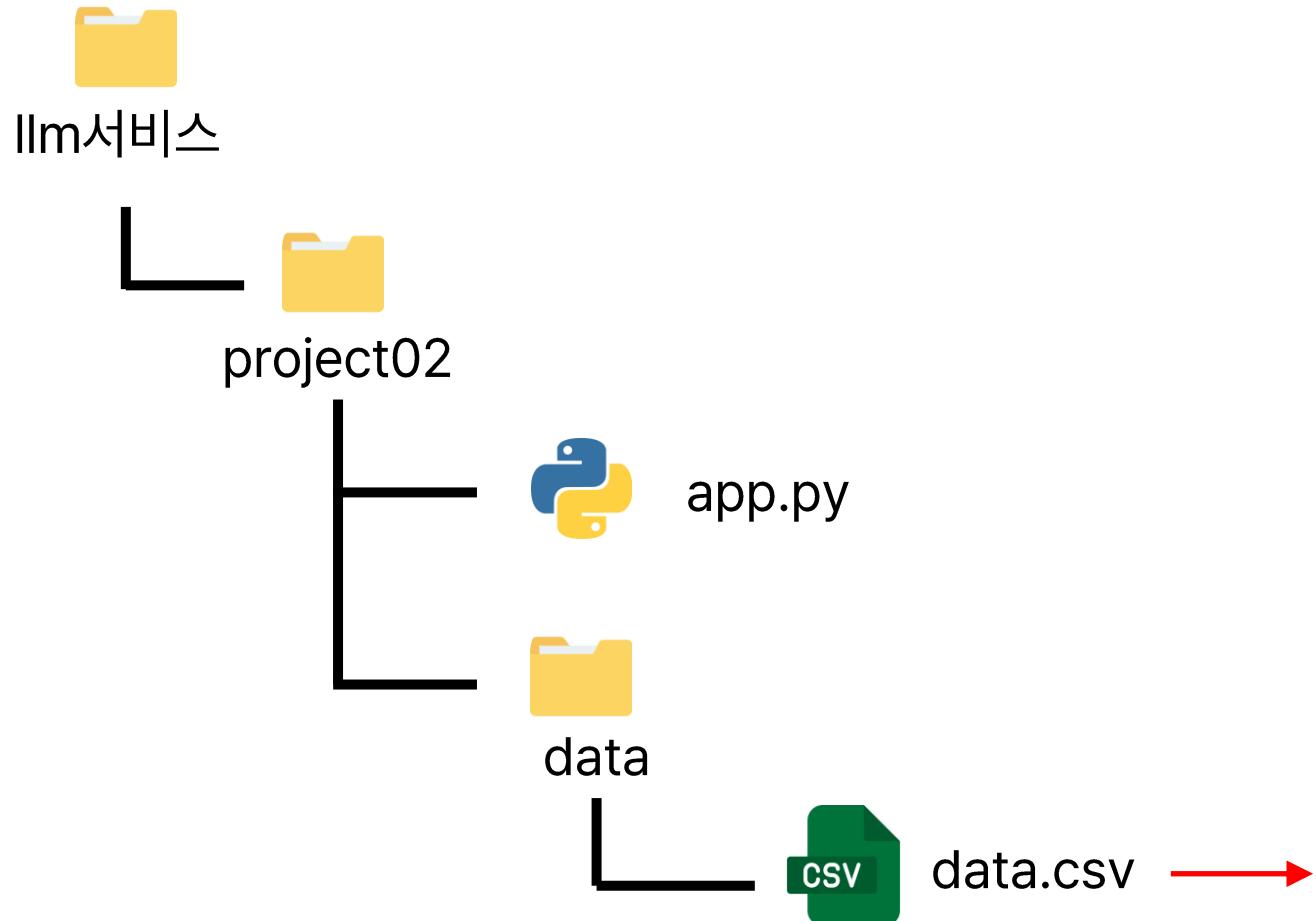
Section 1. LLM 기반 챗봇 만들기

Section 1-2. 랭체인 기반 서비스 실습

Section

랭체인기반서비스실습

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.					

Section

랭체인기반서비스실습

소스 코드

app.py

X

project02 > app.py > load_models

```
1 import streamlit as st
2 import pandas as pd
3 from langchain_chroma import Chroma
4 from langchain.schema import Document
5 from langchain_huggingface import HuggingFaceEmbeddings
6 from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline
7
8 # Streamlit UI 설정
9 st.set_page_config(page_title="RAG QA 시스템", layout='wide')
10 tab1, tab2 = st.tabs(["데이터 저장", "질문하기"])
```

Section

랭체인기반서비스실습

소스 코드

```
project02 > app.py > ...
+--+
12 # 1. 모델 및 임베딩 로딩
13 @st.cache_resource
14 def load_models():
15     embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
16
17     model_id = "monologg/koelectra-base-v3-finetuned-korquad"
18     tokenizer = AutoTokenizer.from_pretrained(model_id)
19     model = AutoModelForQuestionAnswering.from_pretrained(model_id)
20
21     qa_pipeline = pipeline(
22         "question-answering",
23         model=model,
24         tokenizer=tokenizer,
25         device=-1,
26         max_length=512,
27         do_sample=False,
28         temperature=0.1,
29         truncation=True
30     )
31     return embedding_model, tokenizer, qa_pipeline
32
33 embedding_model, qa_tokenizer, text_gen = load_models()
```

소스 코드

```
36 # 2. Tab 1: DB/DB 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

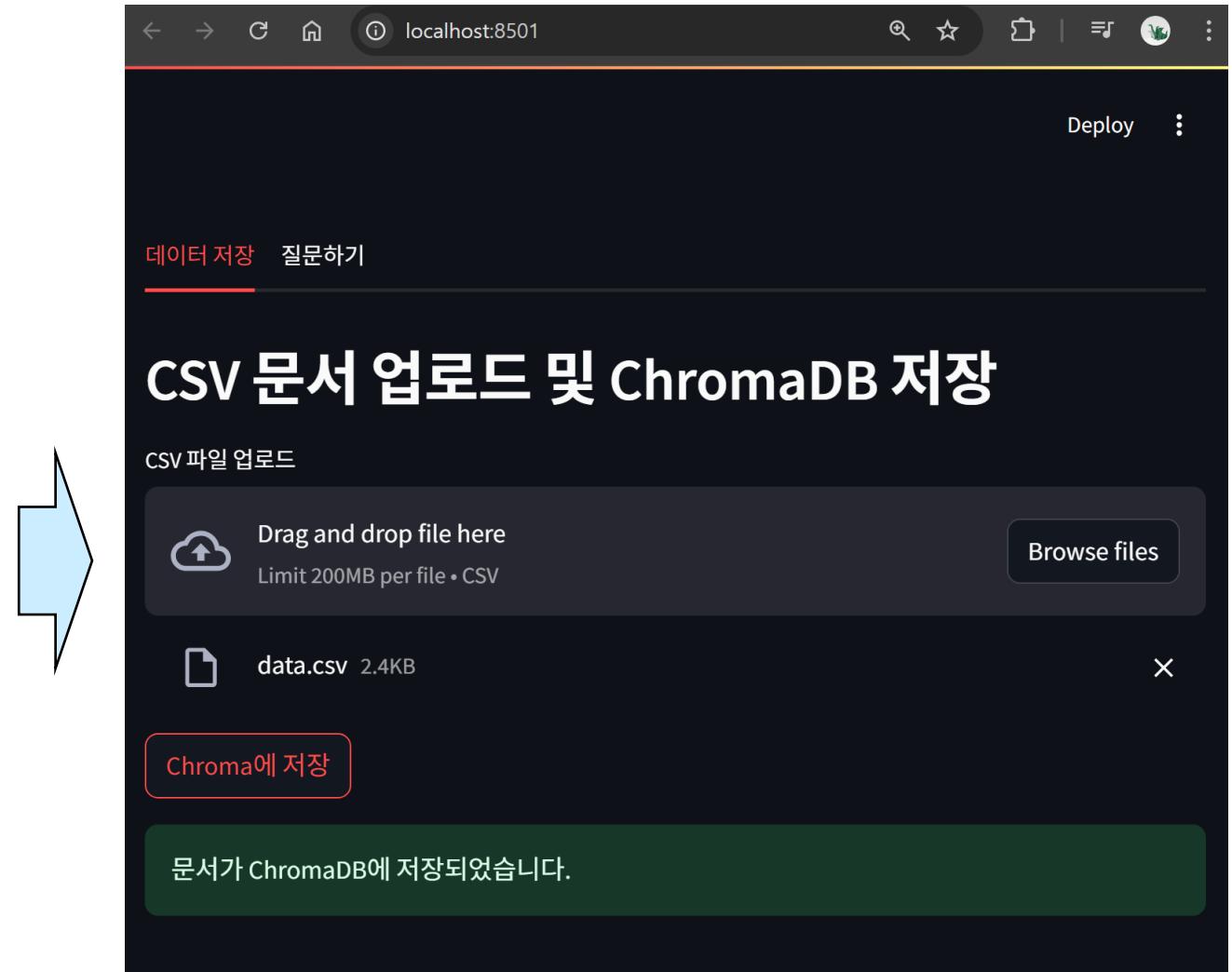
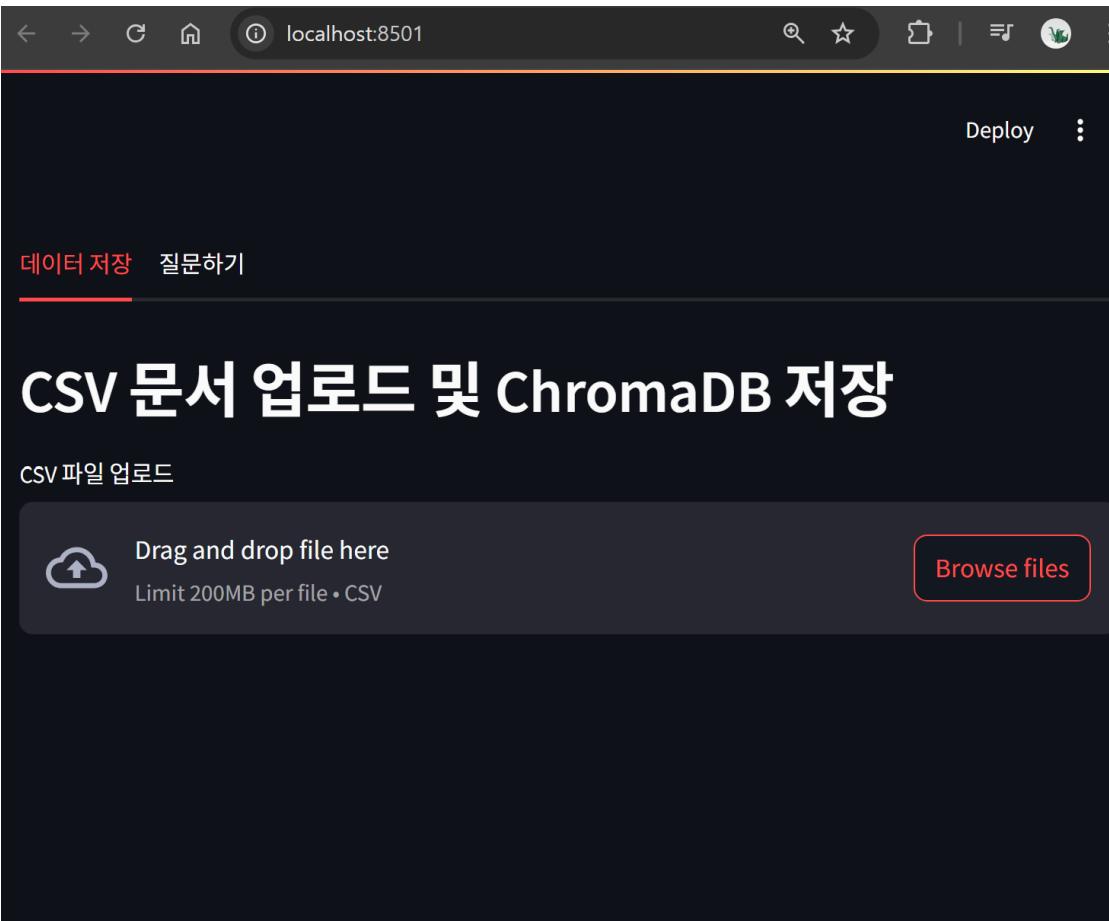
소스 코드

```
project02 > app.py > ...
57     # 3. 탭 2: 질문 응답
58     with tab2:
59         st.header("질문을 입력하고 응답을 확인하세요.")
60         question = st.text_input("질문을 입력하세요:")
61
62         if st.button("답변 생성"):
63
64             if not question:
65                 st.warning("질문을 입력하세요.")
66             else:
67                 retriever = Chroma(
68                     persist_directory=".chromaDB1",
69                     embedding_function=embedding_model
70                 ).as_retriever(search_kwargs={"k": 3})
71                 docs = retriever.invoke(question)
72
73             if not docs:
74                 st.error("유사 문서를 찾지 못했습니다.")
75             else:
76                 best_context = docs[0].page_content
77                 result = text_gen(question=question, context=best_context)
78
79                 st.subheader("정답")
80                 st.write(result["answer"])
81
82                 st.subheader("참고 문서 (Top 3)")
83                 for i, doc in enumerate(docs):
84                     st.markdown(f"***{i+1}.*** {doc.page_content}")
```

Section

랭체인기반서비스실습

결과 확인

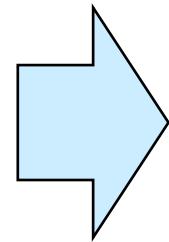


Section

랭체인기반서비스실습

결과 확인

A screenshot of a web application interface. At the top, there is a navigation bar with icons for back, forward, search, and other browser functions. The URL 'localhost:8501' is displayed. Below the navigation bar, there are two buttons: '데이터 저장' (Data Save) and '질문하기' (Ask Question), with '질문하기' being underlined. The main content area has a large heading '질문을 입력하고 응답을 확인하세요.' (Enter a question and check the response). Below this, there is a text input field labeled '질문을 입력하세요:' (Enter a question:). At the bottom left, there is a button labeled '답변 생성' (Generate Answer).



A screenshot of the same web application interface, showing the response confirmation screen. The URL 'localhost:8501' is again at the top. The main content area features a large heading '질문을 입력하고 응답을 확인하세요.' (Enter a question and check the response). Below it, there is a text input field with the placeholder '질문을 입력하세요:' (Enter a question:). A question is entered: 'MES 공정 로그 보관 기간은?'. To the right of the input field is a button labeled '답변 생성' (Generate Answer). Below the input field, the word '정답' (Correct Answer) is displayed in bold, followed by the answer '6개월간' (6 months). Further down, there is a section titled '참고 문서 (Top 3)' (Reference Document (Top 3)) with three numbered points:

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.

GEN AI 인텐시브 과정

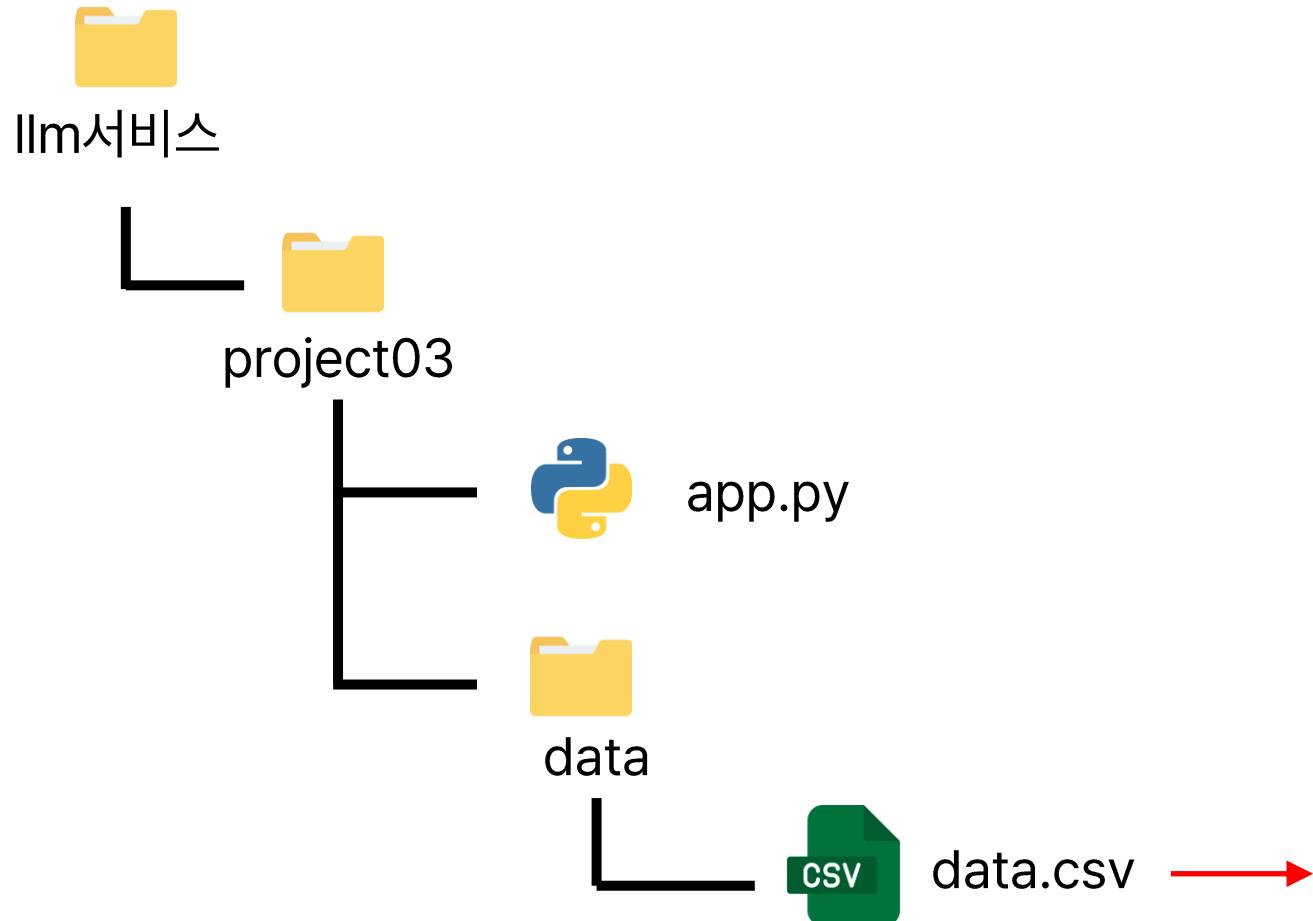
Section 1. LLM 기반 챗봇 만들기

Section 1-3. 멀티턴기반 서비스 실습

Section

멀티턴기반서비스실습

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 > ...

```
1 import pandas as pd
2 import streamlit as st
3 from langchain_chroma import Chroma
4 from langchain.schema import Document
5 from langchain_huggingface import HuggingFaceEmbeddings
6 from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline
7
8 # 1. UI/UX 설정
9 st.set_page_config(page_title="멀티턴 RAG QA 시스템", layout="wide")
10 st.sidebar.header("사이드바 메뉴")
11 mode = st.sidebar.radio("모드를 선택하세요", ["데이터 저장", "질문하기"])
12
```

Section

멀티턴기반서비스실습

소스 코드

```
13 # 2. 임베딩 및 QA 모델 준비
14 @st.cache_resource
15 def load_embedding_model():
16     return HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
17
18 @st.cache_resource
19 def load_qa_model():
20     model_id = "monologg/koelectra-base-v3-finetuned-korquad"
21     tokenizer = AutoTokenizer.from_pretrained(model_id)
22     model = AutoModelForQuestionAnswering.from_pretrained(model_id)
23     pipe = pipeline(
24         "question-answering",
25         model=model,
26         tokenizer=tokenizer,
27         device=-1,
28         max_length=512,
29         do_sample=False,
30         temperature=0.1,
31         truncation=True
32     )
33     return pipe
34
35 embedding_model = load_embedding_model()
36 text_gen = load_qa_model()
```

소스 코드

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

Section

멀티턴기반서비스실습

소스 코드

```
58 # 4. 템 2: 질문 응답
59 elif mode=="질문하기":
60     st.header("질문을 입력하고 대화를 이어가보세요!")
61
62     # 백엔드 DB 불러오기
63     @st.cache_resource
64     def get_retriever():
65         res = Chroma( persist_directory="./chromaDB",
66                     embedding_function=embedding_model
67                 ).as_retriever(search_kwargs={"k": 3})
68         return res
69
70     # 세션 상태 초기화
71     if "messages" not in st.session_state:
72         st.session_state.messages = []
73     if "last_contexts" not in st.session_state:
74         st.session_state.last_contexts = []
75
76     # 기존 대화 내용 출력
77     for msg in st.session_state.messages:
78         with st.chat_message(msg["role"]):
79             st.markdown(msg["content"])
80
81     # 채팅 입력창
82     user_input = st.chat_input("질문을 입력하세요:")
83
84     if user_input:
85         # 사용자 메시지 기록
86         st.session_state.messages.append({"role": "user", "content": user_input})
87         with st.chat_message("user"):
88             st.markdown(user_input)
```

소스 코드

```
84     if user_input:
85         # 사용자 메시지 기록
86         st.session_state.messages.append({"role": "user", "content": user_input})
87         with st.chat_message("user"):
88             st.markdown(user_input)
89
90         # 1. 참고 문서 요청 여부
91         if "참고문서" in user_input or "참고 문서" in user_input:
92             if st.session_state.last_contexts:
93                 ref_response = "### 참고 문서\n" + "\n".join(
94                     f"{i+1}. {ctx}" for i, ctx in enumerate(st.session_state.last_contexts)
95                 )
96             else:
97                 ref_response = "참고 문서가 없습니다. 먼저 질문을 해주세요."
98
99             st.session_state.messages.append({"role": "assistant", "content": ref_response})
100            with st.chat_message("assistant"):
101                st.markdown(ref_response)
```

Section

멀티턴기반서비스실습

소스 코드

```
102
103      # 2. 일반 질문 처리
104  ▼
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  ▼
121      with st.chat_message("assistant"):
122          st.markdown(bot_response)
```

Section

멀티턴기반서비스실습

실행

```
stoic@WIN-7702CJ32RBN:/c/Users/stoic/Documents/work/4_2  
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>

Section

멀티턴기반서비스실습

데이터 저장

The screenshot shows a dark-themed web application interface. At the top, there is a header bar with icons for back, forward, search, and other browser functions, followed by the URL 'localhost:8501'. On the right side of the header are icons for search, star, copy, and more. Below the header, there is a 'Deploy' button and a three-dot menu icon.

사이드바 메뉴

모드를 선택하세요

- 데이터 저장
- 질문하기

CSV 문서 업로드 및 ChromaDB 저장

CSV 파일 업로드

Upload

Drag and drop file here
Limit 200MB per file • CSV

Browse files

Section

멀티턴기반서비스실습

데이터 저장

The screenshot shows a web browser window with the URL `localhost:8501` in the address bar. The page has a dark theme with white text. On the left, there's a sidebar menu titled "사이드바 메뉴" with options: "모드를 선택하세요" (Select mode), "데이터 저장" (selected with a red dot), and "질문하기". The main content area is titled "CSV 문서 업로드 및 ChromaDB 저장". It features a "CSV 파일 업로드" section with a "Drag and drop file here" button, a "Limit 200MB per file • CSV" note, and a "Browse files" button. A file named "data.csv" (2.4KB) is listed with a delete "X" icon. Below this is a button labeled "Chroma에 저장". At the bottom, a green bar displays the message "문서가 ChromaDB에 저장되었습니다." (The document has been saved to ChromaDB).

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

X

Chroma에 저장

문서가 ChromaDB에 저장되었습니다.

Section

멀티턴기반서비스실습

질문하기

The screenshot shows a web browser window with a dark theme. The address bar at the top displays "localhost:8501". The main content area features a large, bold Korean text "질문을 입력하고 대화를 이어가보세요!" (Please enter a question and continue the conversation!). To the left, there is a sidebar menu titled "사이드바 메뉴" (Sidebar Menu) with the following options:

- 모드를 선택하세요
- 데이터 저장
- 질문하기

At the bottom, there is a input field with the placeholder text "질문을 입력하세요:" followed by a right-pointing arrow button.

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:



Section

멀티턴기반서비스실습

질문하기

localhost:8501

Deploy

사이드바 메뉴

모드를 선택하세요

데이터 저장
 질문하기

2. 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
3. 소자 특성 분석 결과는 전자문서 시스템에 등록합니다.

클린룸 입장시 주의사항 알려줘

답변: 반드시

참고문서 알려줘

참고문서

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. 랭그래프기반 서비스 실습

Section

랭그래프기반서비스실습

폴더 구성



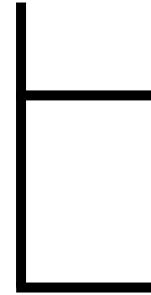
Ilm서비스



project03



app.py



data



data.csv



	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.					

Section

랭그래프기반서비스실습

소스 코드

```
1 import pandas as pd
2 import streamlit as st
3 from langchain.schema import Document
4 from langchain_chroma import Chroma
5 from langchain_huggingface import HuggingFaceEmbeddings
6 from langgraph.graph import StateGraph, END
7 from transformers import AutoTokenizer, AutoModelForQuestionAnswering, pipeline
8
9 # 랭이지 설정
10 st.set_page_config(page_title="LangGraph를 활용한 멀티언 RAG QA 시스템", layout="wide")
11 st.sidebar.header("사이드바 메뉴")
12 mode = st.sidebar.radio("모드를 선택하세요", ["데이터 저장", "질문하기"])
13
14 # 모델 로딩
15 @st.cache_resource
16 def load_embedding_model():
17     return HuggingFaceEmbeddings(model_name="sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")
```

소스 코드

```
19 @st.cache_resource
20 def load_qa_model():
21     model_id = "monologg/koelectra-base-v3-finetuned-korquad"
22     tokenizer = AutoTokenizer.from_pretrained(model_id)
23     model = AutoModelForQuestionAnswering.from_pretrained(model_id)
24     pipe = pipeline(
25         "question-answering",
26         model=model,
27         tokenizer=tokenizer,
28         device=-1,
29         max_length=512,
30         do_sample=False,
31         temperature=0.1,
32         truncation=True
33     )
34     return pipe
35
36 embedding_model = load_embedding_model()
37 text_gen = load_qa_model()
```

Section

랭그래프기반서비스실습

소스 코드

```
42 def build_graph(retriever):
43     def ask_question(state):
44         question = state["question"]
45         docs = retriever.invoke(question)
46         top_docs = docs[:3]
47         best_contexts = [doc.page_content for doc in top_docs]
48         combined_context = "\n".join(best_contexts)
49         result = text_gen(question=question, context=combined_context)
50         return {
51             **state,
52             "answer": result["answer"],
53             "context": best_contexts
54         }
55
56     def get_answer(state):
57         return state
58
59     def ask_reference(state):
60         state["show_reference"] = "참고문서" in state.get("question", "")
61         return state
62
63     def get_reference(state):
64         return state
65
66     def ask_continue(state):
67         state["continue"] = False
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)
```

랭그래프 노드 작성 시에는 @st.cache_resource 사용 안함
왜냐하면 랭그래프에서는 입력이 바뀌면 새롭게 실행되어야 하므로
streamlit 환경에서는 사용자에게 되묻기 어려우므로 단발성 처리

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()
```

Section

랭그래프기반서비스실습

소스 코드

```
99  # 템 1: 데이터 저장
100 if mode == "데이터 저장":
101     st.header("CSV 문서 업로드 및 ChromaDB 저장")
102     uploaded_file = st.file_uploader("CSV 파일 업로드", type="csv")
103
104     if uploaded_file:
105         df = pd.read_csv(uploaded_file, encoding='utf-8')
106         if "text" not in df.columns:
107             st.error("'text' 컬럼이 포함되어야 합니다.")
108         else:
109             docs = [Document(page_content=text) for text in df["text"].tolist()]
110             if st.button("Chroma에 저장"):
111                 Chroma.from_documents(
112                     documents=docs,
113                     embedding=embedding_model,
114                     persist_directory="./chromaDB"
115                 )
116                 st.success("문서가 ChromaDB에 저장되었습니다.")
```

Section

랭그래프기반서비스실습

소스 코드

```
118 # 템 2: 질의 응답
119 elif mode == "질문하기":
120     st.title("LangGraph 기반 RAG QA 시스템")
121
122     @st.cache_resource
123     def get_retriever():
124         return Chroma(
125             persist_directory=".chromaDB",
126             embedding_function=embedding_model
127         ).as_retriever(search_kwargs={"k": 3})
128
129     retriever = get_retriever()
130     app = build_graph(retriever)
131
132     if "messages" not in st.session_state:
133         st.session_state.messages = []
134
135     for msg in st.session_state.messages:
136         with st.chat_message(msg["role"]):
137             st.markdown(msg["content"])
138
139     user_input = st.chat_input("질문을 입력하세요:")
140
141     if user_input:
142         st.session_state.messages.append({"role": "user", "content": user_input})
143         with st.chat_message("user"):
144             st.markdown(user_input)
145
146         result = app.invoke({"question": user_input})
147
148         response = f"**답변:** {result['answer']}"
149         st.session_state.messages.append({"role": "assistant", "content": response})
150
151         with st.chat_message("assistant"):
152             st.markdown(response)
153             with st.expander("참고 문서 보기"):
154                 for i, ctx in enumerate(result["context"], 1):
155                     st.markdown(f"{i}. {ctx}")
```

Section

랭그래프기반서비스실습

결과 화면

The screenshot shows a web browser window with a dark theme. The address bar displays "localhost:8501". The main content area has a dark background with white text.

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

Deploy :

Section

랭그래프기반서비스실습

결과 화면

The screenshot shows a web application interface with a dark theme. At the top, the URL bar displays "localhost:8501". On the right side of the header, there are icons for search, star, refresh, and more, along with a "Deploy" button and a three-dot menu.

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.

This screenshot displays a user interface for managing CSV files and storing them in a ChromaDB. The left sidebar provides navigation options, while the main area focuses on the CSV upload process. A file named "data.csv" has been selected and is ready for storage. A success message at the bottom indicates the document has been successfully stored in ChromaDB.

Section

랭그래프기반서비스실습

결과 화면

The screenshot shows a web application running at `localhost:8501`. The browser's address bar and top navigation bar are visible. On the left side, there is a sidebar menu titled "사이드바 메뉴" with the sub-item "모드를 선택하세요". Below this are two radio button options: "데이터 저장" (unselected) and "질문하기" (selected, indicated by a red dot). The main content area features a large title "LangGraph 기반 RAG QA 시스템" in white text. At the bottom, there is a input field placeholder "질문을 입력하세요:" followed by a right-pointing arrow button.

localhost:8501

Deploy :

사이드바 메뉴

모드를 선택하세요

데이터 저장

질문하기

LangGraph 기반 RAG QA 시스템

질문을 입력하세요: >

Section

랭그래프기반서비스실습

결과 화면

The screenshot shows a web application running at `localhost:8501`. The interface has a dark theme with white text and light-colored cards.

Left Sidebar (사이드바 메뉴):

- 모드를 선택하세요
- 데이터 저장
- 질문하기

Main Content Area:

LangGraph 기반 RAG QA 시스템

Question Card: MES 공정 로그 보관 기간은?

Answer Card: 답변: 6개월간

[참고 문서 보기](#)

Top right corner: Deploy, three-dot menu.

Section

랭그래프기반서비스실습

결과 화면

The screenshot shows a web application running at `localhost:8501`. The interface has a dark theme with a sidebar menu on the left and a main content area on the right.

_sidebar menu:

- 사이드바 메뉴
- 모드를 선택하세요
- 데이터 저장
- 질문하기

main content area:

LangGraph 기반 RAG QA 시스템

Question: MES 공정 로그 보관 기간은?

Answer: 답변: 6개월간

Reference Document View:

- MES 시스템에 기록된 공정 로그는 6개월간 보관됩니다.
- 공정 이탈 발생 시 Shift 리더에게 즉시 보고합니다.
- 소자 특성 분석 결과는 전자문서 시스템에 등록합니다.

Input Field: 질문을 입력하세요: >

Section

랭그래프기반서비스실습

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