제주도_맛집_추천_대화형_AI서비스_개발

"Discover Jeju's Best Eats!"

My Personal Jeju Island Foodie Adventure

Team AlphaStorm

황승현 / 백서연 / 이원병

Index

INDEX

1 Project Overview

2 Data Processing and Analysis

3 RAG Pipeline Development

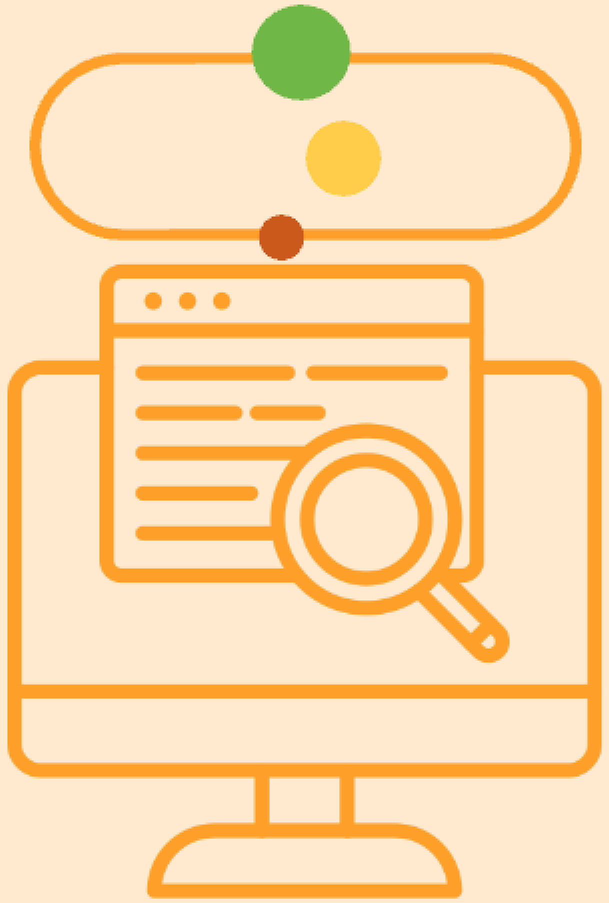
4 Implementation Video

Project Overview

01.

Project Overview

Are you looking for those places everyone has visited at least once—like Udo, London Bagel Museum, Jungmun, and Yeondong?

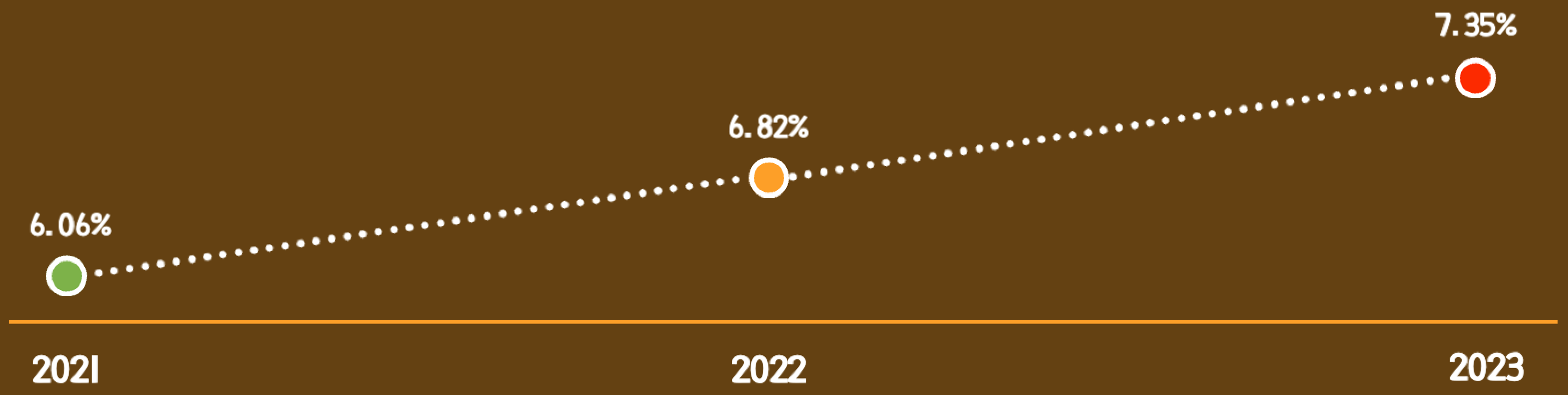


Restaurant Overtourism

What is Restaurant Overtourism?

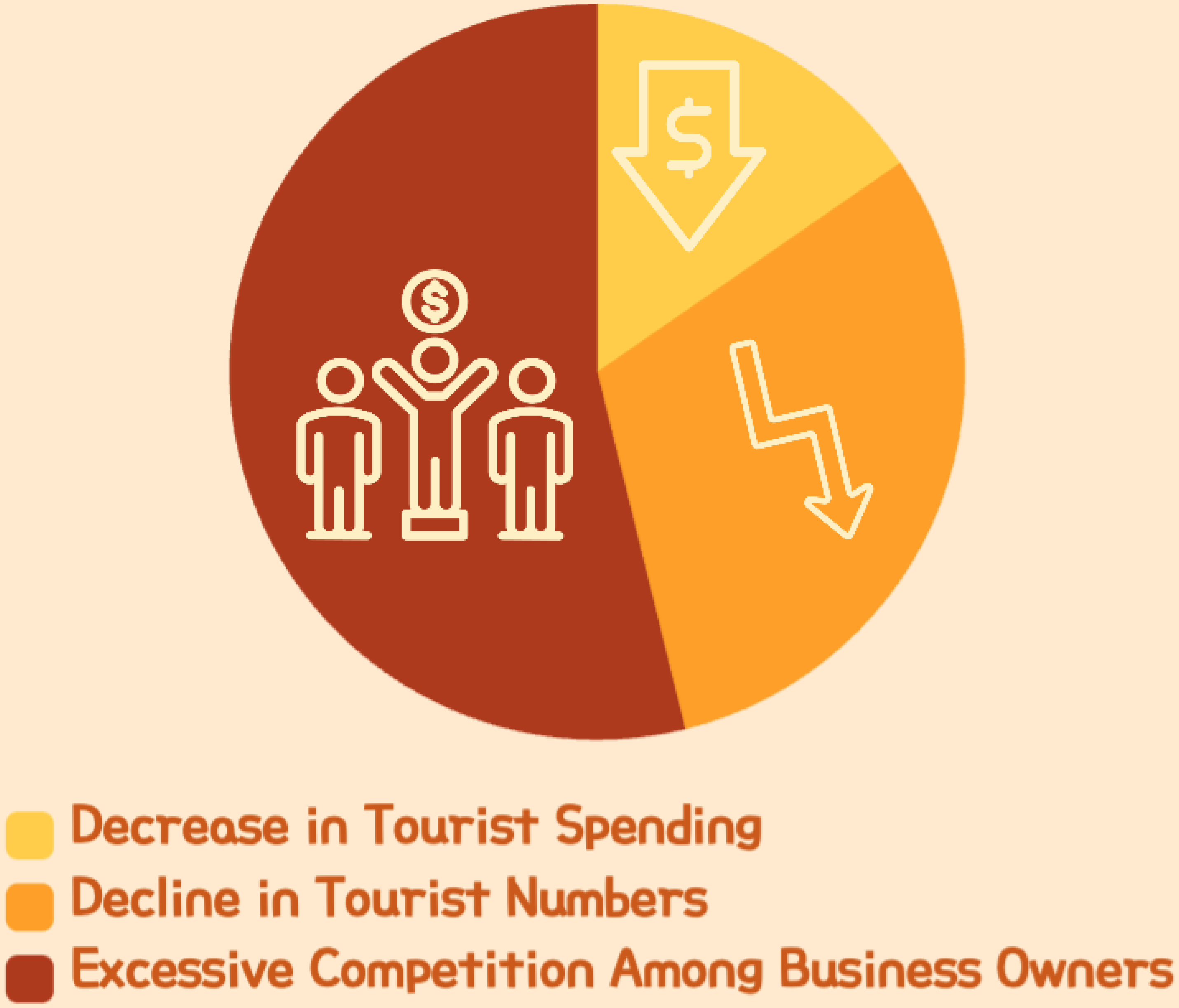
The issue arises when an excessive number of visitors flock to specific popular restaurants or eateries.

Trend of Restaurant Closures in Jeju Island



출처: 행정안전부

Causes of Restaurant Closures in Jeju Island



Restaurant Overtourism

Decrease in Tourist Spending

Shorter stays and lower spending by tourists have reduced overall expenditures.

Decline in Tourist Numbers

Fewer domestic tourists have led to a smaller customer base.

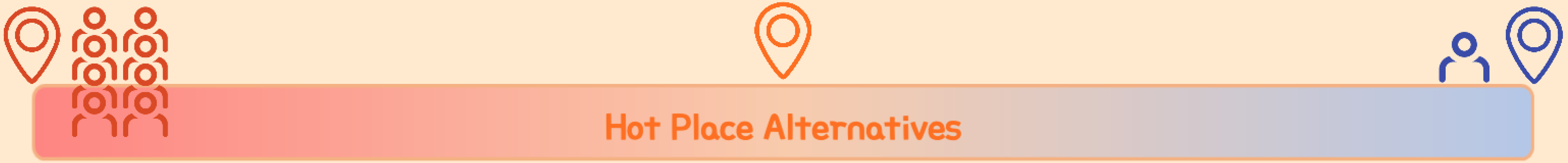
Excessive Competition Among Business Owners

An increase in restaurants post-pandemic has intensified competition.

Restaurant overtourism arises when too many visitors crowd specific popular eateries.

Hot/Cool/Cold Place

Discovering Restaurants Between Hot and Cold Places



Hot Place

Popular and crowded spots

Cool Place

Great food and service, but less known to tourists

Cold Place

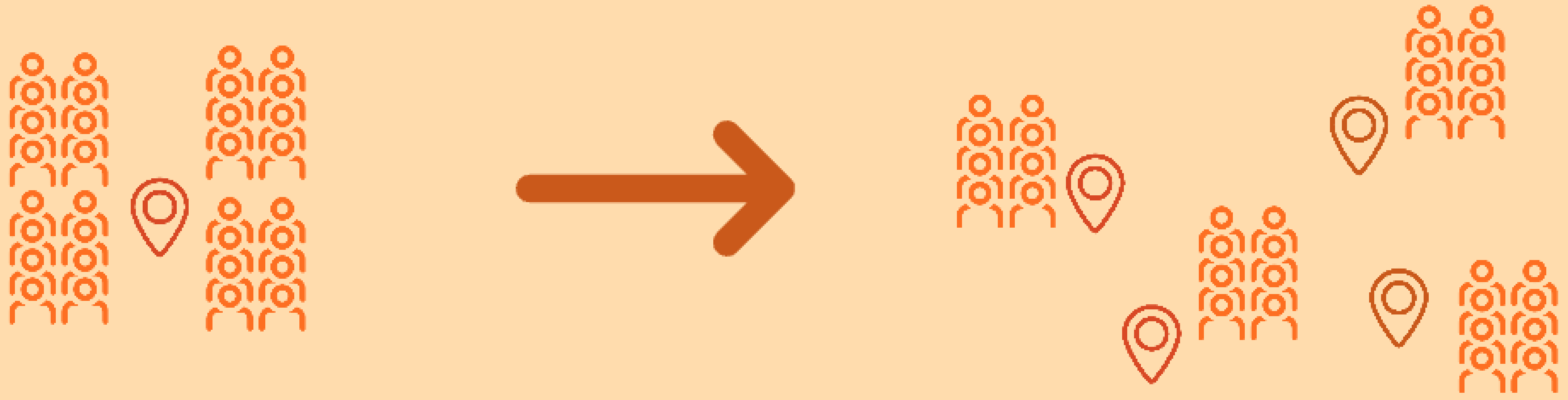
Lesser-known spots that are not widely exposed to tourists

Restaurant Overtourism Solutions

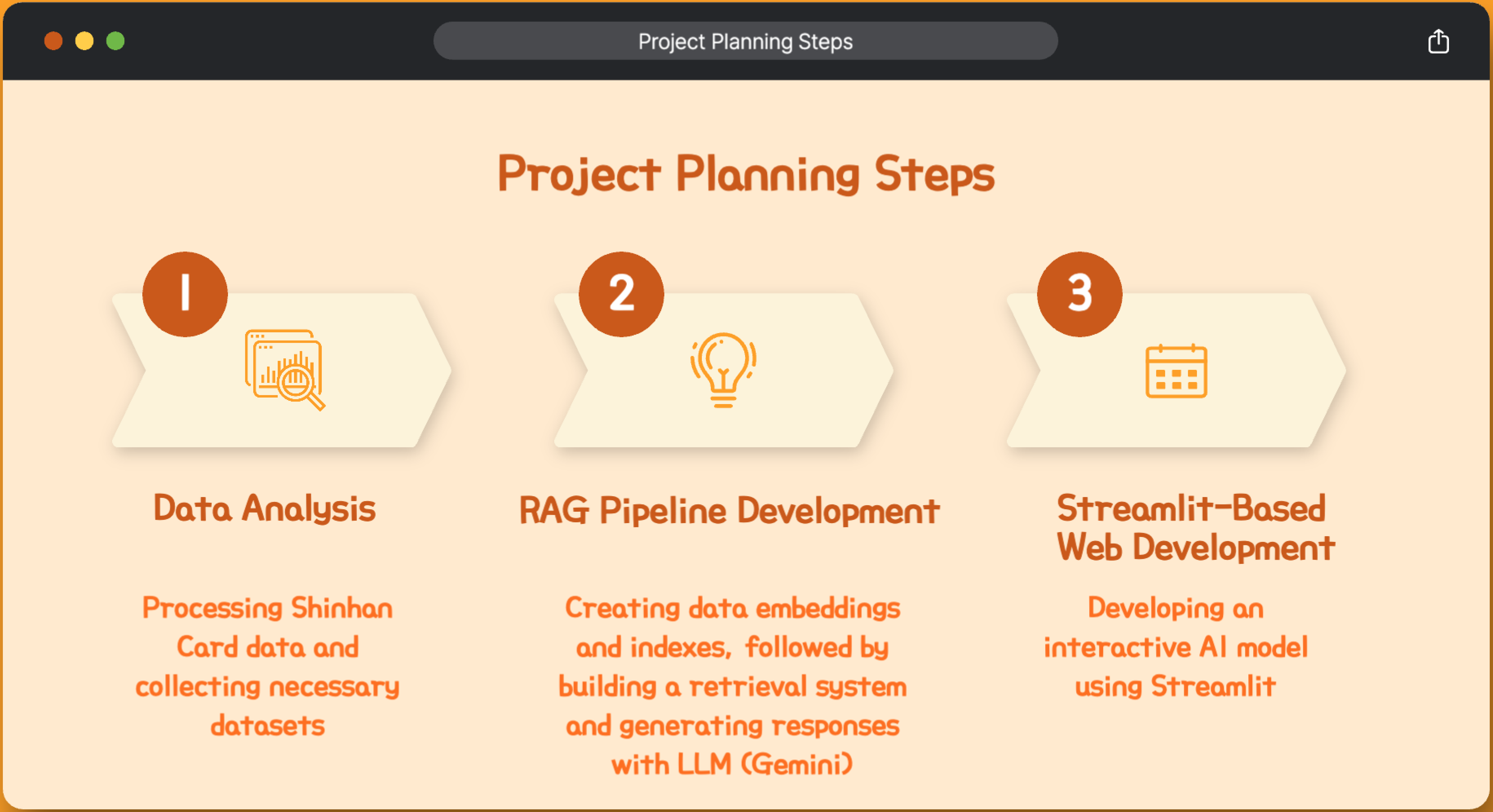
Goal

1. Mitigate excessive competition among business owners.
2. Provide opportunities for diverse restaurants beyond hot places.
3. Offer users access to various dining options tailored to their needs

Preventing Overcrowding and Concentration in Hot Places



Resolving Jeju's restaurant overtourism issues and generating positive effects on the local economy.



Project Planning Steps

1

Data Analysis

Processing Shinhan Card data and collecting necessary datasets

2

RAG Pipeline Development

Creating data embeddings and indexes, followed by building a retrieval system and generating responses with LLM (Gemini)

3

Streamlit-Based Web Development

Developing an interactive AI model using Streamlit

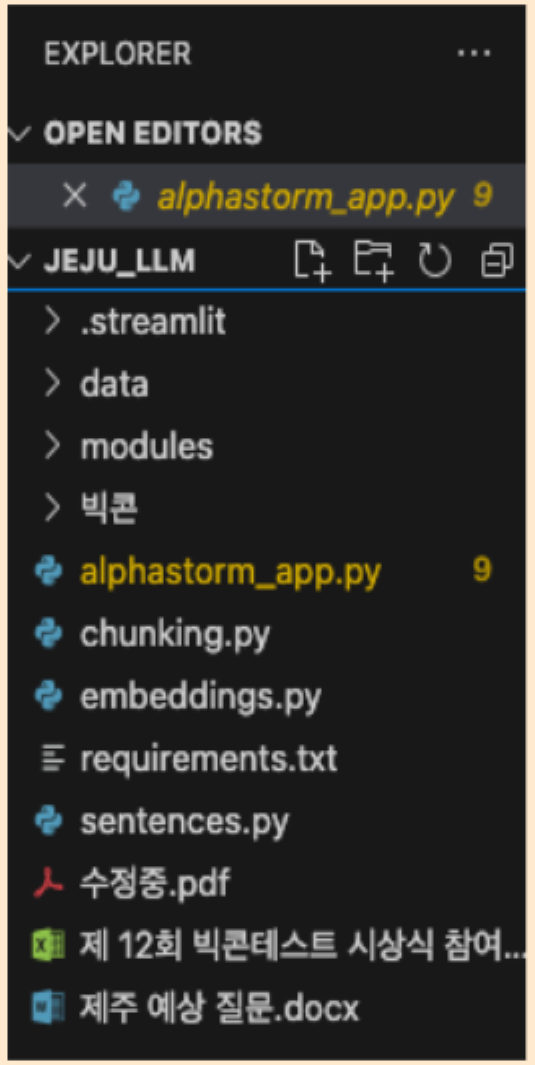
Project Planning Steps

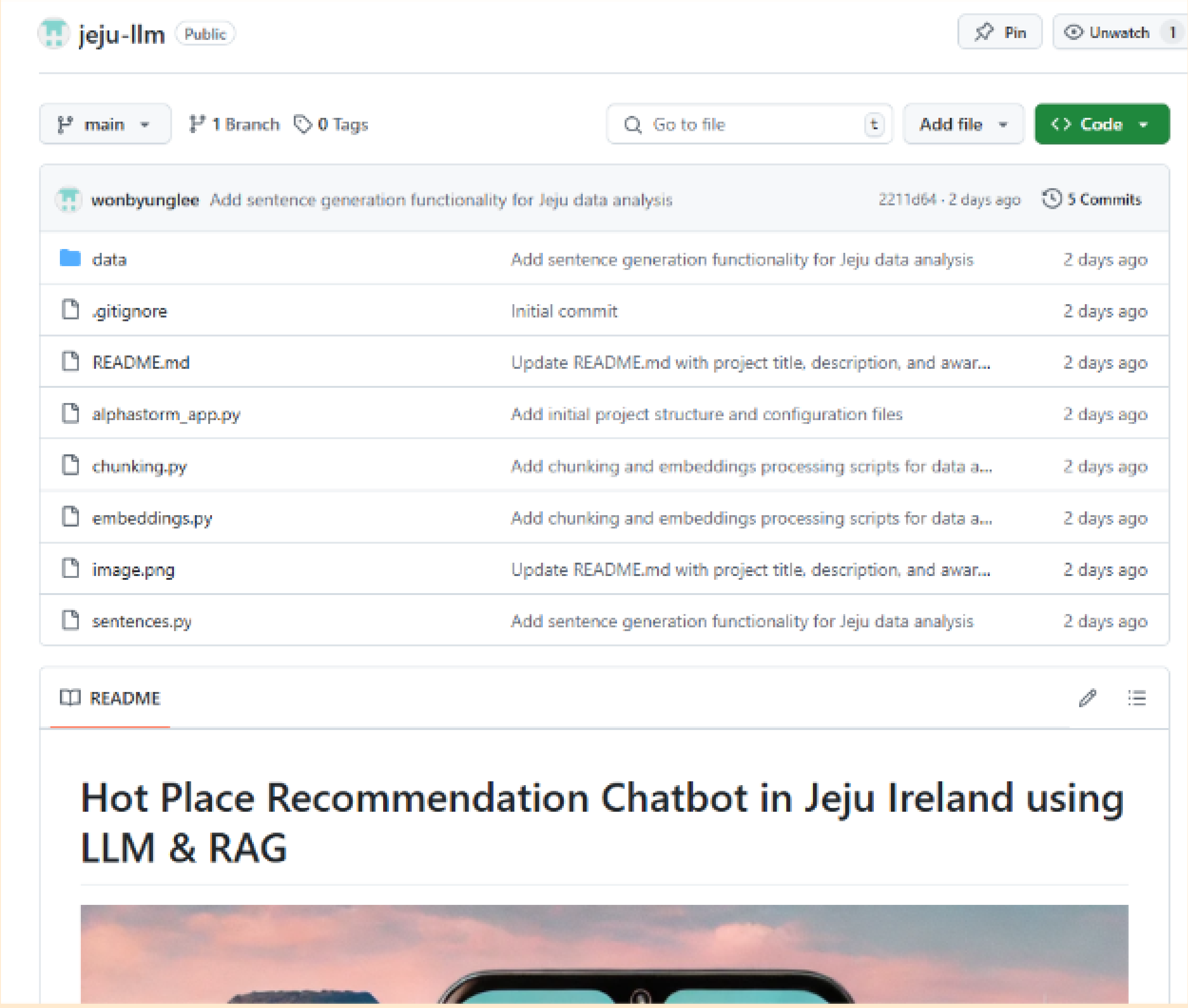
Tools we Used

Visual Studio

GitHub

GitHub Copilot

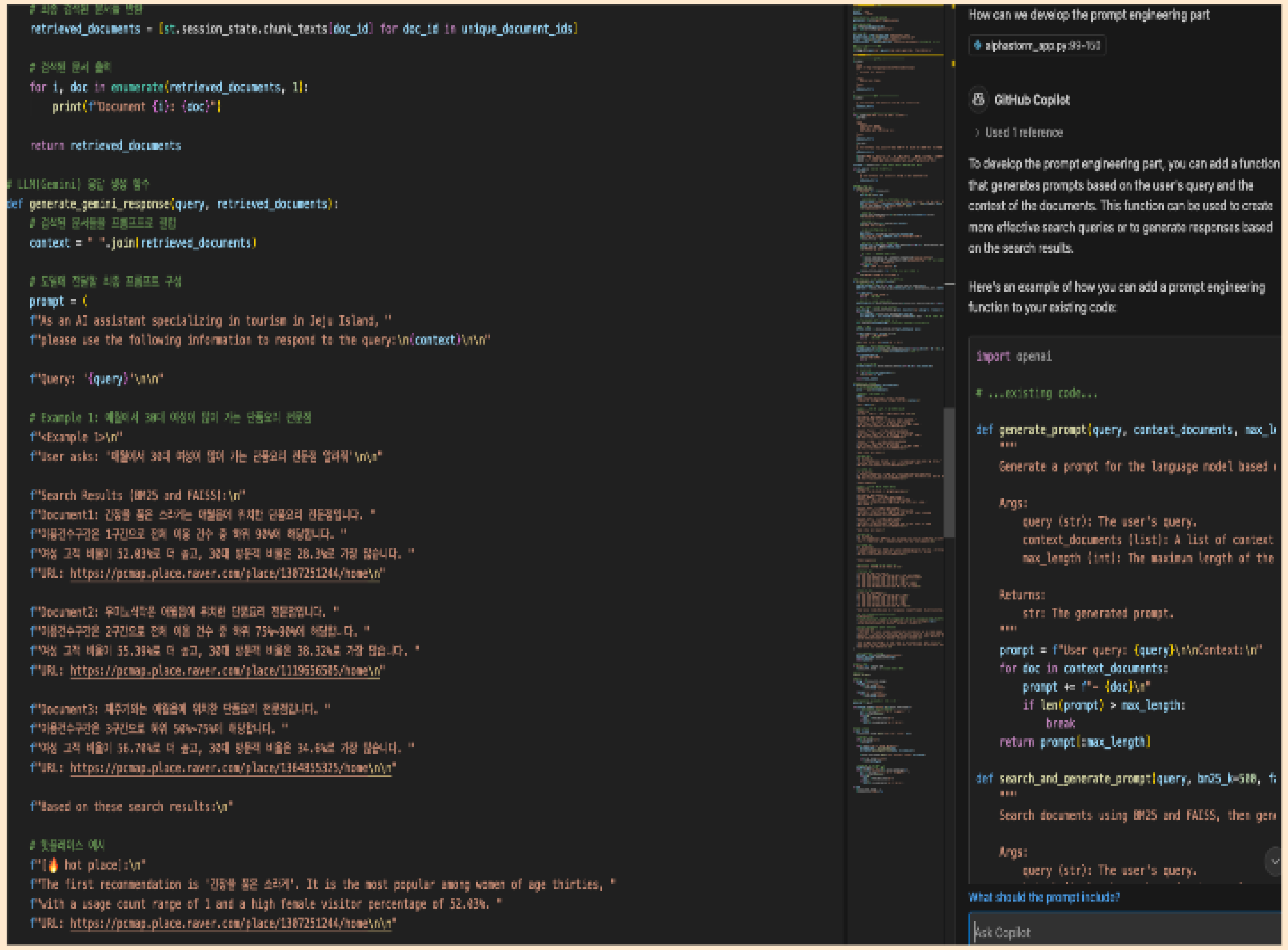




wonbyunglee Add sentence generation functionality for Jeju data analysis · 2211d64 · 2 days ago · 5 Commits

data	Add sentence generation functionality for Jeju data analysis	2 days ago
.gitignore	Initial commit	2 days ago
README.md	Update README.md with project title, description, and awar...	2 days ago
alphastorm_app.py	Add initial project structure and configuration files	2 days ago
chunking.py	Add chunking and embeddings processing scripts for data a...	2 days ago
embeddings.py	Add chunking and embeddings processing scripts for data a...	2 days ago
image.png	Update README.md with project title, description, and awar...	2 days ago
sentences.py	Add sentence generation functionality for Jeju data analysis	2 days ago

Hot Place Recommendation Chatbot in Jeju Ireland using LLM & RAG



```
    # 최종 검색된 문서를 반환
    retrieved_documents = [st.session_state.chunk_texts[doc_id] for doc_id in unique_document_ids]

    # 검색된 문서 출력
    for i, doc in enumerate(retrieved_documents, 1):
        print(f"Document {i}: {doc}")

    return retrieved_documents

# LLM(Gemini) 응답 생성 함수
def generate_gemini_response(query, retrieved_documents):
    # 검색된 문서들을 프롬프트로 결합
    context = " ".join(retrieved_documents)

    # 모델에 전달할 최종 프롬프트 구성
    prompt = (
    f"As an AI assistant specializing in tourism in Jeju Island, "
    f"please use the following information to respond to the query:\n{context}\n\n"

    f"Query: '{query}'\n\n"

    # Example 1: 애월에서 30대 여성이 많이 가는 단품요리 전문점
    f"<Example 1>\n"
    f"User asks: '애월에서 30대 여성이 많이 가는 단품요리 전문점 알려줘'\n\n"

    f"Search Results (BM25 and FAISS):\n"
    f"Document1: 감동을 품은 스리게는 애월읍에 위치한 단품요리 전문점입니다. "
    f"이용건수구간은 1구간으로 전체 이용 건수 중 상위 90%에 해당합니다. "
    f"여성 고객 비율이 52.03%로 더 높고, 30대 방문객 비율은 28.3%로 가장 많습니다. "
    f"URL: https://pcmap.place.naver.com/place/1387251244/home\n"

    f"Document2: 우지노식탁은 애월읍에 위치한 단품요리 전문점입니다. "
    f"이용건수구간은 2구간으로 전체 이용 건수 중 상위 75%~90%에 해당합니다. "
    f"여성 고객 비율이 55.39%로 더 높고, 30대 방문객 비율은 38.32%로 가장 많습니다. "
    f"URL: https://pcmap.place.naver.com/place/1119656585/home\n"

    f"Document3: 제주기와는 애월읍에 위치한 단품요리 전문점입니다. "
    f"이용건수구간은 3구간으로 상위 50%~75%에 해당합니다. "
    f"여성 고객 비율이 56.70%로 더 높고, 30대 방문객 비율은 34.6%로 가장 많습니다. "
    f"URL: https://pcmap.place.naver.com/place/1364855325/home\n\n"

    f"Based on these search results:\n"

    # 핫플레이스 예시
    f"[🔥 hot place]:\n"
    f"The first recommendation is '감동을 품은 스리게'. It is the most popular among women of age thirties, "
    f"with a usage count range of 1 and a high female visitor percentage of 52.03%. "
    f"URL: https://pcmap.place.naver.com/place/1387251244/home\n\n"
```

How can we develop the prompt engineering part

alphastorm_app.py:99-150

GitHub Copilot

› Used 1 reference

To develop the prompt engineering part, you can add a function that generates prompts based on the user's query and the context of the documents. This function can be used to create more effective search queries or to generate responses based on the search results.

Here's an example of how you can add a prompt engineering function to your existing code:

```
import openai

# ...existing code...

def generate_prompt(query, context_documents, max_l
    """
    Generate a prompt for the language model based o

    Args:
        query (str): The user's query.
        context_documents (list): A list of context
        max_length (int): The maximum length of the

    Returns:
        str: The generated prompt.
    """
    prompt = f"User query: {query}\n\nContext:\n"
    for doc in context_documents:
        prompt += f"- {doc}\n"
        if len(prompt) > max_length:
            break
    return prompt[:max_length]

def search_and_generate_prompt(query, bm25_k=500, f
    """
    Search documents using BM25 and FAISS, then gen

    Args:
        query (str): The user's query.
```

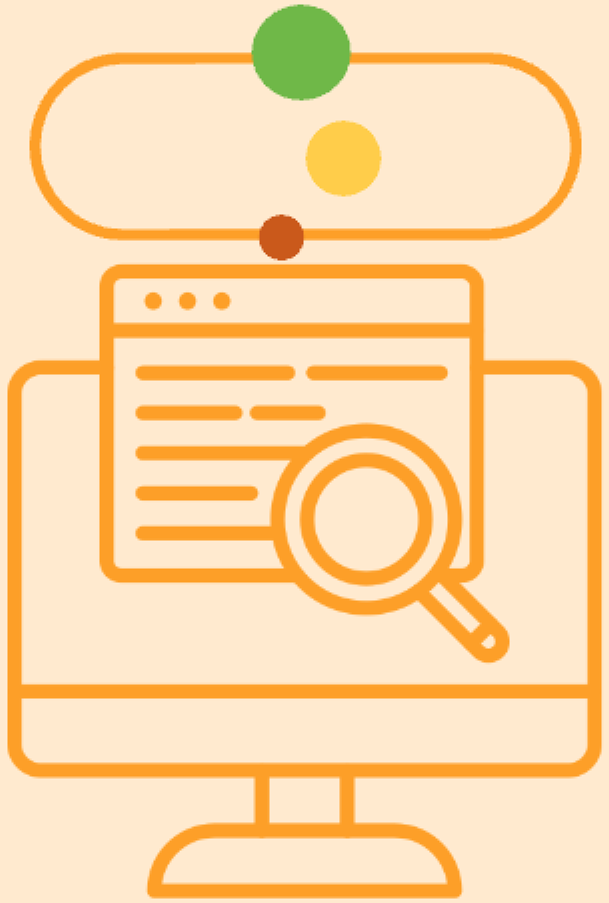
What should the prompt include?

Ask Copilot

RAG Pipeline Development

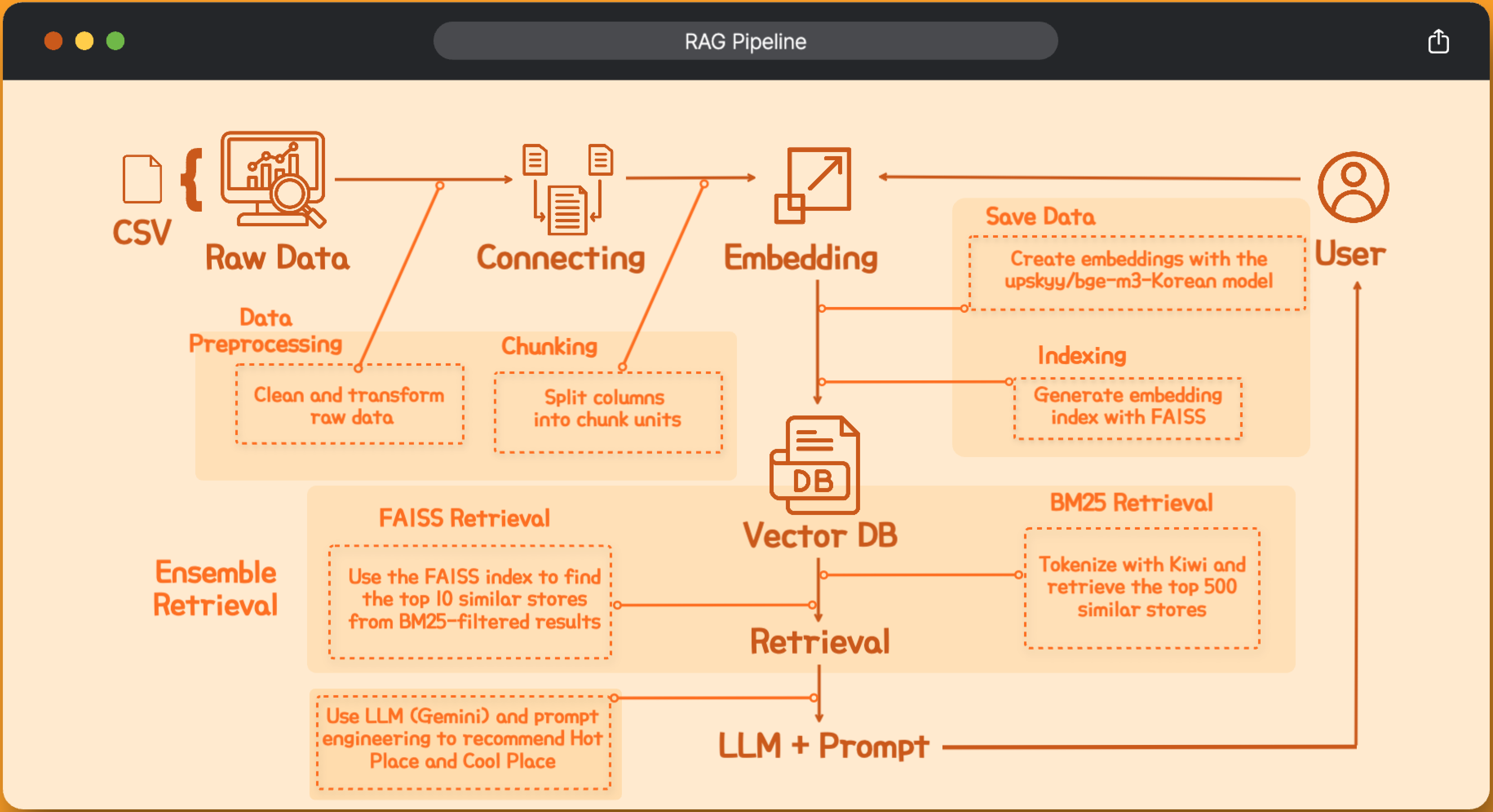
02.

RAG Pipeline Development



Here's an explanation of how the RAG pipeline was implemented!

RAG Pipeline



CSV

Raw Data

Connecting

Embedding

User

Data Preprocessing

Clean and transform raw data

Chunking

Split columns into chunk units

Save Data

Create embeddings with the upskyy/bge-m3-Korean model

Indexing

Generate embedding index with FAISS

Vector DB

Ensemble Retrieval

FAISS Retrieval

Use the FAISS index to find the top 10 similar stores from BM25-filtered results

BM25 Retrieval

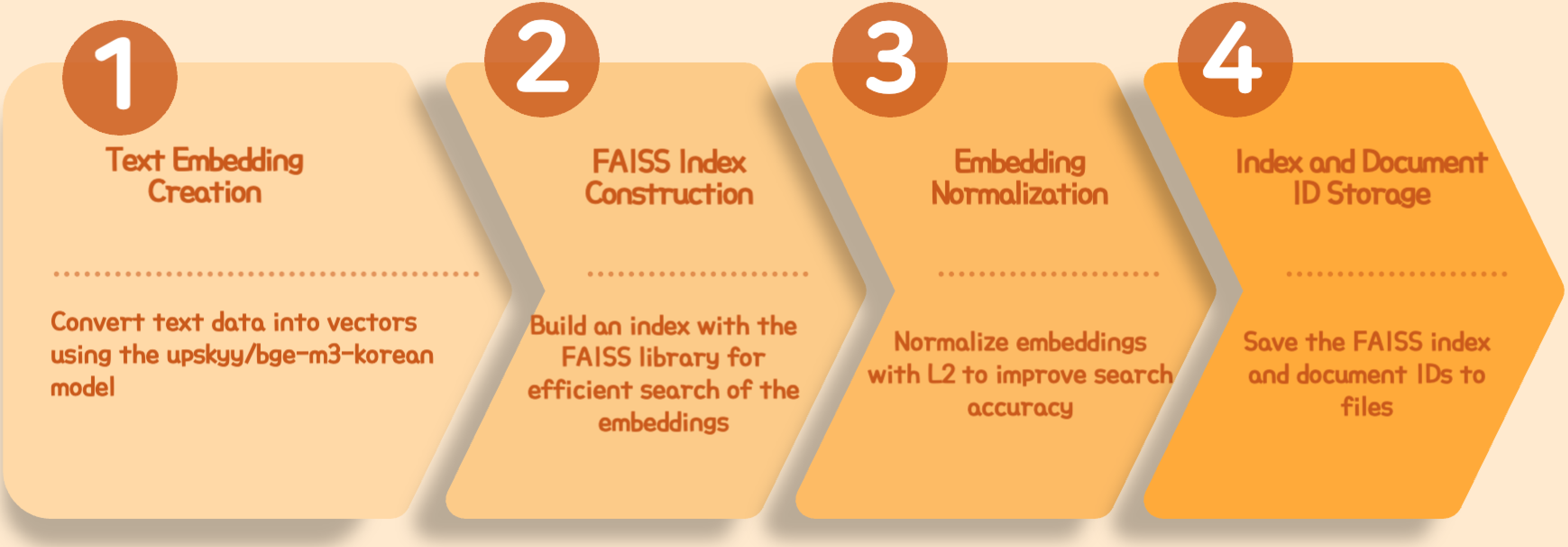
Tokenize with Kiwi and retrieve the top 500 similar stores

Retrieval

Use LLM (Gemini) and prompt engineering to recommend Hot Place and Cool Place

LLM + Prompt

Embedding and Indexing



1

Text Embedding Creation

Convert text data into vectors using the upskyy/bge-m3-korean model

2

FAISS Index Construction

Build an index with the FAISS library for efficient search of the embeddings

3

Embedding Normalization

Normalize embeddings with L2 to improve search accuracy

4

Index and Document ID Storage

Save the FAISS index and document IDs to files

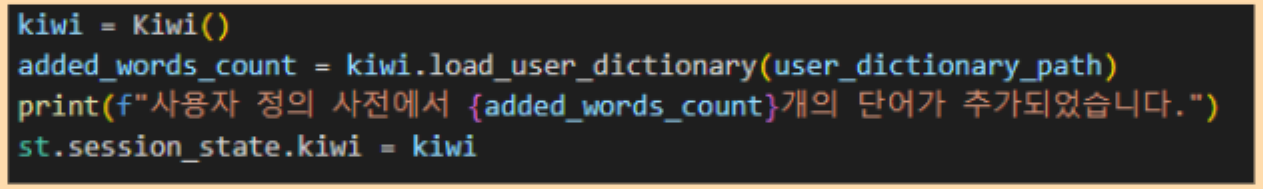
Ensemble Retrieval

Ensemble Retrieval

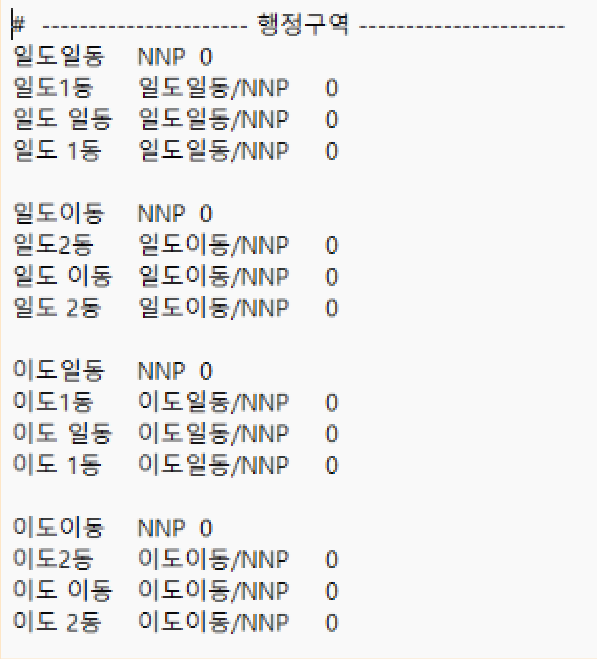
BM25

1. Tokenize search queries using the Kiwi tokenizer (add custom user-defined dictionaries)
2. Search the data by calculating text-based similarity with tokenized queries using the BM25 model
3. Filter the top 500 results using BM25

1. Kiwi Tokenization: Tokenize queries and include custom user-defined dictionaries

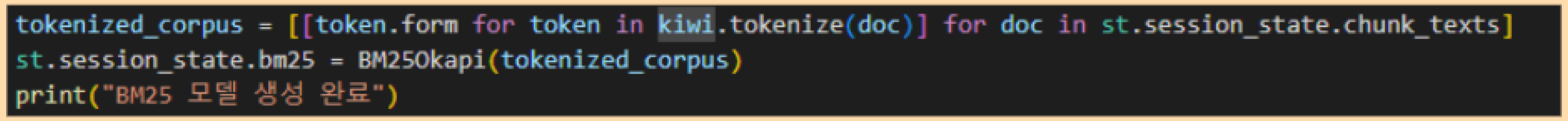


```
kiwi = Kiwi()
added_words_count = kiwi.load_user_dictionary(user_dictionary_path)
print(f"사용자 정의 사전에서 {added_words_count}개의 단어가 추가되었습니다.")
st.session_state.kiwi = kiwi
```



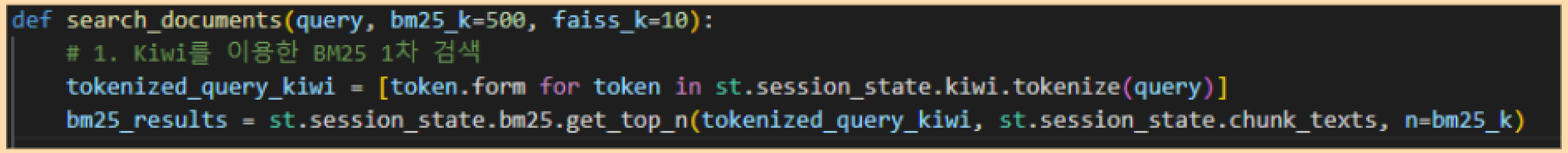
```
# ---------------------- 행정구역 ----------------------
일도일동    NNP  0
일도1동     일도일동/NNP    0
일도 일동   일도일동/NNP    0
일도 1동    일도일동/NNP    0

일도이동    NNP  0
일도2동     일도이동/NNP    0
일도 이동   일도이동/NNP    0
일도 2동    일도이동/NNP    0

이도일동    NNP  0
이도1동     이도일동/NNP    0
이도 일동   이도일동/NNP    0
이도 1동    이도일동/NNP    0

이도이동    NNP  0
이도2동     이도이동/NNP    0
이도 이동   이도이동/NNP    0
이도 2동    이도이동/NNP    0
```

2. BM25 Search: Use tokenized queries to calculate text similarity with BM25.



```
tokenized_corpus = [[token.form for token in kiwi.tokenize(doc)] for doc in st.session_state.chunk_texts]
st.session_state.bm25 = BM25Okapi(tokenized_corpus)
print("BM25 모델 생성 완료")
```

3. Retrieve and filter the top 500 most relevant documents using BM25



```
def search_documents(query, bm25_k=500, faiss_k=10):
    # 1. Kiwi를 이용한 BM25 1차 검색
    tokenized_query_kiwi = [token.form for token in st.session_state.kiwi.tokenize(query)]
    bm25_results = st.session_state.bm25.get_top_n(tokenized_query_kiwi, st.session_state.chunk_texts, n=bm25_k)
```

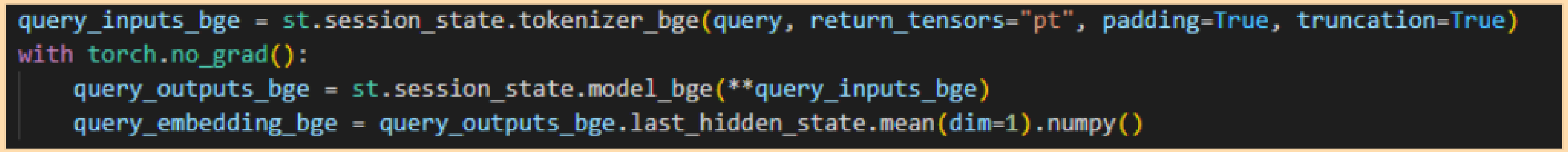
Ensemble Retrieval

Ensemble Retrieval

FAISS

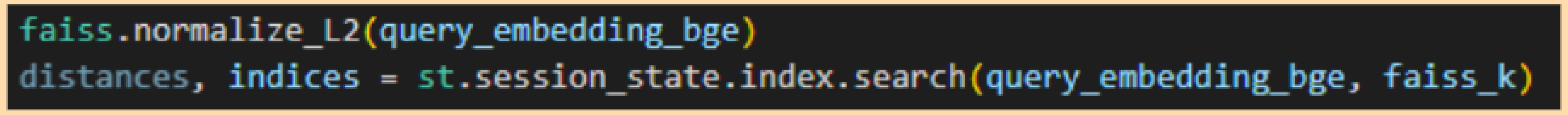
1. Convert the query into an embedding vector using the upskyy/bge-m3-korean model
2. Search the FAISS index for the most similar embeddings from BM25-filtered candidates
3. Use the IDs of the retrieved documents from FAISS to return the final set of relevant documents

1. Convert the query into an embedding vector using the upskyy/bge-m3-korean model



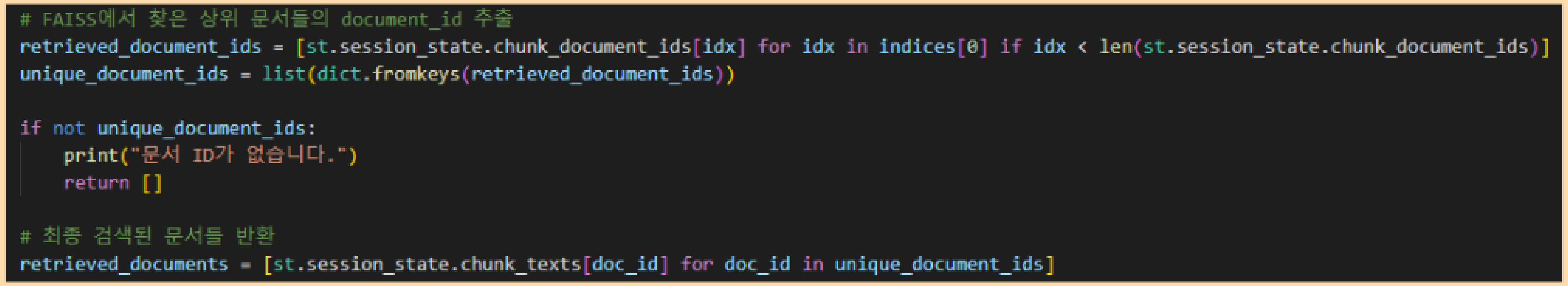
```
query_inputs_bge = st.session_state.tokenizer_bge(query, return_tensors="pt", padding=True, truncation=True)
with torch.no_grad():
    query_outputs_bge = st.session_state.model_bge(**query_inputs_bge)
    query_embedding_bge = query_outputs_bge.last_hidden_state.mean(dim=1).numpy()
```

2. Search the FAISS index for the most similar embeddings from BM25-filtered candidates



```
faiss.normalize_L2(query_embedding_bge)
distances, indices = st.session_state.index.search(query_embedding_bge, faiss_k)
```

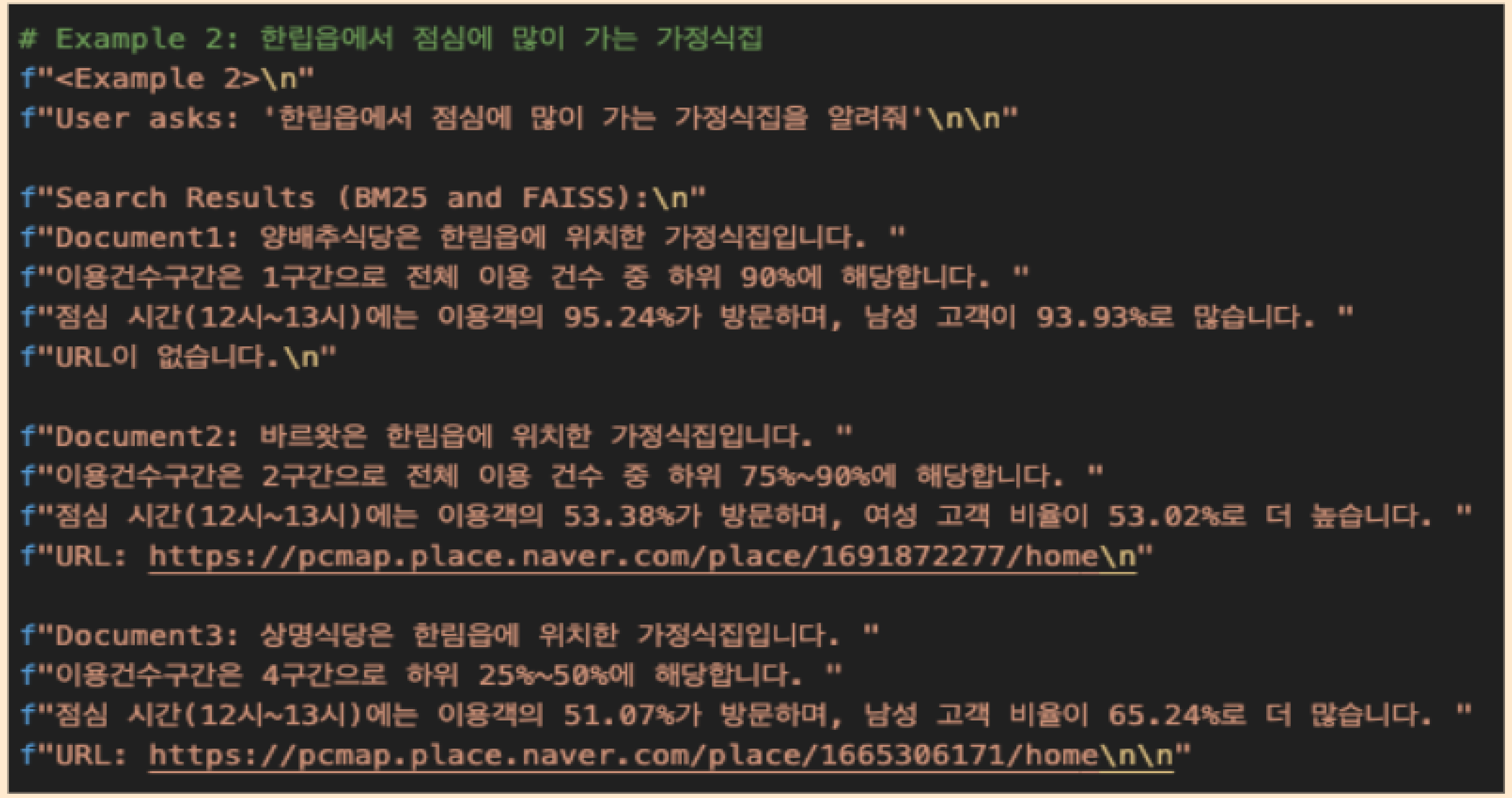
3. Use the document IDs from the FAISS search to return the final set of relevant documents



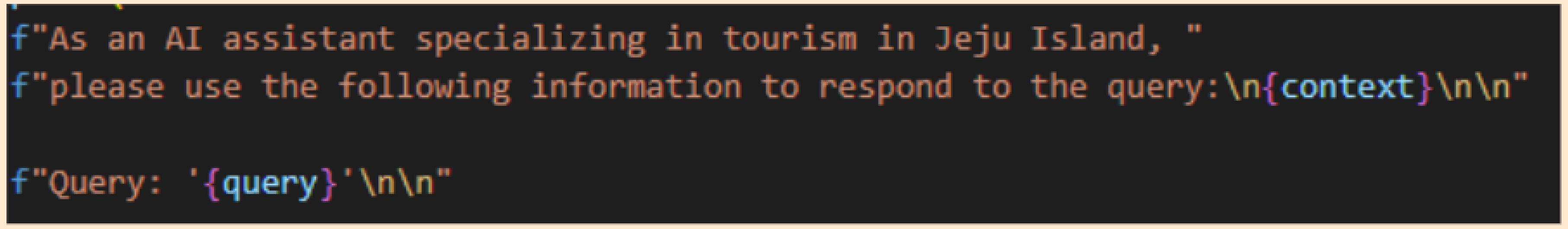
```
# FAISS에서 찾은 상위 문서들의 document_id 추출
retrieved_document_ids = [st.session_state.chunk_document_ids[idx] for idx in indices[0] if idx < len(st.session_state.chunk_document_ids)]
unique_document_ids = list(dict.fromkeys(retrieved_document_ids))

if not unique_document_ids:
    print("문서 ID가 없습니다.")
    return []

# 최종 검색된 문서들 반환
retrieved_documents = [st.session_state.chunk_texts[doc_id] for doc_id in unique_document_ids]
```

Prompt Engineering

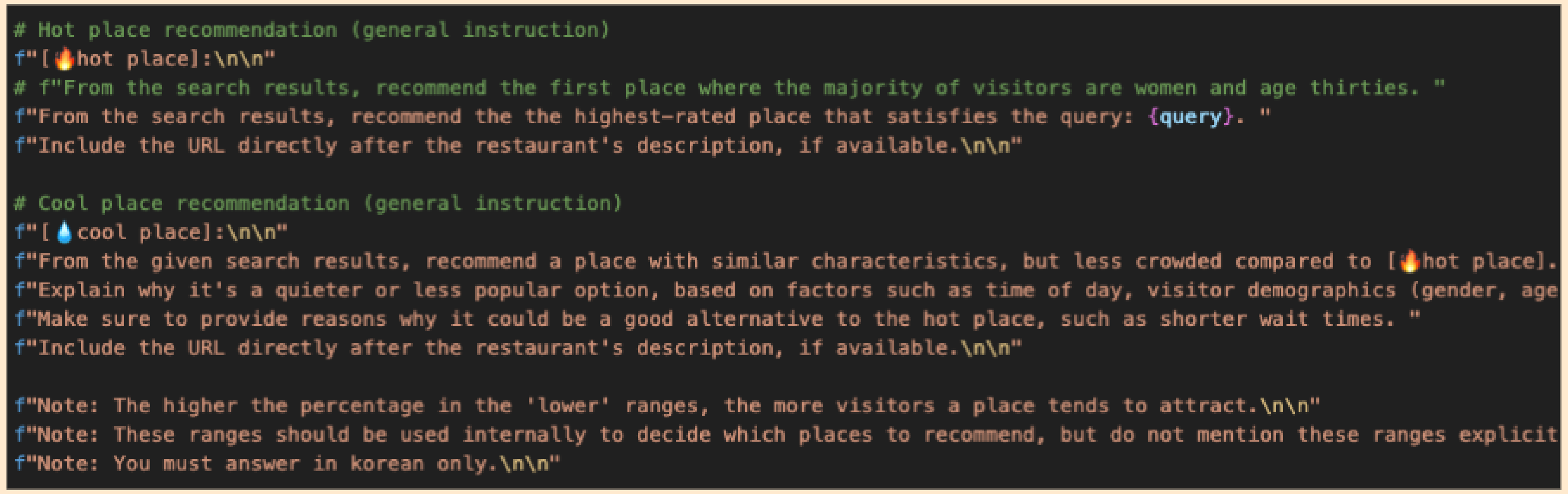
Prompt Engineering



```
# Example 2: 한림읍에서 점심에 많이 가는 가정식집
f"<Example 2>\n"
f"User asks: '한림읍에서 점심에 많이 가는 가정식집을 알려줘'\n\n"

f"Search Results (BM25 and FAISS):\n"
f"Document1: 양배추식당은 한림읍에 위치한 가정식집입니다. "
f"이용건수구간은 1구간으로 전체 이용 건수 중 하위 90%에 해당합니다. "
f"점심 시간(12시~13시)에는 이용객의 95.24%가 방문하며, 남성 고객이 93.93%로 많습니다. "
f"URL이 없습니다.\n"

f"Document2: 바르왓은 한림읍에 위치한 가정식집입니다. "
f"이용건수구간은 2구간으로 전체 이용 건수 중 하위 75%~90%에 해당합니다. "
f"점심 시간(12시~13시)에는 이용객의 53.38%가 방문하며, 여성 고객 비율이 53.02%로 더 높습니다. "
f"URL: https://pcmap.place.naver.com/place/1691872277/home\n"

f"Document3: 상명식당은 한림읍에 위치한 가정식집입니다. "
f"이용건수구간은 4구간으로 하위 25%~50%에 해당합니다. "
f"점심 시간(12시~13시)에는 이용객의 51.07%가 방문하며, 남성 고객 비율이 65.24%로 더 많습니다. "
f"URL: https://pcmap.place.naver.com/place/1665306171/home\n\n"
```



```
f"As an AI assistant specializing in tourism in Jeju Island, "
f"please use the following information to respond to the query:\n{context}\n\n"

f"Query: '{query}'\n\n"
```



```
# Hot place recommendation (general instruction)
f"[🔥hot place]:\n\n"
# f"From the search results, recommend the first place where the majority of visitors are women and age thirties. "
f"From the search results, recommend the the highest-rated place that satisfies the query: {query}. "
f"Include the URL directly after the restaurant's description, if available.\n\n"

# Cool place recommendation (general instruction)
f"[💧cool place]:\n\n"
f"From the given search results, recommend a place with similar characteristics, but less crowded compared to [🔥hot place].
f"Explain why it's a quieter or less popular option, based on factors such as time of day, visitor demographics (gender, age
f"Make sure to provide reasons why it could be a good alternative to the hot place, such as shorter wait times. "
f"Include the URL directly after the restaurant's description, if available.\n\n"

f"Note: The higher the percentage in the 'lower' ranges, the more visitors a place tends to attract.\n\n"
f"Note: These ranges should be used internally to decide which places to recommend, but do not mention these ranges explicit
f"Note: You must answer in korean only.\n\n"
```

Context Setting
Combine query and documents to give the model key info

Recommendation Instructions
Suggest a hot place and alternative cool places

Few-shot Prompting
Provide clear examples for hot place and cool place formats

My Personal Jeju Island Foodie Adventure

Thank you