Technical Deep Dive

RAG

The Pain Points

Complexity

File Search

The Gains

Simplicity

Which side are you on?

Find out now

RAG using Plain python

Step-1: Installation

```
!pip install openai chromadb --quiet
```

Step-2: File uploading

```
from google.colab import files

uploaded = files.upload()
filename = list(uploaded.keys())[0]

with open(filename) as file:
    knowledge_base = file.read()
```

Step-3: File Chunking

```
def fixed_word_chunk(text, chunk_size=50):
    words = text.split()
    return [" ".join(words[i:i + chunk_size]) for i in range(0, len(words), chunk_size)]

knowledge_chunks = fixed_word_chunk(knowledge_base, chunk_size=50)
```

RAG using Plain python

Step-4: OpenAI Embedding Setup

```python
from openai import OpenAI

client = OpenAI()

def get_embeddings(text):
    response = client.embeddings.create(
        input=text,
        model="text-embedding-3-small"
    )
    return response.data[0].embedding
```

Step-5: Storing Embeddings in Chroma DB

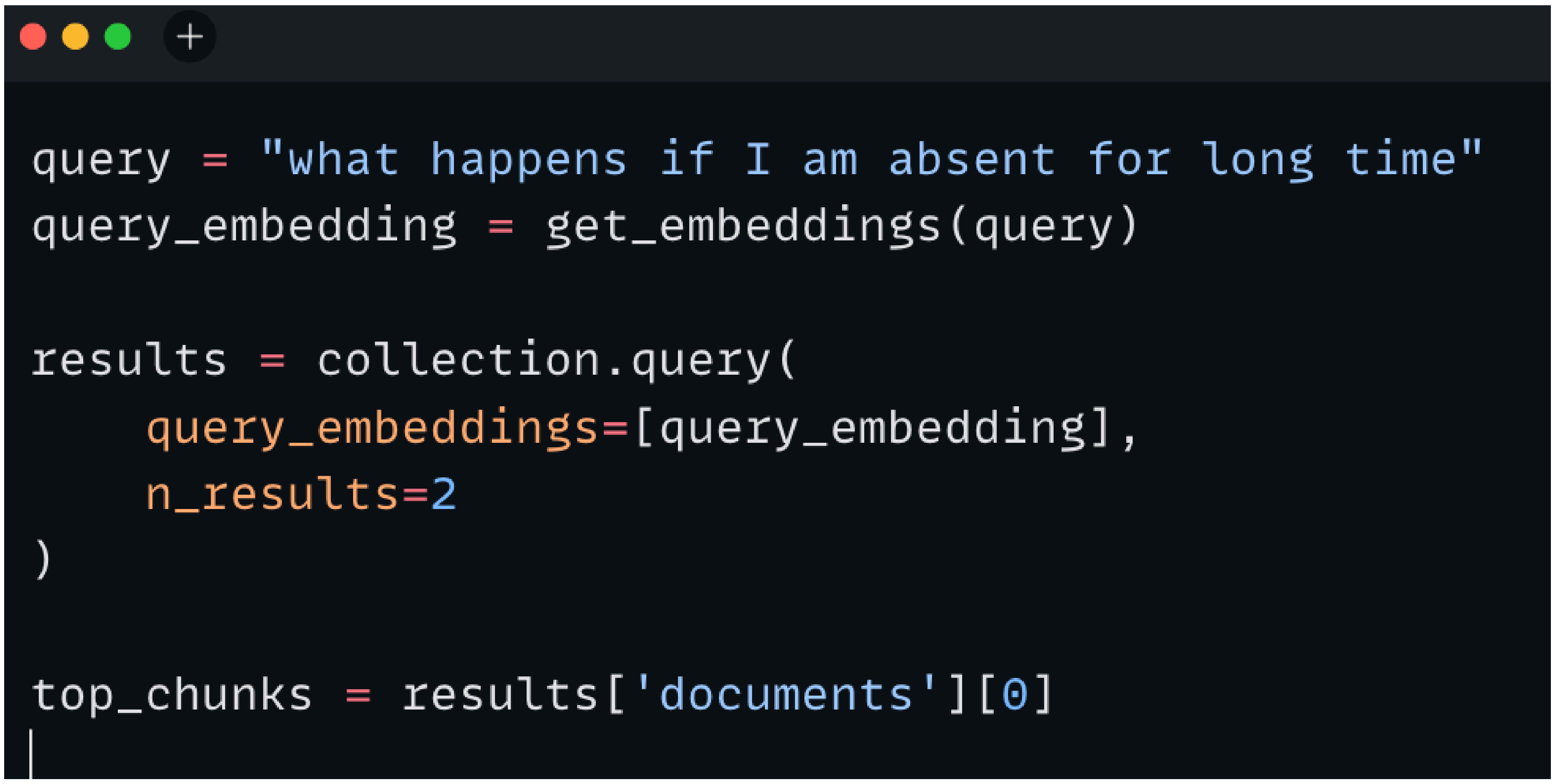
```python
import chromadb
from chromadb.config import Settings

chroma_client = chromadb.Client(Settings(persist_directory="./chroma_store"))
collection = chroma_client.get_or_create_collection(name="my_kb")

for i, chunk in enumerate(knowledge_chunks):
    collection.add(
        ids=[f"chunk-{i+1}"],
        documents=[chunk],
        embeddings=[get_embeddings(chunk)]
    )
```

RAG using Plain python

Step-6: Taking User Query and quering DB

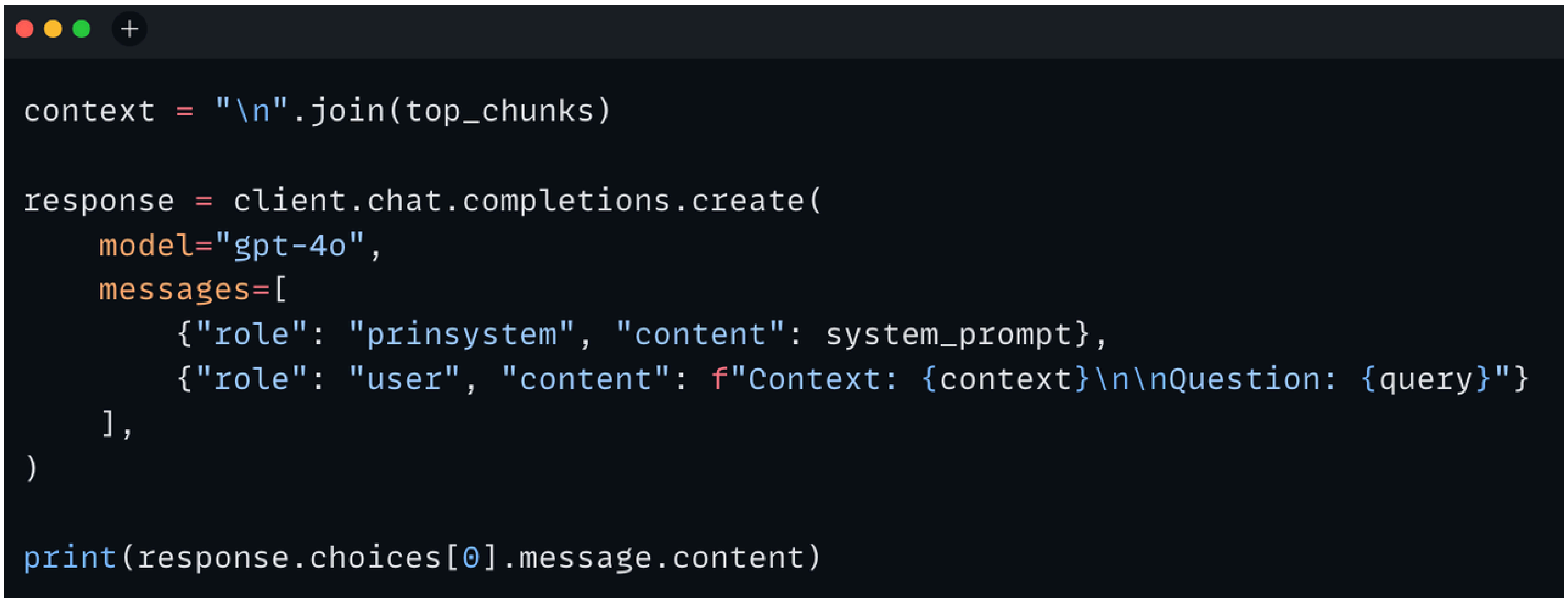


```python
query = "what happens if I am absent for long time"
query_embedding = get_embeddings(query)

results = collection.query(
    query_embeddings=[query_embedding],
    n_results=2
)

top_chunks = results['documents'][0]
```

Step-7: Final GPT Response



```python
context = "\n".join(top_chunks)

response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "prinsystem", "content": system_prompt},
        {"role": "user", "content": f"Context: {context}\n\nQuestion: {query}"}
    ],
)

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

File Search using plain python

Step-1: INSTALL THE SDK and imports

```python
from google import genai
from google.colab import userdata, files
from google.genai import types
import time
import os
```

Step-2: File uploading

```python
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
```

File Search using plain python

Step-3: CREATE THE FILE SEARCH STORE

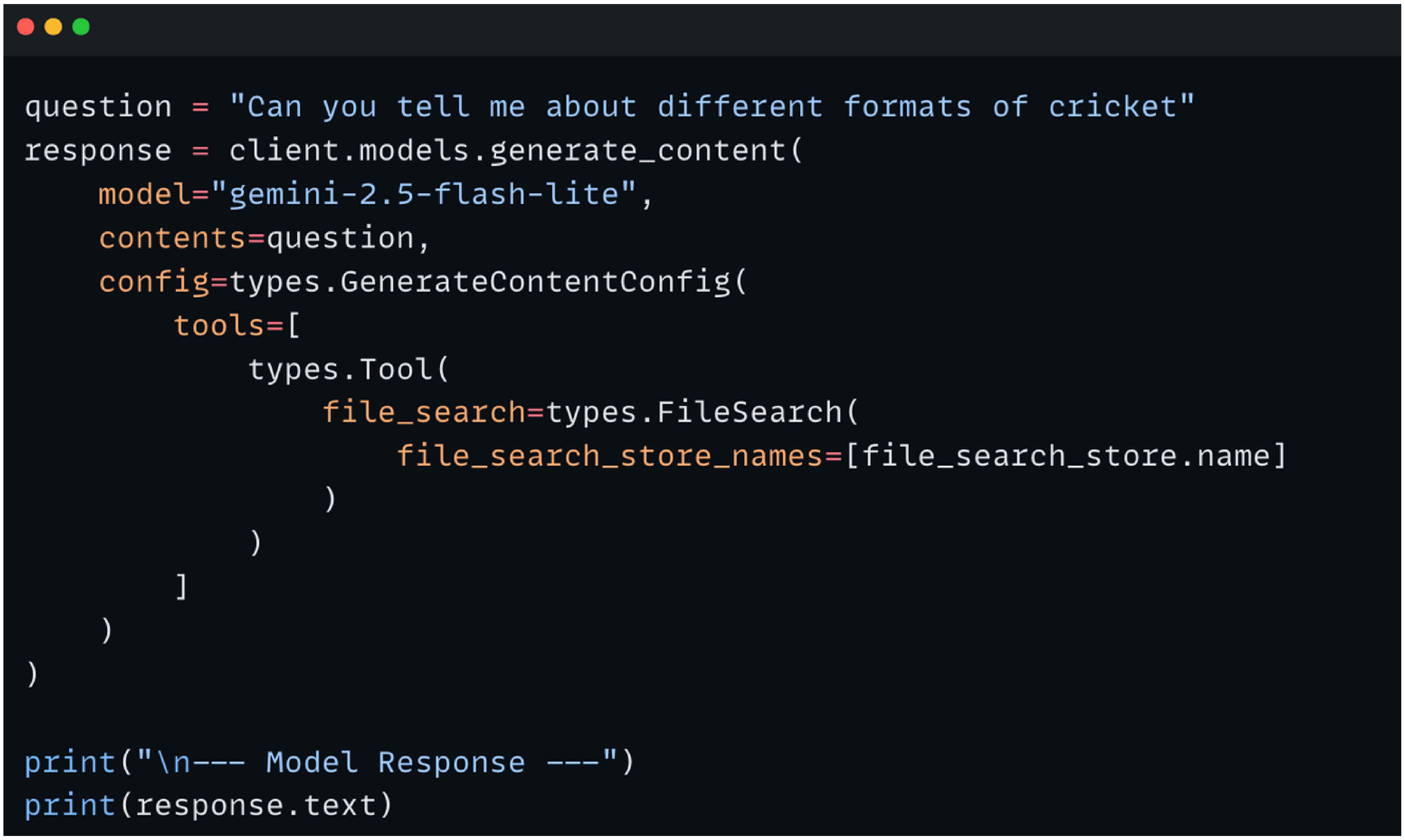
```python
store_display_name = 'Robert-Graves-Store'
file_search_store = client.file_search_stores.create(
    config={'display_name': store_display_name}
)
print(f"Store created with name: {file_search_store.name}")
```

Step-4: UPLOAD AND IMPORT THE FILE

```python
file_display_name = 'Graves-Biography'
operation = client.file_search_stores.upload_to_file_search_store(
    file=file_name,
    file_search_store_name=file_search_store.name,
    config={
        'display_name' : file_display_name,
    }
)
```

File Search using plain python

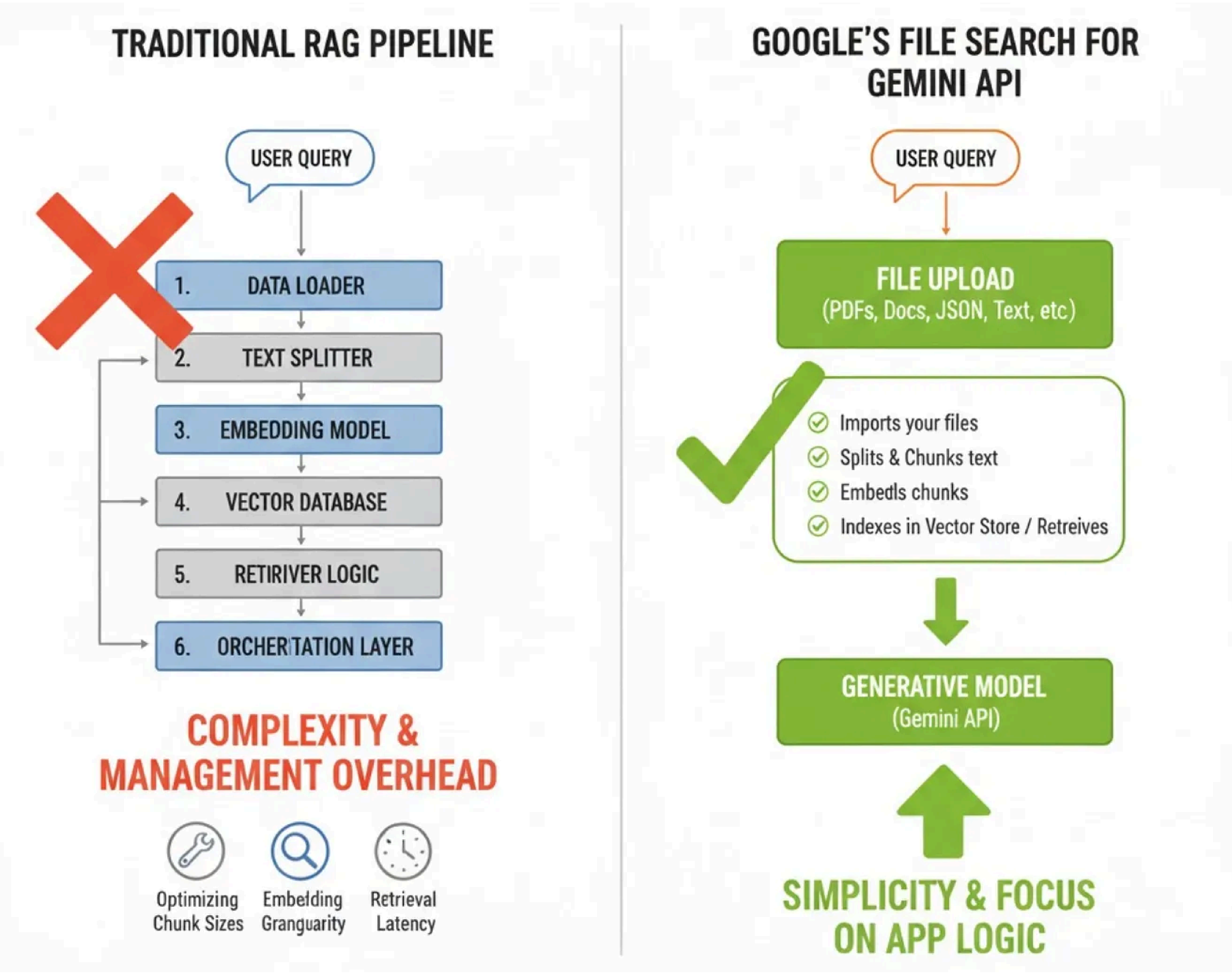
Step-5: ASK A QUESTION ABOUT THE FILE



```python
question = "Can you tell me about different formats of cricket"
response = client.models.generate_content(
    model="gemini-2.5-flash-lite",
    contents=question,
    config=types.GenerateContentConfig(
        tools=[
            types.Tool(
                file_search=types.FileSearch(
                    file_search_store_names=[file_search_store.name]
                )
            )
        ]
    )
)

print("\n--- Model Response ---")
print(response.text)
```

So which one is better?



- **Google's File Search eliminates steps 1 through 5 entirely.**
- **All we need to do is upload files. The system automatically handles everything else**

Advantages of Google File Search

1 Simplified Architecture

2 Cost and Time Efficiency

3 Better Contextual Accuracy

4 Seamless Integration

Want more content like this?

Tap that follow button and stay in the loop!

Like Comment Share Save