In [ ]:

In [1]:
```
import os, pickle, requests, pdfplumber
from dotenv import load_dotenv
from langchain.schema import Document
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.llms import OpenAI
from langchain.chains import RetrievalQA
import gradio as gr

load_dotenv()
API_KEY = os.getenv("OPENAI_API_KEY")
assert API_KEY, "Please set OPENAI_API_KEY in your .env file"
```

```
C:\Users\joshu\anaconda3\envs\rag-env\lib\site-packages\tqdm\auto.py:21: TqdmWarnin
g: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidget
s.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [ ]:

In [ ]:

In [2]:
```
# Cell 2 (simplified): Download one EMA PDF & convert to text

import os, requests, pdfplumber

os.makedirs("data", exist_ok=True)

# Only EMA Bioanalytical Method Validation
url = "https://www.ema.europa.eu/en/documents/scientific-guideline/guideline-bioana
r = requests.get(url, timeout=30)
r.raise_for_status()
pdf_path = "data/ema_bioanalytical_guideline.pdf"
with open(pdf_path, "wb") as f:
    f.write(r.content)

# Convert EMA PDF -> .txt
txt_path = pdf_path.replace(".pdf", ".txt")
pages = []
with pdfplumber.open(pdf_path) as pdf:
    for page in pdf.pages:
        pages.append(page.extract_text() or "")
with open(txt_path, "w", encoding="utf-8") as f:
    f.write("\n\n".join(pages))

print("✅ EMA guideline downloaded and converted to text in ./data/")
```

```
✅ EMA guideline downloaded and converted to text in ./data/
```

In [ ]:

In [ ]:

In [3]:
```python
from langchain.schema import Document
import os

def load_docs(folder="data"):
    docs = []
    for fn in os.listdir(folder):
        if not fn.endswith(".txt"):
            continue
        with open(os.path.join(folder, fn), encoding="utf-8") as f:
            txt = f.read()
        docs.append(Document(page_content=txt, metadata={"source": fn}))
    return docs

all_docs = load_docs("data")
print(f"🔍 Loaded {len(all_docs)} text docs for indexing.")
```

```
🔍 Loaded 1 text docs for indexing.
```

In [ ]:

In [ ]:

In [4]:
```python
# Cell 4: Build & persist FAISS index with a local HF embedder
!pip install --quiet sentence-transformers

from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import FAISS
import pickle

embedder = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
faiss_index = FAISS.from_documents(all_docs, embedder)

with open("faiss_index.pkl", "wb") as f:
    pickle.dump(faiss_index, f)

print("✅ FAISS index built locally (no OpenAI calls) and saved to faiss_index.pkl"
```

```
C:\Users\joshu\AppData\Local\Temp\ipykernel_3512\3554573147.py:8: LangChainDeprecati
onWarning: The class `HuggingFaceEmbeddings` was deprecated in LangChain 0.2.2 and w
ill be removed in 1.0. An updated version of the class exists in the :class:`~langch
ain-huggingface package and should be used instead. To use it run `pip install -U :c
lass:`~langchain-huggingface` and import as `from :class:`~langchain_huggingface imp
ort HuggingFaceEmbeddings``.
  embedder = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
✅ FAISS index built locally (no OpenAI calls) and saved to faiss_index.pkl
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [5]:
```python
# Cell 5: Wire up the RAG chain with GPT-3.5 using the v1.0+ API

from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA

# (If you restarted, reload the FAISS store:)
# with open("faiss_index.pkl", "rb") as f:
#     faiss_index = pickle.load(f)

llm = ChatOpenAI(
    model_name="gpt-3.5-turbo",
    temperature=0,
    openai_api_key=API_KEY
)
retriever = faiss_index.as_retriever(search_kwargs={"k": 5})

rag = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    return_source_documents=True
)

print("✅ RAG chain is ready!")
```

```
C:\Users\joshu\AppData\Local\Temp\ipykernel_3512\774492991.py:10: LangChainDeprecati
onWarning: The class `ChatOpenAI` was deprecated in LangChain 0.0.10 and will be rem
oved in 1.0. An updated version of the class exists in the :class:`~langchain-openai
package and should be used instead. To use it run `pip install -U :class:`~langchain
-openai` and import as `from :class:`~langchain_openai import ChatOpenAI``.
  llm = ChatOpenAI(
✅ RAG chain is ready!
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [6]:
```python
def answer_with_sources(question):
    res = rag({"query": question})
    answer = res["result"]
    sources = "\n".join(f"- {d.metadata['source']}" for d in res["source_documents"
```

```
    return f"**Answer:**\n{answer}\n\n**Sources:**\n{sources}"

iface = gr.Interface(
    fn=answer_with_sources,
    inputs=gr.Textbox(lines=2, placeholder="Ask medical questions…"),
    outputs="markdown",
    title="🩺 Medical RAG Demo",
    description="GPT-3.5 + EMA Bioanalytical Guideline"
)

iface.launch(share=True)
```

```
* Running on local URL:  http://127.0.0.1:7860
* Running on public URL: https://c73119fb403c441137.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run
`gradio deploy` from the terminal in the working directory to deploy to Hugging Face
Spaces (https://huggingface.co/spaces)
```

Medical RAG Demo

GPT-3.5 + EMA Bioanalytical Guideline

question

What parameters and acceptance criteria does the EMA Bioanalytical Method Validation guideline recommend for accuracy and precision?

Clear Submit

Answer: The EMA Bioanalytical Method Validation guideline recommends the following parameters and acceptance criteria for accuracy and precision:

1. **Accuracy:**
    - Within-run accuracy: The mean concentration should be within 15% of the nominal values for the QC samples, except for the LLOQ which should be within 20%.
    - Between-run accuracy: The mean concentration should be within 15% of the nominal values for the QC samples, except for the LLOQ which should be within 20%.
2. **Precision:**
    - Within-run precision: The coefficient of variation (CV) value should not exceed 15% for the QC samples, except for the LLOQ which should not exceed 20%.

Out[6]:

```
C:\Users\joshu\AppData\Local\Temp\ipykernel_3512\3418708132.py:2: LangChainDeprecati
onWarning: The method `Chain.__call__` was deprecated in langchain 0.1.0 and will be
removed in 1.0. Use :meth:`~invoke` instead.
  res = rag({"query": question})
```

In [ ]:

In [ ]:

In [ ]: