In [1]:
```
from PyPDF2 import PdfReader

def extract_handbook_chunks(pdf_path, chunk_size=800, overlap=100):
    reader = PdfReader(pdf_path)
    text = "\n".join(page.extract_text() for page in reader.pages)


    sections = [
        "PROGRAM OVERVIEW",
        "PROGRAM ADMISSION INFORMATION",
        "ELIGIBILITY",
        "APPLICATION",
        "PROGRAM REQUIREMENTS",
        "COMMON CORE COURSES",
        "CONCENTRATION",
        "CPT",
        "OPT",
        "THESIS",
        "CAPSTONE",
        "ACADEMIC PROBATION",
        "LEAVE OF ABSENCE",
        "FINANCIAL ASSISTANCE",
        "RESOURCES",
    ]

    chunks, metadata = [], []
    current_section = "General"

    for line in text.split("\n"):
        if any(sec in line.upper() for sec in sections):
            current_section = line.strip()
        if len(chunks) == 0 or len(chunks[-1]) > chunk_size:
            chunks.append(line)
            metadata.append({"section": current_section})
        else:
            chunks[-1] += " " + line
    return chunks, metadata


chunks, metadata = extract_handbook_chunks("2025-2026-DSAE-MS-Handbook.pdf")
print(f"Extracted {len(chunks)} chunks with metadata")
```

```
Extracted 87 chunks with metadata
```

In [2]:
```
import chromadb
from sentence_transformers import SentenceTransformer

client = chromadb.Client()
collection = client.create_collection("ds_handbook")

embedder = SentenceTransformer("all-MiniLM-L6-v2")

embeddings = embedder.encode(chunks).tolist()
collection.add(
    documents=chunks,
    embeddings=embeddings,
    metadatas=metadata,
    ids=[f"chunk_{i}" for i in range(len(chunks))]
)

print("DSAE Handbook indexed in ChromaDB")
```

```
DSAE Handbook indexed in ChromaDB
```

In [3]:
```
from rank_bm25 import BM25Okapi

tokenized_corpus = [c.split(" ") for c in chunks]
bm25 = BM25Okapi(tokenized_corpus)

def hybrid_retrieve(query, top_k=3):
    q_emb = embedder.encode([query]).tolist()
    dense_results = collection.query(query_embeddings=q_emb, n_results=top_k)

    bm25_scores = bm25.get_scores(query.split(" "))
    bm25_top = sorted(list(enumerate(bm25_scores)), key=lambda x: x[1], reverse=Tru
    bm25_docs = [chunks[idx] for idx, _ in bm25_top]

    docs = list(set(dense_results["documents"][0] + bm25_docs))
    return docs
```

In [4]:
```
import ollama

MODEL_NAME = "llama3.1"    #"mistral", "phi3", "gemma2:9b", "llama3.1"

def llama_answer(query, context):
    prompt = f"You are an ASU academic advising assistant.Use ONLY the context belc
    response = ollama.chat(model=MODEL_NAME, messages=[{"role": "user", "content":
    return response["message"]["content"]
```

In [5]:
```
def rag_chat(query):
    docs = hybrid_retrieve(query)
    context = "\n\n".join(docs)
    answer = llama_answer(query, context)
    print("Context Used:\n", context[:600], "...\n")
    print("Answer:\n", answer)
```

In [7]:
```
rag_chat("I have an internship offer for next summer, but the company wants me to s
```

```
Context Used:
 as a Job Seeker ” to become familiar with Principles for Prof
essional Practice. Students who accept an offer from an organiza
tion and later renege the offer will be prohibited from reques
ting future internship opportunities pending a meeting with the
Assistant Director. Required report A two-page typed minimum final
report is required before a grade and credit is given. The
final report must be submitted to the internship supervisor for
comments and then ...

Answer:
 No, you cannot start working until the summer semester officially starts (1st day
of classes). Exceptions are given to students who provide proper justification fro
m the company supporting this request.
```

In [8]:
```
# Notice that the agent is offering real advising guidance. It's not just quoting t
# It understands the situation described in the question, retrieves the relevant pc
# and then applies that policy to provide a clear and actionable answer.

# Because it is trained on the actual ASU student handbook, the guidance it provide
# is accurate and aligned with university policy — making it significantly more use
# than a standard FAQ chatbot.


# At this stage, the system works in a single-turn setting (ask → retrieve → answer
# A more advanced version is in development where I will add conversational memory,
# allowing follow-up questions and continuous advising sessions across multiple tur
```