```
# Reset pip state and install all required packages
!pip uninstall -y langchain langchain-core langchain-community langchain-text-splitters langchain-mistralai

# Install the latest stable compatible set
!pip install -U langchain langchain-community langchain-text-splitters langchain-mistralai bs4 requests gtts # Added gtts

# Packages for PDF and Web loading
!pip install -qU langchain-unstructured unstructured-client unstructured "unstructured[pdf]" python-magic
!pip install -qU langchain-huggingface
!pip install faiss-cpu
# Using specified stable versions for LangChain components
!pip install langchain==0.3.6 langchain-core==0.3.15 langchain-community==0.3.6

print(" All required packages installed.")
```

```
Found existing installation: langchain 0.3.27
Uninstalling langchain-0.3.27:
  Successfully uninstalled langchain-0.3.27
Found existing installation: langchain-core 0.3.79
Uninstalling langchain-core-0.3.79:
  Successfully uninstalled langchain-core-0.3.79
WARNING: Skipping langchain-community as it is not installed.
Found existing installation: langchain-text-splitters 0.3.11
Uninstalling langchain-text-splitters-0.3.11:
  Successfully uninstalled langchain-text-splitters-0.3.11
WARNING: Skipping langchain-mistralai as it is not installed.
Collecting langchain
  Downloading langchain-1.0.3-py3-none-any.whl.metadata (4.7 kB)
Collecting langchain-community
  Downloading langchain_community-0.4.1-py3-none-any.whl.metadata (3.0 kB)
Collecting langchain-text-splitters
  Downloading langchain_text_splitters-1.0.0-py3-none-any.whl.metadata (2.6 kB)
Collecting langchain-mistralai
  Downloading langchain_mistralai-1.0.1-py3-none-any.whl.metadata (1.9 kB)
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.4)
Collecting requests
  Downloading requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting gtts
  Downloading gTTS-2.5.4-py3-none-any.whl.metadata (4.1 kB)
Collecting langchain-core<2.0.0,>=1.0.0 (from langchain)
  Downloading langchain_core-1.0.2-py3-none-any.whl.metadata (3.5 kB)
Collecting langgraph<1.1.0,>=1.0.2 (from langchain)
  Downloading langgraph-1.0.2-py3-none-any.whl.metadata (7.4 kB)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.12/dist-packages (from langchain) (2.11.10)
Collecting langchain-classic<2.0.0,>=1.0.0 (from langchain-community)
  Downloading langchain_classic-1.0.0-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: SQLAlchemy<3.0.0,>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community)
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (6.0
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (3.1
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/lib/python3.12/dist-packages (from langchain-commu
Collecting dataclasses-json<0.7.0,>=0.6.7 (from langchain-community)
  Downloading dataclasses_json-0.6.7-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.10.1 in /usr/local/lib/python3.12/dist-packages (from langchain-comm
Requirement already satisfied: langsmith<1.0.0,>=0.1.125 in /usr/local/lib/python3.12/dist-packages (from langchain-community)
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (0
Requirement already satisfied: numpy>=1.26.2 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (2.0.2)
Requirement already satisfied: httpx<1.0.0,>=0.25.2 in /usr/local/lib/python3.12/dist-packages (from langchain-mistralai) (0.28
Requirement already satisfied: tokenizers<1.0.0,>=0.15.1 in /usr/local/lib/python3.12/dist-packages (from langchain-mistralai)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.12/dist-packages (from bs4) (4.13.5)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests) (2025.10.5)
Collecting click<8.2,>=7.1 (from gtts)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3-
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langcha
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langcl
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->lang
```

```
import getpass
import os
import faiss
import requests
from bs4 import BeautifulSoup
from collections import defaultdict
```

```
# LangChain and Core Imports
from langchain_mistralai.chat_models import ChatMistralAI
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_community.docstore.in_memory import InMemoryDocstore
from langchain_community.vectorstores import FAISS
from langchain_unstructured import UnstructuredLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain.tools import tool
from langchain_core.documents import Document
from langchain.agents import create_agent
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_core.messages import ToolMessage

# Voice Output Imports
from gtts import gTTS
from IPython.display import Audio, display

# Configuration
VOICE_OUTPUT_FILENAME = "agent_response_audio.mp3"
WEBSITE_SOURCE_NAME = "MPSTME_Website" # Distinct name for website source
PDF_SOURCE_NAME = "SRB_MPSTME.pdf"

# API Keys
# LangSmith
os.environ["LANGSMITH_TRACING"] = "true"
os.environ["LANGSMITH_API_KEY"] = getpass.getpass("Enter your LangSmith API Key (Optional, press Enter to skip): ")

# Mistral
if "MISTRAL_API_KEY" not in os.environ:
    os.environ["MISTRAL_API_KEY"] = getpass.getpass("Enter your Mistral API key: ")

# Unstructured (for PDF processing)
if "UNSTRUCTURED_API_KEY" not in os.environ:
    os.environ["UNSTRUCTURED_API_KEY"] = getpass.getpass("Enter your Unstructured API key: ")


# Model and Embeddings Initialization
model = ChatMistralAI(
    model="mistral-medium-latest",
    temperature=0.7,
)
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")

# FAISS Vector Store Initialization
embedding_dim = len(embeddings.embed_query("hello world"))
index = faiss.IndexFlatL2(embedding_dim)

vector_store = FAISS(
    embedding_function=embeddings,
    index=index,
    docstore=InMemoryDocstore(),
    index_to_docstore_id={},
)
print(" Imports, Keys, Models, and FAISS initialized.")
```

```
Enter your LangSmith API Key (Optional, press Enter to skip): ··········
Enter your Mistral API key: ··········
Enter your Unstructured API key: ··········
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%                                                  349/349 [00:00<00:00, 37.9kB/s]
config_sentence_transformers.json: 100%                                       116/116 [00:00<00:00, 13.6kB/s]
README.md:         11.6k/? [00:00<00:00, 1.21MB/s]
sentence_bert_config.json: 100%                                          53.0/53.0 [00:00<00:00, 4.89kB/s]
config.json: 100%                                                   571/571 [00:00<00:00, 59.7kB/s]
model.safetensors: 100%                                                438M/438M [00:03<00:00, 236MB/s]
tokenizer_config.json: 100%                                           363/363 [00:00<00:00, 21.1kB/s]
vocab.txt:         232k/? [00:00<00:00, 11.4MB/s]
tokenizer.json:         466k/? [00:00<00:00, 24.2MB/s]
special_tokens_map.json: 100%                                            239/239 [00:00<00:00, 26.0kB/s]
config.json: 100%                                                   190/190 [00:00<00:00, 18.7kB/s]
 Imports, Keys, Models, and FAISS initialized.
```

```
# Text Splitter Setup
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=200,
    add_start_index=True,
)

# Data Loading and Splitting (PDF)
file_paths = [f"/content/{PDF_SOURCE_NAME}"]
loader = UnstructuredLoader(file_paths)
docs = loader.load()

# Manually inject the clear filename metadata
for doc in docs:
    doc.metadata["filename"] = PDF_SOURCE_NAME
    doc.metadata["source"] = file_paths[0]

all_splits = text_splitter.split_documents(docs)
vector_store.add_documents(documents=all_splits)
print(f"✅ Loaded {len(all_splits)} chunks from PDF: {PDF_SOURCE_NAME}")


# Data Loading and Splitting (Website)
website_url = "https://engineering.nmims.edu/"
print(f"\nLoading content from college website: {website_url}")

try:
    response = requests.get(website_url, headers={"User-Agent": "Mozilla/5.0"}, timeout=15)
    response.raise_for_status()
    soup = BeautifulSoup(response.text, "html.parser")
    # Scrape relevant text content
    content_divs = soup.find_all(["div", "article"], class_=["content", "main-content", "post-content", "entry-content"])
    text = " ".join(div.get_text(separator="\n", strip=True) for div in content_divs) if content_divs else soup.body.get_text(s

    if not text.strip():
        raise ValueError("Manual scraping resulted in empty content.")

    # Assign clear website metadata
    web_doc = Document(page_content=text, metadata={"source": website_url, "filename": WEBSITE_SOURCE_NAME})
    web_splits = text_splitter.split_documents([web_doc])
    vector_store.add_documents(documents=web_splits)
    print(f" Loaded {len(web_splits)} chunks from Website: {WEBSITE_SOURCE_NAME}")

except Exception as e:
    print(f" Could not load or process website content: {e}. Continuing with PDF-only data.")
```

```
Warning: No languages specified, defaulting to English.
✅ Loaded 1444 chunks from PDF: SRB_MPSTME.pdf

Loading content from college website: https://engineering.nmims.edu/
 Loaded 9 chunks from Website: MPSTME_Website
```

```
# --- Tool Definition (UPDATED for Objective Score) ---
@tool(response_format="content_and_artifact")
def retrieve_context(query: str):
    """Retrieve information to help answer a query."""

    # FIX: Use similarity_search_with_score and increase k to 5
    retrieved_results: list[tuple[Document, float]] = vector_store.similarity_search_with_score(query, k=5)

    # Separate documents and scores
    retrieved_docs = [doc for doc, score in retrieved_results]
    scores = [score for doc, score in retrieved_results]

    # Calculate the objective confidence metric (Average L2 Distance)
    # Lower L2 distance means higher semantic similarity (higher confidence)
    avg_l2_distance = sum(scores) / len(scores) if scores else 0.0

    # Serialize the full content for the LLM to read and use
    serialized = "\n\n".join(
        (
            f"Source: {doc.metadata.get('filename', doc.metadata.get('source', 'Unknown'))} (Chunk ID: {i})\n"
            f"Content: {doc.page_content}"
        )
        for i, doc in enumerate(retrieved_docs, 1)
    )

    # UPDATED ARTIFACT: Return the documents and the calculated metric
    artifact = {
        "retrieved_docs": retrieved_docs,
        "avg_l2_distance": avg_l2_distance
    }

    # Pass the metric to the LLM via the serialized text for visibility
    serialized_with_score = f"**RETRIEVAL METRIC: Average L2 Distance = {avg_l2_distance:.4f}**\n\n" + serialized

    return serialized_with_score, artifact

# --- System Prompt Revision (UPDATED for Objective Score) ---
system_prompt = (
    "You are a helpful assistant with access to a tool that retrieves context from a student resource book PDF and a college we
    "The tool provides a **RETRIEVAL METRIC** which is the **Average L2 Distance** of the retrieved chunks. **LOWER DISTANCE me
    "Use this context to answer user queries thoroughly and accurately with detail. "
    "**Start your final answer with the information derived from the context.** "
    "**Your final answer MUST conclude with a section labeled 'Confidence Score' on a new line, displaying the Average L2 Dista
    "**Do NOT include the citation chunk in your response; that will be handled externally.**"
)

# --- Agent Creation ---
tools = [retrieve_context]
prompt = ChatPromptTemplate.from_messages([
    ("system", system_prompt),
    MessagesPlaceholder(variable_name="messages"),
    MessagesPlaceholder(variable_name="agent_scratchpad"),
])
model_with_prompt = model.bind_tools(tools)
agent = create_agent(model_with_prompt, tools)

print("✅ Agent created successfully.")
```

```
✅ Agent created successfully.
```

```
# --- 1. Manual Query Input ---
query = input("Enter your query for the RAG Agent: ")

# Run the agent
result = agent.invoke({
    "messages": [{"role": "user", "content": query}],
    "intermediate_steps": []
})

print("\n--- Agent Execution Complete ---")
```

```
# --- 2. Final Answer and Citation Extraction Logic  ---
final_answer_content = "Could not retrieve a final answer."
citation_sources = set()
avg_l2_distance = None # NEW: Variable to store the objective score

if isinstance(result, dict) and 'messages' in result:
    final_ai_message = result['messages'][-1]
    if hasattr(final_ai_message, 'content'):
        final_answer_content = final_ai_message.content

    # Extract ALL unique retrieved documents and the score from the ToolMessage artifact
    for message in reversed(result['messages']):
        if isinstance(message, ToolMessage) and hasattr(message, 'artifact') and message.artifact:
            artifact = message.artifact

            # FIX: Extract the objective metric
            if isinstance(artifact, dict) and 'avg_l2_distance' in artifact:
                avg_l2_distance = artifact['avg_l2_distance']

            # Extract retrieved documents for citation
            if isinstance(artifact, dict) and 'retrieved_docs' in artifact:
                retrieved_docs: list[Document] = artifact['retrieved_docs']

                # Use a set to collect only the unique file/source names
                for doc in retrieved_docs:
                    # Use the clean 'filename' metadata (WEBSITE_SOURCE_NAME or PDF_SOURCE_NAME)
                    source_name = doc.metadata.get('filename', doc.metadata.get('source', 'Unknown Source'))
                    citation_sources.add(source_name)
            break

# --- 3. Final Formatted Output Display ---
print("\n Agent Final Answer (Formatted):")
# Print the LLM's full response (Answer + Confidence Score)
print(final_answer_content)

# FIX: Add the objective Confidence Score block separately
if avg_l2_distance is not None:
    print("\n---")
    print("**Objective Retrieval Metric (Semantic Confidence):**")
    print(f"Average L2 Distance (Lower score = Higher Confidence): {avg_l2_distance:.4f}")

# Add the Citation Source block reliably
if citation_sources:
    print("\n---")
    print("**Citation Sources (Unique Files/Websites Used):**")
    for source in sorted(list(citation_sources)):
        print(f"* {source}")


# --- 4. Voice Output Integration ---
try:
    # Remove the Confidence Score line and any trailing separators from the text for cleaner audio
    text_to_speak = final_answer_content.split('Confidence Score')[0].strip()

    if text_to_speak:
        print("\n---")
        print("🔊 Generating voice output...")
        tts = gTTS(text=text_to_speak, lang='en', slow=False)
        # Ensure the output file is clean
        if os.path.exists(VOICE_OUTPUT_FILENAME):
            os.remove(VOICE_OUTPUT_FILENAME)
        tts.save(VOICE_OUTPUT_FILENAME)
        # Display the audio player for playback
        display(Audio(VOICE_OUTPUT_FILENAME, autoplay=True))
    else:
        print("\n Voice output skipped: Final answer content was empty or only contained the confidence score.")
except Exception as e:
    print(f"\n Error generating or playing voice output: {e}")

print("\n--- End of Response ---")
```

```
--- Agent Execution Complete ---

 Agent Final Answer (Formatted):
Here are the detailed **library rules and regulations** based on the provided information:

---

### **Library Rules and Regulations**

#### **1. General Rules**
1. **Compliance with Rules**:
   - Use of the library is conditional on observing the **Rules and Regulations**.
   - Users must comply with these rules and any reasonable request or instruction issued by library staff.
   - Failure to comply may result in exclusion from the library, fines, or disciplinary action.
   - The Librarian reserves the right to refer any breaches of rules or improper behavior to the appropriate disciplinary commit

2. **Access Restrictions**:
   - Access to the library is restricted to **staff and students** of the institution.
   - Users must have a **valid ID card** to enter the library.

---

#### **2. Conduct Inside the Library**
3. **Silence and Mobile Phones**:
   - **Silence must be maintained** in all library areas.
   - The use of **mobile phones** is strictly prohibited. Phones must be switched off or set to silent mode.
   - Violations may result in a **fine and/or exclusion** from the library for up to **three weeks**.

4. **Personal Belongings**:
   - **Bags and personal belongings** are not allowed inside the library.
   - Users must store their belongings in designated areas.
   - Unattended items may be removed by library staff.

---

#### **3. Use of Library Resources**
5. **Photography and Recording**:
   - **Photography, filming, videotaping, and audio-taping** are **not allowed** in the library without prior permission from th

6. **Personal Equipment**:
   - Personal equipment (e.g., laptops, cameras) may only be used with the **prior permission** of the Librarian.

7. **Copyright Compliance**:
   - Users must comply with **copyright regulations** when using photocopiers or other library resources.

---

#### **4. Electronic Resources**
8. **Use of Electronic Data**:
   - Data retrieved from the library's electronic resources may only be used for:
     - Teaching
     - Research
     - Personal educational development
     - Administration and management of the institution
   - **Prohibited Uses**:
     - Commercial exploitation of information.
     - Consultancy or services leading to financial gain.
     - Work benefiting external employers (e.g., industrial placements or part-time courses).
   - Users must comply with the **specific requirements of individual data providers**.
   - **Passwords must never be shared** with others.

---

#### **5. Penalties for Violations**
- Failure to comply with these rules may result in:
  - **Fines**
  - **Exclusion from the library**
  - **Disciplinary action** under institutional policies

---

---
**Objective Retrieval Metric (Semantic Confidence):**
Average L2 Distance (Lower score = Higher Confidence): 0.6114

---
**Citation Sources (Unique Files/Websites Used):**
* SRB_MPSTME.pdf

---
 Generating voice output...
```