Words:  
import logging

from typing import Optional, List

from datetime import datetime

from langchain\_core.prompts import PromptTemplate

from models import ChatMessage

logger = logging.getLogger(\_\_name\_\_)

# --- History Formatting Helper ---

def format\_chat\_history(history: Optional[List[ChatMessage]], max\_turns: int = 5) -> str:

"""

Formats the chat history into a string for the LLM prompt.

Limits history to the last max\_turns user/AI pairs.

"""

if not history:

return "No previous conversation history provided."

max\_messages = max\_turns \* 2

start\_index = max(0, len(history) - max\_messages)

recent\_history = history[start\_index:]

formatted\_history = []

for msg in recent\_history:

sender = "User" if msg.sender == "user" else "AI"

formatted\_history.append(f"{sender}: {msg.text}")

if not formatted\_history:

return "No recent conversation history to display."

return "\n".join(formatted\_history)

# --- RAG PROMPTS ---

# 1) Template when NO history is present

RAG\_PROMPT\_TEMPLATE\_NO\_HISTORY = """

[SYSTEM ROLE INSTRUCTIONS]

You are an advanced, retrieval-augmented AI assistant.

You have access to a 'Context' containing information relevant to the user's query.

Answer the user's query as helpfully and accurately as possible, strictly using the provided Context.

If the answer is not in the Context or is incomplete, clearly say: "I don't have enough information to answer."

[CONTEXT]

{context}

[USER QUERY]

{question}

[ASSISTANT INSTRUCTIONS]

1. Read the context carefully.

2. Do NOT invent or fabricate details that are not present in the context.

3. If you do not have sufficient details from the context, say so.

[FINAL ANSWER]:

"""

# 2) Template when history IS present

RAG\_PROMPT\_TEMPLATE\_WITH\_HISTORY = """

[SYSTEM ROLE INSTRUCTIONS]

You are an advanced, retrieval-augmented AI assistant.

You have access to a 'Context' and a history of the conversation.

Use them both to answer the user’s query in a helpful, accurate manner.

If the answer is not in the context or is incomplete, clearly say: "I don't have enough information to answer."

[CHAT HISTORY]

{chat\_history}

[CONTEXT]

{context}

[USER QUERY]

{question}

[ASSISTANT INSTRUCTIONS]

1. Use prior conversation history for additional background or clarifications.

2. Reference the context for verifiable information. Do not fabricate details or references.

3. If insufficient details are provided, explicitly state that you do not have enough information.

[FINAL ANSWER]:

"""

def create\_rag\_prompt\_template(

history: Optional[List[ChatMessage]] = None

) -> PromptTemplate:

"""

Creates the appropriate PromptTemplate based on whether history is present,

following the refined RAG prompts above.

"""

if history and len(history) > 0:

logger.debug("Creating RAG PromptTemplate with history.")

return PromptTemplate(

template=RAG\_PROMPT\_TEMPLATE\_WITH\_HISTORY,

input\_variables=["context", "question", "chat\_history"]

)

else:

logger.debug("Creating RAG PromptTemplate without history.")

return PromptTemplate(

template=RAG\_PROMPT\_TEMPLATE\_NO\_HISTORY,

input\_variables=["context", "question"]

)

# --- FALLBACK PROMPTS ---

FALLBACK\_PROMPT\_TEMPLATE = """

[ROLE: SYSTEM]

You are an AI assistant with broad general knowledge but no specific retrieved context.

[USER QUERY]

{question}

[ASSISTANT INSTRUCTIONS]

1. Answer using your general knowledge and reasoning.

2. If the question seems to require specific context not provided, say:

"I cannot answer accurately without more specific information."

[FINAL ANSWER]:

"""

FALLBACK\_PROMPT = PromptTemplate.from\_template(FALLBACK\_PROMPT\_TEMPLATE)

FALLBACK\_PROMPT\_WITH\_HISTORY\_TEMPLATE = """

[ROLE: SYSTEM]

You are an AI assistant with broad general knowledge. You also have access to the previous chat history.

[CHAT HISTORY]

{chat\_history}

[USER QUERY]

{question}

[ASSISTANT INSTRUCTIONS]

1. Use the conversation history to maintain context and coherence.

2. If the query is highly specific and requires external references not provided, explicitly state that the context is insufficient.

[FINAL ANSWER]:

"""

FALLBACK\_PROMPT\_WITH\_HISTORY = PromptTemplate.from\_template(FALLBACK\_PROMPT\_WITH\_HISTORY\_TEMPLATE)

# --- ANSWER SUFFICIENCY CHECK PROMPT ---

ANSWER\_SUFFICIENCY\_CHECK\_PROMPT\_TEMPLATE = """

[ROLE: SYSTEM]

You are tasked with determining whether an answer derived from a given context is sufficient.

[CONTEXT]

{context}

[QUERY]

{question}

[ANSWER FROM CONTEXT]

{answer\_from\_context}

[INSTRUCTIONS]

1. Determine if the "Answer from Context" fully addresses the "Query" based solely on the "Context."

2. If it does, respond with "SUFFICIENT."

3. If the provided context is incomplete or the answer admits uncertainty, respond with "INSUFFICIENT."

[DECISION]:

"""

ANSWER\_SUFFICIENCY\_CHECK\_PROMPT = PromptTemplate.from\_template(ANSWER\_SUFFICIENCY\_CHECK\_PROMPT\_TEMPLATE)