Chat History

```
from langchain_community.chat_message_histories import ChatMessageHistory

from langchain_core.chat_history import BaseChatMessageHistory

from langchain_core.runnables.history import RunnableWithMessageHistory
```

These imports bring in three important LangChain components:

1. **ChatMessageHistory** –
   A simple implementation that stores a sequence of messages (from user and AI).
   It behaves like an in-memory chat log.
2. **BaseChatMessageHistory** –
   An **abstract base class** (interface) that defines what a "chat message history" must look like.
   Any class that stores messages for a session must implement this interface.
3. **RunnableWithMessageHistory** –
   A **LangChain wrapper** that combines an LLM (or any Runnable) with a message history.
   It automatically tracks the conversation over multiple interactions.

The message history store

1. Creating a **dictionary** to store all session histories.
2. Each user/session will have its own chat history stored in memory.

```
store = {
   "session_1": ChatMessageHistory(),
   "session_2": ChatMessageHistory(),
}
```

3. For reusability – create one user defined function – fetch/create session history

```
def get_session_history(session_id: str) -> BaseChatMessageHistory:
   if session_id not in store:
      store[session_id] = ChatMessageHistory()
   return store[session_id]
```

This fuction ensures that:

- Each unique session_id gets its own ChatMessageHistory object.
- If a session doesn't exist yet, it creates one.
- It returns the ChatMessageHistory object for the given session.

So if two different users are chatting, they won't mix messages:

```
get_session_history("user_123")  # returns one history
```

```
get_session_history("user_456")  # returns another
```

4. Attaching message history to an LLM

```
with_message_history = RunnableWithMessageHistory(llm_obj, get_session_history)
```

This is the key part.

- llm_obj → your actual **LLM instance** (like ChatOpenAI, ChatAnthropic, etc.).
- get_session_history → your function that retrieves the right chat history for a given session.

5. What RunnableWithMessageHistory does:
   It **wraps** your LLM so that every time you call it with a specific session_id, it:
   - Retrieves the chat history (via your get_session_history function).
   - Passes that history to the LLM (so it can respond in context).
   - Updates the chat history with the new user + AI messages automatically.

```
response = with_message_history. invoke (
    {"messages": [{"role": "user", "content": "Hello!"}]},
    config={"configurable": {"session_id": "123"}}
)
```

LangChain:

- Calls get_session_history("123")
- Adds "Hello!" to that session's message log
- Sends the whole conversation to the LLM
- Stores the LLM's response back in the same history

Now when the same session_id "123" talks again, the model sees all previous messages.