MCP Workshop: Exercise 1

Relevant File: src/main.py

Clone the repo from https://github.com/DiegoLigtenberg/workshop-langgraph-mcp

This repo contains an agent in src/main.py. Please get this file running using poetry, docker or podman (see README.MD). The code hosts a web-based chat interface that connects to local and external mcp servers. These mcp servers will run code that exists locally from a filepath (src/local_mcp_servers), or by using a uv package install (in the case for word-mcp).

The package that is installed for the word-mcp can be found on this Github page:
https://github.com/GongRzhe/Office-Word-MCP-Server

These following exercises provide a way to familiarize yourself with chat interfaces with (external) MCP servers. Please start with **exercise 1**. Here you chat with a pre-made chat interface connected to the Vibify MCP server.

From **exercise 2** onwards, you run your own chat interface connected to local/word-MCP server(s).
Accesses to the FastAPI web interface is done at http://localhost:8000/chat or https://127.0.0.1:8000/chat.

Exercise 1: This introduction exercise shows a chat interface connnected to the Vibify website with a Vibify-specific Mcp server. Go to mcp-workshop- server.up.railway.app. To get familiar with the application. Ask the chatbot to:
1) Name a song. 2) What song is streamed the most. 3) Can you increase the streamcount of this song by 10.
Why do you think question 3 is not possible? (hint: what access would you want your mcp to have to your database)

Exercise 2.a: Go to src/main.py. Setup the environment using poetry/docker/podman. Run the file and view the tool calls that are loaded from the Word-mcp server. Check the console output when the server starts and scan the list of all the tools available from the office-word-mcp-server.
*1) How many **tools** are loaded 2) Do you think the LLM has enough context to know the order of tools to call?*

Exercise 2.b: *How could we guide the LLM to have a better understanding of when to use what tool? Hint: system prompt*

Exercise 3: Try to make a query that combines the local weather server and math server (src/local_mcp_servers) and creates a word document with the answer of your query. Make sure to give the word document a name.docx.
For example: "What's the weather in Paris and what's 15 * 7?"
"Create a word document called results.docx with both answers."
What tools did the program use?

Exercise 4: Compare the server configuration for local servers vs external packages.
Where is the uv package installed? How can you make sure to trust the code of the downloaded MCP servers?

Bonus: *Can you come up with a way to create a Sopra Steria business proposal word-document (with clear paragraphs, a table, a conclusion and a reference list)?*

NOTE: you may need to remove the truncate_messages_safely() function as this can require quite some memory