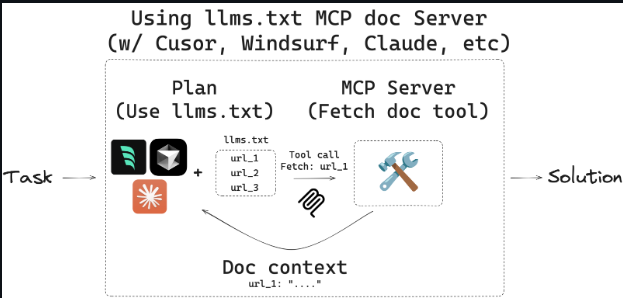
**MCP docs**

1. <https://github.com/langchain-ai/mcpdoc>
2. Its an MCP Server created by Langchain team.
3. It's going to leverage llms.txt of public documentation of packages, for example, LangGraph documentation and it's going to help our agent, whether it's Cursor, Windsurf or Claude desktop, going to give them the ability to fetch the most up-to-date documentation. So this actually gives us the ability to fetch them dynamically because we're going to scrape them from their official website, which is supposed to be up to date.
4. 
5. So first it has access to this llm.txt tool, which has a bunch of URLs and explanation of what do those URL represent, which topic do they cover. So, this is the first step of the MCP server. The second step is to figure out which URL do we need according to the user's question and then to fetch that relevant information by simply making a curl request to a ‘fetch\_docs ’ tool.

**Installation and Running** the mcpdoc server

1. Download UV
2. Clone MCP Github repo
3. Install all dependencies available in UV.log file
4. Test the MCP server locally with your llms.txt file. (we have langgraph llm.txt). It should run the server on port 8082.
5. Run MCP inspector
6. In inspector, connect to SSE server running on port 8082 and check out the tools that they MCP server exposes.
7. We have here two tools, list doc sources, which is going to show us what is the URL to the llm.txt file. So, we can then make HTTP request and fetch the information and scrape that page.
8. Other tool is fetch docs. With the llm.txt as input, it is going to receive a URL and it's going to retrieve all the content from that URL by scraping it. And what we can see here is the content of the LangGraph llm.txt file that we scraped with the fetch\_doc store tool. And we can see here all the URLs from the llm.txt and the agent is going to then extract those URLs and it's going to run the fetch\_docs tools for those URLs to get the relevant documentation dynamically.
9. Now, integrate everything with Claude desktop. Go to setting and configure MCP server. In configuration, the URLs that are going to store the llm.txt are going to be for the LangGraph documentation. Transport layer is STDIO (not SSE), port 8081, specify the absolute path to where the code is stored and restart claude desktop.
10. Now when we search as user “what is langgraph memory?”, it activate the tools.
11. The agent is trying to run the list doc sources tool. And because we initialized this code with the LangGraph documentation (the built-in URL when we initialize the server), we are not running it with any arguments and we can see now the results that we got from this tool - we got that the LangGraph URL, the URL to the llm.txt is this here URL. And now the next tool that should run and is going to be to scrape that URL data.
12. Now we're trying to fetch the documentation with the URL we got from the previous tool. So the URL we're passing into this tool is going to be the LangGraph llm.txt and we're going to scrape the information from there and we're going to see all the information from the llm.txt like content of the llm.txt, topics and the URLs for those topics.
13. The next step that the agent is going to do, it's going to find the relevant URL that we need to scrape. So the URL that talks about memory and it's going to scrape that URL again. It invokes now the fetch docs tool again, however, with a different URL and we are going to scrape now the langgraph/concepts/memory.
14. So, let's go and allow it and get the content, real-time content, this time of memory. So now it's going to fetch the information and here we have now the summary of what's LangGraph memory, however, this time it's grounded in real time information from the official LangGraph documentation.