# HW4: LLMs, RAG :)

## Summary

In this HW, you will:

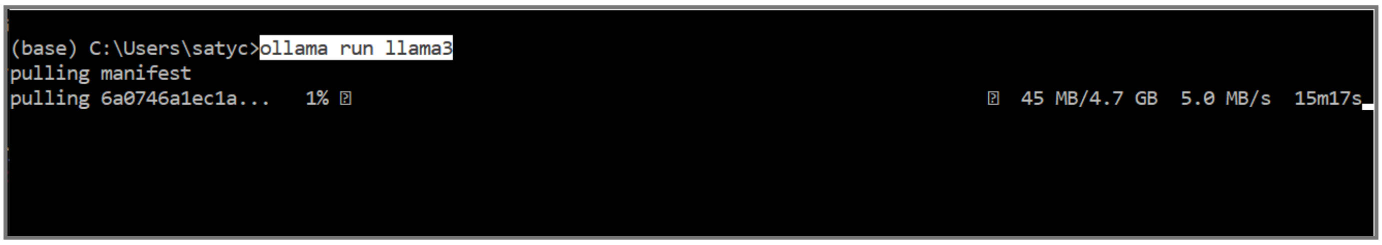
- use 'ollama' to invoke LLMs locally (in your laptop)
- use DataStax's LangFlow visual dataflow system, to do RAG
- use Claude Desktop and MCP to do RAG on local files

Fun!!

---

**Q1.** (2 points)

Install Ollama from https://ollama.com/ - it lets you download and run LLMs locally! After it's installed, do this [to install Meta's llama3 model] - **ollama run llama3**:

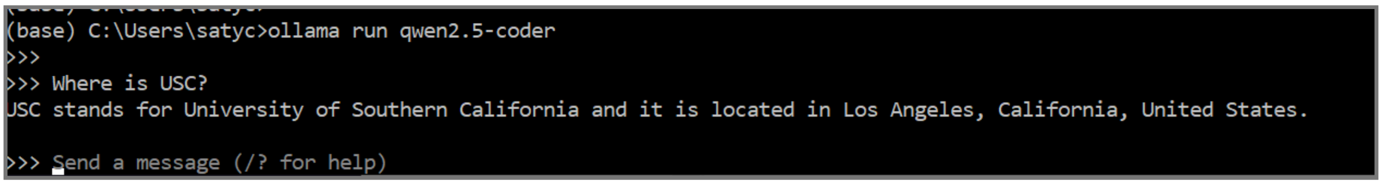


```
(base) C:\Users\satyc>ollama run llama3
pulling manifest
pulling 6a0746a1ec1a...   1% ▯                                                  ▯  45 MB/4.7 GB  5.0 MB/s  15m17s
```

Similarly, get 'gemma2:2b' [do 'ollama run' or 'ollama pull' on it].

After both models are installed, you are ready to start sending prompts to them :)

When you run a model locally, you can prompt it too, like so:



```
(base) C:\Users\satyc>ollama run qwen2.5-coder
>>>
>>> Where is USC?
USC stands for University of Southern California and it is located in Los Angeles, California, United States.

>>> Send a message (/? for help)
```

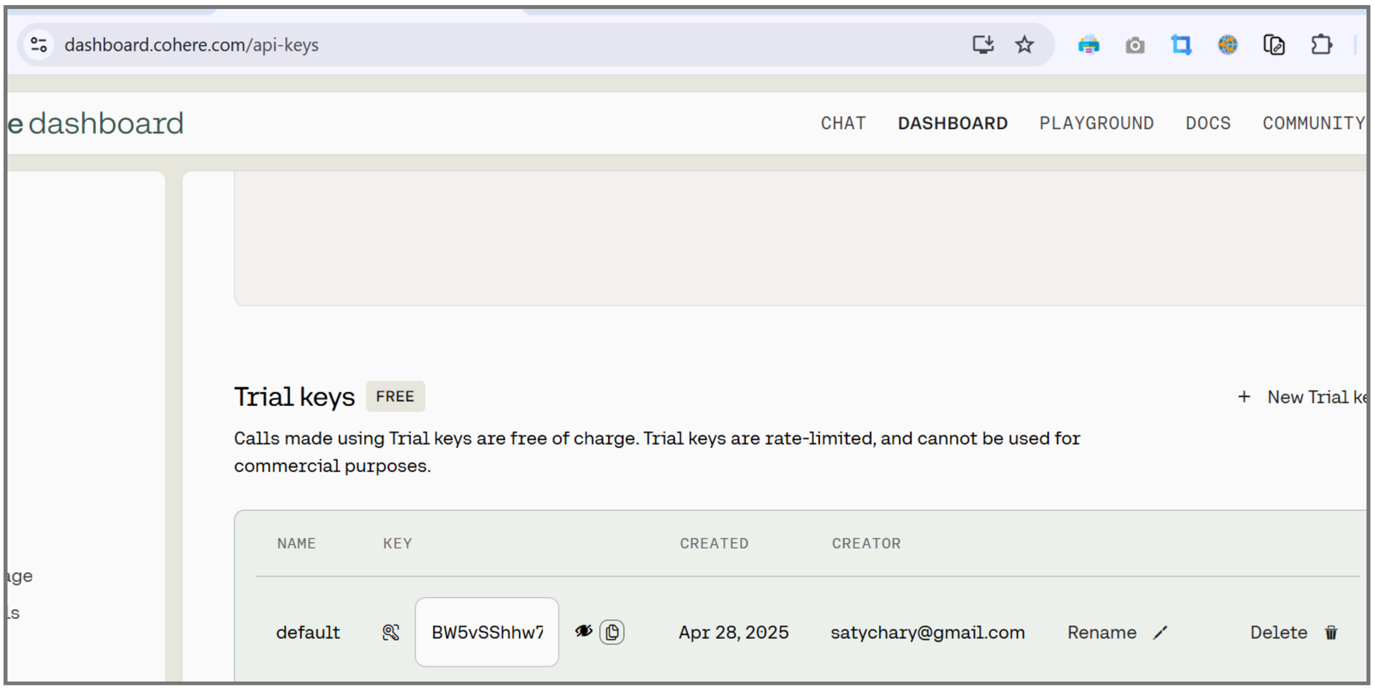
Search for two facts on the command line like shown above, get screenshots.

---

## **Q2.** (4 points)

Now we will switch gears, and do visual dataflow inside the browser :)

Sign up for a free DataStax account at https://astra.datastax.com/signup . Also, get (free) API keys at HuggingFace [look in 'Settings' under your profile on the right], and Cohere (be sure to get the free 'Trial Keys' one):

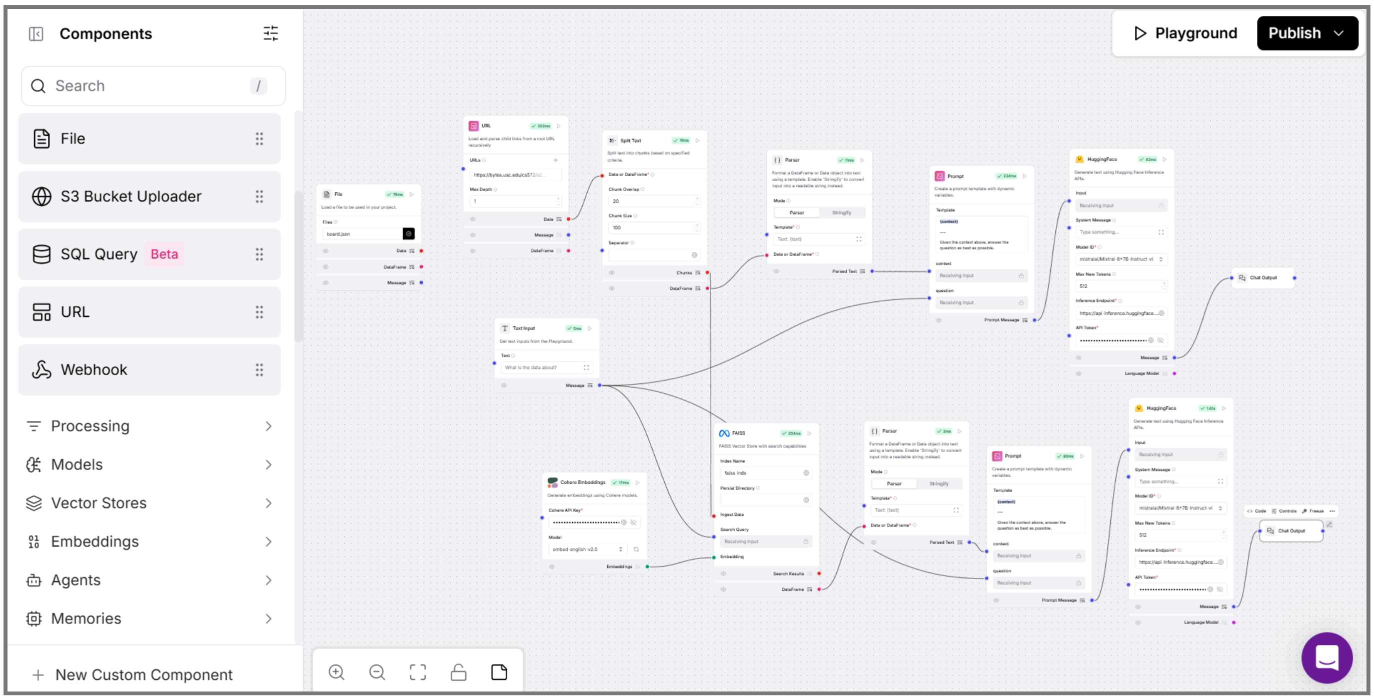


Trial keys FREE

Calls made using Trial keys are free of charge. Trial keys are rate-limited, and cannot be used for commercial purposes.

| NAME | KEY | CREATED | CREATOR | | |
|---|---|---|---|---|---|
| default | BW5vSShhw7 | Apr 28, 2025 | satychary@gmail.com | Rename | Delete |

Start a new 'flow' in DataStax LangFlow - you can start with any template, eg. 'Vector RAG' - but DELETE all the nodes in there, because you will need to replicate the graphs you see in the screenshot below, by populating a blank area with nodes and connections. Have fun dragging and dropping nodes and connecting them, to get (recreate) these two graphs (one at the top, one at the bottom):

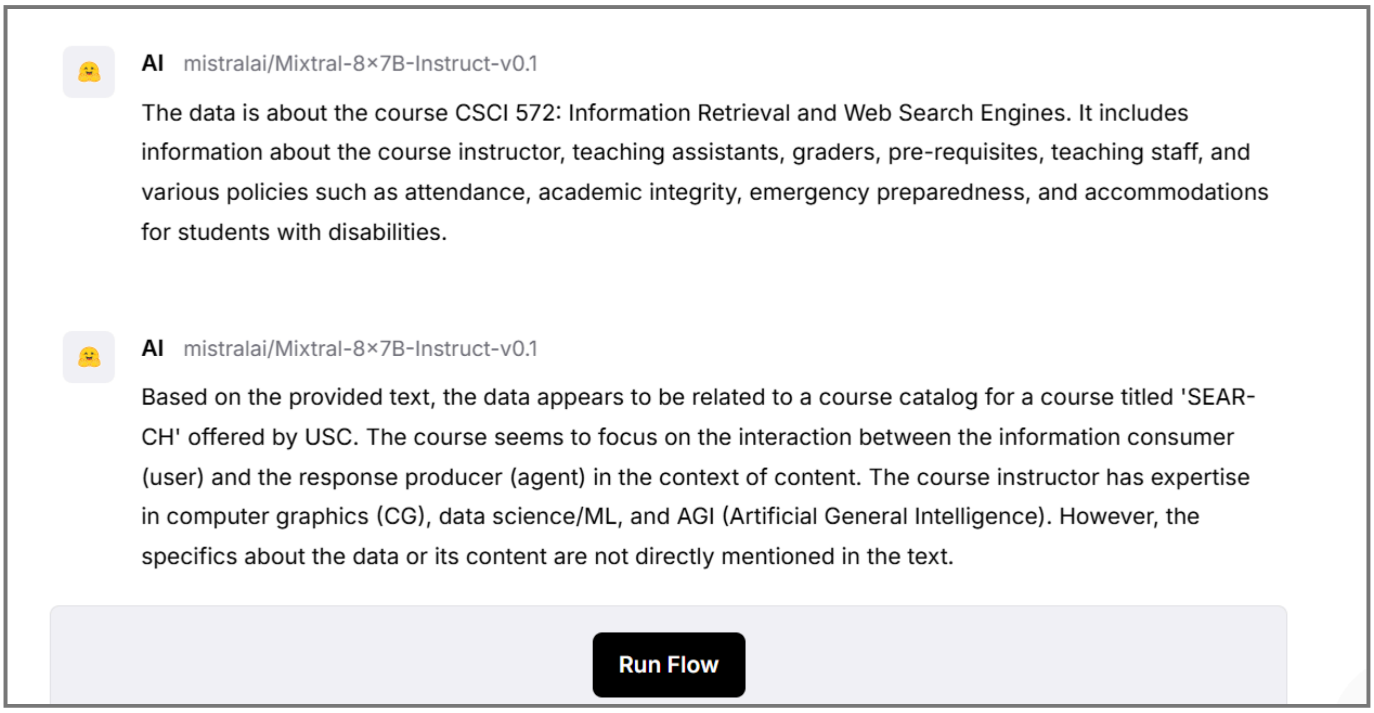


[RMB over the image and display it in a new tab if you want to see it magnified]

In the top graph, I put in https://bytes.usc.edu/cs572/s25-555-search/home/index.html for the URL, in the URL node - to have the page be summarized two ways :) Be sure to put in the API token in the two Hugging Face nodes and in the Cohere node.

In the top graph we're simply fetching URL data and making it be part of the prompt ('context') for the LLM. In the bottom graph we are vectorizing it, then doing RAG on it.

To run the graphs, click on 'Playground', then 'Run Flow'. Here are both the results (amusing and cool, lol):



AI mistralai/Mixtral-8x7B-Instruct-v0.1

The data is about the course CSCI 572: Information Retrieval and Web Search Engines. It includes information about the course instructor, teaching assistants, graders, pre-requisites, teaching staff, and various policies such as attendance, academic integrity, emergency preparedness, and accommodations for students with disabilities.

AI mistralai/Mixtral-8x7B-Instruct-v0.1

Based on the provided text, the data appears to be related to a course catalog for a course titled 'SEAR-CH' offered by USC. The course seems to focus on the interaction between the information consumer (user) and the response producer (agent) in the context of content. The course instructor has expertise in computer graphics (CG), data science/ML, and AGI (Artificial General Intelligence). However, the specifics about the data or its content are not directly mentioned in the text.

Later you can use the 'File' node [which is in the graph above, but not used] instead of the URL node, and read in a document that's on your laptop (.pdf, .txt etc) and ask questions about it. Nice!

Please put in a different URL, and get screenshots of your graphs and the results.

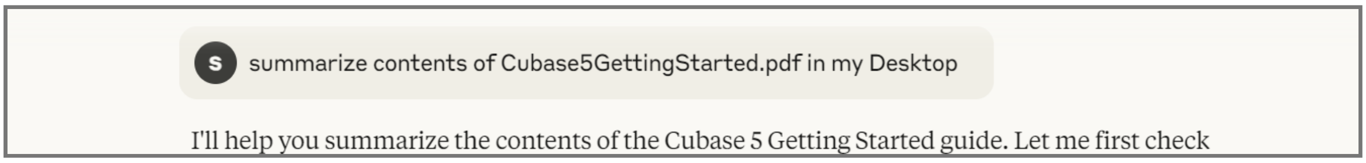
**Q3.** (4 points)

Next, download Claude Desktop, and sign up for a free account (**NO NEED FOR A PAID ONE**). It's a deceptively simple but useful frontend, for sending prompts into LLMs and agentic apps, and displaying outputs.

We're going to do local RAG by querying about a file. For this, we need the filesystem MCP. Locate claude_desktop_config.json on your laptop, then add this to it [obviously you'd change the two 'satyc' paths shown, to your own]:

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "C:\\Users\\satyc\\Desktop",
        "C:\\Users\\satyc\\Downloads"
      ]
    }
```

The above makes it possible to query contents of a file, like so:



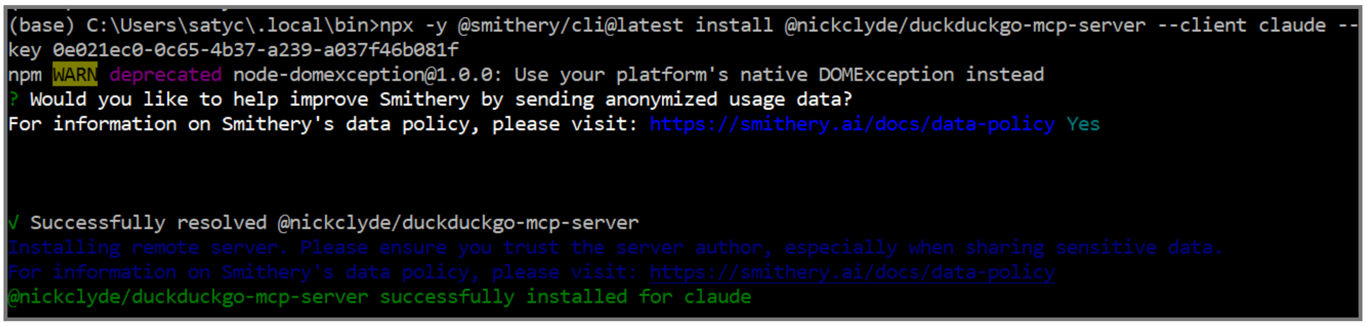
S summarize contents of Cubase5GettingStarted.pdf in my Desktop

I'll help you summarize the contents of the Cubase 5 Getting Started guide. Let me first check

Before you do the above, be sure to quit Claude completely and restart it for the MCP spec to be picked up.

Like above, do a query on one of your files, get a screenshot.

Next, we're going to enable DuckDuckGo (DDG) search, by installing the appropriate 'Node' packages. Be sure to install Node, then do this [run this in a shell: **npx -y @smithery/cli@latest install @nickclyde/duckduckgo-mcp-server --client claude --key 0e021ec0-0c65-4b37-a239-a037f46b081f**]:

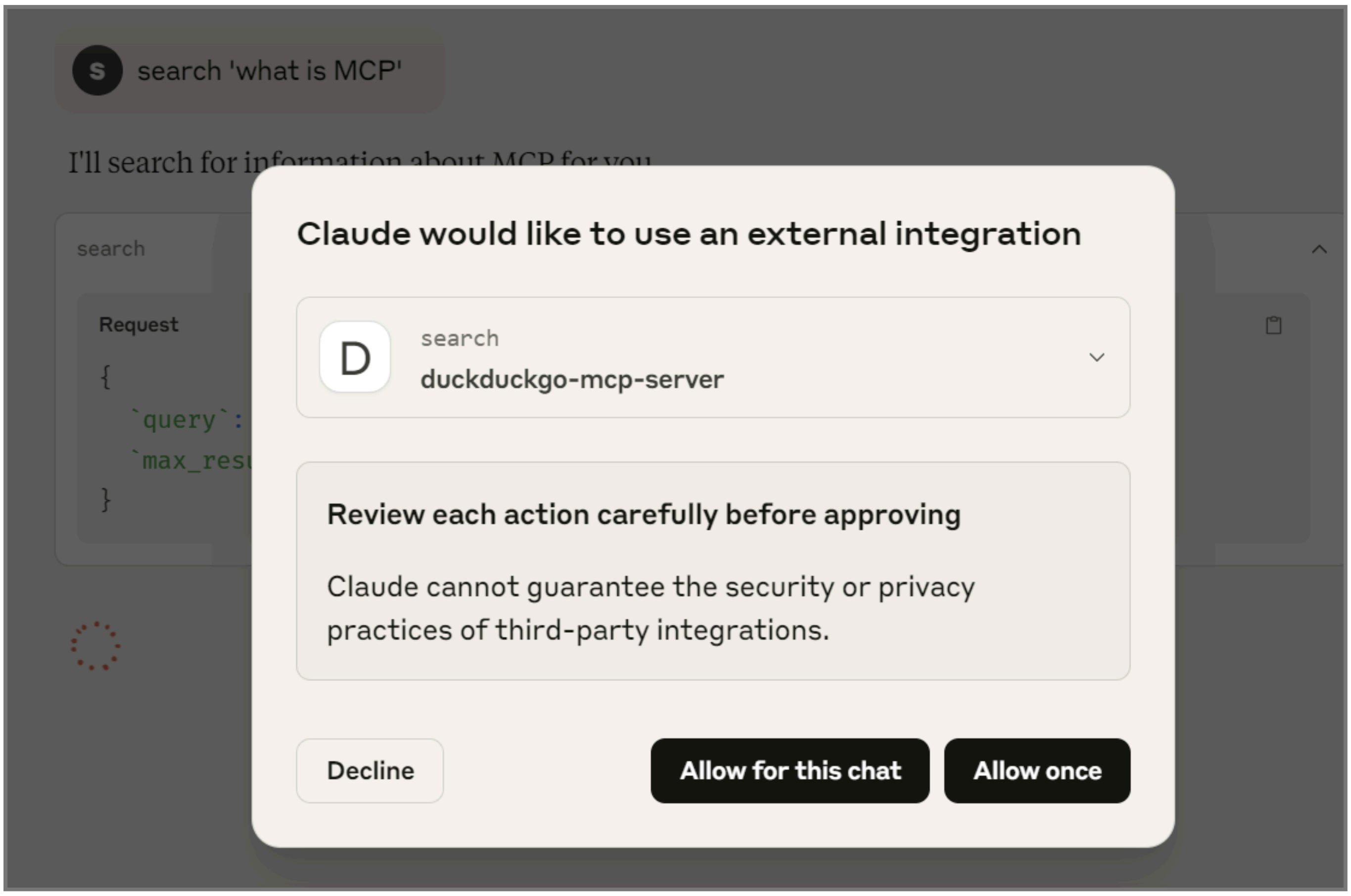


```
(base) C:\Users\satyc\.local\bin>npx -y @smithery/cli@latest install @nickclyde/duckduckgo-mcp-server --client claude --
key 0e021ec0-0c65-4b37-a239-a037f46b081f
npm WARN deprecated node-domexception@1.0.0: Use your platform's native DOMException instead
? Would you like to help improve Smithery by sending anonymized usage data?
For information on Smithery's data policy, please visit: https://smithery.ai/docs/data-policy Yes



√ Successfully resolved @nickclyde/duckduckgo-mcp-server
Installing remote server. Please ensure you trust the server author, especially when sharing sensitive data.
For information on Smithery's data policy, please visit: https://smithery.ai/docs/data-policy
@nickclyde/duckduckgo-mcp-server successfully installed for claude
```

Add (append) this MCP spec to claude_desktop_config.json so that the two specs together look like this:

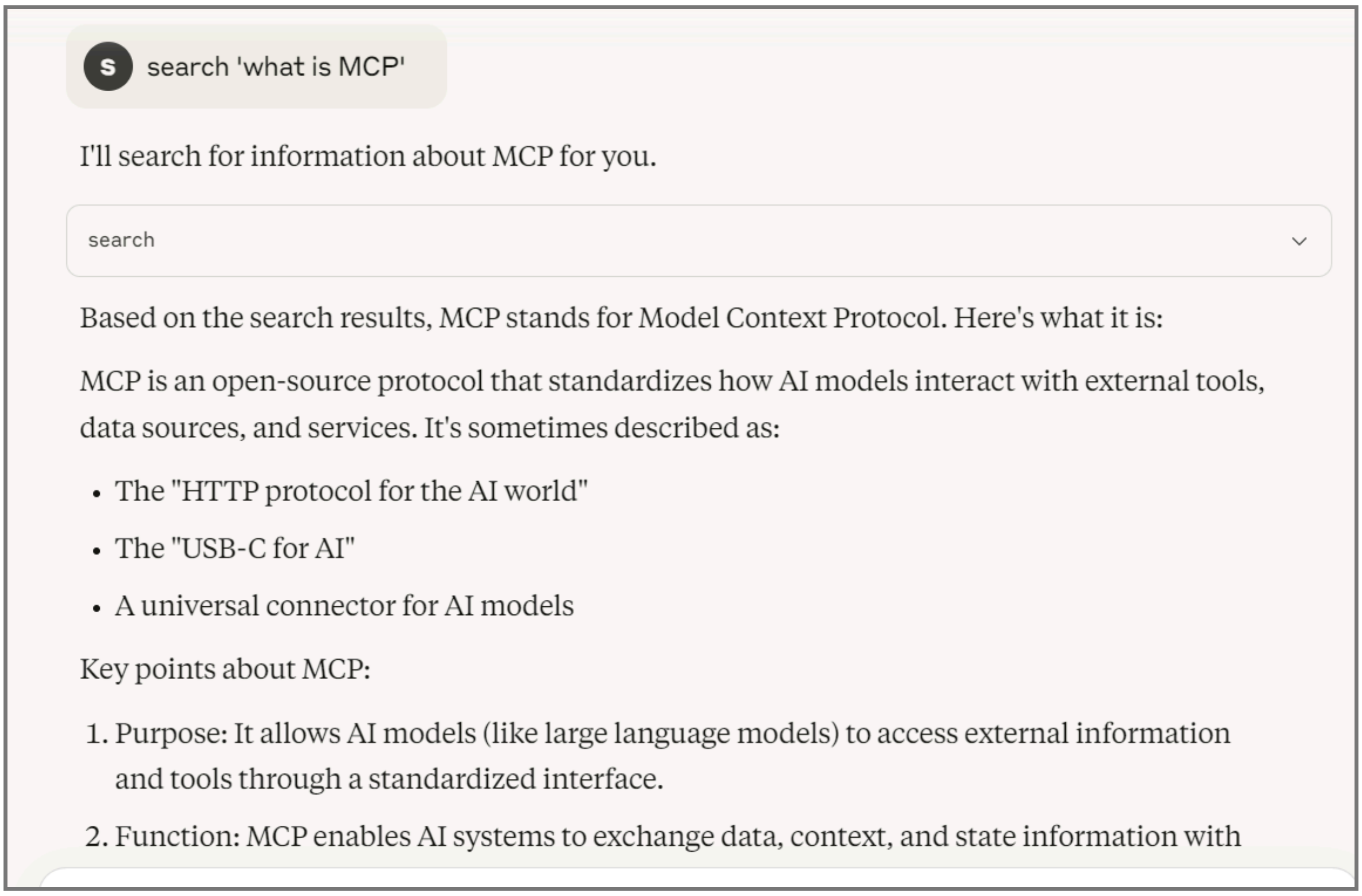
```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "C:\\Users\\satyc\\Desktop",
        "C:\\Users\\satyc\\Downloads"
      ]
    },

    "duckduckgo-mcp-server": {
      "command": "cmd",
      "args": [
        "/c",
        "npx",
        "-y",
        "@smithery/cli@latest",
        "run",
        "@nickclyde/duckduckgo-mcp-server",
        "--key",
        "0e021ec0-0c65-4b37-a239-a037f46b081f"
      ]
    }
  }
}
```

Again, quit and restart Claude Desktop. Now you should be able to do DDG searches!



Review each action carefully before approving

Claude cannot guarantee the security or privacy practices of third-party integrations.



I'll search for information about MCP for you.

Based on the search results, MCP stands for Model Context Protocol. Here's what it is:

MCP is an open-source protocol that standardizes how AI models interact with external tools, data sources, and services. It's sometimes described as:

- The "HTTP protocol for the AI world"
- The "USB-C for AI"
- A universal connector for AI models

Key points about MCP:

1. Purpose: It allows AI models (like large language models) to access external information and tools through a standardized interface.
2. Function: MCP enables AI systems to exchange data, context, and state information with

Perform two searches, get screenshots.

---

Simply submit screenshots for Q1, Q2, Q3, all as a single .zip file.

After this course, explore all three above, MUCH more! Eg. look at all these MCP servers you can enable: https://github.com/modelcontextprotocol/servers

---

## Getting help

There is a hw4 'forum' on Piazza, for you to post questions/answers. You can also meet w/ the TAs, CPs, or me.

Have fun! These (LLMs, vectorizing, RAG, Python-based UI spec...) are really (REALLY!!!) useful pieces of 'tech' to know. In the coming years, LLMs are sure to make their way into EVERY app/site/backend/system/process.