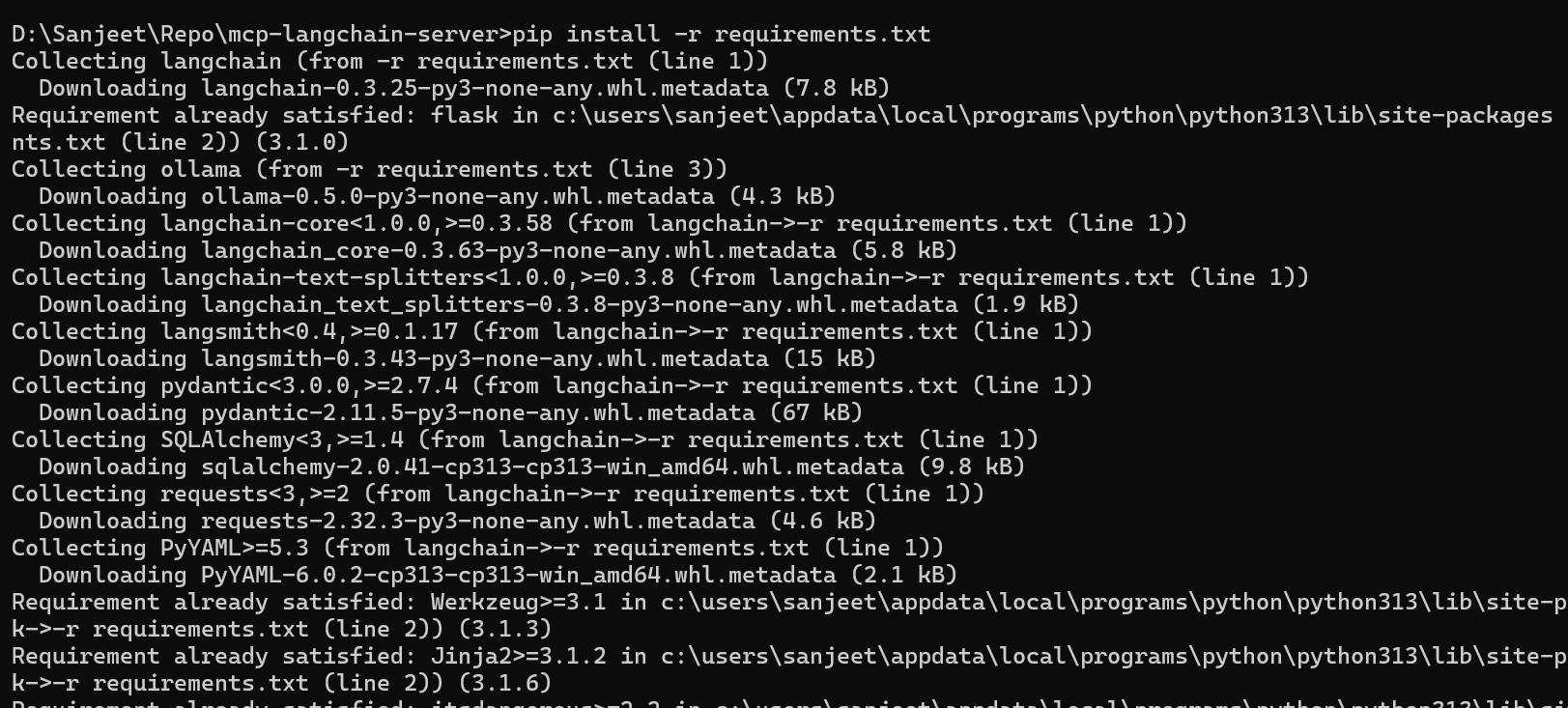
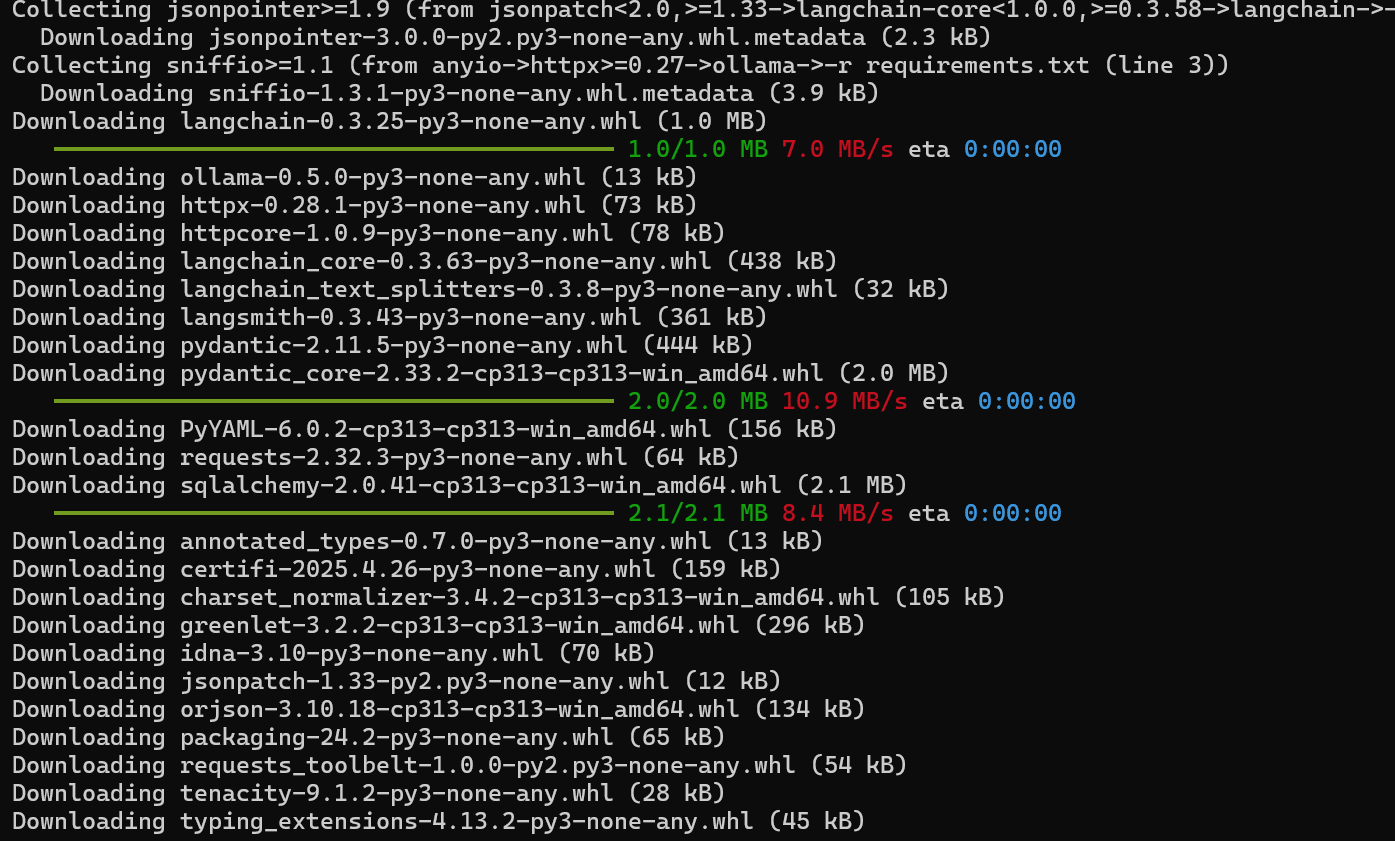
Here’s a **ready-to-run MCP-style LangChain + Ollama** Python project that:

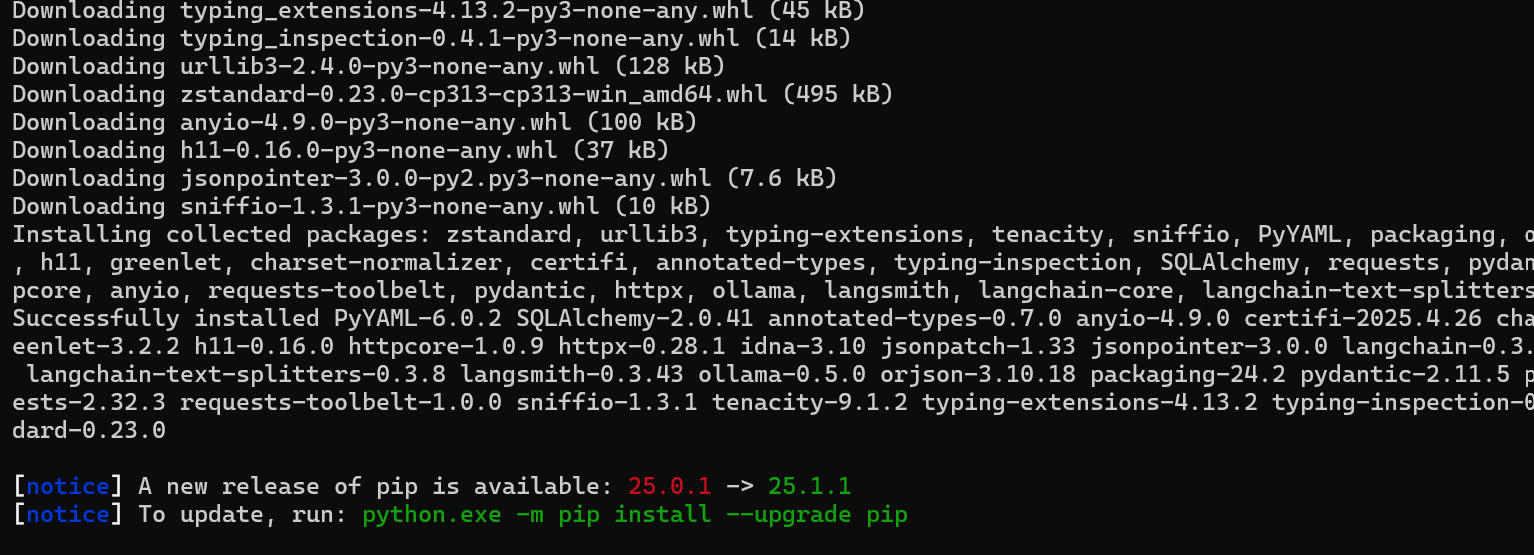
* Runs completely **free and locally**
* Accepts MCP-style { action, params } via a Flask API
* Routes to a **local LLM (via Ollama)** for answers

💡 Install with: pip install -r requirements.txt

Run this in regular command prompt or vscode terminal







**Notes**

* Make sure **Python** is installed. You can check with:

python –version

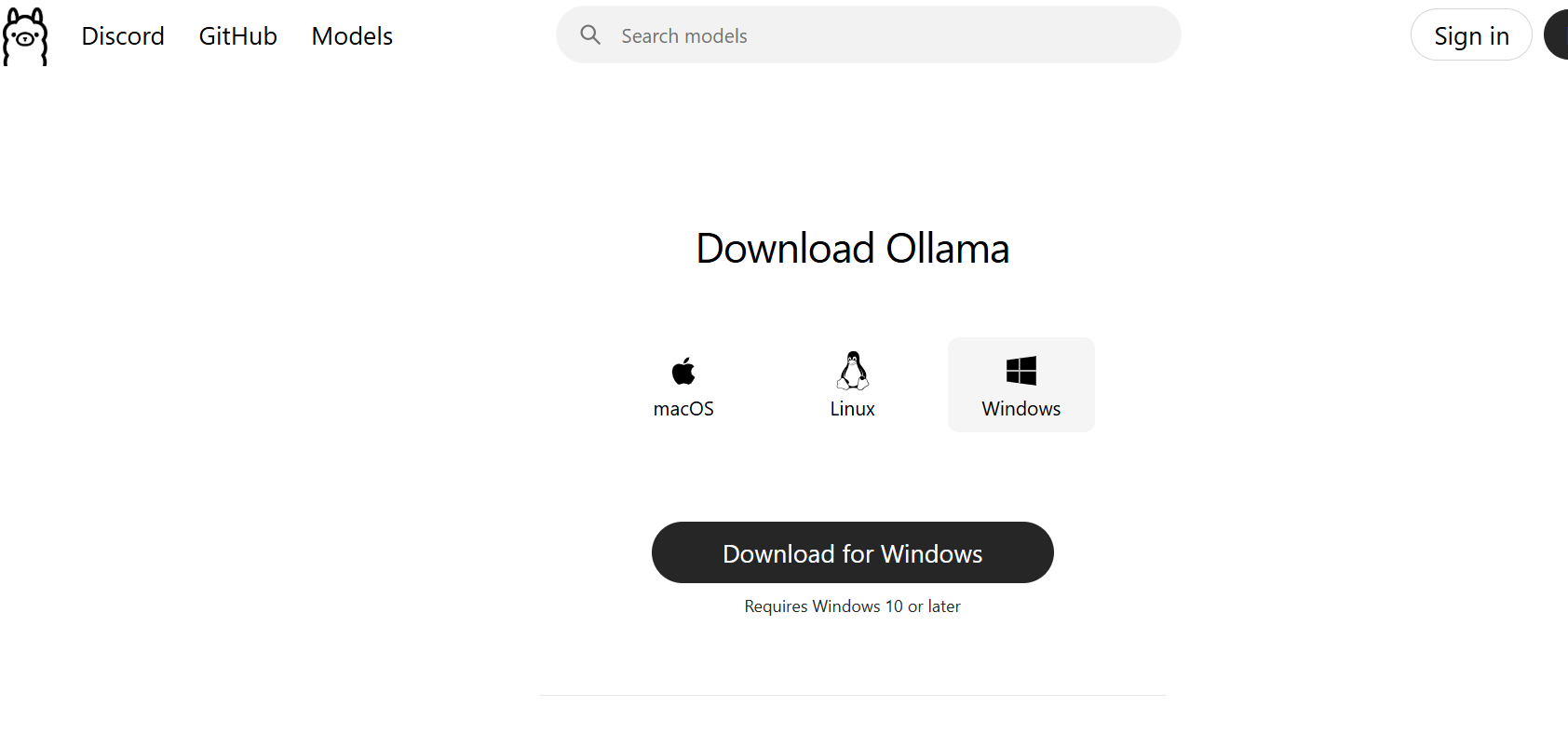
If you're using **Python 3 and pip is mapped to pip3**, you can run:

pip3 install -r requirements.txt



<https://ollama.com/download>

<https://github.com/ollama/ollama>

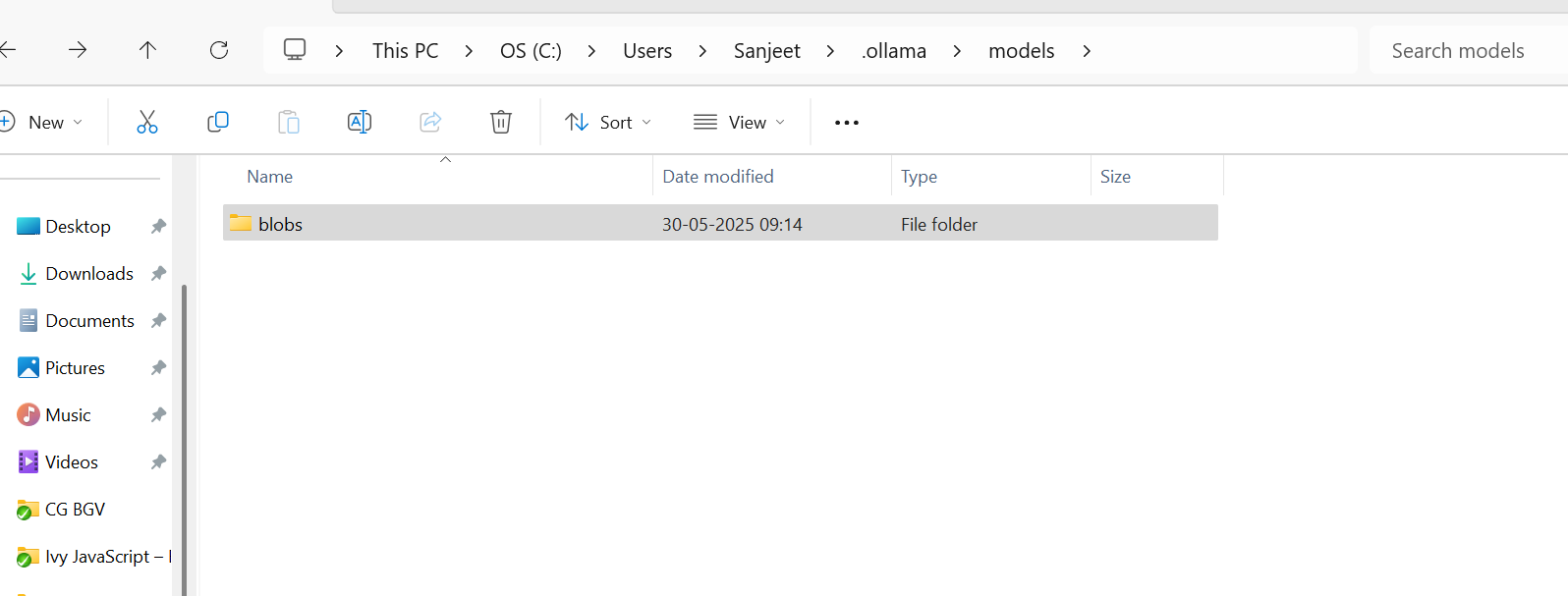


**Run It**

1. Make sure Ollama is installed and running:

ollama run llama3

5 GB installation size



**1. Verify Ollama is Installed and in PATH**

* Open Command Prompt and run:

ollama –version

* If you see the version → proceed to next step.
* ❌ If you see "command not recognized", then **Ollama is not added to your system PATH**.

**➤ Fix PATH (if needed)**

* Go to:

Settings → System → About → Advanced system settings → Environment Variables

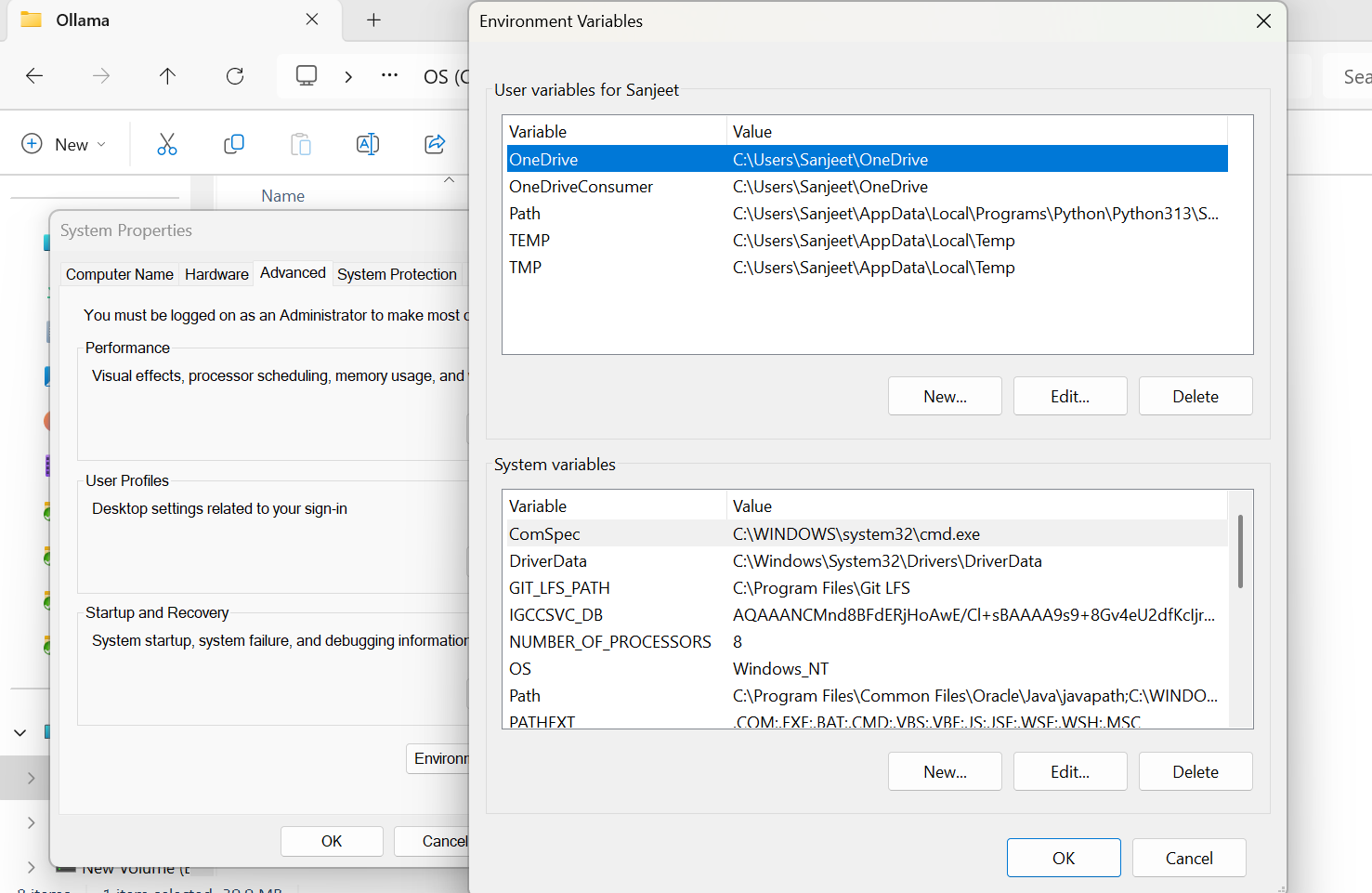
Under **System variables**, find Path, click **Edit**, then click **New**, and add the folder where ollama.exe is installed.

Typically:

C:\Users\<your-username>\.ollama\bin

(or)

C:\Users\Sanjeet\AppData\Local\Programs\Ollama



Click OK, restart Command Prompt, and try:

ollama –version

**2. Start the Ollama Service**

If ollama is installed but not running the server, start it manually:

ollama serve

**3. Run the Model**

In a new Command Prompt:

ollama run llama3

This will:

* Download the model (first time only, ~4GB for Llama 3 8B)
* Start an interactive prompt

**Next Steps You Can Try**

Now that Ollama works, here are a few useful commands:

**🔸 Run LLaMA 3:**

**List downloaded models:**

ollama list

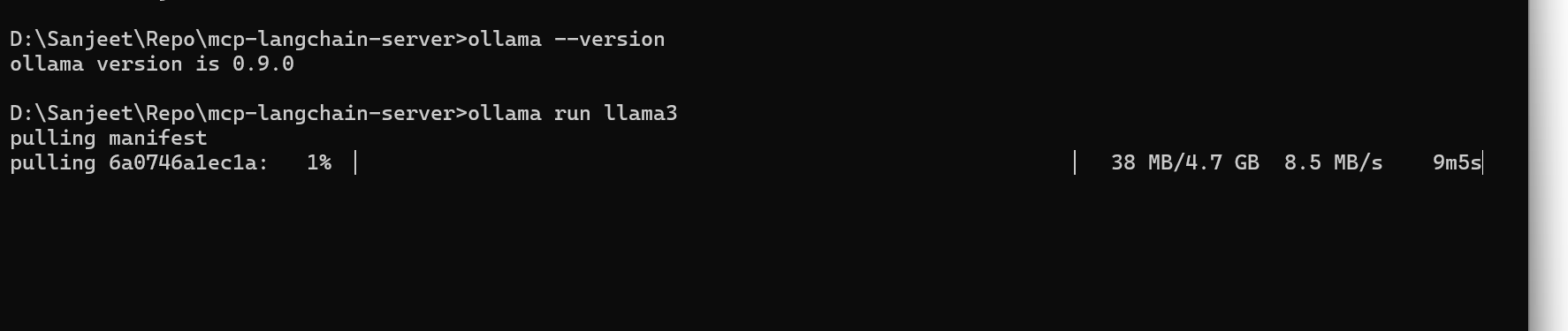
**🔸 Pull other models:**

ollama pull mistral

ollama pull codellama

**🔸 Stop running model:**

ollama stop





means that your system doesn't have enough **RAM (Memory)** available to run the llama3 model locally with Ollama.

**What You Can Do**

**Option 1: Try a Smaller Model**

Use a smaller model that needs less memory. Try:

ollama run mistral

* mistral only needs ~4 GB of RAM and still performs quite well for general tasks.

Or even smaller:

ollama run gemma:2b

* gemma:2b uses just ~2–3 GB RAM.

You can explore more models: https://ollama.com/library

**Option 2: Close Other Apps**

Try to **free up RAM** by:

* Closing Chrome, VS Code, and other memory-heavy apps
* Restarting your system before running ollama run llama3

Then try again.

**Option 3: Use Ollama with a GPU (if available)**

If your system has a dedicated GPU, Ollama can use it. However, this also needs sufficient **GPU memory** (VRAM) — at least 6 GB VRAM.

**Option 4: Use Ollama in the Cloud**

If local doesn't work due to hardware limits, you can:

* Run Ollama in a **cloud VM** (e.g., on AWS, Paperspace, or Colab with Docker)
* Use hosted models (e.g., via OpenAI, Groq, TogetherAI, etc.)

Would you like me to recommend a **lightweight GenAI setup** for your current system specs (2.7 GB available RAM)?

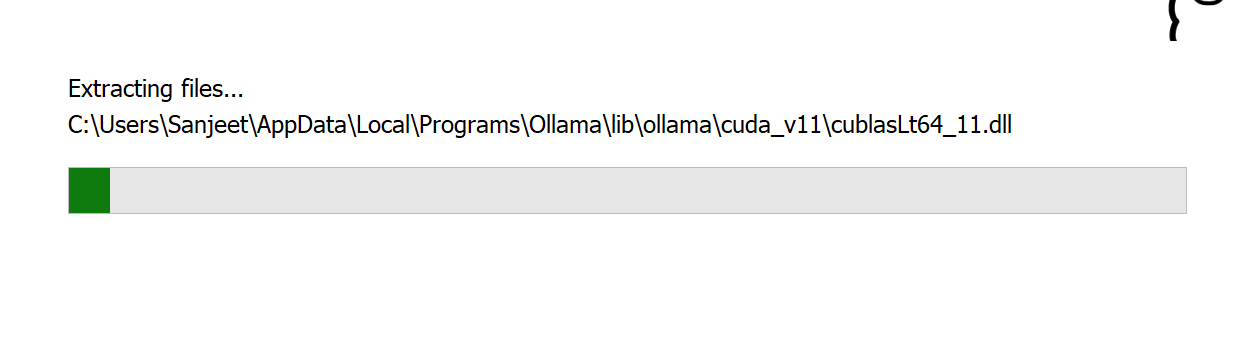
**Why?**

LangChain + Ollama integration works through an **API-compatible local server** that Ollama runs (by default at http://localhost:11434). The model name (like llama3, mistral, or gemma:2b) is just a **parameter passed at runtime** — your code doesn't need to change **unless** you explicitly set the model name in Python.

**LLaMA 3 still needs at least ~6 GB of RAM free**, and ideally more (8–12 GB total system RAM recommended).

**✅ Tip: Run in PowerShell or Windows Terminal**

If Command Prompt gives issues, try running in **PowerShell** or **Windows Terminal**.



Start your MCP server:

python app.py

Test it via Postman or curl:

curl -X POST http://localhost:4000/mcp/action \

-H "Content-Type: application/json" \

-d '{"action": "get\_time", "params": {}}'