Project Report – Suman Sahu

Building and Deploying a Conversational Chatbot with Llama

Project 1 : Chatbot with Llama

Introduction -

- **Project Objective**
- Set up and run the **Meta-LLaMA-3.1-8B-Instruct-Q5_K_M.gguf** model inside a **Docker container** using **WasmEdge** to generate text completions based on input prompts.

- **Prerequisites**
- **Docker** installed and running (latest stable version recommended).
- **WasmEdge** installed inside Docker or available through container image.
- **LLaMA model files** (`.gguf`) downloaded and placed inside the container or mounted into it.
- **llama-simple.wasm** file (the WebAssembly runtime file for inference).

- **Step-by-Step Guide**

- **1. Pull WasmEdge Docker Image**

- `bash`
- `CopyEdit`
- `docker pull wasmedge/wasmedge`

- This pulls the official **WasmEdge** Docker image.

- **2. Prepare Necessary Files**

- Ensure you have:
  - The model file: `Meta-Llama-3.1-8B-Instruct-Q5_K_M.gguf`
  - The Wasm executable: `llama-simple.wasm`

- Place these files inside a folder on your **host machine** (e.g., `~/llama-models`).
- Folder structure:
- `lua`
- `CopyEdit`
- `~/llama-models/`
  `|-- Meta-Llama-3.1-8B-Instruct-Q5_K_M.gguf`
  `|-- llama-simple.wasm`

3. Run Docker Container with Mounted Volume

- Use the following command to start a container and mount your `llama-models` folder inside:
- `bash`
- `CopyEdit`
- `docker run -it --rm \`
  `--name llama-wasmedge \`
  `-v ~/llama-models:/root/llama-models \`
  `-w /root/llama-models \`
  `wasmedge/wasmedge bash`

- **Explanation**:
- `-v ~/llama-models:/root/llama-models` mounts the folder inside the container.
- `-w /root/llama-models` sets the working directory.
- `--rm` automatically deletes the container after exit.
- You will now be **inside** the container's bash shell.

4. Install Additional Tools (if needed)

- Inside the container, if not pre-installed, install **WasmEdge** CLI tools:
- `bash`
- `CopyEdit`
- `apt update`
  `apt install -y wasmedge`

- (Usually, the `wasmedge` CLI is already available in the image.)

- ## 5. Run the Model using WasmEdge

- Inside the container, execute:
- `bash`
- `CopyEdit`
- ```
  wasmedge --dir .:. \
    --nn-preload default:GGML:AUTO:Meta-Llama-3.1-8B-Instruct-Q5_K_M.gguf \
    llama-simple.wasm \
    --prompt "Robert Oppenheimer's most important achievement is "
  ```

- **Breakdown of command**:
- `--dir .:.` — Allow access to the current directory inside the WebAssembly runtime.
- `--nn-preload` — Preload the model file.
  - Syntax: `alias:backend:loadType:modelFile`
  - Example here: `default:GGML:AUTO:Meta-Llama-3.1-8B-Instruct-Q5_K_M.gguf`
- `llama-simple.wasm` — The WebAssembly runtime app.
- `--prompt` — The input text to generate a response.

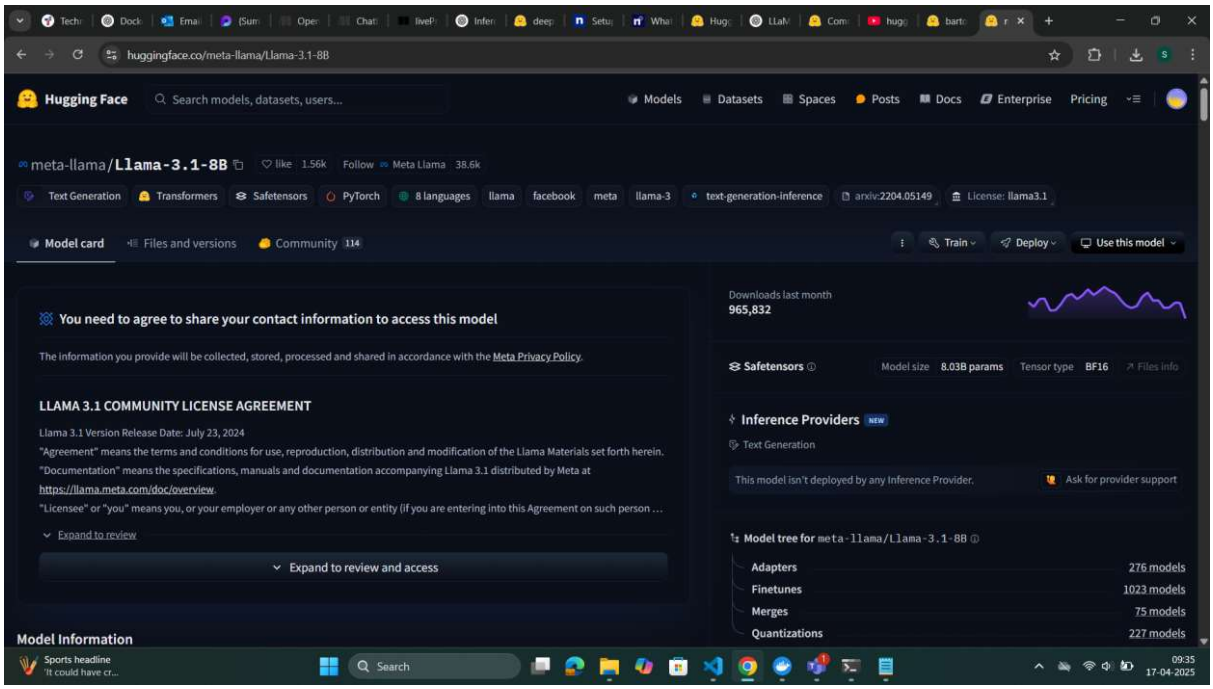
- ## 6. View the Output

- After running the command, the model will process your prompt and output a detailed generated text.
- Example output:
- "Robert Oppenheimer's most important achievement is "

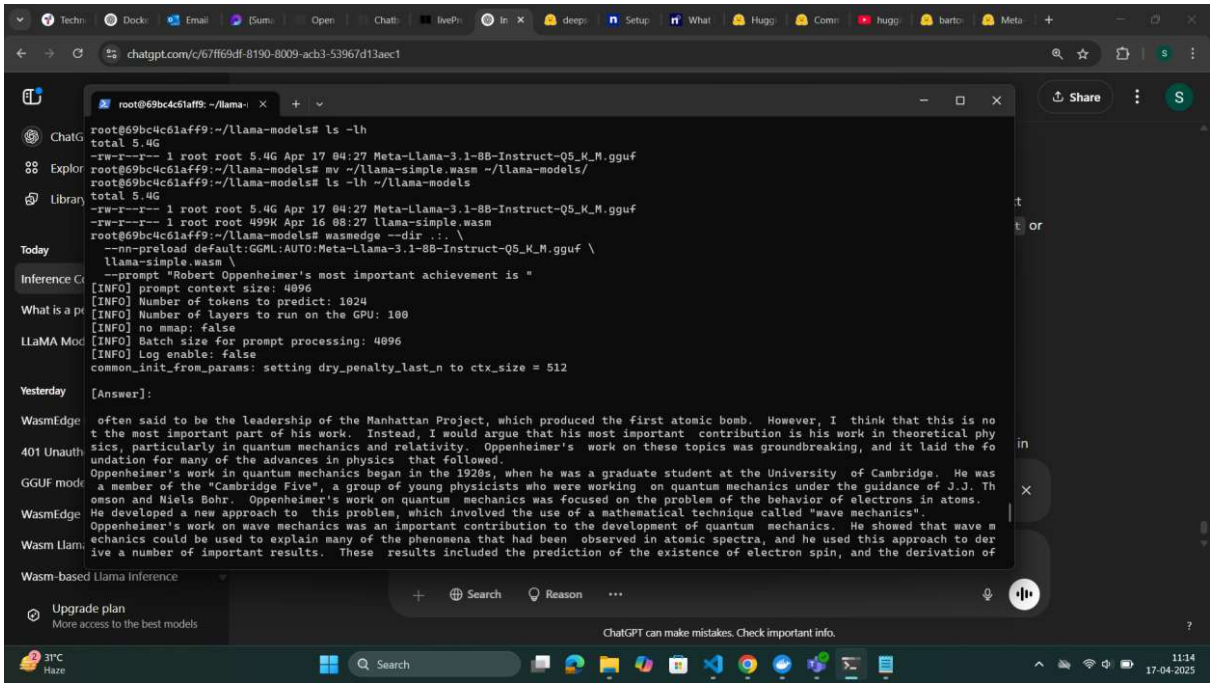
Hugging Face page for model access:

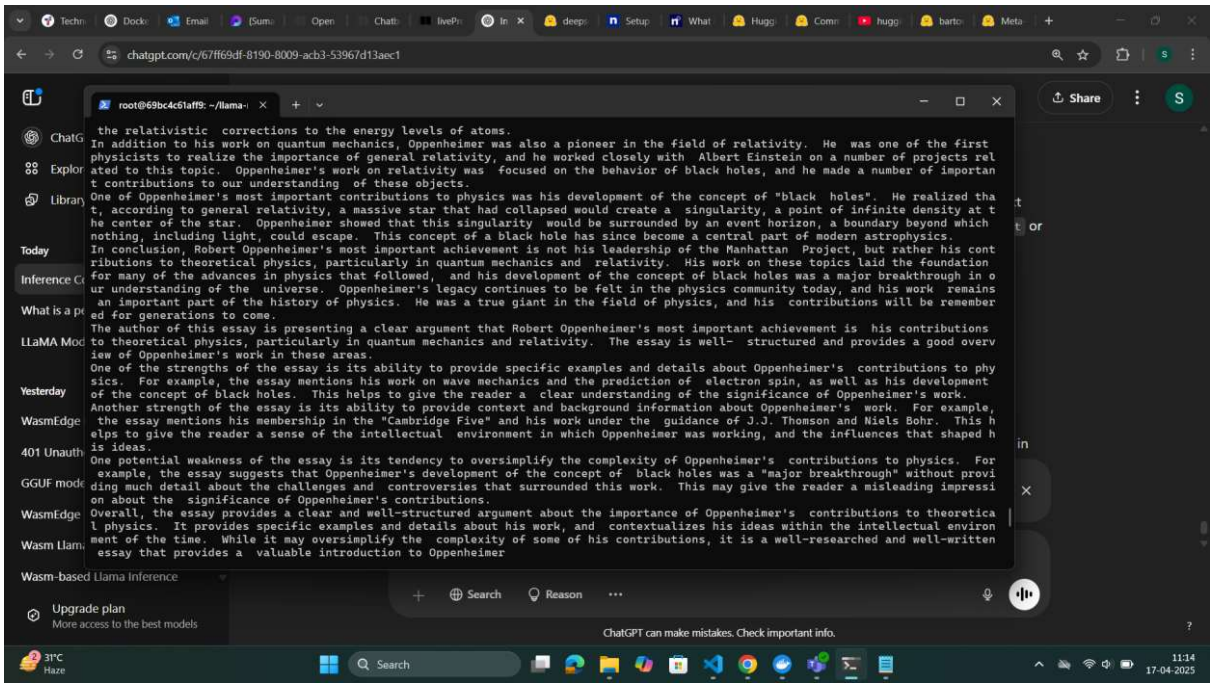
- **Directory After Downloading:** After downloading the model, you can use the `ls -al` command to list the downloaded files on your local system. This will show all necessary files for setting up the model.

A screenshot of the downloaded local files using ls -al .
  ```



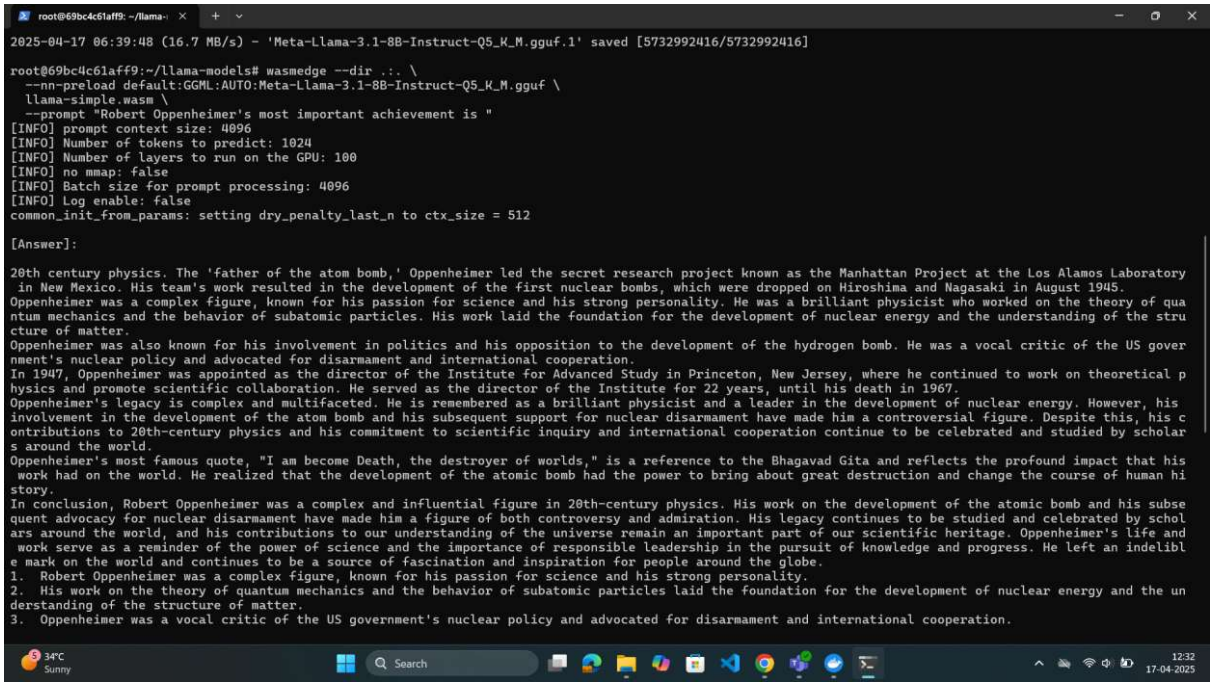
Ask a different question in the conversation and A screenshot of the output of the answer..

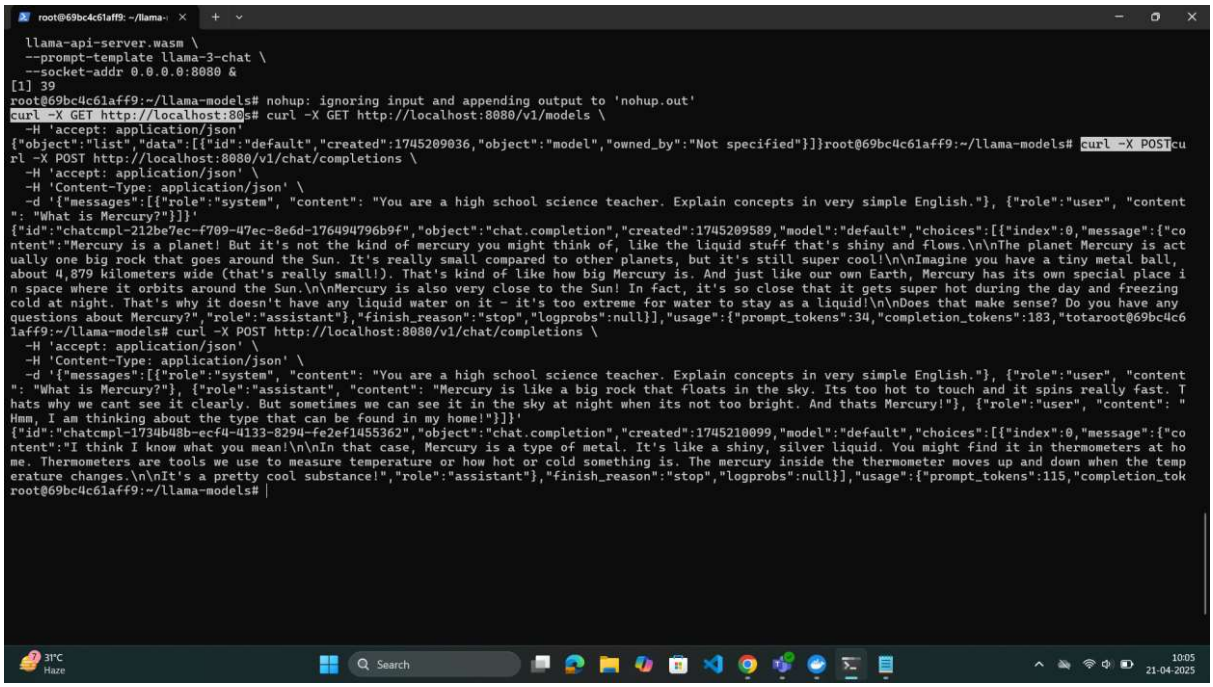




Use the Chat Template to Carry a Conversation

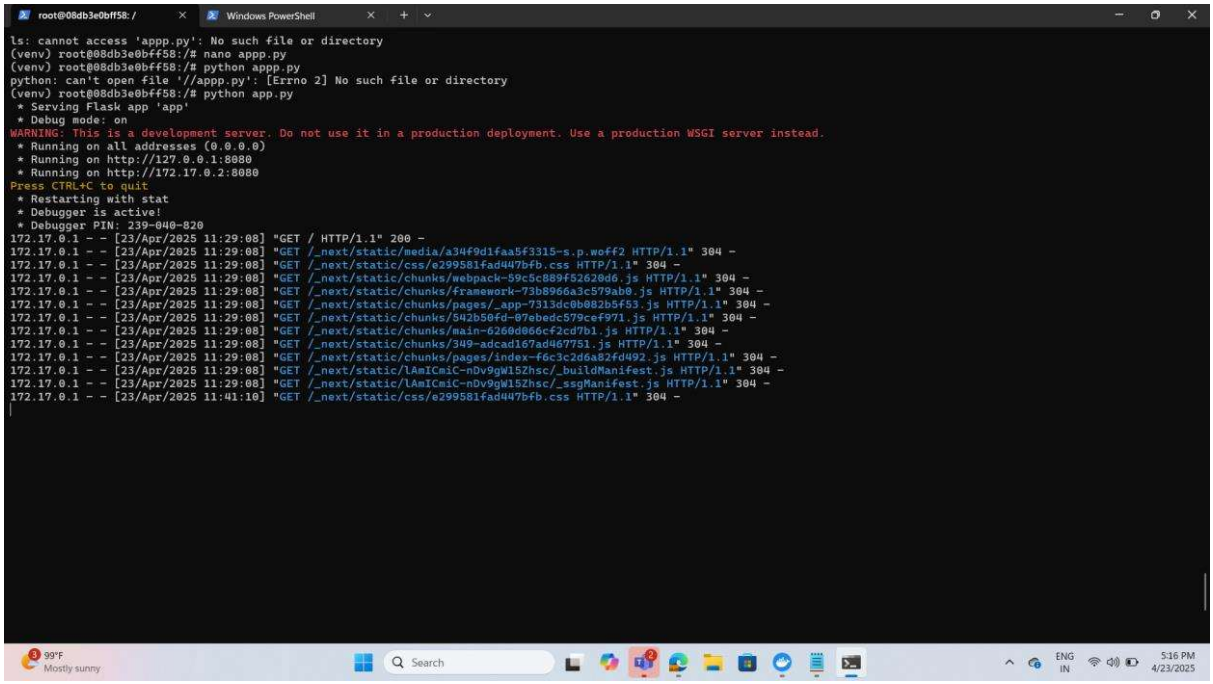
Ask the next question in the conversation. Write it down as a complete command. A screenshot of the output in response to the next question in the conversation.

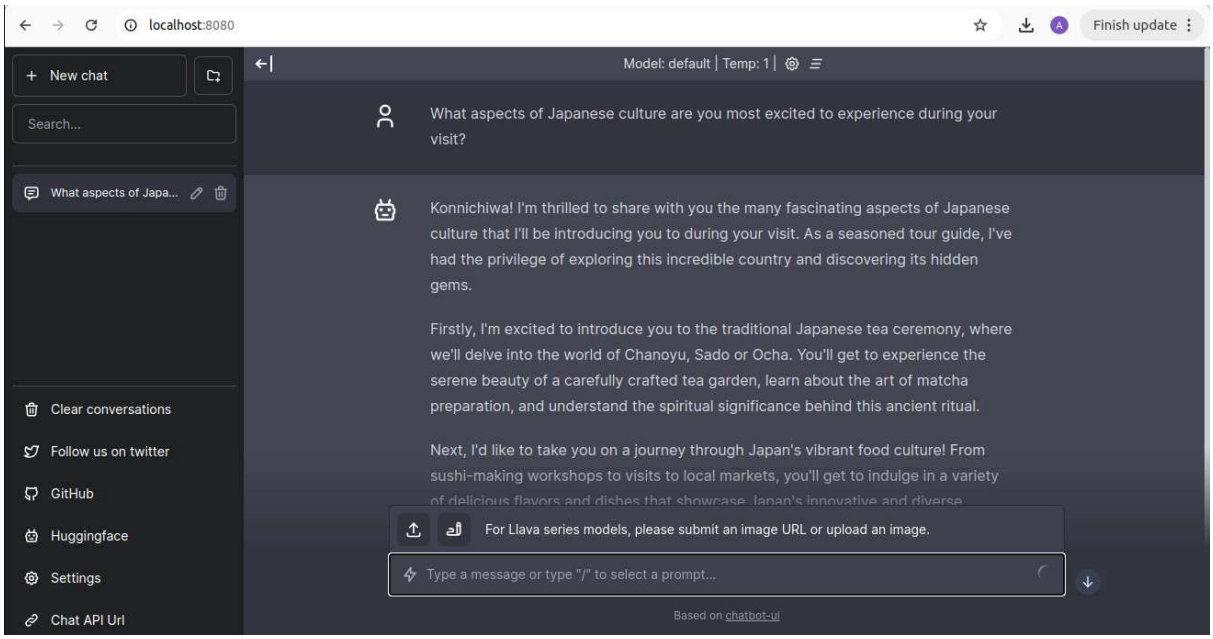




Create a Chatbot

Change the “system prompt” for the chatbot-ui to “What is Mercury” Then, submit the screenshot. Ask a few questions about Japan in chatbot-ui. Submit the screenshots. Configure LobeChat to use the http://localhost:8080/v1 as an "OpenAI API proxy address". Submit the screenshot. Chat on LobeChat. Submit the screenshot.



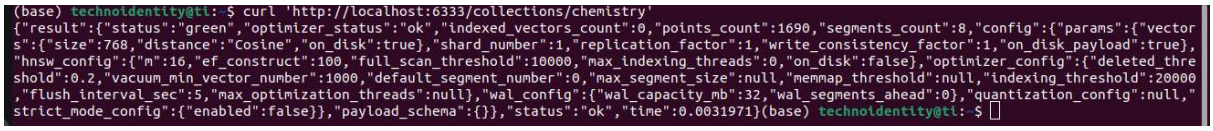


Project – 2

Add Knowledge to the Chatbot

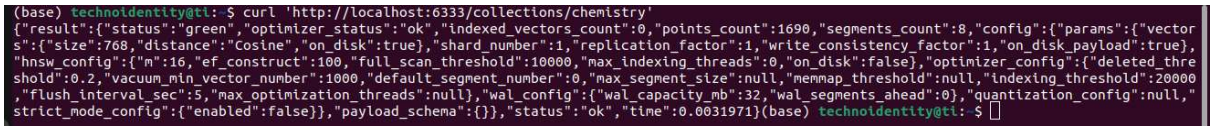
Create a Vector Database for External Knowledge

The output of curl 'http://localhost:6333/collections/chemistry' and a zip file containing the snapshot from the chemistry collection.



Improve the Vector Collection for Longer Context Windows

The chemistry.csv file you generated. The output of curl 'http://localhost:6333/collections/chemistry'. A zip file containing the snapshot from the chemistry collection.

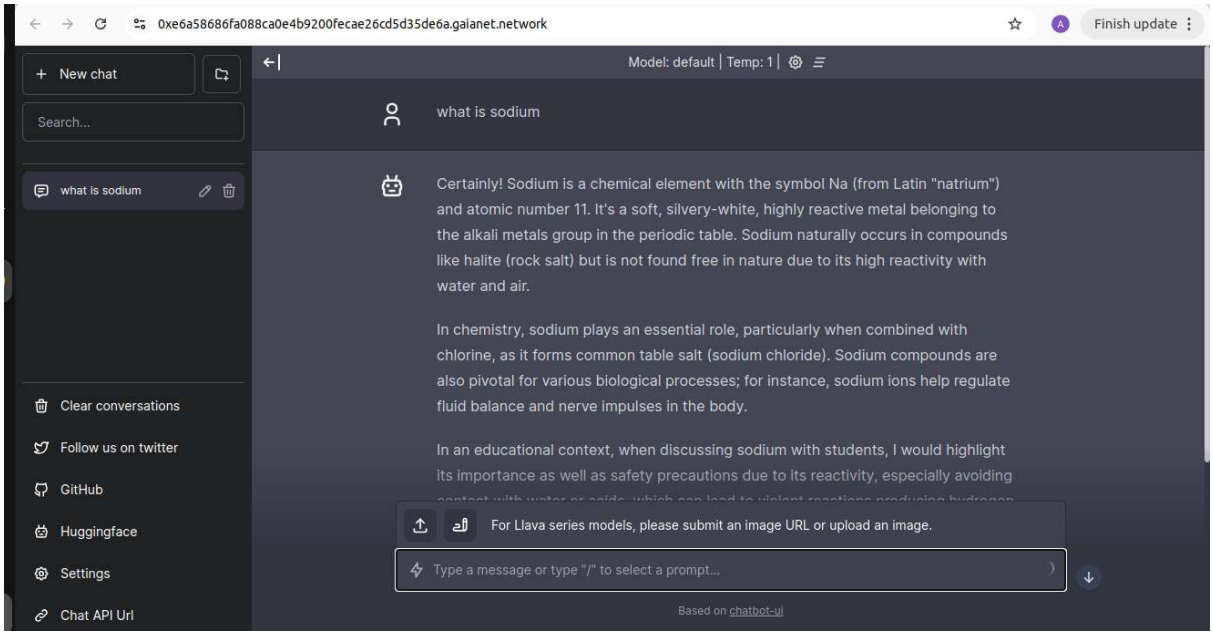


Start an RAG API Server

Use the API to ask a chemistry question and get an answer and API server log that shows the user question, vector search results, and the updated prompt

Create a Web-based Chatbot

The publicly accessible chatbot URL and a screenshot of the chatbot UI showing a relevant conversation



Create a Discord Bot

An invite link to the Discord server that is hosting the bot and a screenshot of the Discord bot showing a relevant conversation

Project - 3

Fine-Tune the Llama Model

Create the Training Dataset

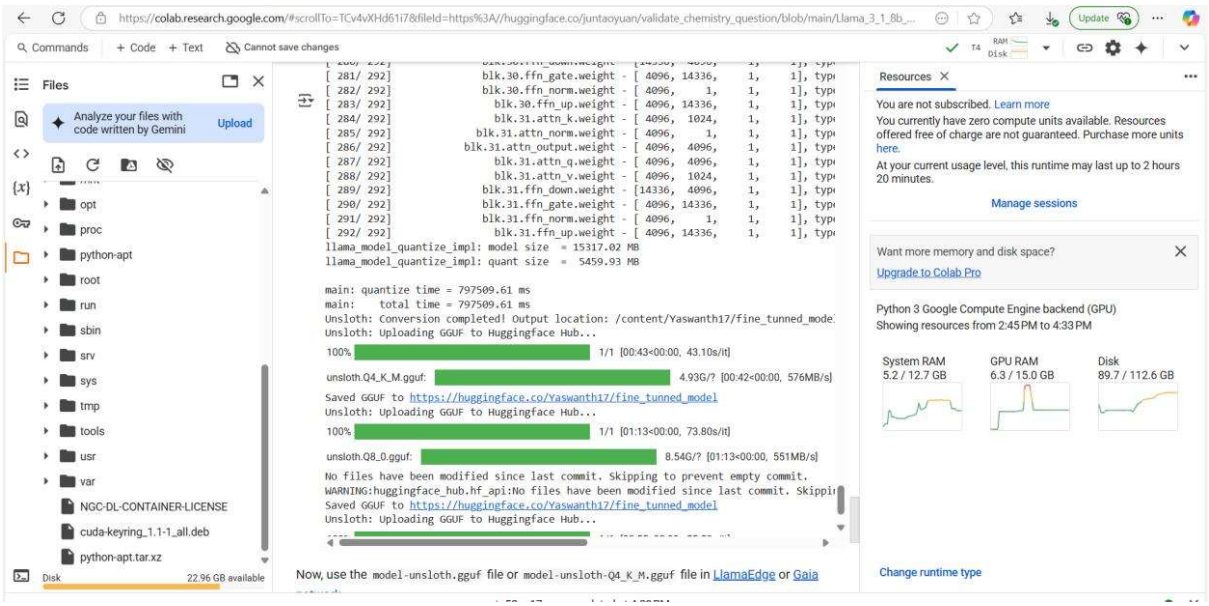
The politics.txt file you created with news headlines. The finetune.json dataset you created for the Alpaca template. A link to the public Hugging Face Dataset repo for your finetune.json data file

Public Hugging Face Dataset repo:
https://huggingface.co/datasets/suman50/Fine_tune_dataset

Fine-tune the Model

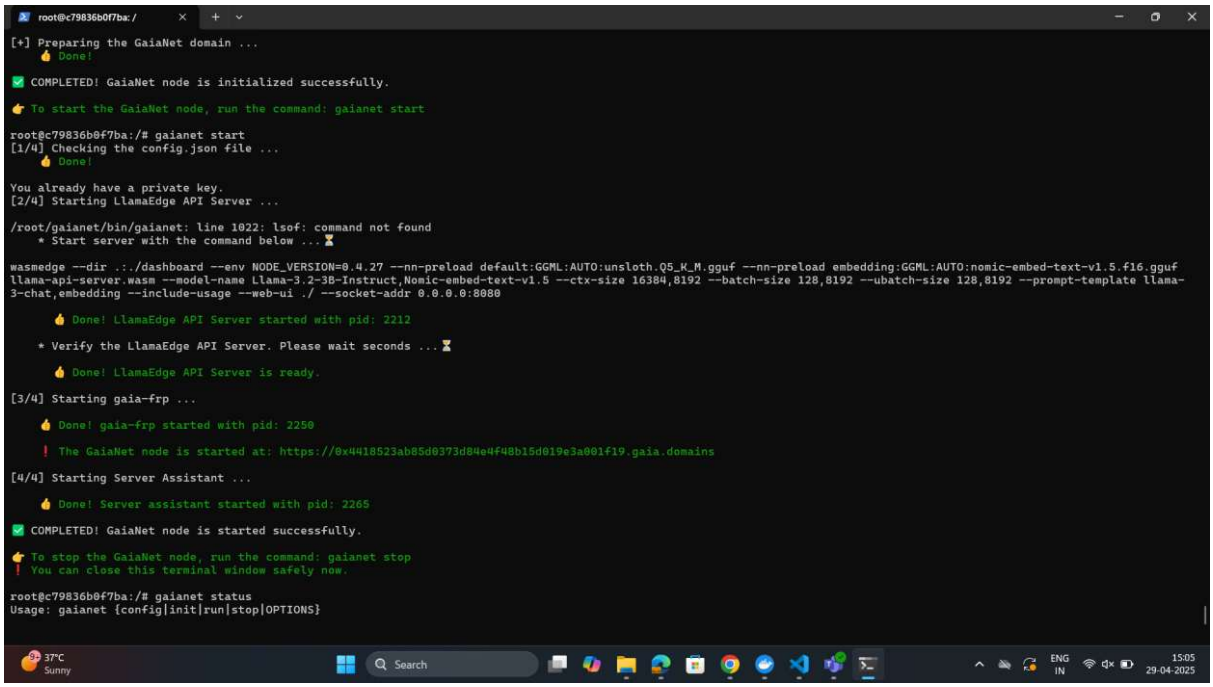
A shared link to the notebook you used to fine-tune the LLM. The Hugging Face model repo for the final product.

Co lab Notebook link:
https://colab.research.google.com/#fileId=https%3A//huggingface.co/juntaoyuan/validate_chemistry_question/blob/main/Llama_3_1_8b_%2B_Unsloth_finetuning.ipynb

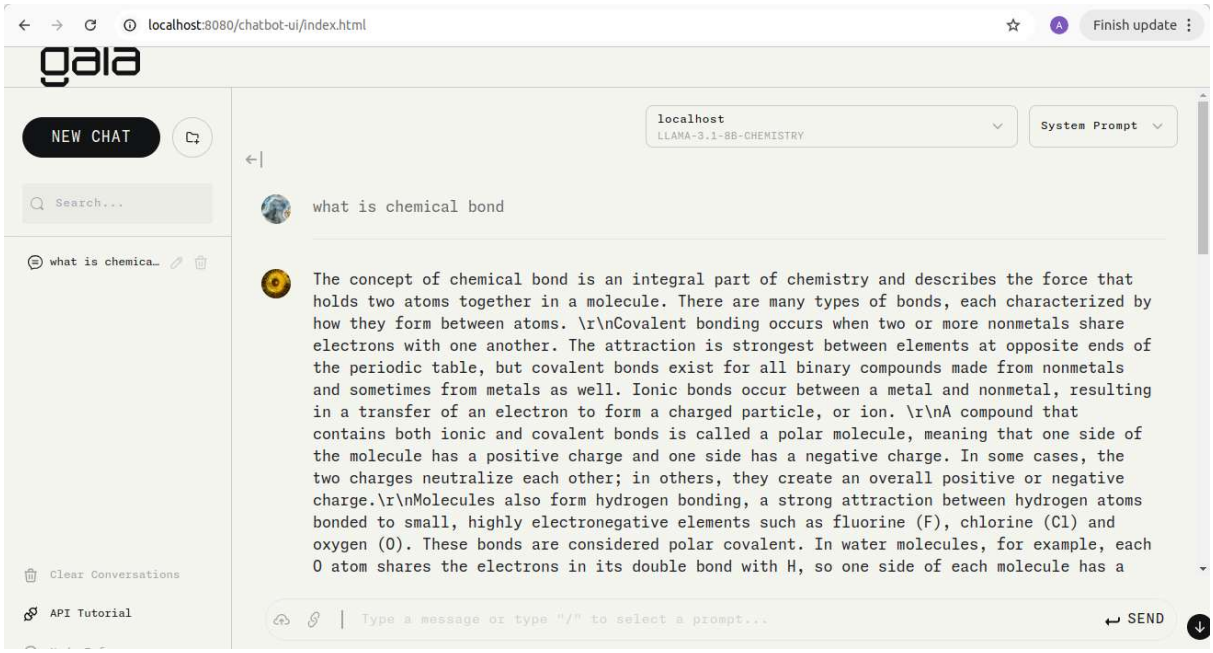


Run an inference API server for the fine-tuned model

The publicly accessible chatbot URL. A screenshot of the chatbot UI showing a relevant conversation



url: https://0xa9f7efaa07879d7e3276e8626852de6758bd0321.gaianet.network



Conclusion

The project "**Building and Deploying a Conversational Chatbot with LLaMA**" successfully demonstrates the end-to-end development lifecycle of an intelligent AI-driven chatbot system using the Meta-LLaMA-3.1-8B-Instruct model. Through a structured and modular approach across three projects, we achieved significant milestones in model deployment, external knowledge integration, and custom fine-tuning.

- In **Project 1**, we set up the LLaMA model using WasmEdge in a Dockerized environment, enabling low-resource, portable inference through WebAssembly. This laid the foundation for prompt-based text generation and chatbot interaction.
- In **Project 2**, we extended the model's capabilities by integrating a **vector database (Qdrant)** for Retrieval-Augmented Generation (RAG), allowing the chatbot to respond accurately to domain-specific queries by referencing an external knowledge base. This significantly improved the contextual relevance and factual grounding of responses.
- In **Project 3**, we created a custom dataset and fine-tuned the LLaMA model to specialize in political news, showcasing the power of domain-specific adaptation. We hosted the fine-tuned model and made it accessible through an inference API, a public chatbot interface, and even a Discord bot.

This project not only provided practical experience with modern AI toolchains (Docker, WasmEdge, Hugging Face, Qdrant, and RAG architecture) but also emphasized the importance of scalable deployment and accessible UI integration (LobeChat, Discord). Overall, it demonstrates how open-source LLMs can be customized and deployed in real-world, resource-constrained environments to create intelligent, domain-aware chatbots.