LLM Council Local Deployment

Project Overview

This final project is inspired by Andrej Karpathy LLM Council concept: instead of relying on a single Large Language Model (LLM), multiple LLMs collaborate by answering, reviewing and synthesizing responses to a user query.

In the original implementation, the system relies on cloud-based models accessed via OpenRouter. **Your task is to refactor and extend this system so that the entire LLM Council runs locally**, using tools such as Ollama, GPT4All, Llamafile, Hugging Face or similar local inference frameworks.

The end result should be a **distributed multi-LLM system** in which multiple machines communicate via a network to form an LLM Council, resulting in a summarized final output generated by a dedicated **Chairman LLM**.

What Is the LLM Council?

The LLM Council is a multi-stage reasoning system:

- **Multiple LLMs answer the same user query independently**
- **LLMs anonymously review and rank each other's answers**
- **A Chairman LLM synthesizes all outputs into a final response**
- The user can inspect intermediate model outputs

This approach emphasizes **diversity of reasoning**, **self-critique** and **aggregation of perspectives**.

Council Workflow

Stage 1: First Opinions

- The user submits a query.
- Each LLM in the council generates an answer **independently**.
- All responses are displayed in a **tabbed interface** so users can inspect them individually.

Stage 2: Review & Ranking

- Each LLM is shown the responses from the *other* LLMs.
- Model identities are **anonymized** to avoid bias.
- Each LLM ranks the responses based on: accuracy and insight

Stage 3: Chairman Final Answer

- A designated **Chairman LLM**:
  - Receives all original responses
  - Receives all review rankings
- The Chairman synthesizes everything into a **single final response** presented to the user.

Project Goal

Your objective is to **refactor the original LLM Council implementation** so that:

- ❌ Cloud-based APIs (OpenRouter, OpenAI, etc.) are removed
- ✅ All LLMs run **locally**
- ✅ The system supports **multiple machines communicating via REST APIs**
- ✅ The full council workflow (Stages 1–3) functions end-to-end

This project focuses on the development of a distributed AI architecture and the practical deployment of local LLMs.

Mandatory Technical Requirements

1. Local LLM Execution

You must replace OpenRouter with **local inference** using **one** of the following:

- Ollama (**recommended**)
- GPT4All
- Llamafile
- Hugging Face
- LangChain

2. Distributed Architecture

- Each **team must run at least 3 LLMs on separate pc**
- The **Chairman LLM must run on a separate machine**
- All machines must communicate using **REST APIs** (e.g. Ollama API)
- **At least two council LLMs must run on different machines.**
    - **Solo project:** A single student may run all LLMs locally on one PC, provided sufficient hardware resources are available.
    - **Group of 2 students:**
        - The **Chairman LLM** must run on one PC.
        - All **council LLMs** must run on a second, separate PC.
    - **Group of 3 to 5 students:**
        - The **Chairman LLM** must run on one dedicated PC.
        - At least **two council LLMs must run on separate PCs**, with **each LLM running on its own machine**.

3. Chairman Separation

- The Chairman must:
    - Run as a **separate service**
    - Have its own model instance
    - Only synthesize responses (not generate first opinions)

👥 Team Requirements

- Teams of **1 to 5 students**

- All members must be from the **same TD group** e.g. All team members must belong to the same TD group (e.g., all from CDOF1). Students from different TD groups (e.g., CDOF1 and CDOF3) are not allowed to form a team.
- Collaboration and task distribution must be evident
- You may work alone but **not more than 5 members per team**

Deliverables

Each team must submit **one single submission** on **DVL** (in the *Final Project Submission* folder) **by the deadline of the final TD session** containing:

1. **Source Code** the complete refactored implementation of the **LLM Council**.
2. **Documentation**
    - A `README.md` file including:
        - Group member names and TD group number
        - Project overview
        - Setup and installation instructions
        - Instructions to run the demo
    - A **short technical report** describing:
        - Key design decisions
        - Chosen LLM models
        - Improvements made over the original repository
          If you want it **more concise** or **more formal**, I can refine it further.
3. **Live Demo**
    - End-to-end execution during the **last TD session**
    - Show:
        - Multiple machines
        - Council responses
        - Review stage
        - Chairman final answer

Evaluation Criteria during demo

- **Code Quality**: Clean structure, modularity, readability
- **Functionality**: Fully working 3-stage council workflow
- **Improvements**: Enhancements beyond the original repo
- **Documentation**: Clear setup guide, architecture, explanation
- **Teamwork & Demo**: Clear roles, smooth live presentation
- 📌 **Final evaluation will take place during the last TD session**

Optional Enhancement Ideas (Bonus)

Any additional frontend improvement beyond the base implementation will be considered a **plus**. Examples include, but are not limited to:

- Model health checks & heartbeat monitoring
- Token usage estimation
- Load and availability status per model
- Improved tab view:
    - Color-coded responses

    - Collapsible panels
    - Side-by-side comparison
    - Diff highlighting between outputs
- Model performance dashboard:
    - Latency per model and response-time tracking
    - Ranking results
    - Indicators for model status (running, idle, unavailable)
- Dark / Light mode UI
- Clear visualization of the council workflow (Stage 1 → Stage 3)

> *Any **original improvement or feature not listed above** that enhances usability, clarity, performance monitoring or user experience will be **positively evaluated**.*

Learning Outcomes

By completing this project, you will:

- **Work collaboratively in a team** to refactor and extend an existing codebase
- **Understand, analyze, and modify a real-world multi-LLM system**
- Design and implement **distributed AI systems**
- Deploy, configure and manage **local LLMs**
- Gain **hands-on engineering experience** beyond simple prompt engineering

Generative AI Usage & Penalties

- The use of **Generative AI tools (LLMs)** is **allowed** for this project.
- **Any use of Generative AI must be explicitly declared** in the project documentation.
- Teams must include a **"Generative AI Usage Statement"** specifying:
    - Which tools/models were used
    - For what purpose (e.g., code refactoring, documentation writing, debugging, design, etc.)
- This declaration must be included in the **project documentation** (e.g., in the `README.md` or technical report).

⚠️ Penalty Policy

- **Failure to declare the use of Generative AI** or attempting to hide such usage, will result in a **severe penalty**.
- If Generative AI usage is detected and **not disclosed** the project may receive a **significantly reduced grade or be considered non-compliant**.

> *Transparency is mandatory. Using Generative AI responsibly and declaring its use will **not** negatively impact your evaluation.*