MLE Take-Home Assignment: LLM Inference Benchmarking

## Part 1: Deploy an Open-Source LLM

Your first task is to set up a local inference server for an open-source Large Language Model.

- **Choose an Inference Server: Select one of the following high-performance inference servers:**
  - **vLLM**
  - **Text Generation Inference (TGI)**
  - **SGLang**
  - **Or any other open-source LLM serving framework you prefer.**
- **Choose a Model: Deploy the `meta-llama/Meta-Llama-3-8B-Instruct` model (or a comparable open-source model if you have technical constraints).**
- **Deliverable: In your `README.md`, briefly explain your choice of inference server. Include the commands or `Dockerfile` required to spin up your server with the chosen model. Ensure the server exposes an API endpoint for inference (e.g., a REST API compatible with OpenAI's format).**

## Part 2: Benchmark the Deployed Model

Next, you'll measure the performance of the LLM server you just deployed. 📈

- **Choose a Benchmarking Framework: Use a tool specifically designed for LLM performance testing. We recommend one of the following, but you are free to choose another:**
  1. **GuideLLM**
  2. **LLMPerf**
- **Define the Benchmark: Your benchmark should measure the following key performance metrics under varying loads (e.g., different numbers of concurrent users/requests):**
  1. **Throughput: The number of output tokens generated per second.**
  2. **Latency: The time-to-first-token (TTFT).**
  3. **ITL - Inter Token Latency**
  4. **E2E - End to end latency**
- **Deliverable: Document the steps and commands needed to run the benchmark against your server. Store the raw results in a structured format (e.g., CSV, JSON) within your repository.**

## Part 3: Visualize and Analyze the Results

Finally, present your findings in a clear and insightful way. The goal is to communicate the performance characteristics of your setup. 📊

- Create Visualizations: Generate at least two meaningful graphs from the benchmark data you collected. We suggest:
    - A graph showing throughput (output tokens/sec) vs. the number of concurrent requests.
    - A graph showing time-to-first-token (ms) vs. the number of concurrent requests.
- Provide Analysis: In your `README.md`, embed the generated graphs. Below the graphs, write a brief analysis (2-3 paragraphs) that answers the following:
    - What do the results tell you about the performance of your serving setup?
    - Where do you observe performance bottlenecks (e.g., does latency increase significantly after a certain number of users)?
    - What is one potential optimization you would explore next to improve performance?
- Deliverable: Your final `README.md` should contain the embedded visualizations and your written analysis. The code used to generate the graphs (e.g., a Python script using Matplotlib/Seaborn or a Jupyter Notebook) must also be included in your repository.

Submission Guidelines

- Repository: Please create a single public Git repository containing all your work.
- README: Your `README.md` should be the main entry point, clearly explaining how to set up the environment, run your code, and interpret the results.
- Code: Ensure your code is clean, well-commented, and easily runnable. Include a `requirements.txt` file or similar dependency list.