AI Code Challenge: Quantized Local Language Model with Retrieval Augmented Generation

## Challenge Overview

Welcome to the AI code challenge!

Bluestaq is forming an AI team. We will always do what is right even when it’s hard. This team will be high-speed and low drag. Like a well-coordinated elite unit, we set clear expectations, communicate seamlessly, and move together with precision and purpose, ensuring that every challenge is met with determination and excellence. We pick each other up and keep moving. If you want in on this team, approach this challenge with creativity, intensity, focus, and determination.

Your task is to develop a quantized local language model with Retrieval Augmented Generation (RAG) capabilities. The model should be able to run entirely on a laptop and interact via the command line. This includes the entire architecture – no cloud resources allowed. This challenge will test your skills in machine learning, natural language processing, and software development.

### Objectives

* Develop a Quantized Language Model: Implement or utilize a pre-trained language model that has been quantized to run efficiently on a laptop.
* Integrate Retrieval Mechanism: Implement a retrieval mechanism to augment the generation capabilities of the language model.
* Command Line Interaction: Create a command-line interface (CLI) to interact with the model.
* Robustness and Efficiency: Ensure the model is robust and efficient, capable of handling various queries within reasonable time and resource constraints. RAM and CPU usage will be monitored during interaction.

### Scope and Expectations

#### 1. Quantized Language Model

* Model Selection: Choose a suitable pre-trained language model that can be quantized or already is quantized. Bonus points for designing and implementing this and/or explaining why or why not it was implemented.
* Quantization: If possible, apply techniques to reduce the model size and improve inference speed, such as 8-bit or 16-bit quantization.
* Validation: Ensure the quantized model maintains acceptable performance compared to its original form. Bonus points for providing a small test set with evaluation criteria and results.

#### 2. Retrieval Mechanism

* Corpus Creation: Create or utilize an existing text corpus for retrieval purposes.
* Retrieval Algorithm: Implement a retrieval algorithm (e.g., BM25, dense retrieval using sentence embeddings, keyword vector search, or other approach that you see fit.) to fetch relevant documents or passages from the corpus based on a query.
* Integration: Combine the retrieval mechanism with the language model to enhance its generation capabilities. Bonus points for properly sourcing each generated chunk. If you use an empirical approach and provide those results, this will be heavily weighted in your assessment.

#### 3. Command Line Interface

* Input Handling: Design the CLI to accept queries from the user.
* Prompt Engineering: Designing and implementing intelligent methods to reduce uncertainty from the user such as asking questions for query reformulation and RAG will be heavily weighted in your assessment.
* Output Display: Display the generated responses in a user-friendly format.
* Error Handling: Implement error handling to manage invalid inputs or unexpected behaviors.
* Guardrails: Design and implement constraints on what topics can and cannot be discussed with the model.

#### 4. Robustness and Efficiency

* Performance Testing: Test the model to ensure it runs efficiently on a standard laptop with limited resources. Assume Bluestaq laptop specifications at a maximum.
* Response Time: Aim for a response time that balances speed and accuracy, ideally under a few seconds per query. Quantized models under 4GB can achieve this performance on Bluestaq laptops.
* Documentation: Provide clear documentation on how to set up, run, and interact with the model. “Time-to-local-host" is going to be an important factor in this assessment. Ideally, a shell script that can be run on a Linux OS for a complete install will be considered the gold standard. It is OK to assume a certain version and distribution of Linux.

### Deliverables

1. Code Repository**:** A Bluestaq Bitbucket repository containing all the source code, organized and well-documented. I will set up the private repository for you and me once you have confirmed your interest. This way your challenge solution cannot be copied.
2. Model Files: Pre-trained and quantized model files or API instructions necessary to install and run the application.
3. Command Line Interface: The CLI tool for interacting with the model.
4. Documentation: Comprehensive documentation covering:
   1. Instructions for setting up the environment and dependencies. Shell script that automates this end-to-end is highly desirable and will be weighted in your assessment.
   2. How to run the CLI tool.
   3. Examples of usage and expected outputs. Experimental results on evaluation are highly desirable and will be weighted in your assessment.
   4. Description of the retrieval mechanism and how it integrates with the language model. An architecture diagram highly preferred so we can walk through it during the 1-on-1 challenge submission debrief. We will be whiteboarding...
   5. Any additional features or considerations. We will have a 1-hour whiteboard discussion on your implementation, limitations, and future directions.

### Evaluation Criteria

The implementation should meet the specified objectives and perform as expected, demonstrating correctness. Efficiency is crucial, with the model running effectively on a Bluestaq laptop while maintaining acceptable performance and response times. The CLI should be user-friendly and well-documented, ensuring usability. Innovation in quantization, retrieval, or overall design approaches will be highly valued. Additionally, the solution must handle a variety of inputs gracefully, demonstrating robustness and reliability.

If you are selected, you will join the Bluestaq AI team full-time.