Machine Learning Engineer – LLM Task Assignment

Company: Artikate Studio, Artikate Private Limited
Task Title: LLM-Powered Fact Checker with Custom Embedding-Based Retrieval
Submission Deadline: 24 hours
Offer: Based on overall performance and final interview

Objective

Build a lightweight system that can analyze a short news post or social media statement, extract key claims or entities, and verify them against a vector database of verified facts using a **Retrieval-Augmented Generation (RAG)** pipeline. The final output should classify the claim as:

- ✅ **True**
- ❌ **False**
- 🤷 **Unverifiable**

Input Sample

> "The Indian government has announced free electricity to all farmers starting July 2025."

Your system should extract the key claim ("free electricity to all farmers"), check it against trusted data, and return a verdict with reasoning.

What You Need to Build

1. Claim/Entity Detection

- Use NLP/transformer models (e.g., spaCy, HuggingFace) to extract key claims, statements, or named entities from input text.

2. Trusted Fact Base

- You will be provided with or can create a small database (CSV/text) of 30–50 short, **verified statements or government press releases**.

Example: https://www.pib.gov.in/ViewRss.aspx

- You must chunk and embed these using a model like `sentence-transformers`, OpenAI embeddings, etc.

3. Embedding & Retrieval System

- Store the embedded facts in a **vector database** (e.g., FAISS, ChromaDB, Pinecone).
- For each extracted claim, retrieve the **top-k most similar facts**.

4. LLM-Powered Comparison

- Construct a prompt to send to an LLM (e.g., OpenAI GPT-3.5/4, Cohere, or open-source like Mistral).
- Prompt should **compare the original claim against retrieved facts**, and classify the result into one of the three verdicts with reasoning.

5. Final Output

The response should include:

```
{
  "verdict": "Likely True",
  "evidence": ["Retrieved statement 1", "Retrieved statement 2"],
  "reasoning": "The retrieved statements match the main claim closely, including the timeline and scope of the policy."
}
```

Bonus (Optional but Appreciated)

- Build a minimal **Streamlit/Gradio UI**
- Include confidence scoring or a "Was this helpful?" feedback toggle
- Use a local LLM for comparison instead of OpenAI
- Filter vague/unverifiable claims using a scoring threshold

What We're Evaluating

- NLP and embedding-level thinking
- Your understanding of how to engineer an LLM system (not just prompt it)
- Clean, modular Python code
- Ability to chain together ML/NLP components into a working pipeline
- Practicality of your solution: can it work in the real world?

Deliverables

- A GitHub repo or zipped folder with your:
    - Python code (Jupyter Notebook or `.py` files)
    - Clean README with setup & usage instructions
    - Sample input/output files
- A **5–7 minute video walkthrough** (YouTube unlisted link or Google Drive)
- Streamlit/Gradio link or screenshots of the app

Submission Instructions

Please send your solution and video walkthrough link to wellfound chat.
Use subject line: **LLM Engineer Task – [Your Name]**

Deadline

24 hours

If you have any questions during the task, feel free to email or message on Wellfound chat.

We look forward to seeing how you approach this challenge!