Plato – RAG System Engineering Assignment

This assignment evaluates your ability to engineer a reliable and deterministic Retrieval-Augmented Generation (RAG) system for Plato's AI hiring engine. You are provided with:

1. 1. `universal_jobs_shard_01.jsonl.gz` – 500 jobs with 10 applicants each. Each applicant includes a full CV and structured JSON evaluation.
2. 2. `generate_universal_jobs_dataset.py` – Script that generated the dataset (do not reuse this logic directly).
3. 3. `README_universal_jobs_dataset.txt` – Overview of the schema and generation strategy.

Objective: Build a RAG pipeline that takes a resume and a job description and produces a score using the provided strict JSON prompt and structure. Your system must: Embed and retrieve job + resume context Integrate with an LLM using the provided evaluation prompt Ensure scoring consistency: output must stay within **1% margin** across repeated runs Apply any custom resume parsing rules if provided Produce responses in the **exact JSON format** described

Prompt Format:

Use the detailed scoring prompt provided separately. Your system should dynamically inject resume, job, and parsing rule variables. Responses must include **no extra text**—only the structured JSON.

Evaluation Criteria:

Technical correctness and LLM integration Scoring accuracy and JSON fidelity Consistency (<1% margin of error across runs) Explanation quality and realism (do red flags and deductions make sense?) Effective use of embeddings, vector store, and retrieval logic Clarity and documentation (code + README)

Submission Instructions:

- Provide a GitHub repo or zipped folder containing: - `/data`: Any processed data files or intermediate chunks - `/src`: Your main code (retrieval, prompting, scoring) - `/notebooks`: Optional experimentation or evaluation notebooks - `README.md`: Setup instructions, model selection, and how determinism was enforced - Include 2–3 test cases showing identical scoring across multiple runs

Good luck! Your work on this assignment will reflect how well you can build advanced, production-ready AI systems under real-world constraints.