📘 GENAI FACTORY – RAG (RETRIEVAL-AUGMENTED GENERATION) EXECUTION FLOW

🎯 MỤC TIÊU

Hiểu rõ toàn bộ quy trình kỹ thuật của RAG trong **GenAI Factory**, từ giai đoạn xử lý offline (Indexing) đến xử lý online (Inference), và vai trò của từng thành phần trong hệ thống.

⚙️ 1. GIAI ĐOẠN 1 – INDEXING PIPELINE (OFFLINE)

Mục tiêu của giai đoạn này là **biến dữ liệu thô (Raw Documents)** thành **Tri thức có thể truy vấn (Vector Embeddings)**.

Bước	Hành động Kỹ thuật	Cấu phần Factory	Kết quả
1. Tải Dữ liệu (Loading)	Job MLOps tải tài liệu thô (PDF, DOCX, JSON) từ S3 hoặc Storage.	`infra/storage/s3_bucket_config.yaml`	Dữ liệu gốc.
2. Phân đoạn (Chunking)	Chia nhỏ tài liệu dài thành các đoạn nhỏ (ví dụ: 512 tokens) có chồng chéo (overlap).	Job tiền xử lý (qua Dockerfile.trainer).	Danh sách các đoạn văn bản đã được tối ưu hóa.
3. Tạo Vector (Embedding)	Chuyển từng đoạn văn bản thành Vector nhúng (Embedding Vector) bằng mô hình `.embed()`.	`shared_libs/atomic/llms/openai_llm.py` (async_embed)	Danh sách Vector và Metadata.
4. Đánh chỉ mục (Indexing)	Lưu trữ Vector và Metadata vào **Vector Database** để phục vụ tìm kiếm.	`infra/storage/vector_db_config.yaml`	Vector Index (Tri thức đã lập chỉ mục).

🔷 **Kết quả:** Hệ thống tạo ra một kho tri thức đã được số hóa, sẵn sàng cho truy vấn ngữ nghĩa.

⚡ 2. GIAI ĐOẠN 2 – INFERENCE PIPELINE (ONLINE)

Đây là giai đoạn **thực thi thời gian thực**, diễn ra khi người dùng gửi câu hỏi qua API.

Bước	Hành động Kỹ thuật	Cấu phần Factory	Vai trò & Kết quả
A. Tiếp nhận	Nhận `user_query` và kiểm tra an toàn đầu vào.	`assistant_service.py`, `safety_pipeline.py`	Query đã được xác thực.

Bước	Hành động Kỹ thuật	Cấu phần Factory	Vai trò & Kết quả
Query			
B. Query Embedding	Nhúng `user_query` thành **Query Vector** bằng mô hình nhúng.	`openai_llm.py` (async_embed)	Vector hóa câu hỏi.
C. Retrieval	Truy vấn Vector DB để tìm k Vector gần nhất (Semantic Search).	`retriever_tool.py` (BaseTool Hardening)	Truy xuất k đoạn tài liệu liên quan.
D. Re-ranking (Tùy chọn)	Lọc và sắp xếp lại các đoạn trích để lấy top tài liệu chất lượng nhất.	`rag_pipeline.py`, `retriever_config.yaml`	Grounded Documents.
E. Augmentation (Prompt Construction)	Kết hợp Hướng dẫn (Instruction) + Grounded Docs + Query thành **Prompt hoàn chỉnh**.	`rag_prompt.py`	Prompt cuối cùng gửi tới LLM.
F. Generation (LLM Inference)	Gửi Prompt đến LLM để tạo phản hồi có nền tảng.	`openai_llm.py` (async_generate, Retry/Fallback)	Câu trả lời chính xác, có cơ sở dữ liệu.
G. Output Validation	Kiểm tra lại Toxicity và PII trước khi gửi người dùng.	`safety_pipeline.py`	Câu trả lời an toàn và có ngữ cảnh.

🔷 **Kết quả:** Người dùng nhận được phản hồi chính xác, có cơ sở và an toàn.

🧩 TỔNG KẾT

RAG là một **chuỗi xử lý đa tầng** được Hardening với **Async I/O + Resilience + Retry/Fallback**, biến dữ liệu nội bộ tĩnh thành **tri thức động**, sẵn sàng phục vụ truy vấn theo thời gian thực.

✅ **Indexing Pipeline:** Chuẩn bị tri thức (Offline).
✅ **Inference Pipeline:** Truy vấn và phản hồi (Online).

Kết quả là một hệ thống **GenAI Production-Grade**: nhanh, chính xác và đáng tin cậy.