📘 **Canvas: Phân Tích Chuyên Sâu – Kỹ Thuật Chịu Lỗi (Resilience) trong GenAI Factory**

🎯 Mục tiêu: Biến Hiệu Suất thành Độ Tin Cậy

Trong khi kỹ thuật **Bất đồng bộ (Async I/O)** mang lại hiệu suất cao, thì **Resilience Engineering (Kỹ thuật chịu lỗi)** lại biến GenAI Factory trở thành một hệ thống **cấp độ sản xuất (Production-grade)** có khả năng chịu đựng và phục hồi khi các dịch vụ bên ngoài gặp sự cố.

Cơ chế Resilience được triển khai thông qua: - `BaseLLMWrapper` → Retry, Backoff, Exception Handling. - `LLMFactory` → Fallback & Circuit Breaker logic.

I. 🔁 Cơ chế Retry (Thử lại) & Backoff trong BaseLLMWrapper

Mục tiêu là **xử lý tự động các lỗi tạm thời (Transient Errors)** mà không cần can thiệp thủ công từ lập trình viên.

Thành phần	Vai trò kỹ thuật	Cơ chế hoạt động
Exceptions	Định nghĩa phân loại lỗi (`LLMRateLimitError`, `LLMServiceError`).	Khi `openai_llm.py` nhận phản hồi lỗi từ API (429/5xx), nó raise các ngoại lệ tương ứng.
@RETRY_STRATEGY	Bộ kích hoạt Retry (decorator áp dụng cho `_protected_async_call`).	Thư viện `tenacity` giám sát phương thức này. Khi gặp lỗi, nó tự động thử lại theo quy tắc được định nghĩa.
Exponential Backoff	Kiểm soát tần suất thử lại.	Thời gian chờ tăng theo cấp số nhân (1s, 2s, 4s, 8s...) kèm **Jitter (ngẫu nhiên)** để tránh hiện tượng quá tải đồng loạt (Thundering Herd Problem).

💡 **Giá trị Hardening:** Việc gói logic này trong `BaseLLMWrapper` giúp mọi LLM được tạo bởi Factory có khả năng **tự phục hồi**, không cần các lập trình viên pipeline viết thêm code xử lý lỗi.

II. 🔄 Cơ chế Fallback (Dự phòng) trong LLMFactory và BaseLLMWrapper

Cơ chế này hoạt động như **Circuit Breaker (Bộ ngắt mạch)** để xử lý các lỗi nghiêm trọng không thể khôi phục bằng Retry (ví dụ: lỗi 4xx cố định, cấu hình sai, hoặc API bị sập hoàn toàn).

Bước	Thành phần	Vai trò kỹ thuật
1. Cấu hình	`LLMFactory.create_llm()`	Khởi tạo `primary_llm` (ví dụ: OpenAI) và `fallback_llm` (ví dụ: Hugging Face nội bộ).
2. Liên kết	`primary_llm.set_fallback_llm(fallback_llm)`	Gắn mô hình dự phòng vào mô hình chính thông qua phương thức trong `BaseLLMWrapper`.
3. Kích hoạt (Switch)	`BaseLLMWrapper.async_generate()`	Sau 5 lần Retry thất bại, khối `try...except` sẽ chuyển hướng cuộc gọi sang `fallback_llm`.
4. Kết quả	Trả về phản hồi dự phòng.	Hệ thống trả về kết quả từ mô hình nhỏ hơn nhưng vẫn đảm bảo **dịch vụ không gián đoạn**.

💡 **Giá trị Hardening:** Fallback loại bỏ rủi ro phụ thuộc vào một nhà cung cấp duy nhất (**Vendor Lock-in**) và đảm bảo **tính sẵn sàng (Availability)** của hệ thống.

III. 🧩 Tóm tắt Chiến lược Resilience

Cơ chế	Mục tiêu	Lớp bảo vệ
Retry	Xử lý lỗi tạm thời (mạng, rate limit).	*Tự phục hồi cho lỗi nhỏ.*
Fallback	Xử lý lỗi nghiêm trọng hoặc lỗi vendor.	*Lớp dự phòng và ngắt mạch (Circuit Breaker).*
Async I/O	Đảm bảo hiệu suất trong mọi điều kiện.	*Nền tảng mở rộng và không nghẽn.*

🚀 Kết luận: Kỹ Thuật Xây Dựng Độ Tin Cậy trong GenAI Factory

Sự kết hợp giữa **Retry**, **Exponential Backoff** và **Fallback** biến Factory thành một hệ thống **chịu lỗi, ổn định và đáng tin cậy**:

✅ *Tự động phục hồi lỗi tạm thời (Retry).*
✅ *Duy trì dịch vụ với cơ chế dự phòng (Fallback).*
✅ *Hoạt động ổn định 24/7, sẵn sàng cho môi trường doanh nghiệp.*

Tóm lại: Resilience chính là “xương sống vô hình” giúp GenAI Factory vận hành liên tục – ngay cả khi thế giới bên ngoài gặp lỗi.