📘 **Canvas: Vai Trò của API Token, RAG, Prompt Engineering và Fine-Tuning trong GenAI Factory**

🎯 **Mục tiêu:** Hiểu cách số lượng API Token ảnh hưởng đến chất lượng, khả năng mở rộng và cách RAG, Prompt Engineering, Fine-Tuning hỗ trợ bù đắp.

I. 🔢 Ảnh hưởng của Số Lượng Token API

Số lượng Token API bạn mua từ các nhà cung cấp (như OpenAI hoặc Anthropic) quyết định trực tiếp **chất lượng** và **khả năng mở rộng** của GenAI Factory ở hai khía cạnh: 1. **Kích thước Ngữ cảnh (Context Window)**
2. **Thông lượng (Throughput)**

A. 🧠 Giới Hạn Ngữ Cảnh (Context Window)

Giới hạn Token Thấp	Ảnh hưởng đến Chất lượng Hệ thống
Giảm độ chính xác RAG	Không thể chèn đủ tài liệu đã truy xuất, dễ bỏ sót ngữ cảnh quan trọng và tăng rủi ro ảo giác.
Mất ngữ cảnh hội thoại	Lịch sử hội thoại (`memory_service.py`) phải tóm tắt thường xuyên, khiến mô hình quên chi tiết quan trọng.
Giới hạn lập luận Agent	Agent (ví dụ: ReAct) phải giới hạn `max_steps` hoặc `timeout`, làm giảm khả năng xử lý vấn đề phức tạp.

B. ⚡ Giới Hạn Tốc Độ Yêu Cầu (Rate Limit / Throughput)

Giới hạn Token/Phút Thấp	Ảnh hưởng đến Khả năng Mở rộng
Nút thắt cổ chai	Khi lưu lượng truy cập tăng, dễ gặp lỗi 429 (Too Many Requests).
Hiệu suất kém	Cơ chế Async I/O không thể phát huy hết, khiến thông lượng giảm và độ trễ tăng.

II. 🧩 RAG & Prompt – Cơ Chế Bù Đắp cho Giới Hạn Token

RAG (Retrieval-Augmented Generation) và **Prompt Engineering** là hai kỹ thuật giúp tối ưu hóa việc sử dụng Token, đảm bảo chất lượng mà vẫn tiết kiệm chi phí.

A. 🗄️ RAG – Bù cho Giới Hạn Ngữ Cảnh

Kỹ thuật RAG	Vai trò Bù Đắp Token
Truy xuất thông minh	Thay vì chèn toàn bộ tài liệu (30,000 tokens), RAG chỉ chọn các đoạn trích liên quan (500–1000 tokens).
Tập trung và chính xác	Đảm bảo mỗi Token được sử dụng đều có giá trị, duy trì factual grounding và giảm ảo giác.

B. 📝 Prompt Engineering – Tối Ưu Hóa Ngữ Cảnh & Chi Phí

Kỹ thuật Prompt	Vai trò Bù Đắp Token
Tóm tắt bộ nhớ	`memory_orchestrator.py` tóm tắt hội thoại dài, giữ lại ngữ cảnh cốt lõi với ít Token hơn.
Few-Shot Learning	`fewshot_prompt.py` sử dụng vài ví dụ chất lượng cao thay vì hướng dẫn dài dòng.
Dự đoán Token (Cost Control)	`base_prompt.py` ước lượng lượng Token trước khi gọi API để kiểm soát chi phí và tránh lỗi vượt giới hạn.

III. 🧬 Fine-Tuning – Chiến Lược Nâng Cao về Chất Lượng

Fine-Tuning (Tinh chỉnh) là quá trình huấn luyện lại mô hình nền tảng (GPT-3.5, Llama 2,...) trên tập dữ liệu nhỏ, chất lượng cao của riêng bạn để:

Mục tiêu	Tác dụng
Chuyên môn hóa lĩnh vực	Dạy mô hình hiểu sâu về quy trình hoặc ngôn ngữ ngành (ví dụ: phê duyệt tín dụng).
Định dạng / Phong cách đầu ra	Buộc mô hình phản hồi theo tone hoặc định dạng cố định (JSON, ngôn ngữ pháp lý,...).

🔷 *Fine-Tuning thay đổi trực tiếp trọng số mô hình, khác với RAG/Prompt chỉ tác động tạm thời khi inference.*

IV. 🔄 So Sánh Giữa Token, RAG, Prompt và Fine-Tuning

Tiêu chí	Fine-Tuning	RAG	Prompt Engineering	API Token
Mức độ thay	Vĩnh viễn (thay	Tạm thời	Tạm thời (định hướng	Không thay đổi

Tiêu chí	Fine-Tuning	RAG	Prompt Engineering	API Token
đổi	đổi trọng số).	(thêm dữ liệu vào input).	output).	(chỉ mở rộng khả năng xử lý).
Dữ liệu sử dụng	Dữ liệu có cấu trúc (input-output).	Tài liệu tham khảo.	Hướng dẫn, ví dụ.	Kích thước đầu vào/ra.
Thời điểm áp dụng	Offline (trước triển khai).	Online (thời điểm query).	Online (thời điểm query).	Mọi lúc.
Giá trị cốt lõi	Cải thiện năng lực nội tại mô hình.	Bổ sung tri thức cập nhật.	Điều khiển hành vi mô hình.	Kiểm soát khả năng và tốc độ.

V. 🤝 Mối Quan Hệ Tương Hỗ Trong GenAI Factory

Fine-Tuning không thay thế mà **bổ trợ** cho RAG và Prompt.

Kỹ thuật	Vai trò Bổ Trợ
Fine-Tuning → RAG	Giúp mô hình hiểu tài liệu truy xuất chính xác và ít nhiễu hơn.
Fine-Tuning → Prompt	Giảm độ dài Prompt, tiết kiệm Token mà vẫn duy trì phong cách và định dạng đầu ra.

Tổng kết:
- **RAG/Prompt = Kỹ thuật Hiệu Suất** → Giải quyết vấn đề bằng cách tối ưu đầu vào.
- **Fine-Tuning = Kỹ thuật Chất Lượng** → Giải quyết vấn đề bằng cách cải thiện bản thân mô hình.

VI. 💡 Khi Nào Sử Dụng Mỗi Chiến Lược

Chiến lược	Thời điểm Sử dụng	Chi phí	Thời gian
API Token	Luôn cần thiết để inference.	Chi phí vận hành (OPEX).	Tức thì.
RAG/Prompt	Nên dùng đầu tiên để tối ưu nhanh.	Chi phí token vận hành.	Tức thì.
Fine-Tuning	Khi Prompt/RAG không đạt yêu cầu độ chính xác hoặc	Chi phí phát triển (GPU,	Dài hạn (chu kỳ R&D).

Chiến lược	Thời điểm Sử dụng	Chi phí	Thời gian
	định dạng.	labeling).	

VII. 🚀 Khuyến Nghị Chiến Lược cho GenAI Factory

1. **Bắt đầu với RAG + Prompt Engineering** để đạt hiệu quả nhanh và tiết kiệm chi phí.
2. **Theo dõi chất lượng & độ ổn định** dưới tải thực tế.
3. **Thực hiện Fine-Tuning** khi cần tối ưu phong cách, định dạng hoặc độ chính xác.

💡 *Mua Token = Quyết định OPEX (chi phí vận hành).*
💡 *Fine-Tuning = Quyết định CAPEX (đầu tư phát triển / R&D).*

Cân bằng hợp lý giữa Token, RAG, Prompt và Fine-Tuning chính là chìa khóa giúp dự án GenAI Factory tiến hóa từ chatbot thử nghiệm thành **nền tảng trí tuệ nhân tạo cấp doanh nghiệp**.