📘 GENAI FACTORY – QUY TRÌNH VẬN HÀNH CỦA AGENT (REACT LOOP)

🎯 MỤC TIÊU

Hiểu rõ cách **Agent** vận hành giúp bạn kiểm soát hoàn toàn khả năng giải quyết vấn đề của hệ thống **GenAI**.
Trong **GenAI Factory**, Agent được triển khai theo mô hình **ReAct (Reasoning and Acting)** — cho phép LLM không chỉ suy nghĩ mà còn **tương tác với thế giới bên ngoài** thông qua các **Tools**.

⚙️ 1. NỀN TẢNG: VÒNG LẶP REACT (THOUGHT → ACTION → OBSERVATION)

Agent (ReActAgent) hoạt động dựa trên **vòng lặp ra quyết định liên tục**, được định nghĩa trong `loop()` (base_agent.py) và mở rộng trong `react_agent.py`.

Giai đoạn	Vai trò	Chức năng (LLM/Agent)	Output của LLM (Quan trọng!)
1. Thought (Suy nghĩ)	LLM Reasoning (Lập luận)	LLM phân tích câu hỏi, các công cụ có sẵn và lịch sử, sau đó xác định bước tiếp theo.	Thought: “Tôi cần lấy dữ liệu từ DB.”
2. Action (Hành động)	Agent Acting (Thực thi)	Agent Parse đầu ra của LLM → trích xuất tên Tool và tham số, sau đó gọi `run()` của Tool.	Action: `SQLTool(query='SELECT Q3_Revenue')`
3. Observation (Quan sát)	Tool Result (Kết quả)	Tool thực thi và trả về kết quả (ví dụ: dữ liệu, lỗi, thông báo). Agent ghi lại kết quả này.	Observation: “Kết quả DB trả về là 10 triệu USD.”
Loop	State Update	Observation được thêm vào `self.history` và gửi lại cho LLM trong Prompt tiếp theo.	Lặp lại từ Thought.
Final Answer	Task Complete	Khi LLM cho rằng đủ thông tin, nó kết thúc vòng lặp bằng câu trả lời cuối cùng.	Final Answer: “Doanh thu Q3 là 10 triệu USD.”

🧠 2. CÁC THÀNH PHẦN KIẾN TRÚC THAM GIA

Để vòng lặp ReAct hoạt động, **Agent** cần sự phối hợp của **bốn thành phần chính** – tất cả đều dựa trên các Interface chuẩn hóa:

🔷 A. Agent (Người Điều phối)

- **Lớp:** `ReActAgent` kế thừa `BaseAgent` (`base_agent.py`).
- **Chức năng:** Chứa logic vòng lặp (`loop`), lưu trữ lịch sử (`self.history`), và Parse đầu ra của LLM để kích hoạt Tool.
- **Cấu hình:** Khởi tạo với `BaseLLM` (bộ não) và danh sách `BaseTool` (các công cụ hành động).

🔷 B. LLM (Bộ não)

- **Lớp:** `OpenAILLM` hoặc `HuggingFaceLLM` kế thừa `BaseLLM` (`base_llm.py`).
- **Chức năng:** Cung cấp năng lực suy luận (Reasoning). Nhận Prompt (câu hỏi, lịch sử, mô tả Tool) → trả về Thought/Action.

🔷 C. Prompt (Ngôn ngữ Giao tiếp)

- **Lớp:** `ReActPrompt` kế thừa `BasePrompt` (`react_prompt.py`).
- **Chức năng:** Buộc LLM tuân thủ định dạng nghiêm ngặt (*Thought: ...: ...*). Nếu sai định dạng → Agent không Parse được hoặc cần cơ chế tự sửa lỗi.

🔷 D. Tool (Cánh tay)

- **Lớp:** Tool cụ thể (ví dụ: `SQLTool`, `CalculatorTool`) kế thừa `BaseTool` (`base_tool.py`).
- **Chức năng:** Tương tác với thế giới thực (Database, API, File System). Agent chỉ cần gọi `tool.run(input_data)`.

🔄 3. PHÂN TÍCH CHUYÊN SÂU: PHƯƠNG THỨC `loop()`

`loop(self, query: str)` là **trái tim của Agent**, mô tả luồng luân chuyển giữa các thành phần:

```
self.history = []
# 1️⃣ Khởi tạo ngữ cảnh: Câu hỏi + Mô tả Tools
context = self.react_prompt.render({...})
```

```
# [2] Vòng lặp chính (tối đa max_loops)
for _ in range(self.max_loops):
    # 3.1 Kích hoạt suy luận (Thought/Action)
    current_thought = self.llm.generate(context)
    self.history.append(current_thought)

    # 3.2 Kiểm tra điều kiện dừng
    if "Final Answer:" in current_thought:
        return current_thought.split("Final Answer:")[1].strip()

    # 3.3 Thực hiện Hành động (Act)
    action_name, action_input = self._parse_action(current_thought)
    observation = self.tools[action_name].run(action_input)

    # 3.4 Ghi nhận Quan sát (Observe)
    self.history.append(f"Observation: {observation}")

    # 3.5 Cập nhật ngữ cảnh cho vòng tiếp theo
    context = self.react_prompt.render({
        "agent_history": "\n".join(self.history),
        ...
    })
```

KẾT LUẬN

Agent không phải là LLM. Nó là một **phần mềm quản lý trạng thái (State Management Software)** sử dụng LLM như **bộ điều khiển logic (Control Logic)**.

Agent luân phiên chuyển quyền giữa **LLM (suy nghĩ)** và **Tool (hành động)**, giúp hệ thống:

✅ Giải quyết các vấn đề nhiều bước (multi-step).
✅ Tự động hóa quy trình phức tạp.
✅ Vượt ra khỏi giới hạn của LLM chỉ biết sinh văn bản.

> 💡 **Agent = Brain + Memory + Tools + Logic Loop → Autonomous Intelligence.**