DATE : 11 Jan 2026
By: MOHAMMED SIMAAN AFTAB
Gmail : simaanaftab467@gmail.com
Contact: +91 6366676678
Github: https://github.com/simaan467

AI DATA FETCHING AGENT(LLM BASED)

AI Data-Fetching Agent – Submission Explanation This project demonstrates a small but complete AI agent system designed to plan its actions, use tools, evaluate its own performance, and improve over multiple runs. It was intentionally kept lightweight and transparent so that any reviewer can understand what the agent is doing and why.

1. **What the Agent Does** : The agent accepts a user question, creates a step-by-step plan using an LLM, and then executes the plan using two tools: a Chroma vector database and a small external API. After collecting information, it combines the results into a final answer.

2. **Why It Makes Mistakes First :** The agent is allowed to make errors in early attempts. Each run is checked by an evaluator that looks for errors such as skipping a tool, calling tools in the wrong order, or ignoring tool output. All mistakes are logged in a simple JSON memory file.

3**. How the Agent Improves**: On future runs, the agent reads its past mistakes and adjusts its planning prompt. Over time, it begins to plan more carefully and avoids the earlier errors. This shows a clear improvement cycle and demonstrates that the agent is learning from experience.

4. **Setup Instructions:** These steps help run the project smoothly:

• You can download the full code from GitHub: https://github.com/simaan467/AI-Data-Fetching-Agent

• After downloading, install all required packages: pip install -r requirements.txt
• The project uses Google's Gemini API (free tier). Please use your own Gemini API key when running the agent.
• The ChromaDB will initially have no data. You can populate it by running: python seed.py
• After seeding, you can generate the final output using: python main.py

4**. Tools Used :**

• ChromaDB for local vector search

• A free city information API (Wikipedia Summary API)

• Google Gemini (gemini-flash-latest) for planning and reasoning

• A custom evaluator to check rule based correctness

• A simple error logging memory system

5. **Why This Satisfies the Assignment** The project shows an understanding of agent planning, tool orchestration, self evaluation, and incremental improvement. Each part of the agent is easy to follow, and the improvement over multiple runs is visible in the logs. This makes it a reliable demonstration of practical agent design.