Our First Approach

1

In our initial implementation we adopted LangChain to orchestrate multiple agents— Semantic Interpreter, Document Generator, Visualization Agent—leveraging memory, prompt chains, and planner tools. While powerful, this approach introduced significant complexity, especially due to:

- Deep abstraction layers → hard to trace execution flow
- Prompt interference and unpredictable fallback behavior
- Slow debugging cycles in chained-agent pipelines

That's why We moved to another solution: from LangChain to Swarm

Swarm

OpenAI

Swarm is an SDK developed by OpenAI (unofficial, but built around OpenAI's APIs), designed to create multi-agent systems in a simple and efficient way.

As our architecture stabilized into four well-defined agents **(Conversational, Prompt Engine, Data, Visualization + Explanation)**, we transitioned to Swarm for the final implementation:

Each agent is now a transparent, message-driven interface
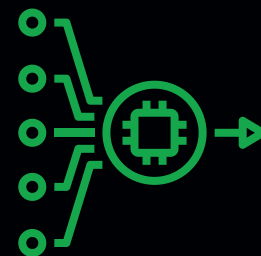
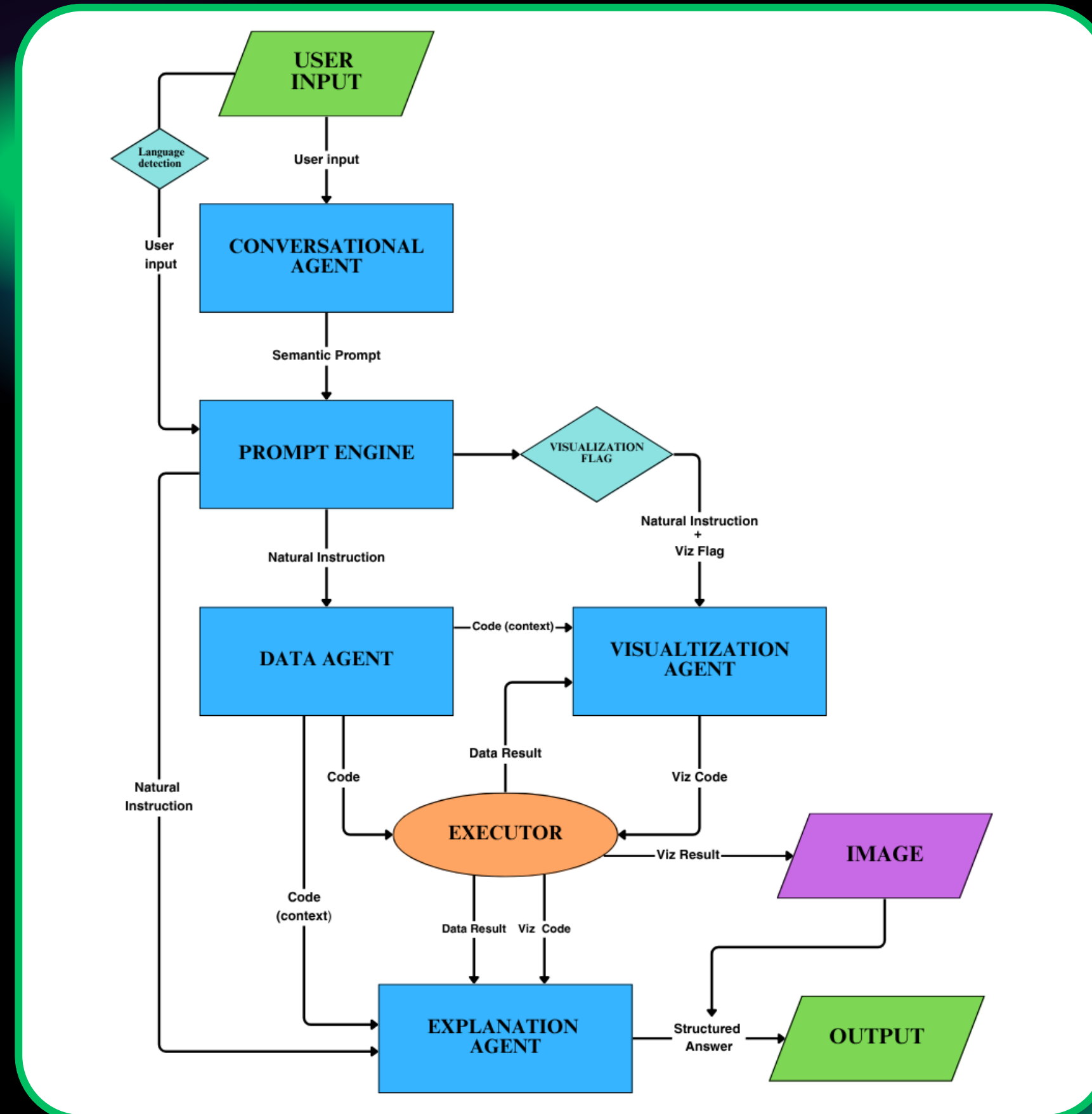Easier debugging, clearer logs, and fully deterministic outputs

Executed in sequence via a controlled pipeline **(Swarm.run()** logic in **SwarmAgentSystem)**

Full control over prompt formats and instruction routing

An in-depth walkthrough of our system's workflow, directly reflecting the underlying code architecture
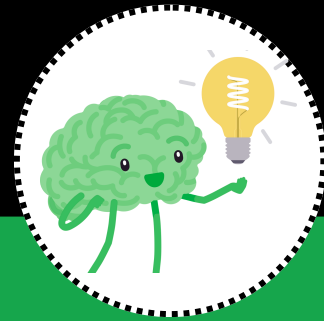
Enhancements

Memory Support

Swarm does not natively support memory injection like langchain. To overcome this limitation, a custom mechanism called **combined_input** was implemented, which concatenates previous instructions with new ones in a controlled manner. (A **minimal** but efficient temporary memory mechanism, well-suited for the specific needs of this project.)

This approach allows the system to efficiently recognize **follow-up** questions, maintaining the context of previous interactions.
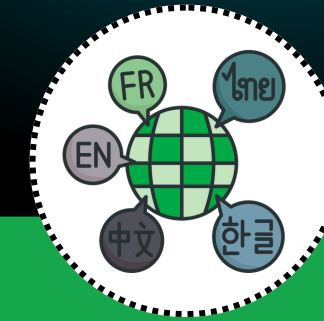
Multilingual Input Support

By using the **langdetect** library, the system automatically detects whether the question is in Italian or English and generates responses in the same language, **improving accessibility** for analysts and public sector officials.

This enables multilingual interaction, allowing users to engage with the system in their **preferred language** while receiving coherent responses

Agent Interface

**UI
Streamlit-Powered**

**User Friendly
Interface**

Choose language, ask questions, get insights & download charts

Multi-Agent System Data Analysis in NoiPA

REPLY

AI based System capable of analyzing Public Administration data in natural language: **Salaries, Income, Accesses, Commuting**

Choose the language and see some examples 💡

Powered by OpenAI · Swarm · LlamaIndex · Streamlit

💬 Fai una domanda sui dati/ ask something...

Link: https://triple3-rfr6bswwrbu2wpqfgwvwag.streamlit.app/

Performance and Evaluation
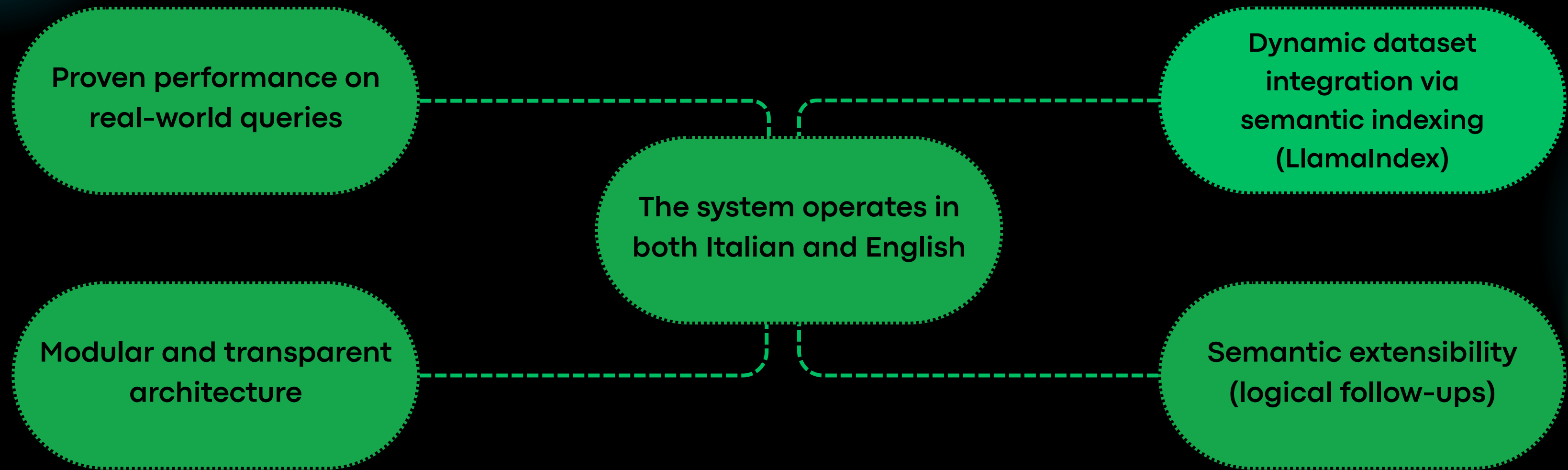
ACCESS

SALARY

COMMUTING

INCOME

Question	Agent Response	ChatGPT/EDA Response
How many users have accessed the system for each authentication method, grouped by age group?	☑	☑
What is the average minimum age for each administration?	☑	☑
How are the users distributed across the different values of the administration column?	☑	☑
How many users have a minimum age greater than 40 and a maximum age less than 60?	☑	☑
How are the users distributed across the different ranges of KMs?	☑	☑
Are there administrations in which all the users are commutants?	☑	☑
For the Lazio region, how are citizens distributed by percentage across their income ranges?	☑	☑
Are there regions where certain income brackets are disproportionately represented compared to the national average?	☑	☑

In the vast majority of cases, the agent accurately interprets the questions and provides correct answers. This holds true both for queries targeting individual datasets, as shown in the table, and for more complex questions that require integrating information across multiple datasets. The consistency and correctness of the responses demonstrate the agent's strong capability in handling diverse data analysis tasks effectively.

🎯 Point of strengths 🏋️

Proven performance on real-world queries

Dynamic dataset integration via semantic indexing (LlamaIndex)

The system operates in both Italian and English

Modular and transparent architecture

Semantic extensibility (logical follow-ups)

Weakness

Despite the high number of solid performances, we observed that the response time may not be immediate, and the system may struggle in fully applying multiple complex conditions.