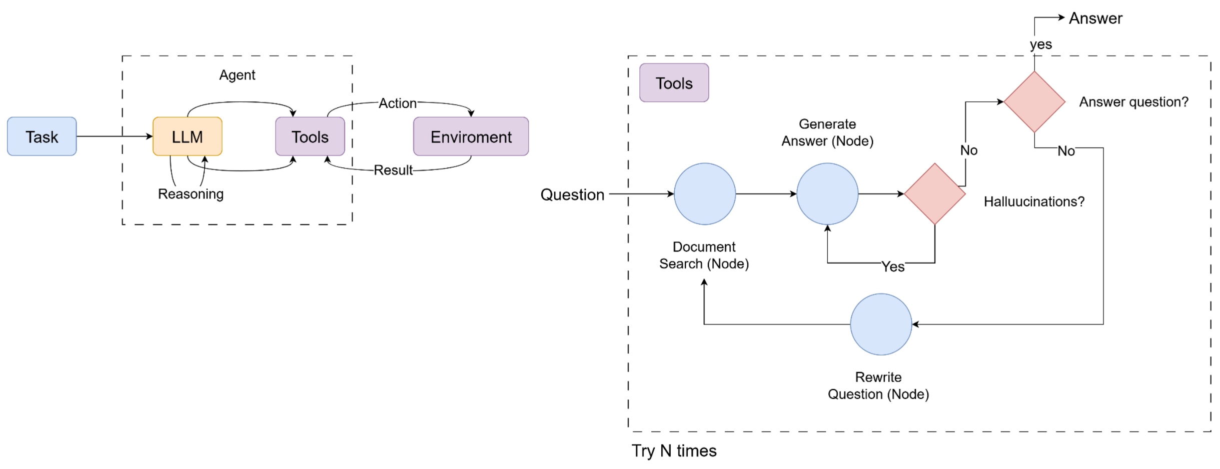
Agnos-Agent

Demo Access : *https://agnos-agent-hbdghnp5uwfrgeojdmfxhy.streamlit.app/*



Agnos-Agent is a chatbot designed for advisory and document-based question answering. It is built on the **ReAct framework**, utilizing **LangChain** and **LangGraph** as its workflow engine. This architecture allows the chatbot to follow a **Reason → Act → Observation** cycle, mimicking human-like decision-making, and continuously improving answer relevance and accuracy.

The system integrates **two main workflows**:

1. **ReAct Workflow** – handles reasoning, action selection, and iterative decision-making.
2. **RAG Workflow (Retrieval-Augmented Generation)** – handles document-based question answering with hallucination detection.

These workflows work together to produce **factually grounded** and **contextually relevant** responses.

ReAct Workflow

The ReAct workflow follows a **cyclic pattern**:
Thought → Action → Observation → Thought …

Purpose: Allows iterative reasoning and dynamic action selection.

1. **Task Input** – User submits a question.
2. **Reasoning Phase** – LLM generates a "thought" on how to approach the task.
3. **Action Selection** – LLM chooses a tool or action based on reasoning.
4. **Action Execution** – Selected tool interacts with the environment or dataset.
5. **Observation** – Result from action is returned to the LLM.
6. **Iteration** – Process repeats until the task is complete or maximum iterations are reached.

RAG Workflow (Retrieval-Augmented Generation)

Purpose: Generate answers based on retrieved documents while preventing hallucinations.

Key Steps:

1. **Document Search** – Retrieve relevant documents based on the user's query.
2. **Answer Generation** – Produce an answer using the retrieved context.
3. **Hallucination Check** – Ensure the answer is grounded in the retrieved documents.

4. **Answer Quality Check** – Validate if the generated answer addresses the question accurately.
5. **Question Rewriting (Optional)** – Reformulate the query if previous attempts fail.
6. **Finalize Response** – Send the verified answer to the user.

The workflow incorporates a **self-corrective flow** to ensure high-quality responses. If a hallucination or a low-quality answer is detected, the system iteratively decides whether to regenerate the answer or rewrite the query, improving the context for retrieval. This iterative process ensures that the final answer is both factually correct and contextually relevant. In addition, the workflow leverages **LLM as a judge**, where specialized LLM-based graders, such as `GradeHallucinations` and `GradeAnswer`, evaluate the generated response. These graders use keywords and the content of retrieved documents to score the answer, guiding the workflow to either finalize the response or trigger further corrective actions, thus maintaining accuracy and reliability.

Tools & Frameworks

Tool / Framework	**Purpose / Role**	**Notes / Integration**
Streamlit	User interface for chat interactions	Connects frontend to backend LLM workflows
LangGraph	Manages agent's flow using ReAct decision-making	Coordinates nodes, iteration, and conditional execution
LangChain	Orchestrates LLM calls, tools, and memory	Integrates GPT-4o, ChromaDB, and other tools into a single workflow
OpenAI (GPT-4o)	Core LLM for intent understanding and response generation	Used in both ReAct and RAG workflows
ChromaDB	Vector database for semantic embeddings	Stores embeddings of competencies, documents, and roles for retrieval efficiency