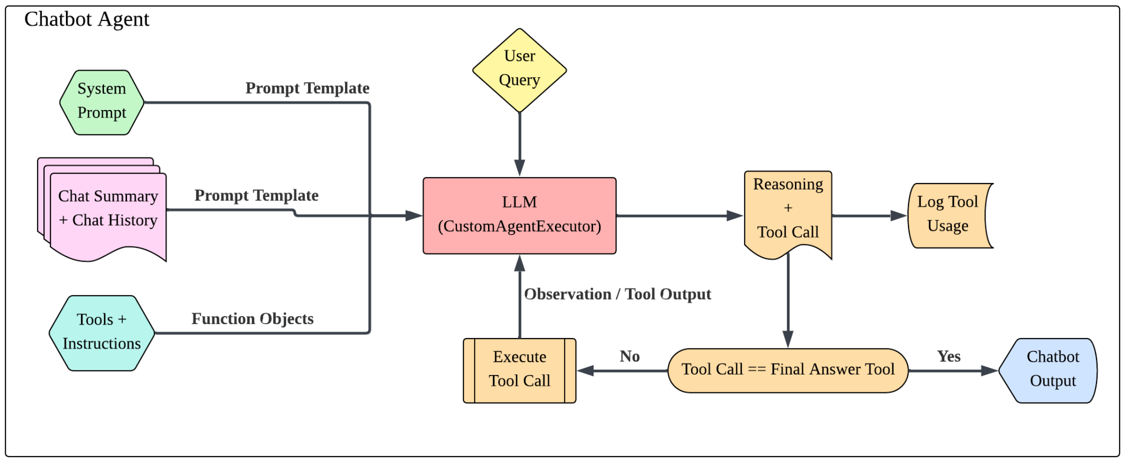

Overview

The chatbot uses a reactive planner loop managed by the "CustomAgentExecutor" class. It combines the system prompt, chat history, and tools to guide the LLM in deciding whether to call a tool or give a final answer. The executor runs tool calls, feed results back to the LLM, and stops once the "final_answer" tool is invoked and returns the chatbot's final response.

Intent recognition is LLM-assisted. The LLM suggests actions via tool calls rather than a comprehensive hand-coded intent classifier. This keeps the planner lightweight and robust across wide intent spaces.

1. Input assembly

- The controller constructs the prompt from three sources:
  - System Prompt (instructions)
  - Compacted chat summary + recent chat history (memory)
  - User query.
- It also exposes available tools as function objects with tool descriptions so the LLM can choose a tool.

2. LLM proposes next action

- The LLM reasons and returns a structured output:
  - Either a tool call (with tool name + arguments) or a final answer.
  - This is the agent's proposed action. (In my setup I instruct the LLM to always use tools this includes a tool to output the final answer.)

3. Planner decision / minimal verification

- The controller inspects the LLM output and decides the concrete action to take:
  - If the LLM requested a tool, the controller will execute the function tool call.
  - If the LLM returned a "final_answer" tool call, the controller will stop and return that as the assistant response.
  - If initial query is ambiguous or has missing information, the LLM may ask the user a clarifying question. The LLM will choose to use the "final_answer" tool to output the question.
- The controller performs light validation of tool arguments (sanitization and sanity checks using pydantic) before execution.

4. Execute tool & observe result

- When a specified tool is invoked (e.g., final_answer, multiplication, etc). The controller receives the tool output (observation), and appends it along with its tool call id to the agent scratchpad / memory.

5. Loop or finish
    - The controller feeds the observation back into the LLM (via the scratchpad), so the LLM can continue reasoning with the new evidence.
    - The loop repeats until the LLM issues a “final_answer” tool call. Then the controller returns the final response to the user.
6. Logging

- After each tool call request by the LLM. The tool call along with its details are logged and stored in memory for potential further reference.
- Details logged:
    - Session ID
    - Iteration (agent tool call iteration value)
    - Query
    - Tool Name
    - Tool Arguments
    - Tool Output
    - Timestamp