LangChain Agentic Workflow

Architecture and Working Explained

Introduction

- LangChain's agentic workflow combines language model reasoning with external tools to enable dynamic, multi-step problem solving.

Architecture Overview

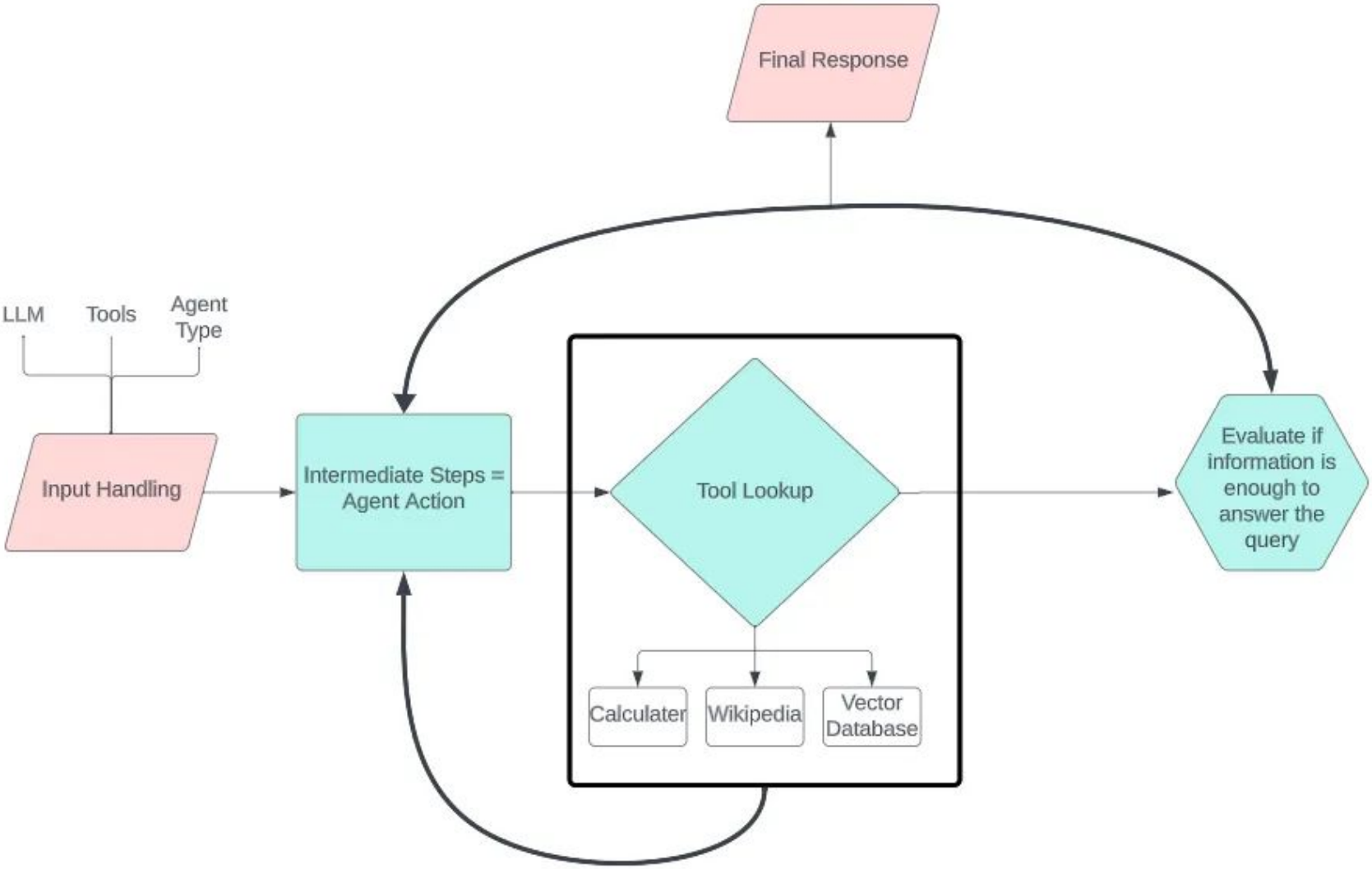
- 1. Agent: Core decision-maker using language models (e.g., GPT-4).
- 2. Tools: External functionalities like databases, APIs, or Python code.
- 3. Agent Executor: Orchestrates interaction between agent, tools, and user.
- 4. Toolkits: Groupings of tools for specific use cases.

Components in Detail

- 1. **Agent**: Uses prompts and logic to decide actions.
- 2. **Tools**: Perform specific tasks (e.g., Search API, Python execution).
- 3. **Toolkits**: Bundles of tools optimized for particular workflows.
- 4. **Agent Executor**: Iteratively manages the reasoning and tool invocations.

Working of Agent Executor

- 1. Initialization: Configures agent with tools and prompt templates.
- 2. Input Handling: Processes user queries.
- 3. Action Loop: Agent decides actions iteratively.
- 4. Tool Invocation: Executes tools and retrieves results.
- 5. Final Response: Combines results into a user-friendly output.


Final Response
LLM
Tools
Agent Type
Input Handling
Intermediate Steps = Agent Action
Tool Lookup
Calculater
Wikipedia
Vector Database
Evaluate if information is enough to answer the query

Illustrative Workflow: Product Return Example

1. **User Query**: `"Return my product which I ordered?"`
2. **Steps Taken by the Agent**:
   - **Input Handling**:
     - Identifies the intent to return a product and gathers required information (e.g., Order ID).
     - Prompt: `"Please provide your order ID so I can assist you with the return process."`
   - **Order Status Check**:
     - Calls `get_order_status_tool` with the corrected input format.
     - Example: `Action Input: order_id = 123`
     - Response: `"Delivered"`
   - **Return Policy Retrieval**:
     - Calls `search_return_policy` to fetch the return policy.
     - Example Output:

```kotlin
return_policy:
Our return policy allows returns within 30 days of purchase.
For Tablet, the return policy allows return within 10 days of purchase.
```

   - **Order Info Retrieval**:
     - Calls `get_order_info_tool` to retrieve order details using the order ID.
     - Example Output:

```arduino
{
  'customer_name': 'John Doe',
  'order_date': '2022-01-01',
  'products': ['Laptop', 'Monitor'],
  'total': 1500.0
}
```

   - **Return Eligibility Check**:
     - Evaluates the return conditions based on the return policy and order details.
     - Finds that the products (Laptop and Monitor) are not eligible for return because the order date is beyond the 30-day return window.
3. **Final Output**:
   - The agent informs the user: `"Unfortunately, the products in your order (Laptop and Monitor) are not eligible for return as the purchase date is beyond the 30-day return window."`

Features of LangChain Workflow

- 1. Dynamic Reasoning: Adaptive decision-making.
- 2. Multi-Tool Support: Seamless integration with diverse tools.
- 3. Error Handling: Graceful handling of tool failures.
- 4. Extensibility: Easy to add new tools or customize agents.

Common Use Cases

- 1. Question Answering: Retrieve information from documents or APIs.
- 2. Code Execution: Perform computations using Python or similar tools.
- 3. Database Querying: Execute SQL commands.
- 4. Web Search: Fetch real-time data.
- 5. Complex Reasoning: Solve multi-step problems.

Conclusion

- LangChain's agentic workflow combines the power of language models with external tools to solve complex problems effectively, offering flexibility, scalability, and advanced reasoning capabilities.