M-Lab CARTE AI Workshop 2025

Visual Workflow Building

Inspired by Flowise Agentic RAG Tutorial

Overview

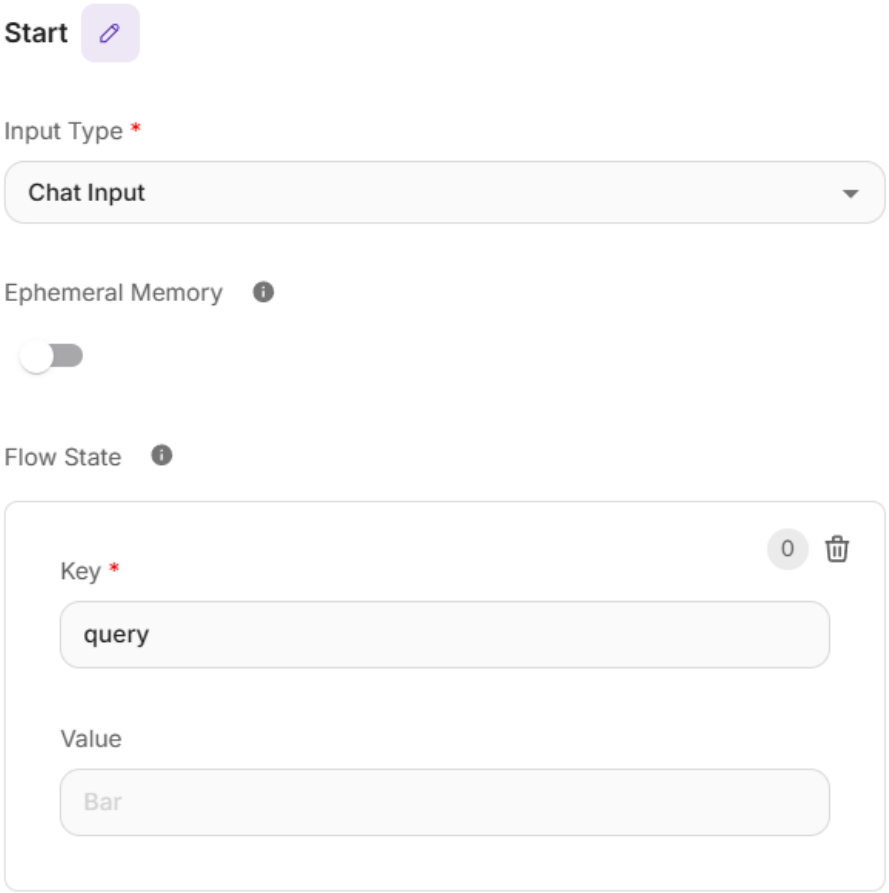
- Today you'll build a **RAG system** (Retrieval-Augmented Generation)
- RAG combines document search with AI generation
  - The model retrieves relevant information from your documents, then uses that context to answer questions
- This lets AI answer questions about *your* data, not just what it learned in training

What is Flowise?

- Flowise is a visual workflow builder for AI agents—drag and drop instead of writing code
- Lets you connect LLMs, document stores, prompts, and logic into complete AI applications
- Perfect for prototyping and testing AI workflows quickly
- Today: we've pre-configured instances for each table—one per team

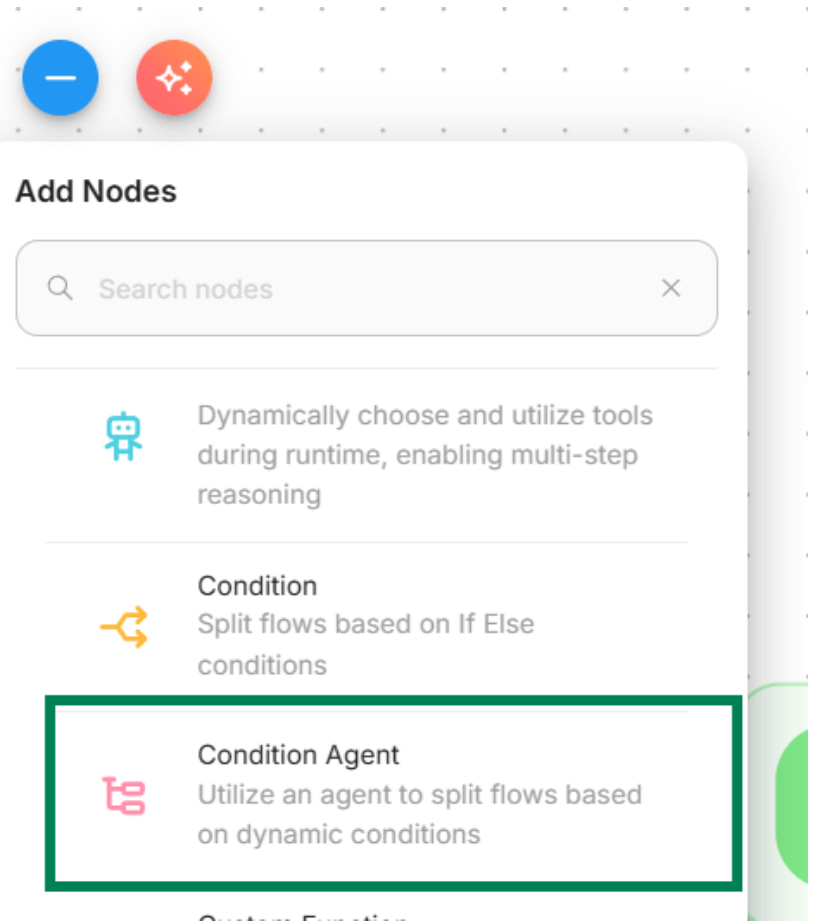
Step 1: Setting Up the Start Node

- Double click the Start node
  - This is the entry point for your workflow
- **Input Type**: Select 'Chat Input' to accept user questions
- **Flow State**: Add a state variable with key "`query`" and leave the value blank
  - The LLM will update this "`query`" variable as your request moves through the workflow



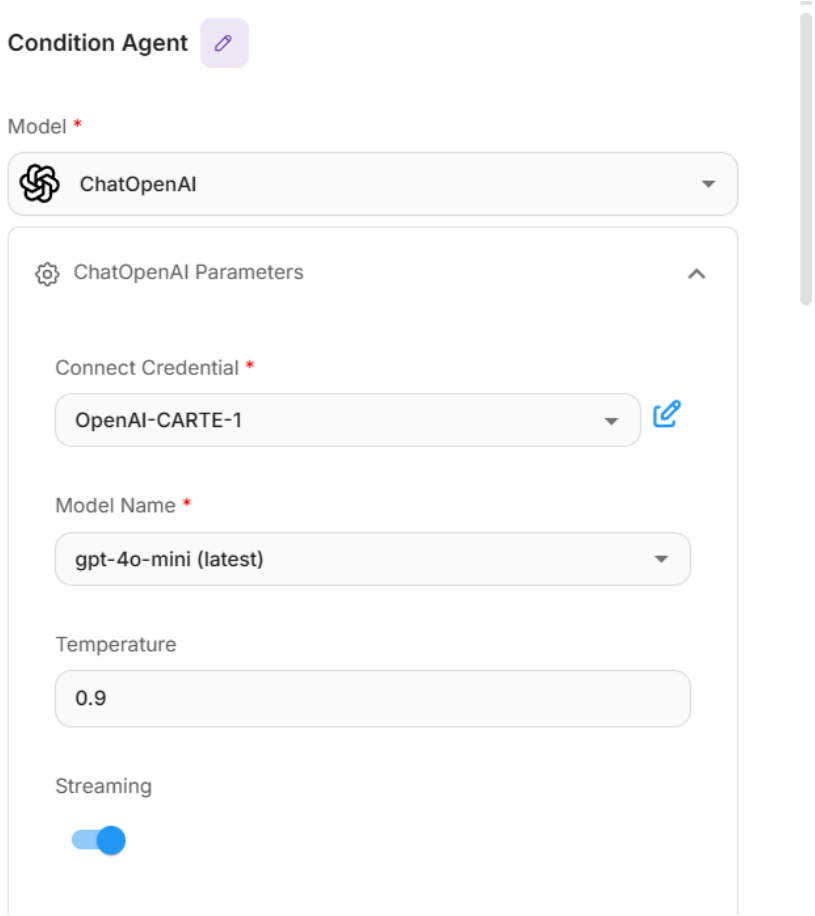
Step 2: Adding Query Validation

- Add a Condition Agent node and connect it to the Start node
- This checks whether to send the question to the RAG agent or not
- Drag it to the canvas, connect it, then double click to configure



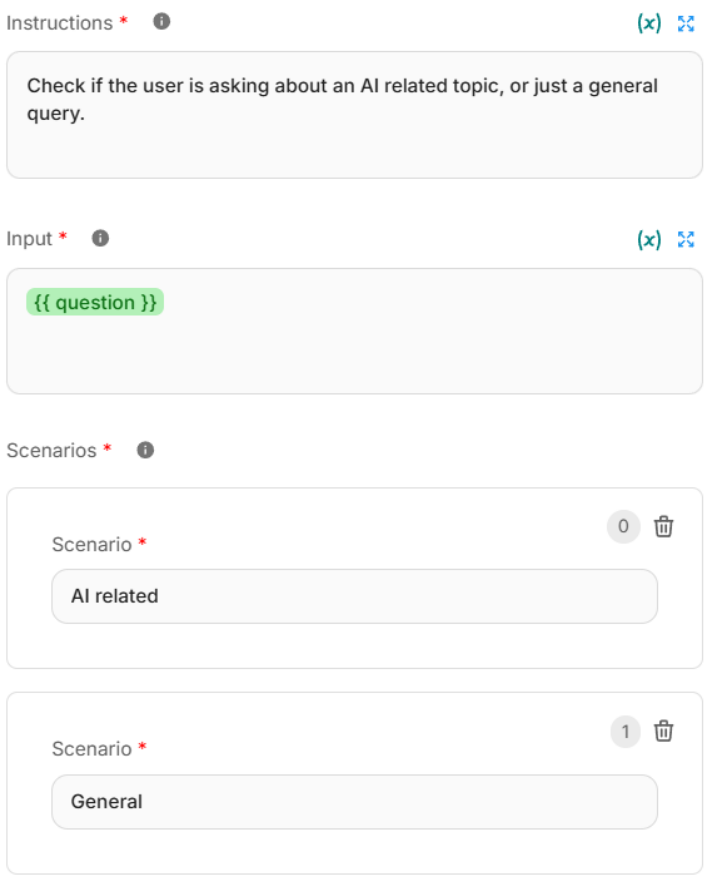
Step 2: Adding Query Validation

- We're using OpenAI's LLMs for this lab
- Select "ChatOpenAI" as the model
- Open the "ChatOpenAI Parameters" tab and set the credential to OpenAI-CARTE-x
  - For example: OpenAI-CARTE-1



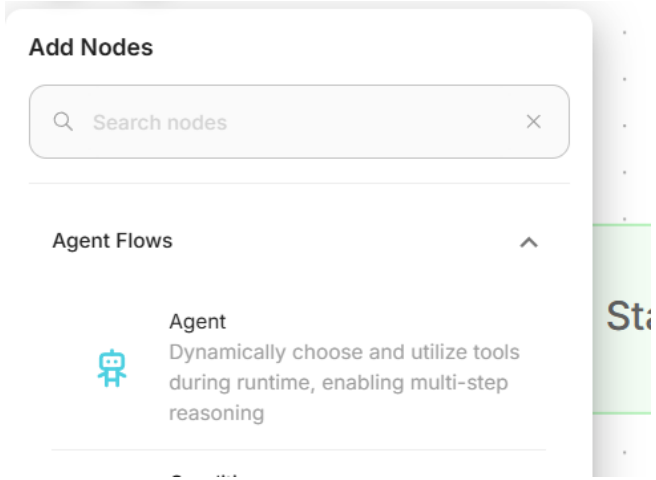
Step 2: Adding Query Validation (ctd)

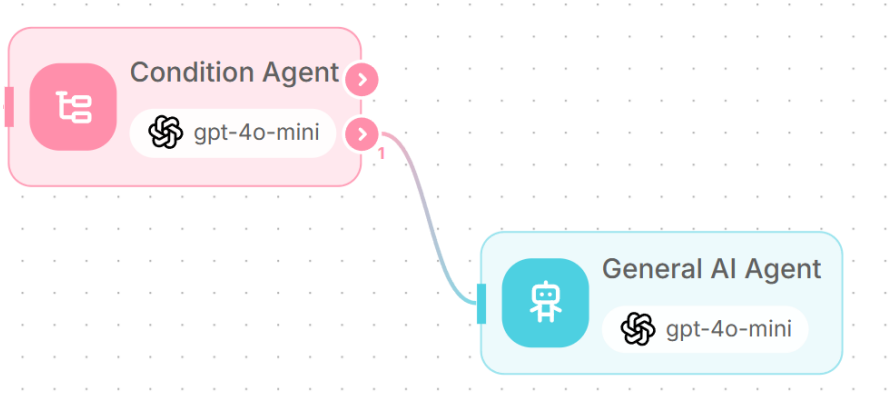
- **Instructions**: "Check if the user is asking about AI related topics, or just a general query."
- **Scenarios**:
  - AI Related
  - General
- This routes questions to different branches based on topic



Step 3: General Response Branch

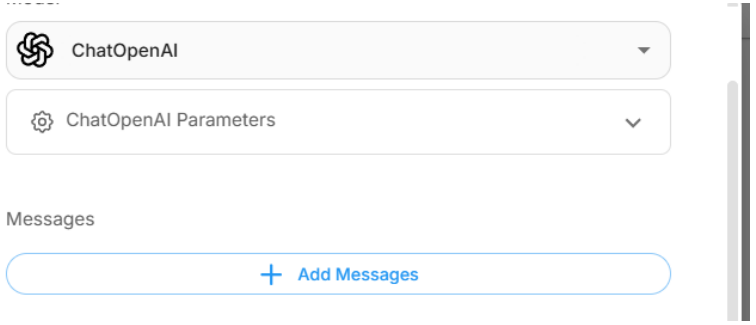
- Drag an Agent and connect it to the bottom of the Condition Agent
- This is your "general answer" agent

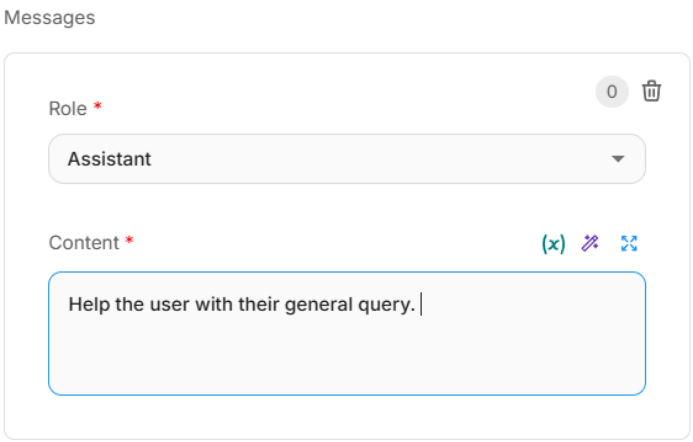




Step 3: General Response Branch (ctd)

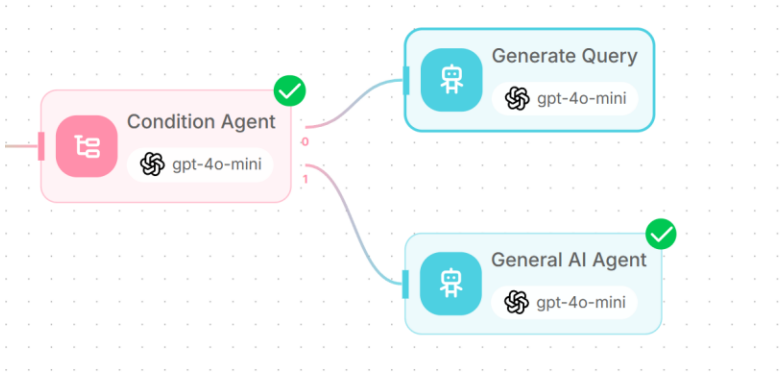
- Configure the model with ChatOpenAI
- Click Add Messages
- Set the role to Assistant and add a generic message in the content box
- This sets the system prompt for this agent

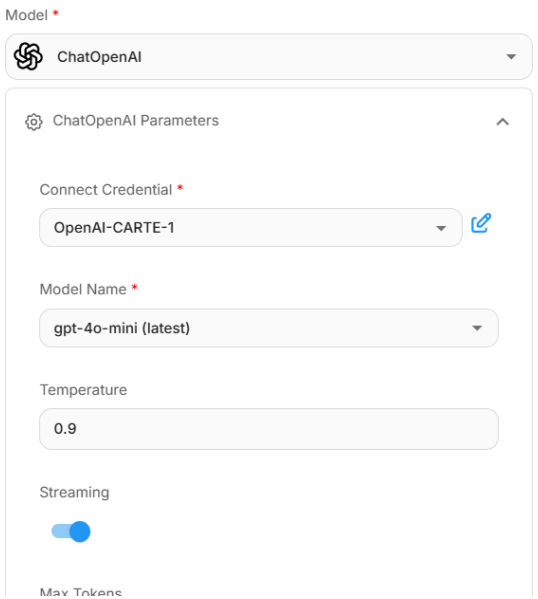




Step 4: Configure Query

- Add an agent to generate a search query
- Drag another Agent and connect it to the Condition Agent
- Configure the model like before (ChatOpenAI, add credentials)

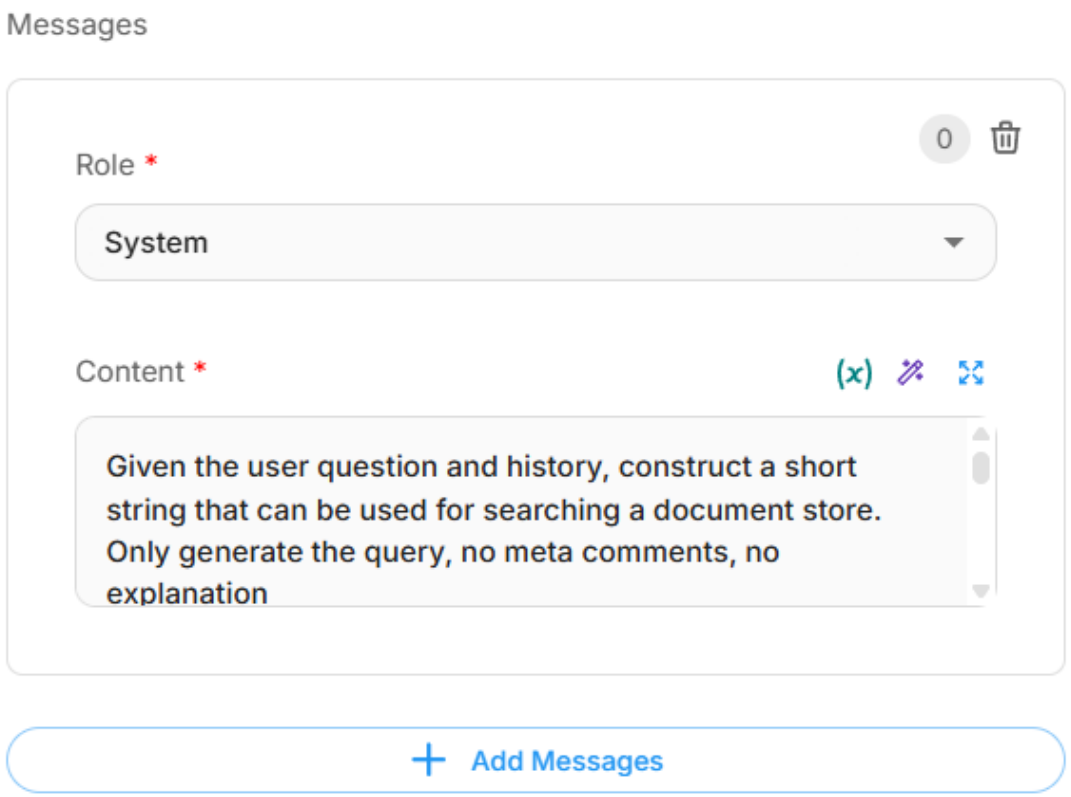




Step 4: Configure Query

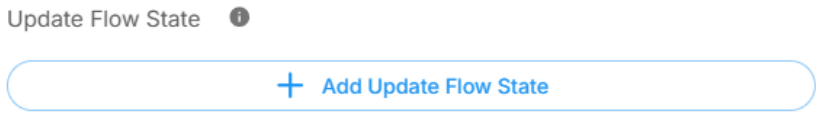
- Click “Add Message” and set the role to system.
- The content is as follows:

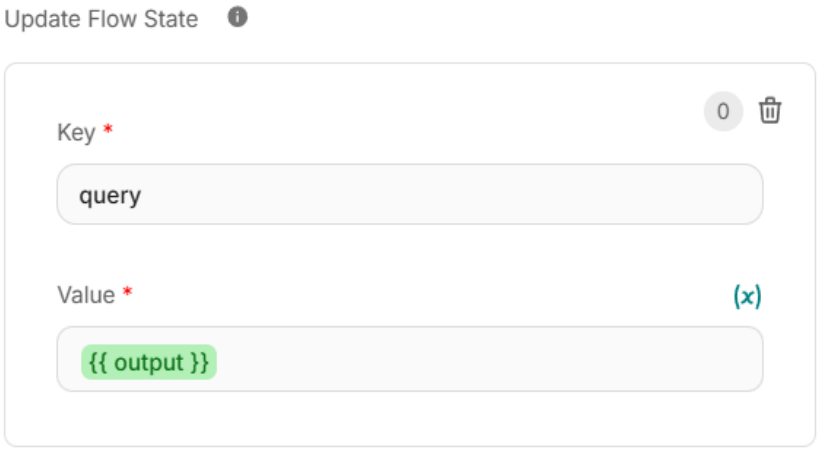
```
Given the user question and history, construct a short string
that can be used for searching a document store. Only generate
the query, no meta comments, no explanation

Example:

Question: what are the events happening today?

Query: today's event

Example:

Question: how about the address?

Query: business address of the shop

Question: {{ question }}

Query:
```



Step 4: Configure Query

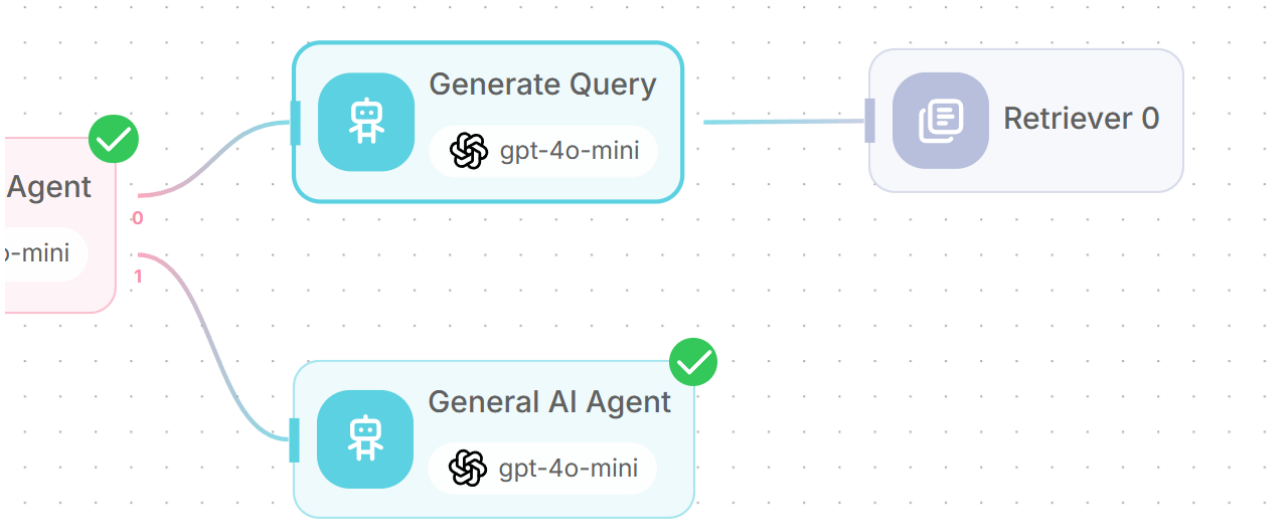
- Scroll down and click "Update Flow State"
- Add the node output to the flow state from Step 1
- Set Key to “`query`” and Value to “`{{ output }}`”
- This updates the query field using the LLM's output

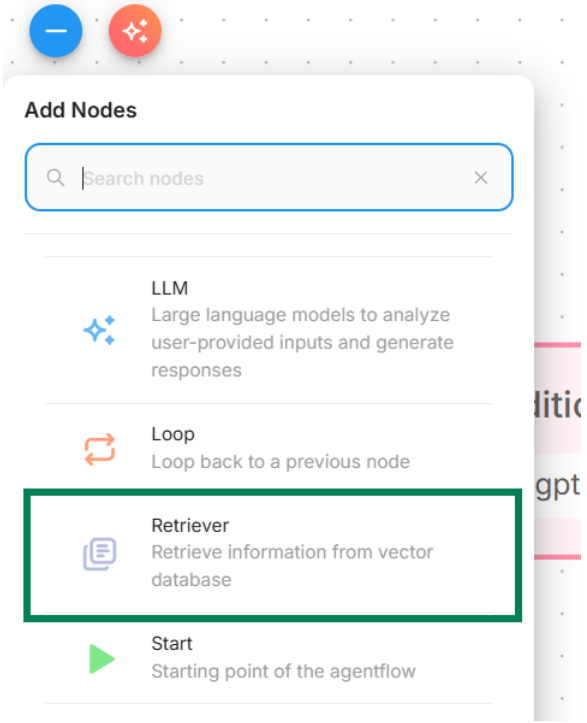




Step 5: Add Document Retriever

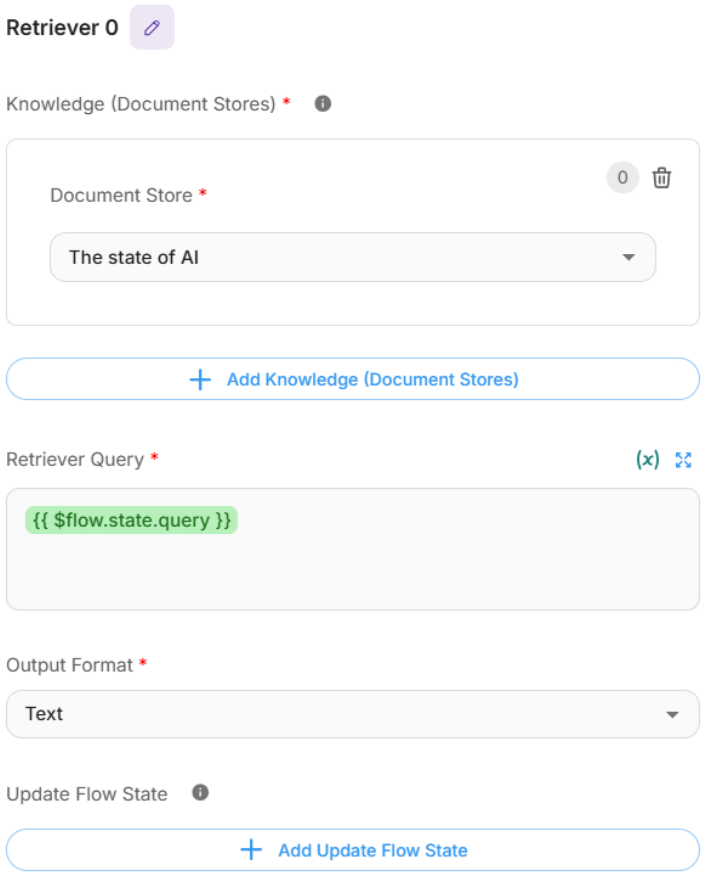
- Drag and Drop the “Document Retriever” onto the canvas.
- Connect it to the query generation agent.





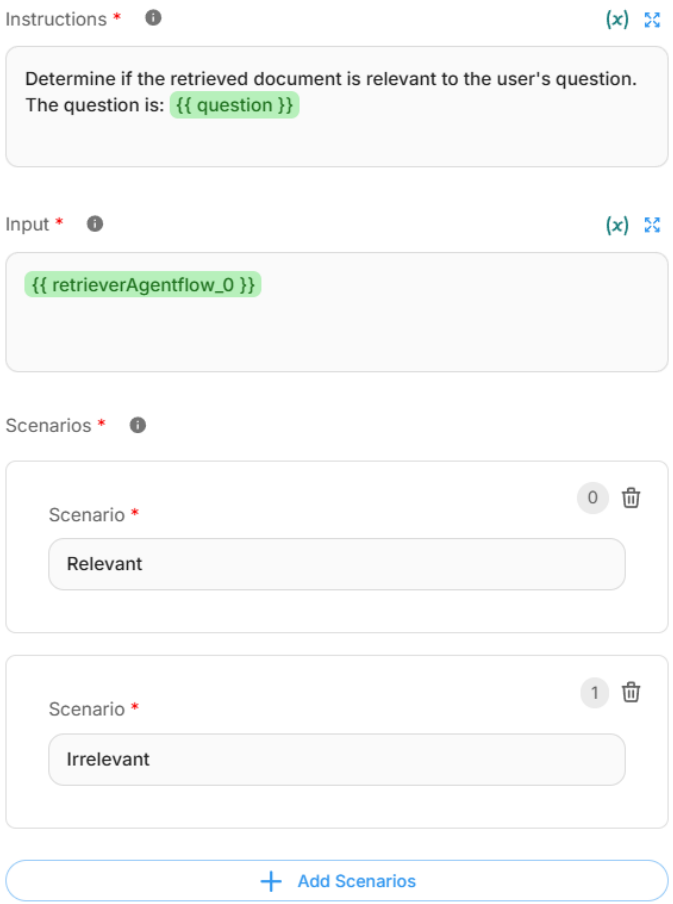
Step 5: Add Document Retriever

- Double click the retriever.
- Set the document store as “the state of AI”
- Set the retriever query as “`Retrieve information related to {{$flow.state.query}} or AI in general`”.
  - `$flow.state.query` is the query generated in the previous step.

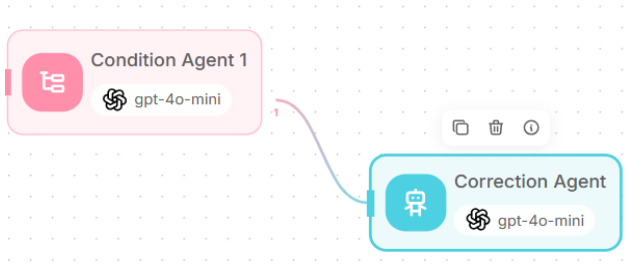


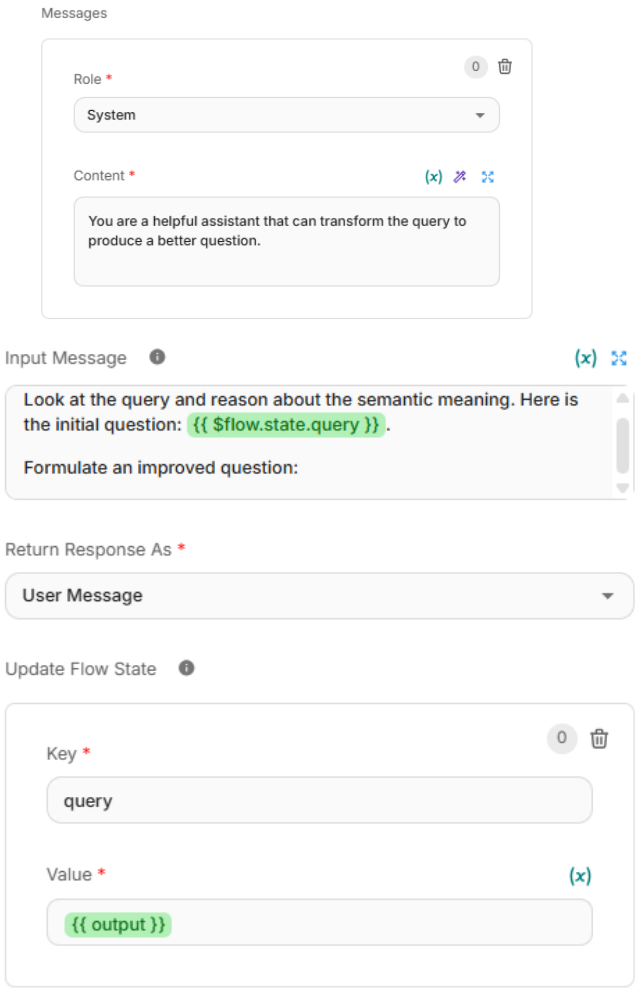
Step 6: Add Document Relevance Check

- Now we will setup a check to make sure the retrieved document is relevant.
- Create a condition agent and connect it to the retriever.
- Instruction: Determine if the retrieved document is relevant to the user's question. The question is: `{{ question }}`
- Input: `{{ retrieverAgentflow_0 }}`
  - Document fetched in previous step



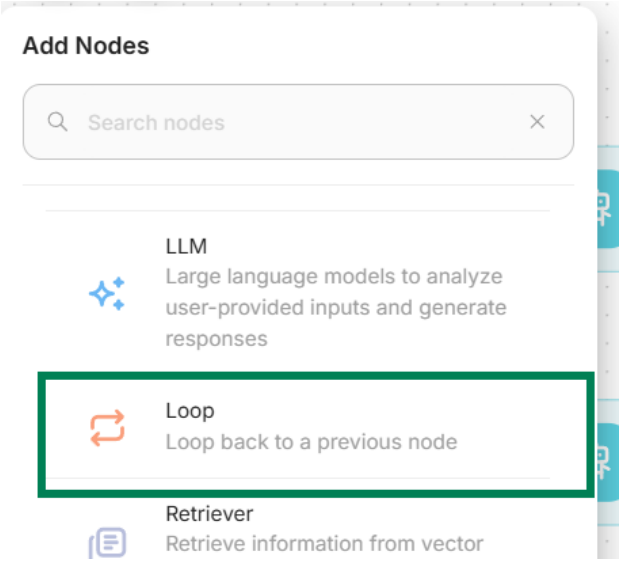
Step 7: Add Loop

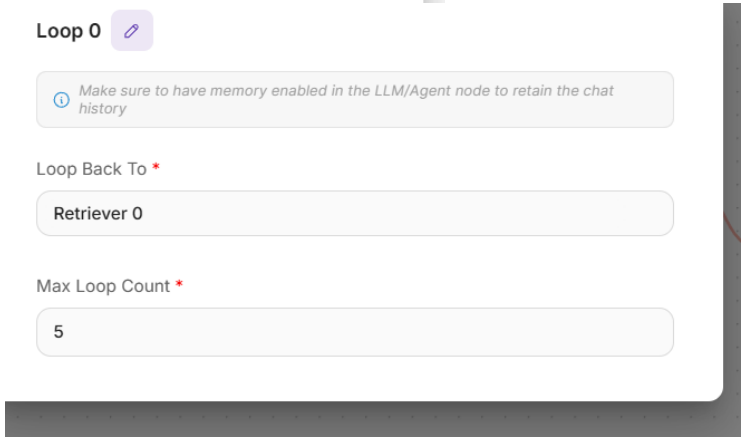




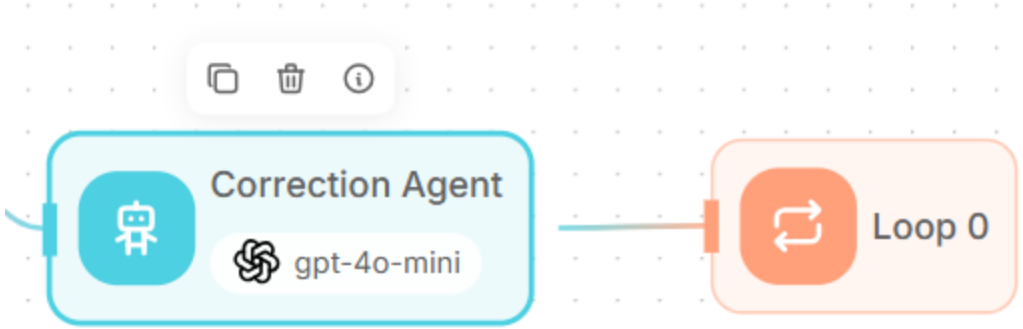
- To the negative condition, add a new Agent. This will edit the query for us.
- Provide the old query to the agent and have it update the flow state "query"
- Input message:
  - Look at the query and reason about the semantic meaning. Here is the initial question: {{ $flow.state.query }}.
  - Formulate an improved question:

Step 7: Add Loop





- To the agent, add a loop block and have it loop back to the retriever agent.
- Set Max loop count to 5.



Step 8: Add Final Output Agent

- To the positive condition, add a new agent.
- Add a system prompt / message:
  - "You are a helpful assistant”
- Add the user prompt with the previously found information:
  - Given the question: {{ question }}. And the findings: {{ retrieverAgentflow_0 }}. Output the final response