ISSN: 2473-4901
v11 n6461

Teaching Case

Building Advanced Multi-Agent Chatbots: A Low-Code/No-Code Approach

Frank Lee
flee@gsu.edu

Harmanprit Kaur
hkaur7@student.gsu.edu

J. Mack Robinson College of Business
Georgia State University
Atlanta, Georgia 30303, USA

Hook

This tutorial outlines a no-code methodology for developing a multi-agent chatbot for financial analysis.

Abstract

This tutorial provides a methodically organized and technically detailed guide for building a multi-agent financial analysis chatbot using Langflow on Astra. The chatbot uses Retrieval-Augmented Generation (RAG), modular agents, and real-time data integration to offer dynamic, context-aware decision support for financial queries. Although this tutorial is based on financial analytics, its architecture and design are broadly useful across many fields. The chatbot systematically employs specialized agents to research market trends, gather real-time data, and generate evidence-based investment insights.

Keywords: Multi-Agent Chatbots, Retrieval-Augmented Generation, Low-code platform

ISSN: 2473-4901
v11 n6461

Building Advanced Multi-Agent Chatbots: A Low-Code/No-Code Approach

Frank Lee and Harmanprit Kaur

1. INTRODUCTION

Low-code or No-code platform
Low-code or No-code platform
Low-code development employs a visual, model-driven approach that minimizes manual coding, enabling faster and more cost-effective application delivery compared to traditional programming (Aveiro et al., 2023). Similarly, no-code platforms allow users to build applications through intuitive drag-and-drop interfaces without writing code. Both approaches aim to accelerate process automation and scalability by reducing development time and resource demands. While these platforms enhance responsiveness to market needs and lower complexity, they may offer limited customization for complex, highly specialized solutions (Trigo et al., 2022).

Multi-agent chatbots
Multi-agent chatbots represent a significant advancement over traditional single-agent systems by enabling more dynamic and domain-diverse interactions (Aksar et al., 2023). Unlike single-agent models, which are typically confined to narrow tasks, multi-agent architectures integrate specialized agents to enhance conversational depth, responsiveness, and user engagement. However, this complexity necessitates advanced orchestration and coordination mechanisms to maintain coherence and optimize user experience (Calvaresi et al., 2023).

Although this tutorial is based on financial analytics, its architecture and design are broadly useful across many fields. The chatbot systematically employs specialized agents to research market trends, gather real-time data, and generate evidence-based investment insights.

Key system components include:
- Real-time market data retrieval via Yahoo Finance
- Natural language synthesis using OpenAI's GPT-4
- Contextual web research through Tavily AI Search
- Modular agent orchestration via Langflow's visual interface
- Scalable deployment with Astra DB's vector database for memory, RAG pipelines, and user-context tracking

Objectives
The main goals of this tutorial are:
- Learn how to navigate and set up Langflow within the Astra DB environment
- Retrieve and combine external financial data through public APIs
- Use prompt engineering strategies to guide LLM-based agents
- Build a fully functional Langflow-based chatbot for financial analysis
- Modify and deploy a domain-specific, interactive LLM assistant

2. UNDERSTANDING LANGFLOW ON ASTRA

Definition and Functionality
Langflow is a web-based, low-code platform for developing LangChain applications. It allows developers to visually build workflows using connected components like LLMs, prompts, retrievers, and tools, simplifying backend complexity with an easy-to-use node-based interface.

Why Use Astra LangFlow
- Langflow, integrated with Astra DB—a cloud-native, vector-supported NoSQL database, enables seamless deployment of scalable LLM applications.
- This combination supports: serverless, cloud-native operations; high-performance vector search for embedding memory; real-time data integration using RAG workflows; and deployment without backend engineering overhead.

Advantages of Astra Langflow
- Langflow on Astra provides several key advantages:
- Building complete AI workflows with visual components
- Managing memory securely and scalably using vector search

- Accelerating prototyping with reusable templates and no-code tools
- Supporting real-time data processing essential for financial and trend analysis

PROJECT ACTIVITY

The chatbot architecture consists of the following components:

Component	Purpose
Chat Input Node	Receives user queries
Tavily AI Search	Search engine optimized for LLMs and RAG, aimed at efficient, quick, and persistent search results
Yahoo Finance Tool	Fetched real-time financial data using yfinance
Calculator	Perform basic arithmetic operations on a given expression.
URL Tool	Load and retrieve data from specified URLs. Supports output in plain text, raw HTML, or KSON, with options for cleaning and separating multiple outputs.
Agent 1 (Researcher)	Uses OpenAI GPT to analyze Tavily search results. The agent is instructed to identify patterns, market sentiment, and trends from search data
Agent 2 (Financial Analyst)	Interprets Yahoo Finance output to extract financial indicators
SambaNova Node	Generates the final user-facing response
Combine Text Nodes (x2)	Two separate Combine Text nodes are used to merge the output of each agent into a unified input string
Chat Output	Displays the final chatbot response to the user in natural language

3. SETTING UP LANGFLOW ON ASTRA

Accessing Langflow

1. Navigate to Langflow (Figure 1; figures are in the appendix) on Astra by opening a web browser and visiting:

https://astra.datastax.com/langflow/d53da360-33e2-4eed-a97c-0b9bae5b0134

2. Log in to your DataStax Astra account.

   - If you don't have an account, please complete the registration process, which includes providing your full name, email address, and creating a password. Or log in with your Google account
   - Verify the email address and log in

3. Create a new Langflow project by selecting "New Flow" at the top-right of the interface
   - This will open the template selection screen designed to help users get started with different LLM-based workflows

4. Explore available templates

Astra Langflow offers a wide array of pre-built templates organized by use case and methodology. These include:

- Basic Prompting: A minimal workflow to perform prompt-based tasks using an OpenAI model
- Vector Store RAG: A prebuilt Retrieval-Augmented Generation setup for contextual search and memory
- Simple Agent: A basic agent framework for decision-based tasks

On the left side, categories like Assistants, Content Generation, Q&A, Prompting, and RAG are also available for further exploration.
These templates act as practical references, demonstrating that Langflow can be used to create assistants for financial analysis, research synthesis, content writing, and more.

Figure 2 illustrates Langflow's template selection screen showing "Basic Prompting," "Vector Store RAG," and "Simple Agent" templates, with the sidebar displaying categorized options under "Use Cases" and "Methodology."

5. Select "Blank Flow" for this tutorial

While templates are useful for inspiration or quick prototyping, this tutorial aims to help students understand Langflow in detail. Therefore, choose the "Blank Flow" option in the bottom-right corner of the screen. Building the chatbot from scratch will give insight into how each part functions, how data moves between modules, and how to expand the architecture for other uses beyond financial analysis.

4. BUILDING THE FINANCIAL ANALYSIS CHATBOT FROM SCRATCH

This section guides students in building a functional chatbot in Langflow using a visual, no-code interface. While our example emphasizes

financial data, the process can be applied across various fields. Students will learn to capture user input, generate LLM-based responses, retrieve real-time data from Yahoo Finance, Tavily AI Search, and combine these elements into an interactive system. By the end, each student will have created a basic chatbot capable of answering queries with both language generation and live data.

Integrating the Chat Input and LLM Component

Objective: Create a simple user flow where a typed message is processed by a large language model (LLM) to generate a coherent response. This forms the foundation of the chatbot. Before integrating with external data, we want the model to reply to user input using its built-in knowledge and reasoning abilities.

Step 1: Add the Chat Input Component

- In the left panel, click on Components.
- Click "Input" then drag and drop the "Chat Input" node into the workspace
- Drag the Chat Input node into the center workspace

Figure 3 shows the Chat Input in the Canvas, which allows users to enter queries into the chatbot interface, such as, "What is Apple's stock price today?"

Step 2: Add Yahoo Finance Node

- In the left panel, click on Components.
- Under Tools, drag and drop the Yahoo Finance node into the canvas

- In the node settings:

- Turn the Tool Mode on above Yahoo Finance
- Set the Data Method to "get_news" or "get_stock_summary" depending on the use case.
- Set the Number of Items to 5.
- Leave the Edit Tools box blank unless customization is required.

Figure 4 shows the Yahoo Finance component.

Step 3: Add Tavily AI Search

- In the left panel, click on Components.
- Under Tools, drag and drop the Tavily AI Search node into the canvas

- Create a Tavily AI Search API Key:

- Turn the Tool Mode on above Tavily AI Search
- Go to the following link, create an account, and create an API key: https://app.tavily.com/
- Input your API key into "Tavily API Key"

Figure 5 shows the Tavily AI Search website.

Figure 6 displays the Tavily AI Search component in Langflow with Tool Mode toggled on.

Step 4: Add URL node

- In the left panel, click on Data.
- Under Data, drag and drop the URL node into the canvas
- Turn the Tool Mode on above Tavily AI Search
- Click on the Controls
  - Toggle on "Output Format"
  - Toggle off "URLs" and "Max Depth"

Figure 7 presents the URL component with Tool Mode toggled on.

Figure 8 shows the Calculator Node and its available controls.

Step 5: Add Calculator node

- In the left panel, click on Tools.
- Under Tools, drag and drop the Calculator node into the canvas
- Turn the Tool Mode on above Calculator

Figure 9 illustrates the Calculator Node placed within the Canvas.

Step 6: Create a SambaNova API Key

- Sign in using your credentials, or click Sign up if you don't already have an account
- After logging in, you'll land on the Dashboard
- In the left-hand sidebar, click on API Keys
- Click the Create API Key button on the right

Important Reminder:

- API keys are partially hidden after creation, so copy and store the full key in a secure location (such as a password manager or encrypted notes app) immediately. You won't be able to view the full key again later.

Figure 10 shows the SambaNova Cloud interface.

Figure 11 displays the SambaNova "Create API Key" screen.

Step 7: Add Agent node 1

- In the left panel, click on Agents.

- Under Agents, drag and drop the Agent node into the canvas

Input the following:

- Model Provider: SambaNova
- Model Name: Meta-Llama-3.3-70B-Instruct
- Sambanova API Key: *Insert Personal API Key*
- Agent Instructions: *"You are a financial agent, You are a helpful assistant, and you have access to several tools. You must provide insights given the user question, you can get links to the latest news, and you must open each link and research relevant information.*

  {input}

  Think step by step. Do not call a tool if the input depends on another tool output you do not yet have. Do not try to answer until you get all the tools' output. If you do not have an answer yet, you can continue calling tools until you do. Your answer should be in the same language as the initial query. Ensure the tool calls are proper parsable json return the URLs used to create your response in the final answer"

Step 7.1: Connecting Chat Input & Tool Nodes to Agent 1

- Find the Chat Input node in your flow.
- Drag a connection from its output circle to the input port (labeled "input") on the left side of Agent 1.

This allows Agent 1 to receive the user's query from the chatbot interface.

- To equip Agent 1 with the right capabilities, you need to connect both the URL Toolset and Calculator Toolset nodes to the agent's Tools input port (the small circle labeled "tools" on the left side of the agent node).
- Drag a connection from the output circle (usually on the right side) of the URL Toolset node to the tools input circle on Agent 1.
- Repeat the same step for the Calculator Toolset node — connect its output to the tools input on Agent 1.
- Once connected, Agent 1 can now use both the URL retriever and the calculator in its workflow.
- Tip: If you don't see the "tools" port on Agent 1, make sure it's configured as a multi-tool agent that accepts tool input connections.

Figure 12 shows the connections for Agent 1.

Step 8: Add Agent node 2

- In the left panel, click on Agents.
- Under Agents, drag and drop the Agent node into the canvas

Input the following:

- Model Provider: SambaNova

Model Name: Meta-Llama-3.3-70B-Instruct

- Sambanova API Key: *Insert Personal API Key*
- Agent Instructions: "*You are a research agent, expert in web research, you must provide insights given the user question {input} Think step by step Do not call a tool if the input depends on another tool output that you do not have yet. Do not try to answer until you get all the tools output, if you do not have an answer yet, you can continue calling tools until you do. Your answer should be in the same language as the initial query. Ensure the tool calls are proper parsable json"*

Step 8.1: Connecting Chat Input & Tavily Search Toolset to Agent 2

Locate the Chat Input node in your flow.

- Drag a connection from the output circle of Chat Input to the "input" port on the left side of Agent 2.
- This allows Agent 2 to process the same user prompt passed to Agent 1, supporting multi-agent collaboration.
- Find the Tavily AI Search Toolset node in your flow.
- Drag a connection from its output to the "tools" input port on Agent 2 (usually on the left side).
- This equips Agent 2 with real-time AI web search capabilities powered by Tavily.

Figure 13 illustrates the connections for Agent 2.

Step 9: Add Combine Text Node 1

- In the left panel, click on Processing.
- Under Processing, drag and drop the Combine Text node into the canvas.
  - This node will merge the responses from both agents into one unified output.

Step 9.1: Connecting Agent 1 & Agent 2 to Combine Text 1
Locate Agent 1 in your flow.

- Drag a connection from the Response port of Agent 1 to the First Text input on the Combine Text node.
- This sends Agent 1's output to the first input field.

Locate Agent 2 in your flow.

- Drag a connection from the Response port of Agent 2 to the Second Text input on the Combine Text node.
- This sends Agent 2's output to the second input field.
- In the Delimiter field of the Combine Text node, type: \n\n

The Delimiter field defines how the two agent responses are separated; using \n\n inserts a clear line break for improved readability in the final output.

Figure 14 shows the connections for Combine Text 1.

Step 10: Add Combine Text Node 2

- Repeating step 9: In the left panel, click on Processing.
- Under Processing, drag and drop the Combine Text node into the canvas.
- This node will merge the responses from both agents into one unified output.

Step 10.1: Connecting Chat Input & First Combine Text to Second Combine Text Node

- The second Combine Text node allows you to pair the user's original input with the agents' findings, creating a rich prompt for summarization or final response generation.
- Locate the Chat Input node in your flow.
    - Drag a connection from the output of Chat Input to the First Text input on the new Combine Text node.
- This passes the original user query as the first part of the message.
- Locate the first Combine Text node (created in Step 9).
- Drag a connection from its Combined Text output to the Second Text input on the new Combine Text node.
    - This sends the merged agent responses as the second part of the message.
- In the Delimiter field of this second Combine Text node, enter the following: *\n\nContext: \n*
    - This Delimiter separates the user's question from the agent responses and labels the second section clearly as contextual information.

Screenshot 15 displays the connections for Combine Text 2.

Step 11: Add SambaNova Node

- In the left panel, click on Agents.
- Under Agents, drag and drop the SambaNova node into the canvas.

Step 11.1: Configure the SambaNova Node

- Connect "Combine Text 2" port to SambaNova's Input port
- In the SambaNova node settings:

Model Name: Meta-Llama-3.3-70B-Instruct

- Sambanova API Key: *Insert Personal API Key*
- System Message: "*Copy and paste the following prompt: "You are a helpful assistant who writes financial reports based on user query. You will get a financial and research summary. Please write a financial report or answer using the provided context and input text only.*

  {input}

  Your answer should be in the same language as the initial query."

Step 11.2: Connect Combine Text 2 to SambaNova

- Locate Combine Text Node 2 (created in Step 10).
- Drag a connection from its Combined Text output to the Input port of the SambaNova node.
- This ensures the assistant receives both the original user query and the contextual information derived from both agents.

Figure 16 shows the SambaNova Node.

Step 12: Add Chat Output Node

- In the left panel, click on Output.
- Drag and drop the Chat Output node into the canvas.

Step 12.1: Connect SambaNova to Chat Output

- Drag a connection from the Message output port of the SambaNova node to the Chat Output node.
- This allows the final, context-aware response generated by SambaNova to be shown directly to the user in the chat interface.

Figure 17 shows the Chat Output, and Figure 18 shows the Full Build.

5. TEST YOUR CHATBOT USING THE PLAYGROUND

Steps:

- Once your entire flow is built and all components are connected, it's time to test your chatbot.
- Open the Playground. In the top right sidebar, click on Playground.
- This will open a chat-style interface where you can interact with your flow in real time.
- Type a financial question in the chat input box and press Enter.
- The message will travel through the entire Langflow pipeline:
    - Agents will collect and analyze data using Yahoo Finance and Tavily Search.
    - Combine Text nodes will format the responses and context.
    - SambaNova will generate the final response.
    - Chat Output will display the assistant's full reply.
- Sample Questions to Try - Here are some realistic prompts you can use to evaluate your chatbot:
    - *"What's the latest news about Tesla's stock performance?"*
    - *"Is Apple a good investment this quarter?"*
    - *"Summarize the market trends for Nvidia and AMD over the last week."*
    - *"Give me a financial report on Microsoft based on current news and stock data."*
    - *"What are the biggest risks for investing in Google right now?"*

Figure 19 shows the Playground questions.

6. CONCLUSION

This tutorial guides you through the step-by-step creation of a multi-agent financial analysis chatbot using Langflow on Astra. By integrating real-time data sources like Yahoo Finance and Tavily AI Search, coordinating agent collaboration, and delivering outputs through SambaNova's LLM, you've built a scalable and intelligent system capable of generating live financial insights into natural language reports. The finished chatbot not only demonstrates key principles of Retrieval-Augmented Generation (RAG) and agent-based design but also shows how low-code tools like Langflow can simplify developing domain-specific AI solutions. While this build focuses on financial analysis, the architecture can be adapted to various other fields. As a teaching case, this tutorial helps students bridge conceptual understanding with practical implementation. It encourages hands-on learning while prompting critical evaluation of AI toolchains and real-world deployment considerations. Future adaptations may incorporate classroom assessment results and comparative platform analysis.

7. PROJECT

This assignment will guide you through the process of developing a multi-agent AI chatbot using Langflow on Astra, applying the principles and techniques demonstrated in the "Developing a Financial Analysis Chatbot Using Langflow on Astra" tutorial. While the tutorial focuses on financial analysis, you will adapt the architecture to create a chatbot for a domain of your choice.

Submission Requirements:

- Langflow Flow Export: Export your completed Langflow project as a .json file.
- Project Report (2-3 pages): A report detailing your chatbot's design, implementation, and a reflection on the development process.
- Demo Video (3-5 minutes): A screen recording demonstrating your chatbot's functionality.

Instructions:

Part 1: Chatbot Development (Hands-on Implementation)

Choose Your Domain: Select a domain for your AI chatbot. Examples could include:

- Academic research assistant (e.g., summarizing research papers from specific journals)

- Product recommendation engine (e.g., suggesting books, movies, or tech gadgets)
- Health information bot (e.g., providing general information on symptoms, treatments, or healthy living based on reliable sources)
- Local events planner (e.g., finding events based on user preferences and location)
- Real estate assistant (e.g., providing property information and market trends)

Access Langflow on Astra:

- Navigate to Langflow on Astra: https://astra.datastax.com/langflow/d53da360-33e2-4eed-a97c-0b9bae5b0134.
- Log in to your DataStax Astra account (or register if you don't have one).
- Create a "New Flow" and select "Blank Flow" to build your chatbot from scratch.

Integrate Core Components:

- Add a Chat Input Node to receive user queries.
- Add a Chat Output Node to display the final response.
- Integrate a SambaNova Node (or another suitable LLM node if you prefer and have API access) for natural language generation. Configure it with your API key and a system message relevant to your chosen domain.

Incorporate Tools and Agents (Customize for your Domain):

- Using a general web search API (like Tavily AI Search if applicable to your domain).
- Integrating a specialized API (e.g., a book API, movie database API, medical information API, event API). If a direct Langflow component isn't available, you might need to use a generic "URL Tool" or "Python Function" node to interact with your chosen API.
- Consider the URL Tool for loading and retrieving data from specified URLs, with options for plain text, raw HTML, or KSON output.
- Agent 1 (e.g., Data Gatherer/Initial Researcher): Focus on retrieving raw information or performing initial searches using your chosen data retrieval tools.
- Agent 2 (e.g., Synthesizer/Analyzer): Focus on interpreting the data gathered by Agent 1 and formulating insights or structured responses relevant to your domain.
    - Research/Data Retrieval Tool(s): Replace or adapt the Yahoo Finance and Tavily AI Search tools with external data sources relevant to your chosen domain. This might involve:
    - Agents: Design at least two distinct agents, similar to the "Researcher" and "Financial Analyst" agents in the tutorial. Each agent should have a specific role and be configured with appropriate instructions (Agent Instructions) and relevant tools:
    - Optional Tools: Consider if a Calculator or other tools (e.g., a Wikipedia tool if relevant) would enhance your chatbot's capabilities.

Orchestrate Data Flow with Combine Text Nodes:

- Use Combine Text Nodes to merge outputs from your agents and format the input for your final LLM, ensuring a coherent and context-rich prompt.
- Experiment with delimiters to enhance readability and structure.

Connect All Components: Visually connect all your nodes in Langflow, ensuring the data flows logically from user input to tool usage, agent processing, and final output. Refer to the tutorial's screenshots (e.g., Screenshot 16: Full Build) for guidance on connection patterns.

Test Your Chatbot:
Open the Langflow Playground.
Test your chatbot thoroughly with various queries relevant to your chosen domain. Try to ask questions that require your agents to use their integrated tools and process information.
Refine agent instructions, tool configurations, and prompt engineering strategies based on your testing to improve performance and accuracy.

Part 2: Project Report

Write a 2-3 page report that includes the following sections:

Introduction:

- Briefly describe the purpose of your chatbot and the domain you chose.
- Explain why you selected this particular domain and its potential benefits.

Chatbot Architecture and Components:

- Provide a high-level overview of your chatbot's architecture (you can draw a simple diagram if helpful, but it's not strictly required).
- Detail each key component you used (e.g., Chat Input, Chat Output, LLM, specific tools, agents).

- For each tool and agent, explain its specific role and how it contributes to the chatbot's overall functionality.
- Describe any modifications or custom integrations you made compared to the financial analysis tutorial.

Implementation Details:
- Discuss the specific APIs or data sources you integrated and why they were chosen.
- Explain your prompt engineering strategies for directing your LLM-based agents. Include examples of your Agent Instructions and the System Message for your final LLM.
- Describe how you used Combine Text nodes to manage and structure information flow.

Testing and Results:
- Present 3-5 sample questions you used to test your chatbot.
- For each sample question, describe the expected output and the actual output from your chatbot.
- Discuss any challenges you encountered during testing and how you addressed them.
- Highlight the strengths and limitations of your chatbot.

Reflection and Future Work:
- What did you learn from building this chatbot?
- How could your chatbot be improved or extended in the future? (e.g., adding more tools, improving agent logic, integrating memory, enhancing user interface).
- How broadly applicable do you think this architecture and design are to other domains?

Part 3: Demo Video

Create a 3-5 minute screen recording demonstrating your chatbot's functionality.

- Clearly show the Langflow interface and your connected nodes.
- Demonstrate your chatbot responding to at least three different queries, showcasing its ability to use its tools and agents effectively.
- Narrate your demonstration, explaining what you are doing and what the chatbot is performing at each step.

Grading Rubric:

- Chatbot Functionality: (40%) – Chatbot operates as intended, effectively uses integrated tools and agents, and provides relevant responses for the chosen domain.
- Architectural Understanding: (20%) – Demonstrated understanding of Langflow components, agent orchestration, and data flow.
- Report Quality: (20%) – Comprehensive, well-written, and insightful report covering all required sections.
- Video Demonstration: (10%) – Clear, concise, and effective demonstration of the chatbot.
- Creativity/Domain Choice: (10%) – Originality and thoughtful selection of a non-financial domain.

8. REFERENCES

Aksar, B., Rizk, Y., & Chakraborti, T. (2023). TESS: A Multi-intent Parser for Conversational Multi-Agent Systems with Decentralized Natural Language Understanding Models. arXiv.Org, abs/2312.11828. https://doi.org/10.48550/arxiv.2312.11828

Aveiro, D., Freitas, V., Ferreira Da Cunha, E., Quintal, F., & Almeida, Y. (2023). Traditional vs. low-code development: comparing needed effort and system complexity in the NexusBRaNT experiment. https://doi.org/10.1109/cbi58679.2023.10187470

Calvaresi, D., Eggenschwiler, S., Mualla, Y., Schumacher, M., & Calbimonte, J.-P. (2023). Exploring agent-based chatbots: a systematic literature review. Journal of Ambient Intelligence and Humanized Computing, 14(8), 11207–11226. https://doi.org/10.1007/s12652-023-04626-5

Trigo, A., Varajão, J., & Almeida, M. (2022). Low-Code Versus Code-Based Software Development: Which Wins the Productivity Game? 24, 61–68. https://doi.org/10.1109/MITP.2022.3189880

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. The Journal of Machine Learning Research, 3, 993–1022.

ISSN: 2473-4901
v11 n6461

APPENDIX A
Guided Step-by-Step Figures

Due to the file size limit, not all Figures in the Appendix are displayed here. You can download the full Appendix at https://tinyurl.com/lowcodechatbots.

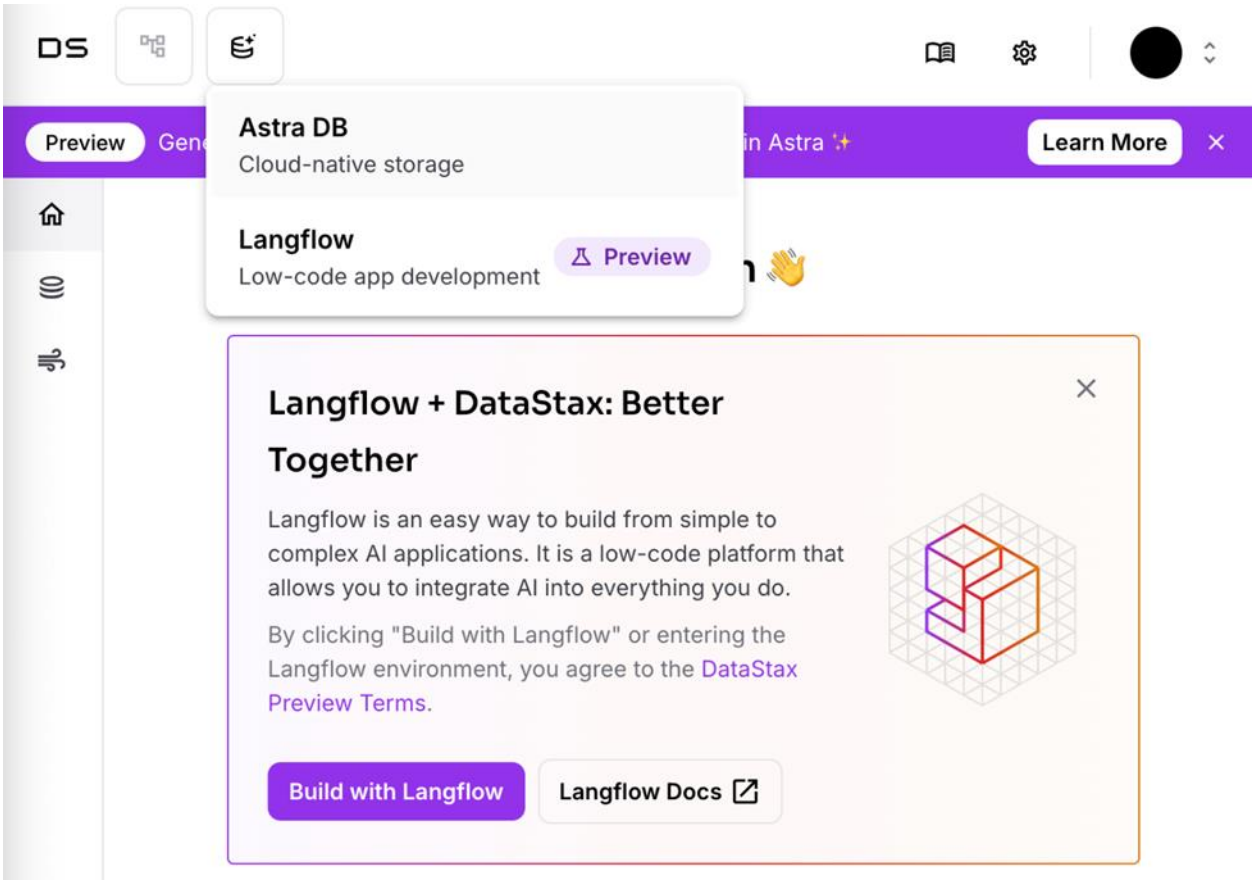

Figure 1. Langflow with Astra DB

ISSN: 2473-4901
v11 n6461

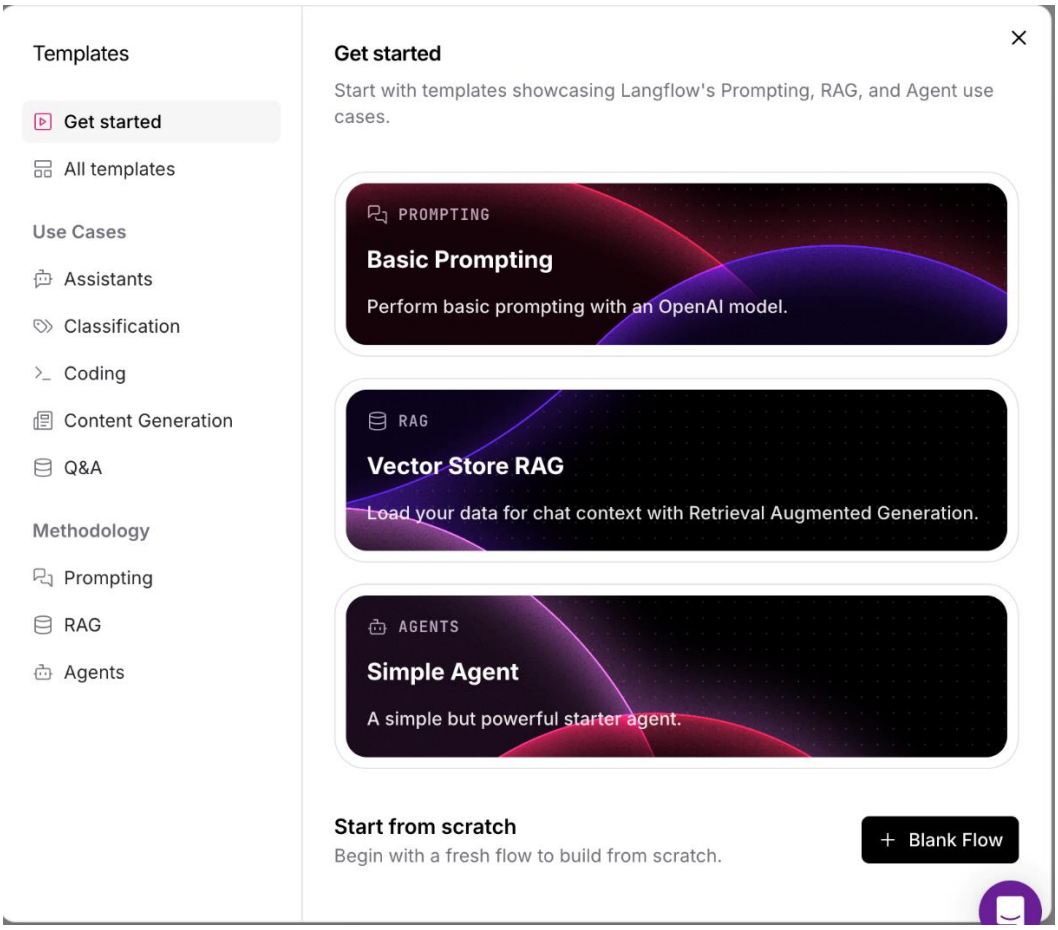

Figure 2. Langflow's template selection screen showing "Basic Prompting," "Vector Store RAG," and "Simple Agent" templates, with the sidebar displaying categorized options under "Use Cases" and "Methodology."

ISSN: 2473-4901
v11 n6461

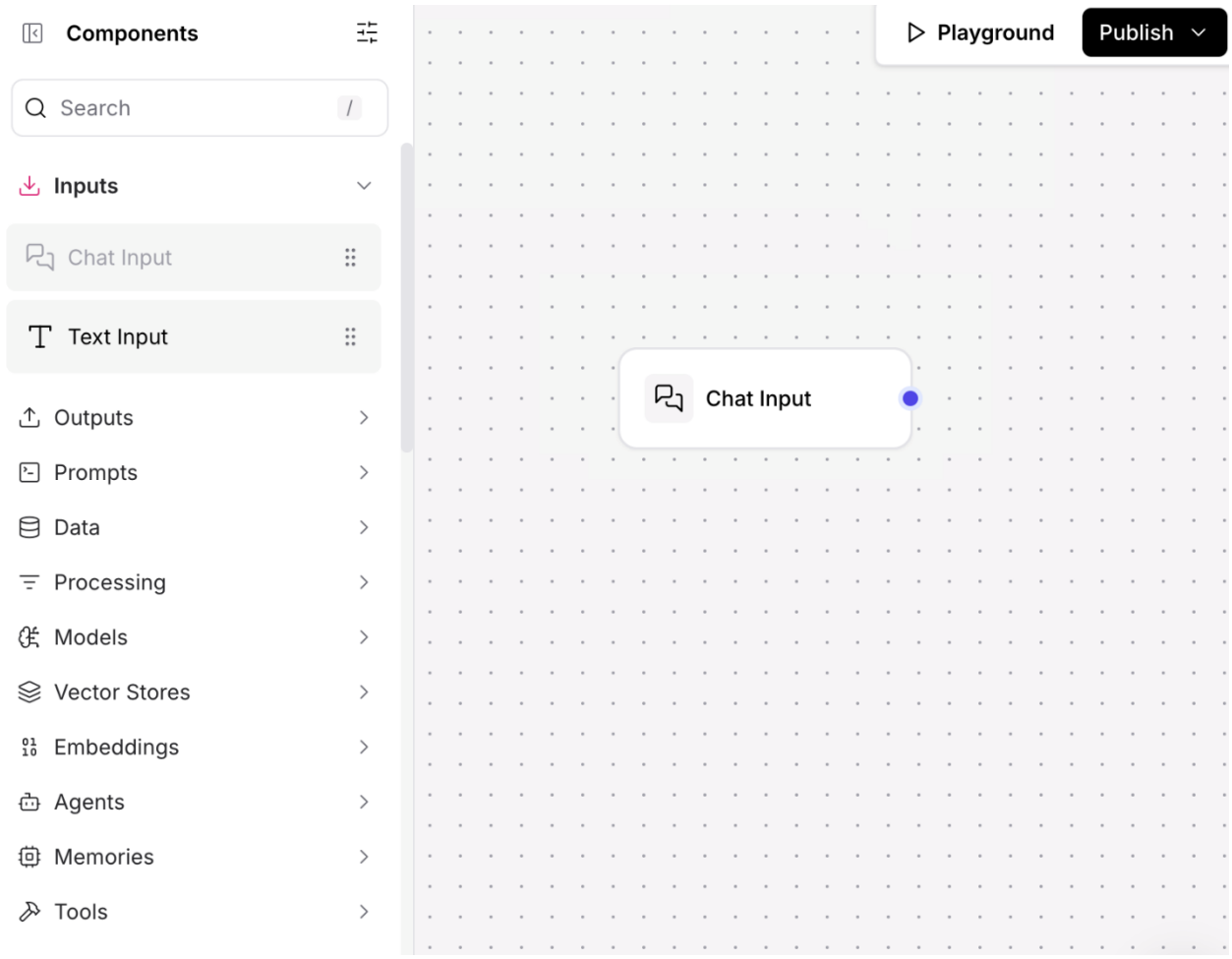

Figure 3. Chat Input in the Canvas

ISSN: 2473-4901
v11 n6461

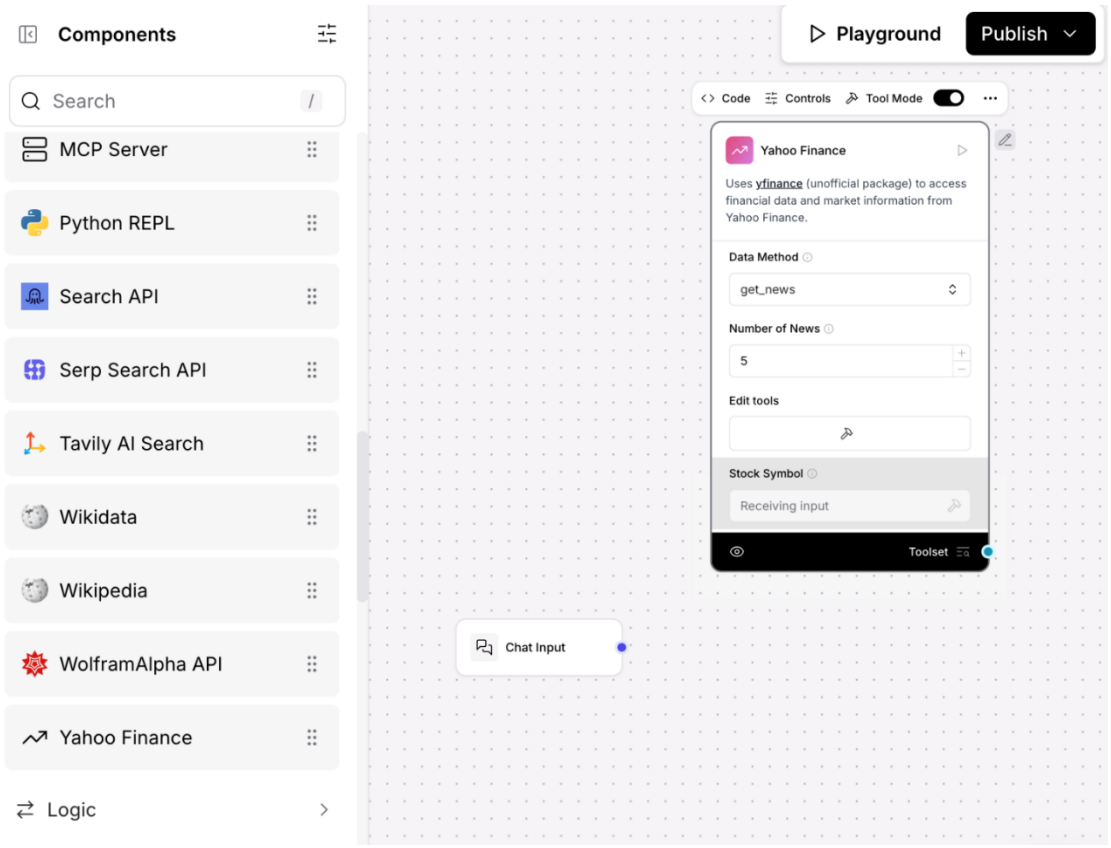

Figure 4. Yahoo Finance

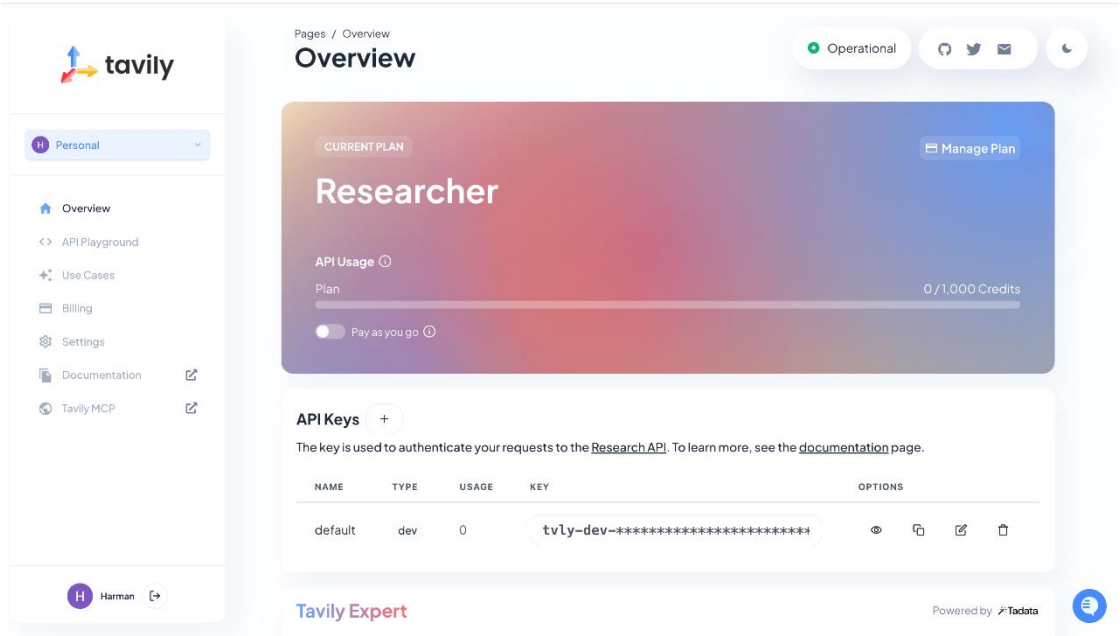

Figure 5. Tavily AI Search Website

ISSN: 2473-4901
v11 n6461

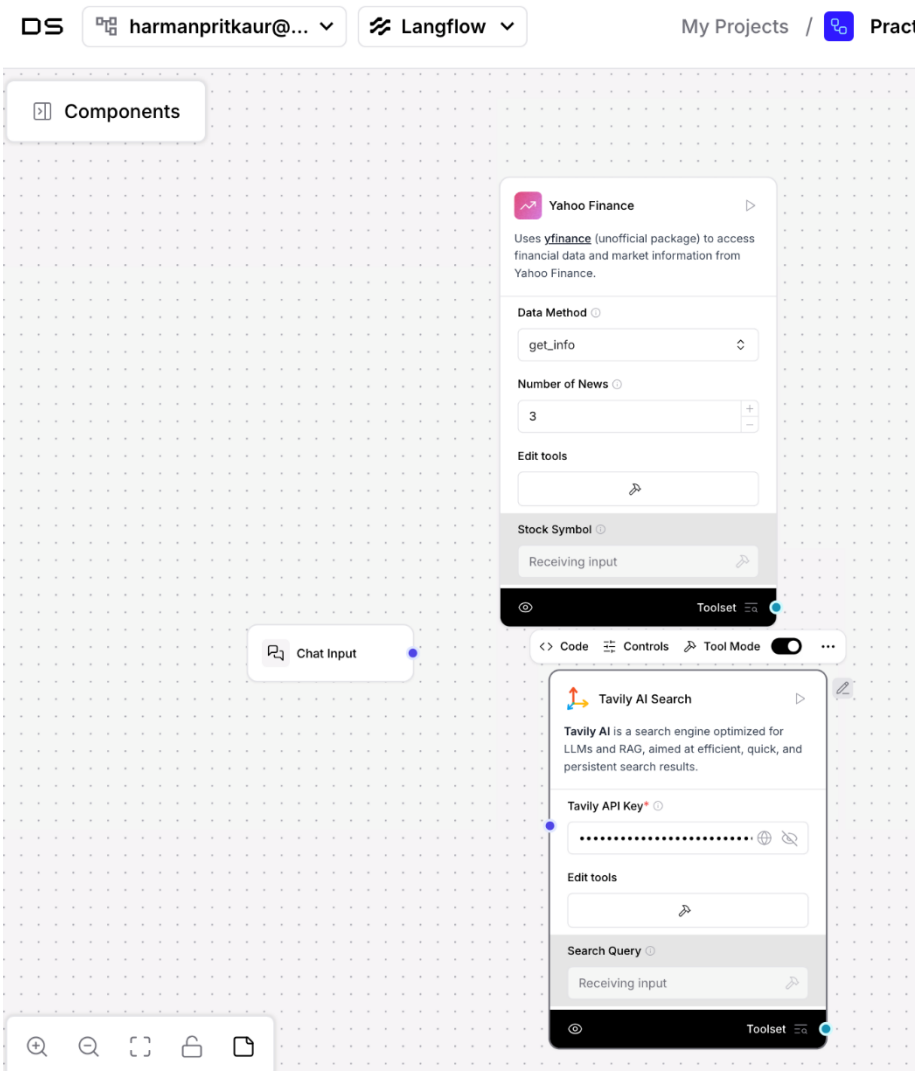

Figure 6. Tavily AI Langflow with Tool Mode "toggled on"

ISSN: 2473-4901
v11 n6461

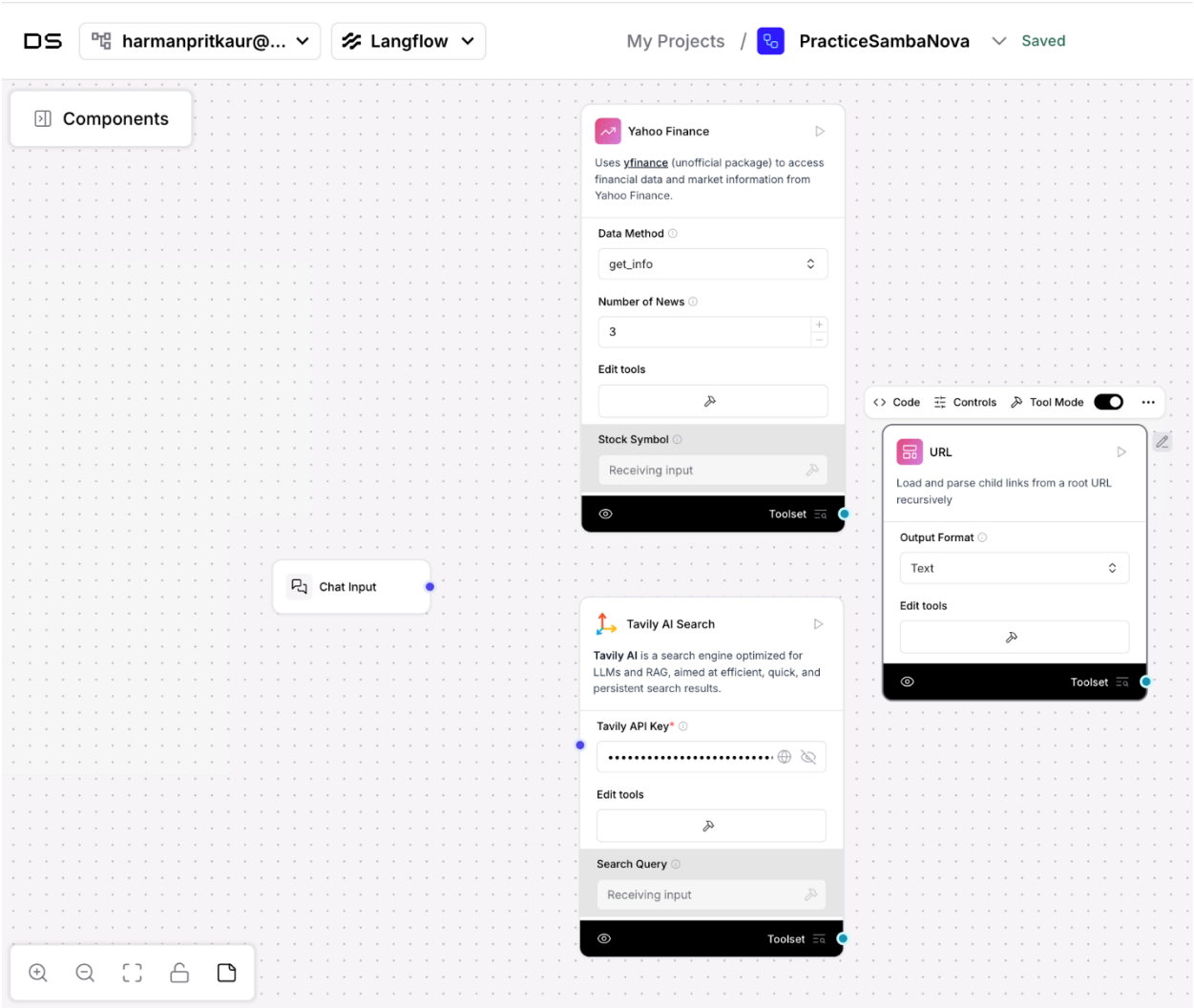

Figure 7. URL component with Tool Mode toggled on

ISSN: 2473-4901
v11 n6461

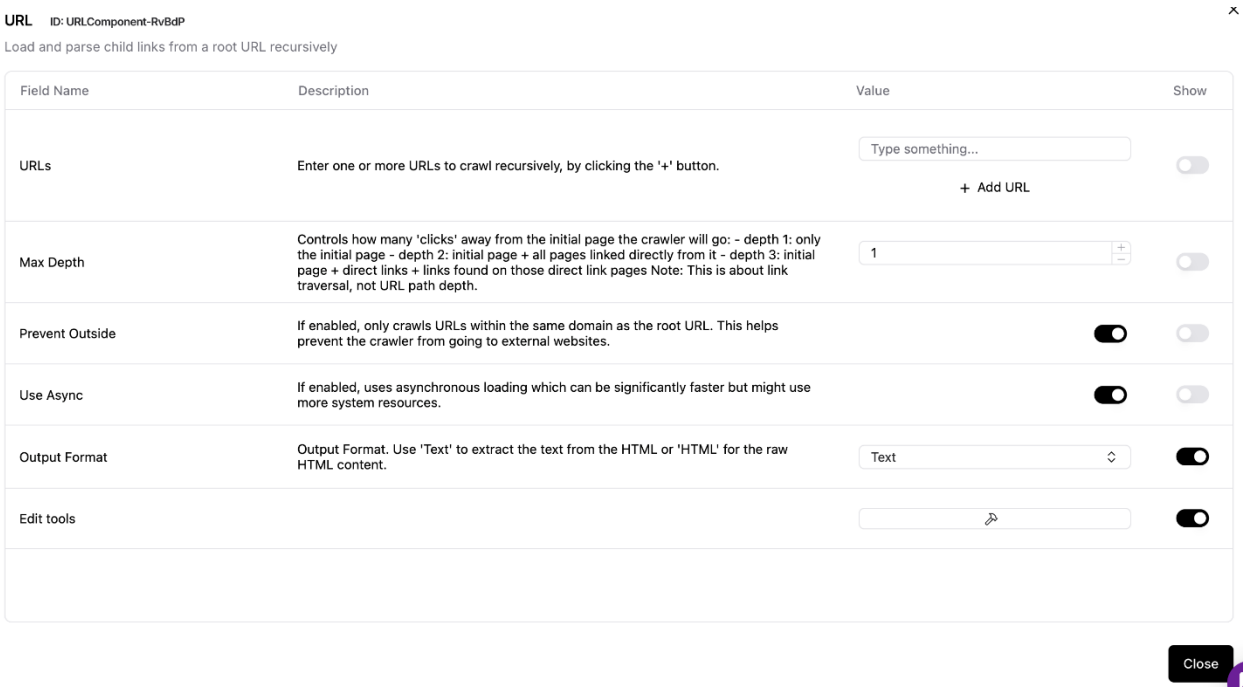

Figure 8. Calculator Node Controls

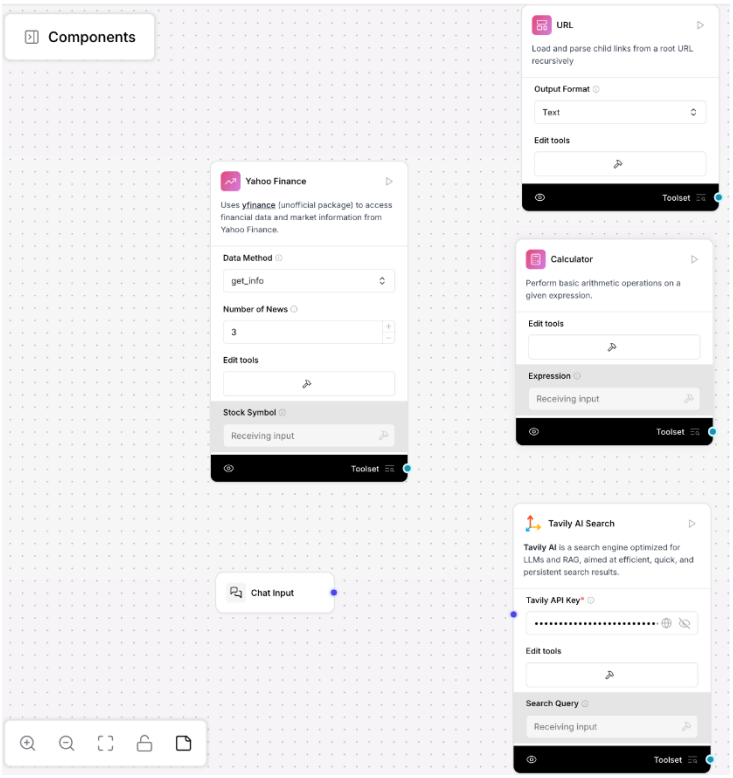

Figure 9. Calculator Node in the Canvas

ISSN: 2473-4901
v11 n6461

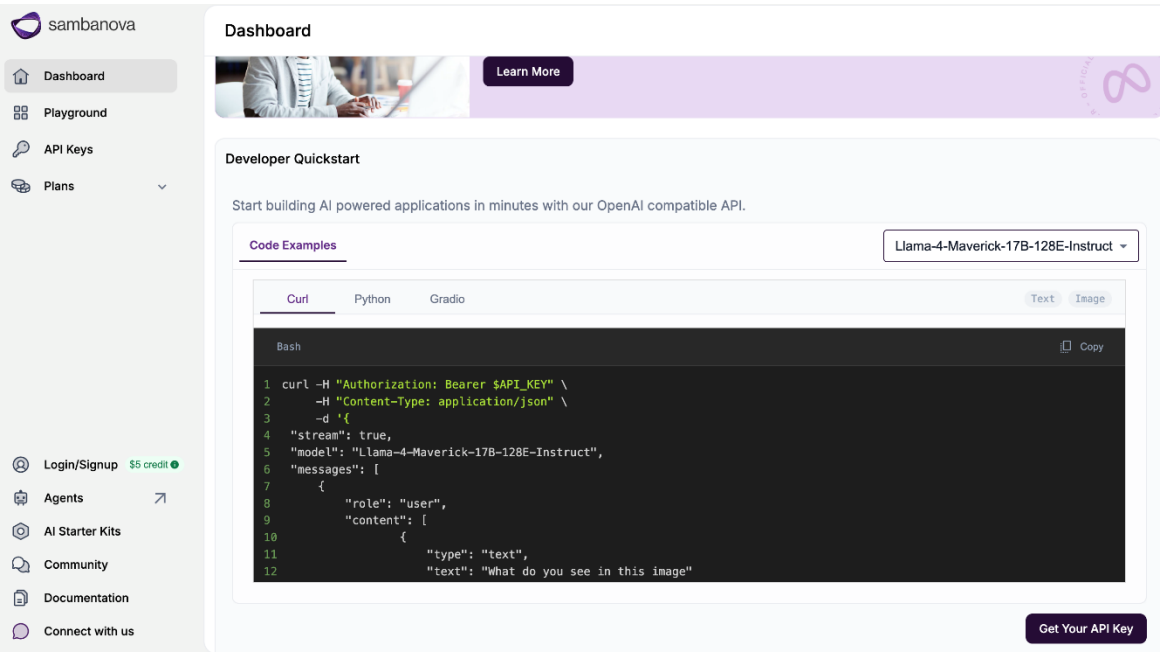

Figure 10. SambaNova Cloud Website

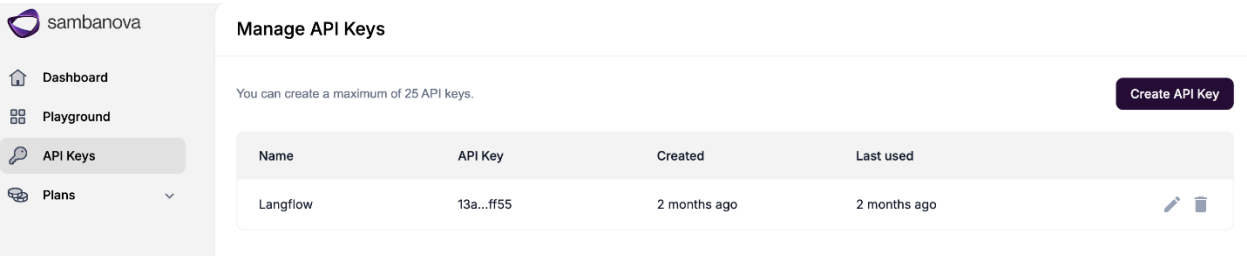

Figure 11. SambaNova "Create API Keys”

ISSN: 2473-4901
v11 n6461

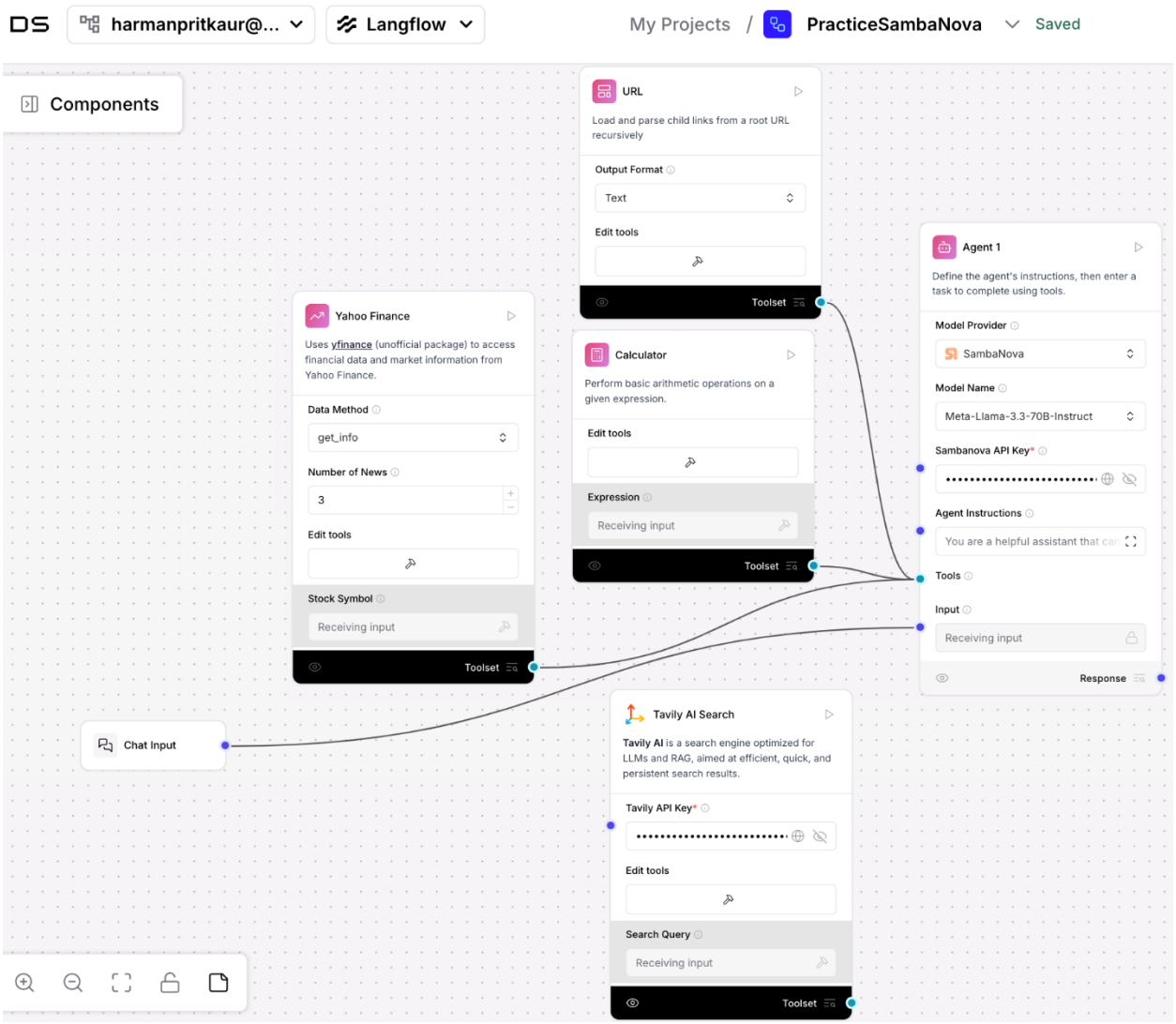

Figure 12. Agent 1 connections

ISSN: 2473-4901
v11 n6461

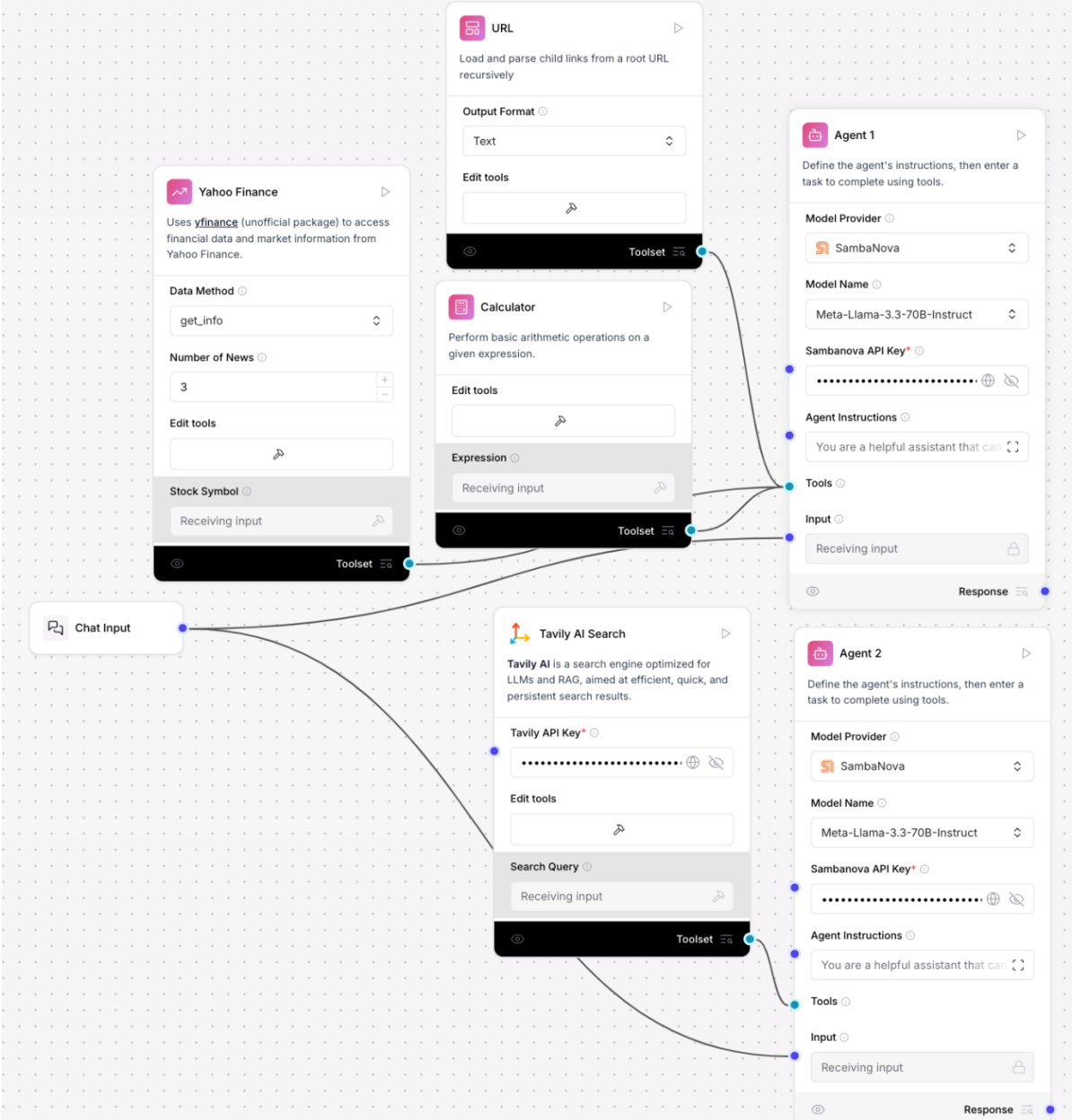

Figure 13. Agent 2 Connections

ISSN: 2473-4901
v11 n6461

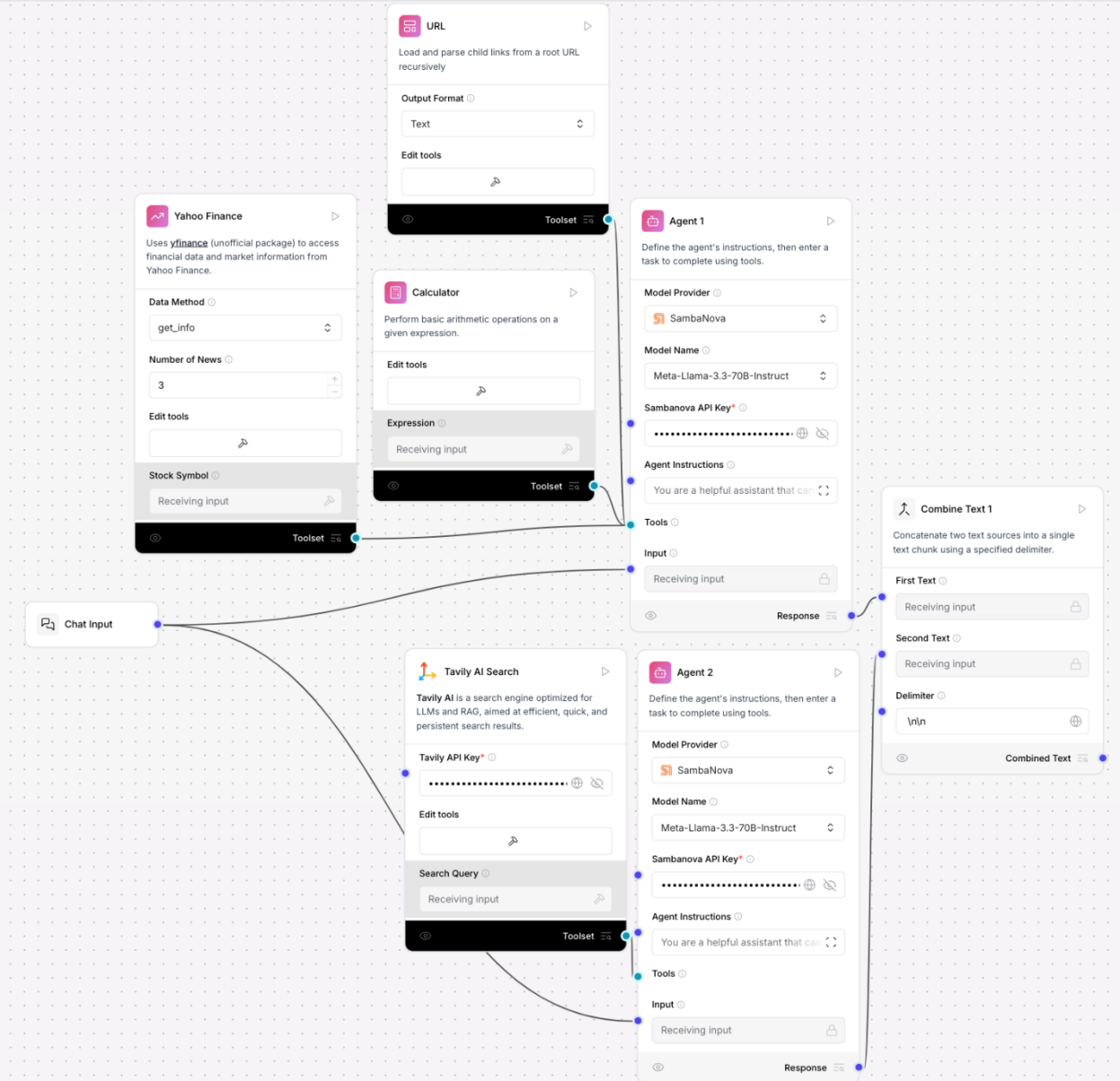

Figure 14. "Combine Text 1" Connections

ISSN: 2473-4901
v11 n6461

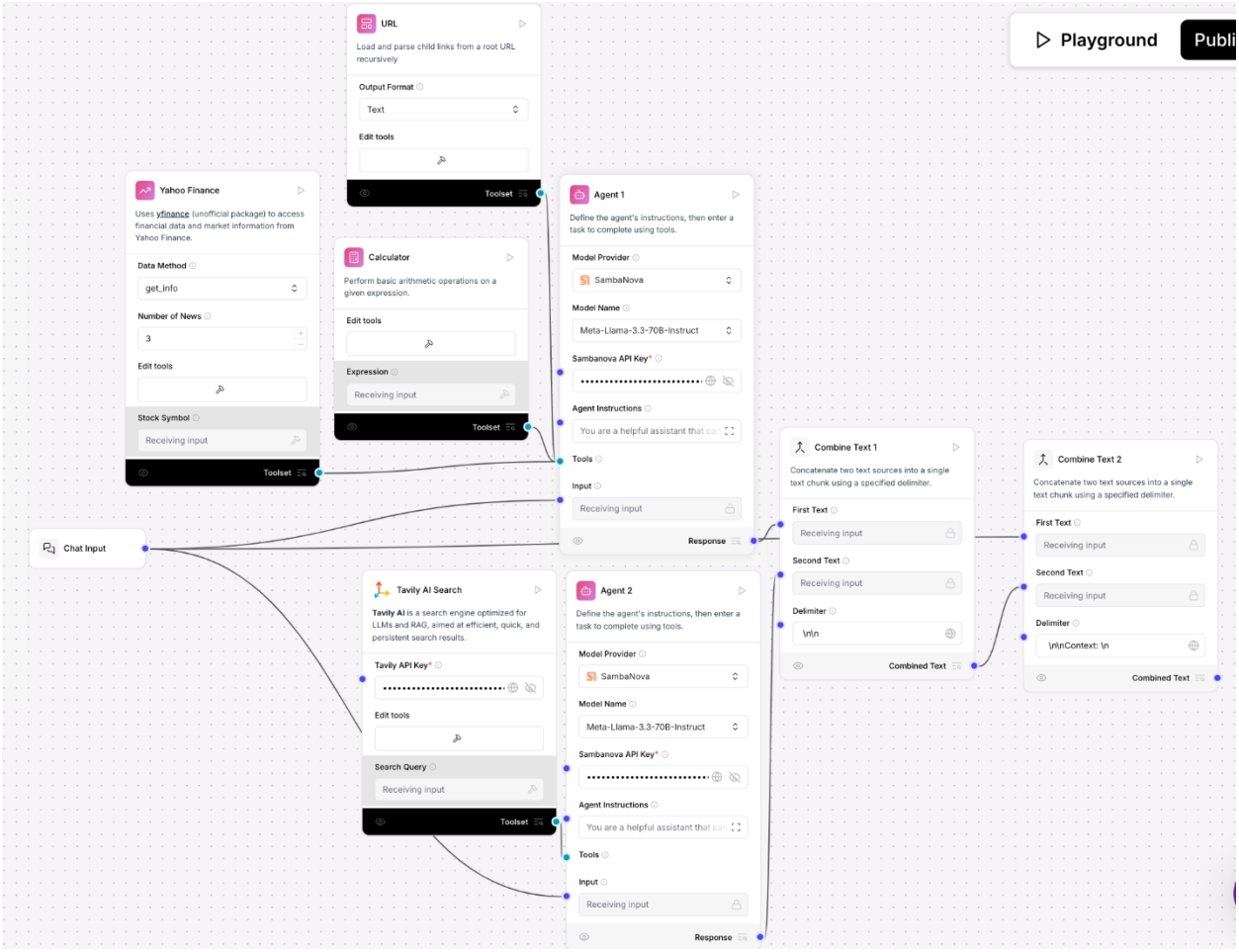

Figure 15. "Combine Text 2" Connections

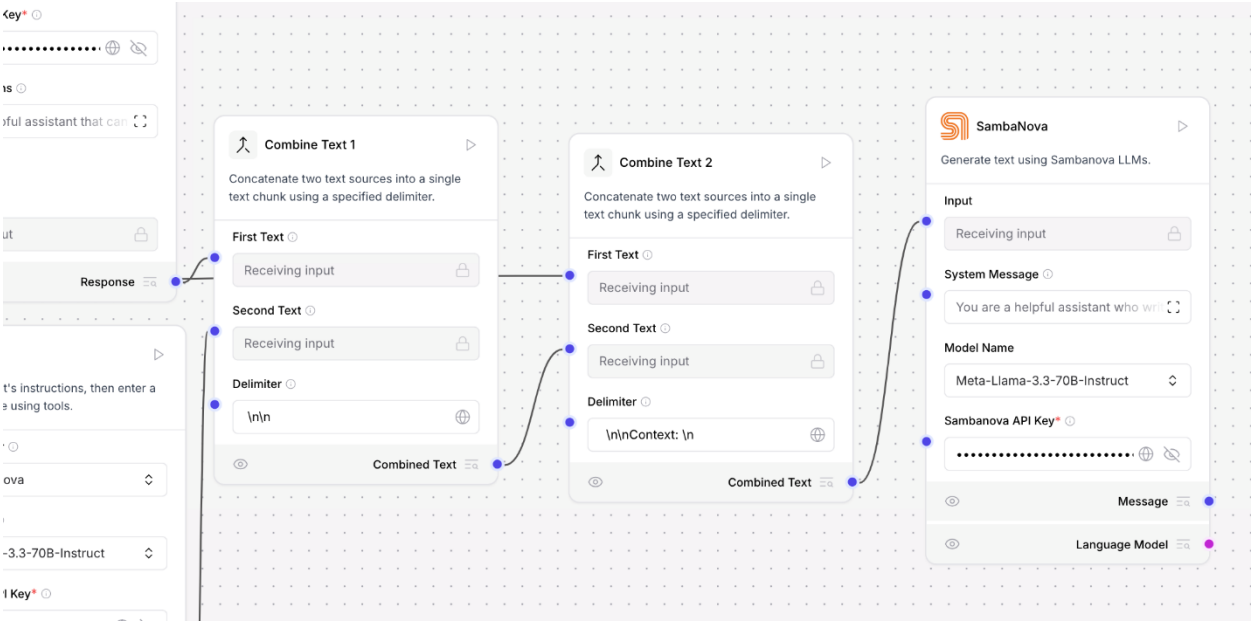

Figure 16: SambaNova Node with Connection

ISSN: 2473-4901
v11 n6461

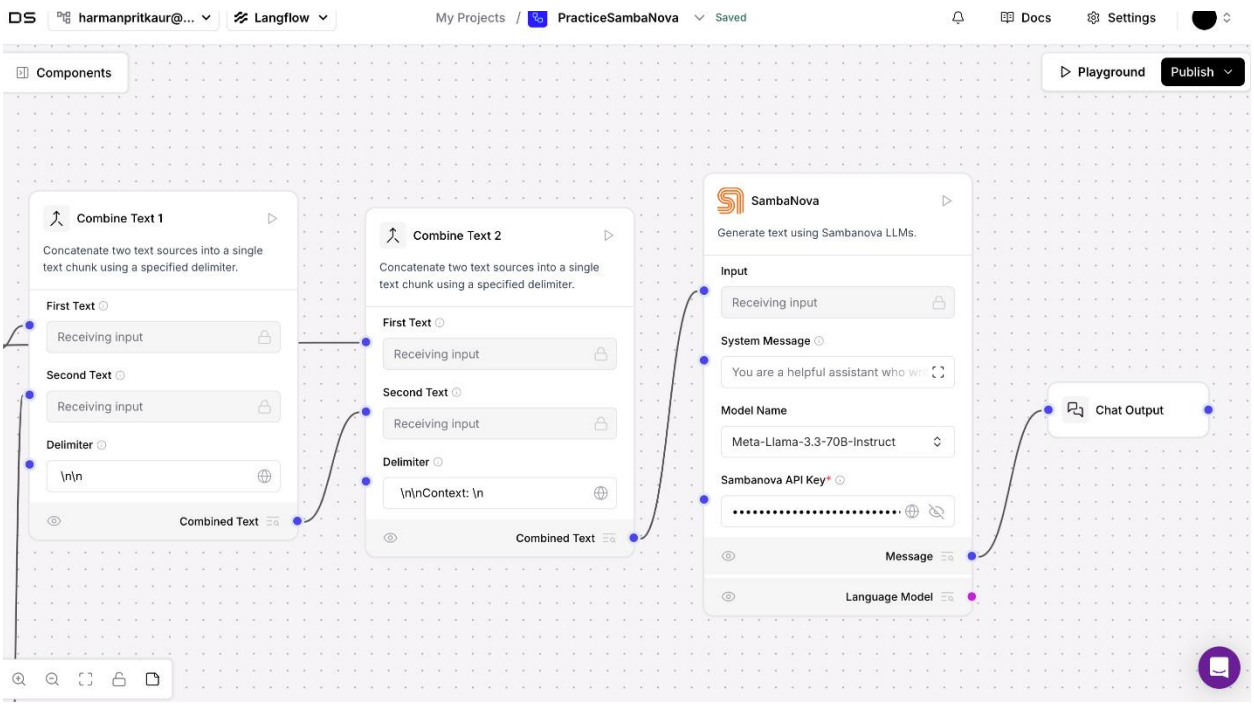

Figure 17: Chat Output Connections

ISSN: 2473-4901
v11 n6461

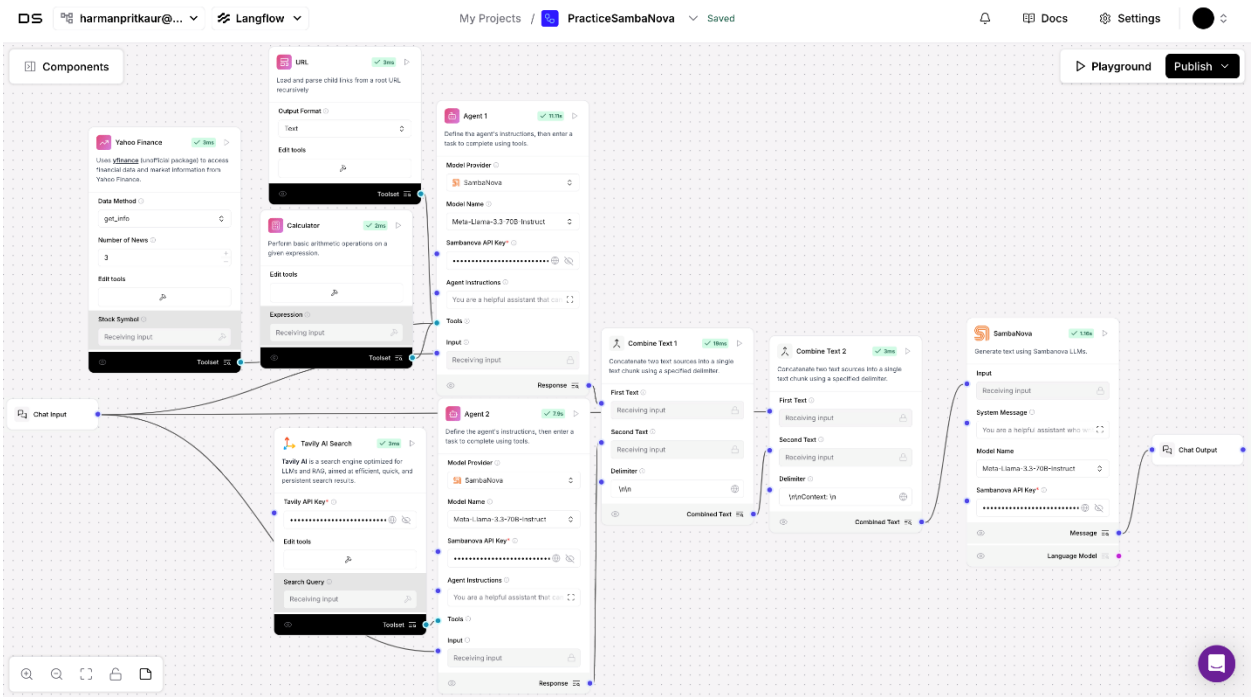

Figure 18: Full Build

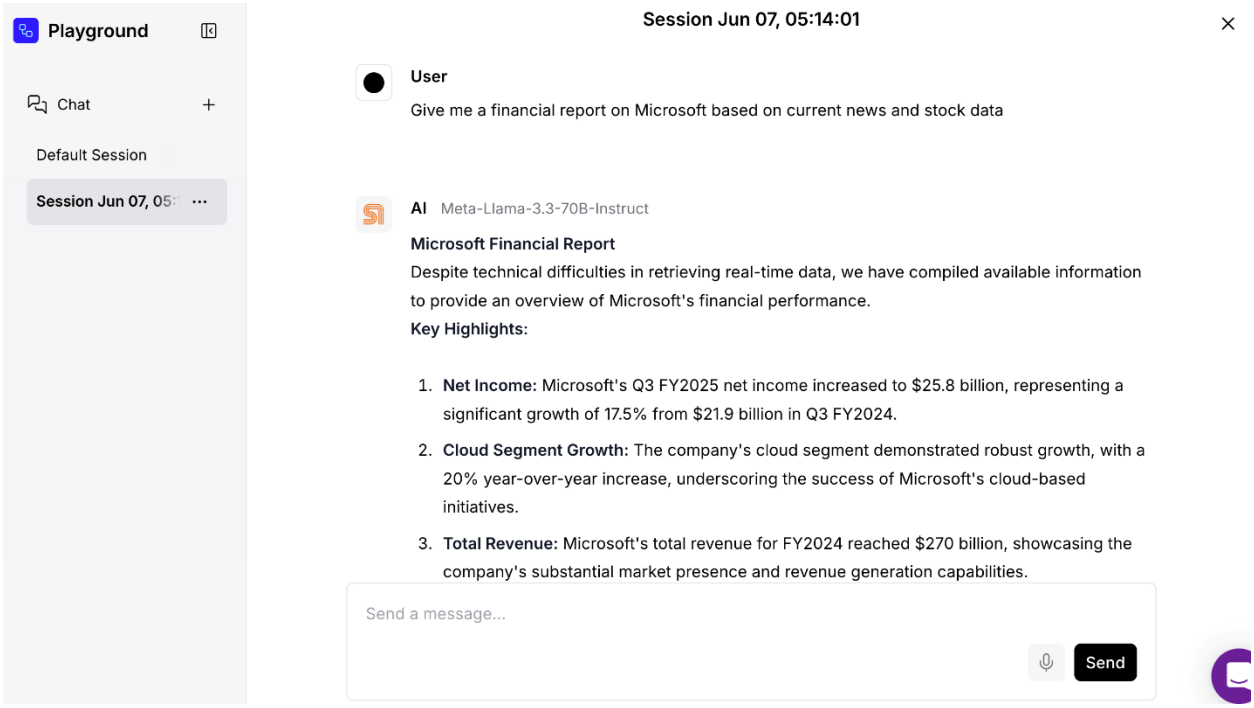

Figure 19: Playground Question w/ real-time data