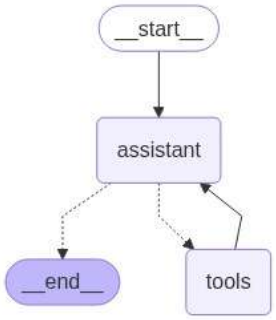
The Rise of the WeatherMind

Although the Python Notebook has some documentation, here is an overview of my work.

Level 1

Started with loading Gemini API and setting up a normal chatbot agent that can reply to messages Then integrated a ToolNode that contained the function to return evaluated expressions.



The assistant will take the prompt and pass necessary information to the function and then give to the user, the evaluated output

Python's 'eval' function was used to evaluate that takes care of the BODMAS rule.

Level 2

Added Weather tool to the toolnode that fetches weather from a particular city and returns it to the assistant (with the help of OpenWeather API).
Added Fashion Recommendation Tool that searches the web for the ongoing real-time fashion trends at a place. This is done with the help of Tavily Search Engine.

Level 3

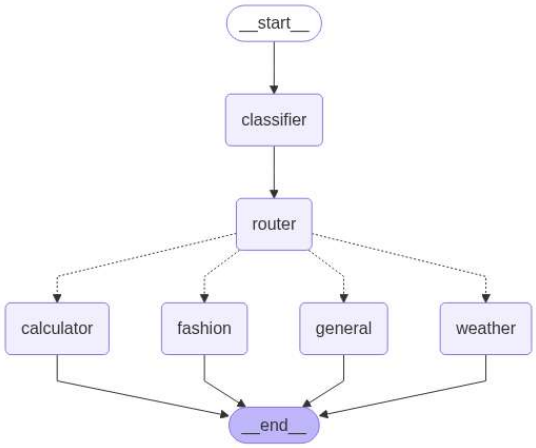
Appropriate and enough prompts were given for each of Calculator Tool, Weather Tool and Fashion Recommendation Tool so that the assistant can judge which tool to choose for a particular prompt by the user.

For adding memory to our chatbot, I passed the chat history along with the messages every time we invoke the graph (or enter a new input). The assistant will go through all the messages previously sent by user, and responded by AI and then work on the final output of the code.

Level 4

Here, I defined different agents for each of calculator, weather and fashion tool. Each agent has its own ToolNode that contains a function. I also added another agent 'general' that takes care of the general queries that are not a part of any of these. Sufficiently understandable prompts are provided for each of the agent. The routing logic goes as:

start -> classifier -> router -> calculator | weather | fashion | general -> end



Here when the chatbot first runs, it enters the classifier and based on the prompt that user entered, it classifies the message as calculator, fashion, general or weather. Then this information is passed on to the router and based on the classification, the router redirects it to the particular agent. Each agent has its own ToolNode from where it fetches the output and then it ends.

A demo chat session has been shown below:

```
You:  Hi, I am Aditya Garg
Assistant: Hi Aditya Garg! It's nice to meet you. How can I help you today?
You:  Can you evaluate 2*5+10/2-2^4/2
Assistant: 7

You:  Great! What is the weather in Guwahati right now?
Assistant: The weather in Guwahati is currently misty with a temperature of 26.95 degrees Celsius, atmospheric pressure of 10
05 hPa, and humidity of 89%.

You:  Can you tell me about the current fashion trends in New York?
Assistant: Based on my search, here are some current fashion trends in New York:

*   **Elevated basics and tailored outerwear:** These were prominent during New York Fashion Week.
*   **Monochromatic Separates:** These were also a key part of the collections shown during Fashion Week.
*   **Y2K-inspired accessories:** Sunglasses and handbags are showing futuristic silhouettes and metallic materials.
*   **Loafers, peplum tops, and chunky sneakers:** These are being spotted on the streets of New York City.
*   **Flats and kitten heels:** Seen during New York Fashion Week.
You:  Oh Nice...do you remember my name?
Assistant: Yes, Aditya Garg. I remember your name.
You:  What expression did I give you to evaluate?
Assistant: You asked me to evaluate the expression `2*5+10/2-2^4/2`.
You:  Which city's weather did you fetch me?
Assistant: The weather in Guwahati: Temperature is 26.95 degrees Celsius, atmospheric pressure is 1005 hPa, humidity is 89%,
and the description is mist.

You:  exit
Goodbye!
```

Here, an example of arithmetic calculation using BODMAS rule is done. Weather conditions in Guwahati have been fetched with great details. Fashion trends in New York are listed.

Then the model is tested if it still remembers the older messages. It still remembers the name, the evaluating expression, or any other conversation that was done in the chat session.