Task 2: The Rise of the WeatherMind

Introduction

As part of the WeatherMind challenge, my goal was to build an intelligent agent system that could understand user queries and respond with appropriate reasoning and data-backed results. The key concept behind this task was **agentic AI** — systems that don't just respond, but decide, use tools, and adapt.

I approached this challenge by progressing through the first two levels, focusing on establishing a strong base: the core LLM-powered agent with tool integration and real-world sensing abilities.

Level 1: Awakening the WeatherMind

The first step in my journey was to give life to the AI — or as the challenge put it, “wake the core.” This meant creating a **LangGraph-powered chatbot** that could interact with a user and perform basic arithmetic using a calculator tool.

To begin with, I set up my environment by installing the required libraries such as `langgraph`, `langsmith`, and `langchain_groq`. Then, I built a simple **stateful graph** using LangGraph, where the flow of conversation is handled using defined nodes.

For the LLM, I used **Groq** to access a large language model. Then, I integrated a **custom calculator tool** that could evaluate arithmetic expressions using BODMAS logic. I made sure the calculator was secure and clean by writing a `safe_eval()` function that carefully parses user inputs.

Finally, I used LangGraph's visualization tools to render the graph — showcasing the skeleton of the AI and how it processes user input through a reasoning flow. This level gave me a great foundation in understanding how LangGraph structures agentic workflows.

Level 2: Senses of the World

After activating the core AI, I moved to the second level: giving the agent **senses of the external world**.

In this phase, I implemented two external tools:

1. **Weather Extractor Tool:**
   I created a tool that could accept a query like “What's the weather in Delhi?” and fetch

live weather data (temperature, condition, etc.) using an open weather API. This involved using Python's `requests` module and formatting the output to make it clean and conversational.

2. **Fashion Recommender Tool:**
   For this tool, I focused on building a prototype that could return trending fashion recommendations based on a given location. Since I didn't have access to a fashion API in this phase, I used a placeholder dictionary to simulate city-based trends. This tool is designed to be easily extended with real-time APIs later.

Both tools were integrated into the LangGraph system. I also started experimenting with **tool routing logic**, so that based on the user's query, the system could decide whether to perform a calculation, get weather data, or return fashion tips.

Key Learnings

These first two levels gave me strong exposure to how agentic AI systems are structured. Unlike traditional chatbots, here I had to:

- Build the logic of how the agent decides *what to do*.
- Integrate external tools and APIs in a reusable way.
- Maintain a clean and modular design, so tools can evolve independently.

LangGraph's visual nature helped me understand how conversations flow, and how tools can be activated intelligently.

Reflections

Working on the WeatherMind project felt like assembling an intelligent machine piece by piece. The process taught me how to combine a language model with decision-making, tools, and real-world perception.

Although I only implemented the first two levels, I now have a working base that can be scaled into a full-fledged multi-agent system in the future. With tool routing logic, memory, and multi-agent collaboration, I see immense potential in where this framework can go next.