DAY – 14

On Day 14, we were introduced to Langflow, a powerful and intuitive visual programming interface built on top of LangChain. Langflow allows users to design, build, and visualize LLM workflows without writing complex backend code. It is particularly useful for those who want to build AI-based applications like chatbots, assistants, and document analyzers with drag-anddrop ease.

Langflow bridges the gap between low-code development and advanced AI pipelines, making it accessible to both beginners and professionals.

1. WHAT IS LANGFLOW?

Langflow is a visual development tool that lets users:

- Create modular, LLM-based applications
- Build chains, agents, and tools visually
- Manage and test prompt flows
- Add file loaders, memory, retrievers, and tools without manual coding

It is built on top of LangChain and is compatible with multiple models like OpenAI's GPT, Google's Gemini, and Anthropic's Claude.

2. SETTING UP LANGFLOW

We learned how to set up Langflow locally using the following steps:

1. Install Python and necessary dependencies.
2. Set up a virtual environment.
3. Run pip install langflow or clone from GitHub.
4. Launch the app using langflow run.
5. Access the Langflow interface through the browser at localhost:7860.

We also configured LLM API keys (like OpenAI/Gemini) within the Langflow UI to start creating flows.

3. TASKS ASSIGNED USING LANGFLOW

We were given four hands-on tasks to understand and apply Langflow's capabilities:

i. Build a Simple FAQ Bot

- Used a basic prompt + LLM component.
- Designed a flow where users ask common questions, and the LLM replies based on a predefined style.
- This task helped us understand the concept of single-turn prompt-response logic.

ii. Add File Loader + Retriever to Create a Document Bot

- Uploaded PDFs using the File Loader component.
- Implemented chunking and vector indexing using a retriever like FAISS or Chroma.
- Added a retriever node to fetch relevant content from the uploaded file when the user asked questions.
- Integrated with LLM to generate responses based on the retrieved chunks.

This demonstrated how we can use Langflow to build document-based QA systems easily.

iii. Integrate Memory for Multi-Turn Conversations

- Added memory components to store previous interactions.
- Connected memory to the LLM block so the conversation became context-aware.
- This was useful for building AI systems that maintain state, remember user inputs, and provide follow-up answers.

iv. Use Agents with Tools

- We built agents capable of calling tools like:
    - Wikipedia Search
    - Calculator
- The agent dynamically decided which tool to use based on the prompt.
- This taught us the ReAct pattern (Reason + Act) used by LangChain agents in Langflow.

4. CAPSTONE: MINI CUSTOMER SUPPORT CHATBOT

As a capstone project, we started working on a Mini Customer Support Chatbot, which included:

- Greeting and identifying user intent
- Searching FAQs or documents
- Performing calculations or tool-based operations (e.g., checking delivery status or pricing)
- Maintaining conversation memory
- Providing personalized and contextual responses

We structured the chatbot using:

- File Loader (for company policy or product documentation)
- Memory + LLM block
- Tool integration (e.g., calculator, search)
- User interface for interaction

This task allowed us to apply everything we learned — from chaining to agents, memory, and tools — in a real-world use case.

CONCLUSION

Day 14 was a hands-on and project-focused session that gave us full exposure to Langflow — a visual tool for designing LLM-based applications. From creating simple FAQ bots to building advanced multi-turn conversational agents with tools and memory, Langflow proved to be an incredibly intuitive and powerful platform.

We also began our capstone project on building a Mini Customer Support Chatbot, where we integrated all the learned concepts into a single, practical AI solution.