# DAY 11

## LangChain & Agents

Exploring Smart AI Workflows using LLMs, Tools, and Memory with LangChain + Gemini

### Introduction to LangChain

LangChain is a powerful Python framework that helps developers build AI applications by combining **Language Models (LLMs)** with **tools, APIs, and memory** into flexible, modular agents.

Instead of just using one AI prompt at a time, LangChain lets us:

- Create workflows (called **Chains**)
- Add tools like calculators or search engines
- Build **Agents** that reason and make decisions
- Maintain **chat memory** for smooth conversations

### Core Concepts Covered

| Concept | Description |
|---|---|
| **LLM** | Large Language Model like Gemini or GPT used to generate intelligent text |
| **Chain** | A sequence of actions combining LLM + logic + tools |

| | |
|---|---|
| **Tool** | A function/API the model can call (search, calculator, translator, etc.) |
| **Memory** | Stores conversation history for context-aware replies |
| **Agent** | An AI brain that decides which tools to use, when, and how |

## Basic Gemini with LangChain

We first tested Gemini in its simplest form using ChatGoogleGenerativeAI:

llm=ChatGoogleGenerativeAI(model="gemini-2.0-flash”,google_api_key=GOOGLE_API_KEY)

response = llm([HumanMessage(content="Explain black holes in simple terms.")])

**Purpose**: Understand how Gemini responds using LangChain’s base wrapper.

## Memory + Conversation Chain

We added **Memory** to allow the AI to remember past messages using ConversationBufferMemory:

memory = ConversationBufferMemory()

conversation = ConversationChain(llm=llm, memory=memory)

Then we asked multiple follow-up questions on topics like AI and Quantum Computing:

topics = ["Artificial Intelligence", "Quantum Computing"]

**Result**: Gemini responded with detailed answers and remembered the previous topic when asked follow-ups like “Tell me a fun fact.”

## Adding Tools for Agent Intelligence

We expanded the functionality by integrating tools like:

- DuckDuckGo Search (DDG)
- Calculator
- Weather API
- English-to-French Translator
- Echo Tool (just repeats input)

Each tool was defined and added to the agent using:

```
from langchain.agents import Tool
tools = [Tool(name="Calculator", func=calculator_tool, description="...")]
```

This gave our agent **real-world abilities**, just like how a human would look up or calculate information before answering.

## Building a Gemini Agent with Tools

Using initialize_agent(), we created an interactive agent that:

- Receives a goal (your input)
- Chooses which tool to use
- Runs calculations, searches, or translations

- ❖ Combines results and replies smartly

```
agent = initialize_agent(
    tools=tools,
    llm=llm,
    agent=AgentType.CONVERSATIONAL_REACT_DESCRIPTION,
    memory=memory,
    verbose=True
)
```

We then used CLI or prompts like:

"What's the weather in Mumbai? Translate 'hello' to French and calculate 273 * 42"

**Outcome**: Gemini figured out which tool was needed for each task and stitched everything into a single response. Magic!

## HTML Output Display

For visualizing the results, we saved the final agent output in a styled HTML format using Python's webbrowser module.

Each action result (search, math, translation, echo) was shown clearly in a structured interface.

File auto-created: langchain_agent_output.html

## Real-Time Chat Agent

We also implemented a **chat-based assistant** using a continuous loop:

```
while True:
    user_input = input("You: ")
    response = conversation.predict(input=user_input)
```

Gemini responded contextually thanks to the memory buffer – just like chatting with a human!

## Advanced Agent Patterns: Multiple Tools + Reasoning

We tested combining multiple tools where Gemini could:

- First **search the web** for a fact
- Then **run a math calculation**
- Return both answers in one go

Example:"Who is the CEO of Tesla, and what is 124 * 15?"

Gemini used DuckDuckGoSearch + Calculator back-to-back!

**Final Takeaways**

| Topic | Key Learning |
|---|---|
| LangChain Setup | Installed LangChain, LLM wrappers, dotenv, tools |
| Conversation Chains | Enabled memory for smart, contextual chat with Gemini |
| Tools & APIs | Added external tools like search, math, translator |

| | |
|---|---|
| Agents | Built intelligent systems that decide which tool to use and when |
| Real-world Interaction | Created end-to-end workflows with tool calling, memory, and responses |
| Output Format | Saved AI responses in clean HTML with date, tool output, and explanation |