Contents

[Multi-Agent 2](#_Toc205070172)

[you can pass the state to the **tool** via *Injected State* annotation 3](#_Toc205070173)

[Two of the most popular multi-agent architectures are: 3](#_Toc205070174)

[Supervisor documentation 4](#_Toc205070175)

[ Durable execution 5](#_Toc205070176)

[Key takeaways: 5](#_Toc205070177)

[Important for error 6](#_Toc205070178)

[Context 7](#_Toc205070179)

[Looked at 7](#_Toc205070180)

[Documentation Track: 7](#_Toc205070181)

A screenshot of a computer

AI-generated content may be incorrect.

All long conversation not looked yet

def call\_model(state: State):  
 trimmed\_messages = trimmer.invoke(state["messages"])  
 prompt = prompt\_template.invoke(  
 {"messages": trimmed\_messages, "language": state["language"]}  
 )  
 response = model.invoke(prompt)  
 return {"messages": [response]}

Trimmer long term

<https://python.langchain.com/docs/how_to/trim_messages/>

from langchain\_core.messages import trim\_messages

**Summary memory**

<https://python.langchain.com/docs/how_to/chatbots_memory/#automatic-history-management>

**❓ Why are we deleting messages?**

Because we just summarized them.

👉 The code says:

“Once history gets long (≥ 4 messages), I’ll summarize all of it into one message — then delete the full message list and only keep the summary.”

AI applications need [memory](https://langchain-ai.github.io/langgraph/concepts/memory/) to share context across multiple interactions. In LangGraph, you can add two types of memory:

* [Add short-term memory](https://langchain-ai.github.io/langgraph/how-tos/memory/add-memory/#add-short-term-memory) as a part of your agent's [state](https://langchain-ai.github.io/langgraph/concepts/low_level/#state) to enable multi-turn conversations.
* [Add long-term memory](https://langchain-ai.github.io/langgraph/how-tos/memory/add-memory/#add-long-term-memory) to store user-specific or application-level data across sessions.

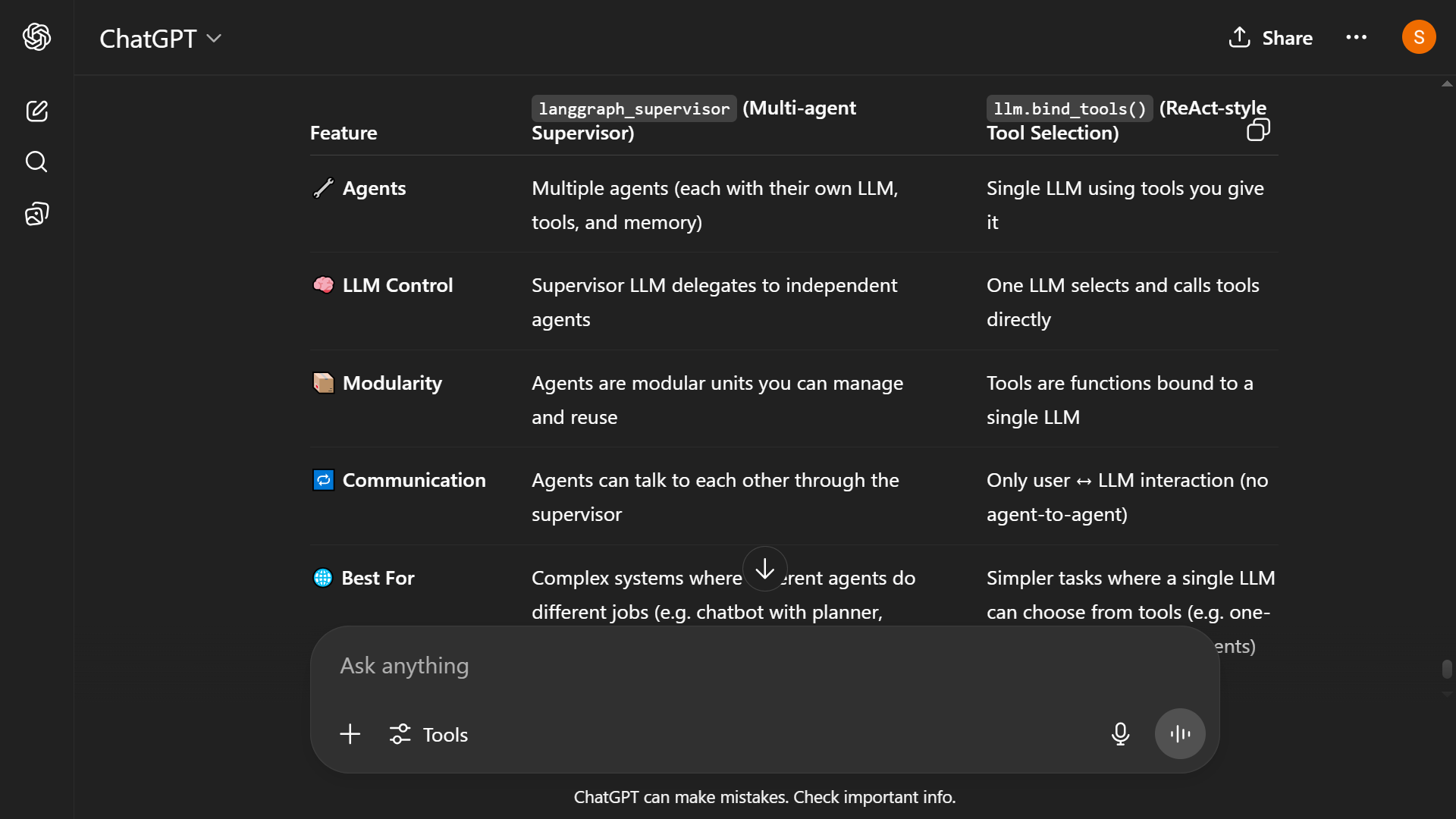
# Multi-Agent

So agent must return command

### you can pass the state to the **tool** via *Injected State* annotation

# Two of the most popular multi-agent architectures are:

* [supervisor](https://langchain-ai.github.io/langgraph/agents/multi-agent/#supervisor) — individual agents are coordinated by a central supervisor agent. The supervisor controls all communication flow and task delegation, making decisions about which agent to invoke based on the current context and task requirements.
* [swarm](https://langchain-ai.github.io/langgraph/agents/multi-agent/#swarm) — agents dynamically hand off control to one another based on their specializations. The system remembers which agent was last active, ensuring that on subsequent interactions, the conversation resumes with that agent.



**✅ Use langgraph\_supervisor when:**

* You want **multiple agents**, each with **distinct prompts, memory, tools, or personality**.
* You need **agent-to-agent communication** or **task delegation**.
* Your system may **scale** to more intelligent or autonomous agents late

#### So i think its preferable

# Supervisor documentation

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* [Durable execution](https://langchain-ai.github.io/langgraph/concepts/durable_execution/): Save progress at key points in the graph execution.

LangGraph's built-in [persistence](https://langchain-ai.github.io/langgraph/concepts/persistence/) layer provides durable execution for workflows, ensuring that the state of each execution step is saved to a durable store. This capability guarantees that if a workflow is interrupted -- whether by a system failure or for [human-in-the-loop](https://langchain-ai.github.io/langgraph/concepts/human_in_the_loop/) interactions -- it can be resumed from its last recorded state

# Key takeaways:

* Needed checkpointer with thread id
* Wrap the non-determisintic things into Tool with @tools so langraph will take care of it
* A screenshot of a computer

  AI-generated content may be incorrect.

# Important for error

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Context

**"Passing outside data"** means you're giving the LangGraph **some useful background info** before it runs — like user inputs, session state, or history — so it performs better.

**Context engineering**

# Looked at

#### LangGraph Cloud: How to Publish your AI Agents in Cloud?

# Documentation Track: