🐝 Swarm Pattern :

🔹 Pehle:

- Hamaray paas **single agents** thay jo ek hi kaam kar sakte thay.
- Agar task complex hota to hume multiple agents manually set karne padte, aur unka coordination mushkil hota tha.

Phir OpenAI ne Swarm Banaya:

- **Goal:** Multi-agent systems ko ek **organized aur safe platform** mein laya jaye
- Taa ke **developers ko khud routing, chaining aur coordination na karna pade**
- Aur agents mil kar **large complex tasks** easily solve kar saken

---

Agents ek banda hai jo tumhari madad karta hai. Swarm ek puri "team" hai jisme bohot agents mil kar tumhara bara task complete karte hain.

Swarm OpenAI ka banaya hua ek software framework hai.
jo developers ko help karta hai **intelligent AI agents banane mein** jo:

- **Soch sakte hain (think)**
- 📋 **Plan kar sakte hain (plan)**
- ⚡ **Apne aap tasks kar sakte hain (do work on their own)**

🎯 Main Purpose:

Swarm ka purpose hai safe aur smart multi-agent system banana, jahan multiple agents mil kar bade tasks solve kar saken.

- Har agent ek **chhoti zimmedari** leta hai
- Phir sab mil kar **complex problem solve** karte hain
- Tum dekh bhi sakti ho ke kaun agent kya kar raha hai (transparency)

Kaise Kaam Karta Hai :

1️⃣ Tum ek **bada task** dete ho (for example: “Ek website bana do”)
2️⃣ Swarm us task ko **chhote tasks** mein tod deta hai
3️⃣ Har chhota task ek **specialist agent** ko assign hota hai

e.g. Web Developer Agent, Designer Agent, Marketing Agent
4️⃣ Agents mil kar kaam complete karte hain
5️⃣ Tum dekh sakti ho real-time mein ke agents kya kar rahe hain

📌 Prompt Chaining:

Prompt Chaining wo technique hai jisme ek prompt ka output, agle prompt ka input banta hai — taake complex tasks step by step solve ho saken.

🍲 Easy Example (Cooking Recipe):

👩‍🍳 Tum AI ko kehti ho:

1️⃣ **Prompt 1:**
"Mujhe ek easy dinner dish batao."
➡ **AI Output:** *"Chicken Biryani"*

2️⃣ **Prompt 2:**
"Chicken Biryani ki ingredients list batao."
➡ **AI Output:** Rice, Chicken, Masala, Oil...

3️⃣ **Prompt 3:**
"Ab in ingredients ke sath recipe steps likho."
➡ **AI Output:** Step 1 wash rice... Step 2 cook chicken...

🎯 Samajhne wali baat

Har step ka **jawab agle step ka input** ban raha hai.
Yehi hai **Prompt Chaining**!

🔀 Routing (Simple Definition):

Routing ka matlab hai: user ke sawaal ko sahi agent ya expert tak bhejna.

Yani system decide karta hai ke **kaunsa agent** is question ka best jawab dega.

📌 Easy Example (School Classroom)

Socho tumhari class mein 3 teachers hain:

👩‍🏫 **Math Teacher**
👨‍🏫 **English Teacher**
👩‍🏫 **Science Teacher**

1️⃣ Student ne poocha:
"2 + 2 kitna hota hai?"
➡ System ne question **Math Teacher** ko bhej diya.

2️⃣ Agla student ne poocha:
"Translate 'Apple' in Urdu."
➡ System ne question **English Teacher** ko bhej diya.

3️⃣ Teesra student ne poocha:
"Water ka formula kya hai?"
➡ System ne question **Science Teacher** ko bhej diya.

🎯 Samajhne wali baat

- **Routing = Sahi sawal ko sahi expert tak bhejna.**
- Is se time bhi bachta hai aur jawab accurate milta hai.

⚡ Parallelization (Simple Definition)

Parallelization ka matlab hai ek hi waqt mein ek se zyada tasks ko chalana — taake kaam jaldi complete ho jaye.

📌 Easy Example (Kitchen Example):

👩‍🍳 Socho tum dinner bana rahi ho:

- Ek dost **rotiyan bel raha hai**
- Dusra dost **sabzi kaat raha hai**
- Tum **daal paka rahi ho**

➡ Teeno kaam **ek sath ho rahe hain** → ye hai **parallelization**.
Agar tum ye teeno kaam **ek ek karke** karti to zyada time lagta.

AI Context Mein Parallelization:

- Agar user ke 3 sawaal aaye
  - Agent 1 → pehla sawaal solve kare
  - Agent 2 → doosra sawaal solve kare
  - Agent 3 → teesra sawaal solve kare
- Sab agents **ek sath** kaam karenge → jawab jaldi mil jayega.

🎯 Key Point:

✅ Parallelization = **Time bachaana** by running multiple tasks at the same time.

🎯 Short Summary:

- **Routing** = Kis agent ko input dena hai (direction choose karna).
- **Parallelization** = Multiple kaam ek sath karna (speed barhana).

🔷 Orchestrator

Orchestrator ek manager hota hai jo decide karta hai ke kaunsa worker kya kaam karega.
Ye **tasks ko distribute** karta hai aur ensure karta hai ke sab sahi chal raha hai.

Evaluator (Nigran/Checker):

Evaluator wo hota hai jo check karta hai ke worker (agent) ka kaam sahi hai ya nahi.
Ye basically **output ko evaluate** karta hai.

⚡ Optimizer (Behtari Lane Wala):

Optimizer wo hota hai jo system ya output ko improve karta hai taake zyada fast, sahi aur efficient ho.

✅ All Concepts in 1-Line Definitions

1. **Prompt Chaining:**

Ek prompt ka output agle prompt ka input banta hai taake kaam step-by-step ho.

2. **Routing:**

User ke input ko sahi agent ya expert tak bhejna.

3. **Parallelization:**

Ek hi waqt mein multiple tasks ko run karna taake kaam jaldi ho.

4. **Orchestrator:**

Manager jaisa system jo decide karta hai kaunsa agent kya kaam karega.

5. **Worker:**

Agent jo asli kaam perform karta hai jo usse assign kiya gaya ho.

6. **Evaluator:**

Wo system ya agent jo kaam ko check karta hai ke sahi hai ya nahi.

7. **Optimizer:**

Wo system jo performance aur output ko aur behtar banata hai.

✨ **Bonus Tip**:
Yeh 7 roles mil kar aik **powerful multi-agent AI system** banate hain 💡

📦 Module Kya Hota Hai?

Python mein module aik file hoti hai jisme functions, classes, aur variables likhe hote hain jise hum import karke dobara use kar sakte hain.

📌 Easy Example

Socho tum ek **Math module** banati ho (maths.py):

# maths.py 	
def add(a, b):
    return a + b
 	
def sub(a, b):
    return a - b
	Phir doosri file mein use karogi:
	
	import maths
	
	print(maths.add(5, 3)) # Output: 8

➡ Ab tumhe bar-bar add aur sub ka code likhne ki zaroorat nahi.

🎯 Key Point

- **Module = Ek reusable file of code**
- Time bachaata hai aur code clean banata hai
- Python ke andar bohot sare **built-in modules** hain (jaise math, os, random).

Base url: https://ai.google.dev/gemini-api/docs/openai?hl=en

google generative baseurl => https://ai.google.dev/gemini-api/docs/openai

base_url="https://generativelanguage.googleapis.com/v1beta/openai/"