Agent :

Agent aik smart digital assistant hota hai jo Large Language Model (LLM) ke upar kaam karta hai. Iska kaam hai user ke input ko samajhna, sochna (LLM ka use karke), zarurat padne par tools ka istemal karna, aur human jese jawab dena. Agent ko aap kuch specific instructions, tools, aur ek model (brain) dete ho jiske zariye wo kaam karta hai. Ye agents aapke apps ka core part hote hain — unko har kaam ke liye train kiya ja sakta hai jaise jokes batana, weather lana, ya form bharna.

Basic Configuration:

Basic configuration ka matlab hai agent ko batana ke wo kis tarah kaam kare. Isme teen tuning parameters hoty hy(yani model ko control karne k steps):

1- **Instructions** — jisme aap agent ko uska behavior ya tone samajhte ho (jaise: “Always speak in urdu”).

2- **Model** — jisme aap decide kartay ho ke agent kis AI brain (LLM) ko use karega, jaise gpt-4, o3-mini, ya gemini.

3- **Model Settings** — agar aap chahte ho ke model zyada creative ho ya controlled ho, to aap model_settings mein tuning parameters(yani model k behaviour ko control karny waly methods) set kartay ho — jaise temperature (creativity), top_p (randomness filtering), aur max_tokens (response ka size).

1- `temperature` – **Kitna creative jawab ho**

- **0.1** → Serious, boring
- **1.0** → Fun, creative, thoda crazy

2- `top_p` – **Words ko filter karna**

- **1.0** → Har word allow
- **0.3** → Sirf top safe words socho: Best 30% ideas pick karo.

3- `max_tokens` – **Jawab ka size**

- **50** → Chhota answer
- **200+** → barra answer socho: Kitne lafz bolne hain?

4- **Tools** — aise Python functions hote hain jinhein agent run kar sakta hai jab use kisi specific task (jaise weather lana) ki zarurat ho.

In sab settings ke zariye agent ko ek smart, focused, aur customized assistant banaya ja sakta hai jo aapke aapke goals ke mutabiq kaam karta hai.

Context:

Ek box jisme student ki info hoti hai, jese uska naam aur wo konsa subject padh raha hai.

Context Class Banate Hain:

```
from dataclasses import dataclass

@dataclass

class StudentContext:

    name: str

    subject: str
```

Ek `StudentContext` naam ka box bana liya hai jisme:

- `name`: student ka naam
- `subject`: konsa subject padh raha hai (Math, Science, etc)

Context aik dataclass hi hoti hy or datclass python ki simple class ki tarah hoti hy bus python ki class me constructors use kar k class define hoti thi is me without constructors kiye class define hoti hai.

Output types:

Agent ka output kya hota hai?

Jab hum OpenAI ke SDK mein `Agent` banate hain, to wo by default sirf text (string) mein jawab deta hai.Jaise: `"Meeting kal hai 3 baje”.`Lekin agar aap chahain ke agent structured tareeqay se response de (jaise ek object ya data format mein), to aap usay bata saktay ho ke kis output type mein data chahiye.

`output_type` ka kya matlab hai?

Aap agent ko yeh keh sakte ho:

> “Mujhe jawab sirf ek specific format mein chahiye – jaise ek form ya dictionary jaisa structure.” Iske liye aap **`output_type`** parameter ka use kartay ho.

Handoffs:

Ek agent ka doosray agent ko kaam de dena.

Jaise:

- Ek agent user se baat karta hai lekin sab kuch nahi janta.
- Agar user booking ke baare mein poochta hai to:
  - Woh usse Booking Agent ke hawale kar deta hai.

- Agar refund poochta hai to:
    - Woh Refund Agent ko kaam de deta hai.

Dynamic Instructions:

Normally jab hum ek Agent banate hai to usko fixed instructions dethy hy:

```
Agent(
  name="Support Agent",
  instructions="Help the user with their queries."
)
```

Static instructions wo hoti hain jo agent ko banate waqt ek fixed text ke roop mein di jati hain, jaise: "Help the user with their query." Ye har user ke liye same hoti hain aur context (jaise user ka naam, mood, ya goal) ko use nahi kar sakti. Dusri taraf, dynamic instructions ek function ke zariye banti hain jo runtime par context ke data (jaise: "User ka naam Mueza hai aur wo confused hai") ko dekh kar instructions banata hai. Is se agent har user ke liye personalized behavior dikha sakta hai. Agar hum chahte hai ke agent user ki situation ke mutabiq sochay, to dynamic instructions ka use zaroori hota hai.

Lifecycle events (hooks):

What are hooks?

Jab hum koi agent banate hai to wo hame normal answer kar deta hai lakin hamaey ye nahi pata hota hy k usny kia kia step kiya aur kaise answer mila to isk liye hum hooks use karty hy.Hooks ek tarika hai agent ke har step pe apna code chalanay ka.Agent jab bhi koi kaam kare (start, complete, error), hamara code bhi chale — jaise message print karna, file mein likhna, ya kuch aur.

Documentation Breakdown:

> “Sometimes, you want to observe the lifecycle of an agent. For example, you may want to log events, or pre-fetch data when certain events occur.”

Explanation:

Kabhi kabhi hum chahenge ke jab agent kaam kare, hum dekh saky ke:

- Wo kab start hua
- Wo kab complete hua
- Kya output aya
- Koi error to nahi aayi

Isko observe karne ke liye hum hooks use karty hy.

> “You can hook into the agent lifecycle with the hooks property.”

Explanation:

Agent banaate waqt ek hooks= naam ka option hota hai.
hum yahan apna custom code de sakty hy jo agent ke har stage par chale.

AgentHooks kya hota hai?

AgentHooks ek template class (yaani base class) hoti hai jo hum use karty hy agar hum chahy ke:

> “Agent ke kaam karne k har step par kuch karna hai — jese logs likhna, data save karna, error track karna.

Guardrails:

Guardrails ka matlab hota hai safety checks yani aise rules ya filters jo ensure karte hain ke user jo input de raha hai woh sahi ho, relevant ho, ya safe ho, before or during agent ka kaam.

Ye ek "security layer" hai jo model ke input ya output par kaam karta hai.

```
def block_if_offensive(input):
    if "stupid" in input:
        return False  # Mat bhejo agent ko
    return True

guardrails = [block_if_offensive]

agent.run(user_input, guardrails=guardrails)
```

Ab agar "stupid" likha hai to input block ho jaayega

Cloning/copying agents:

Jab hum ek Agent bana lete hai (for example, ek pirate-style bolne wala chatbot matlab jiski tone ships waly logo jaisi hoo), to hum usi agent ko duplicate (copy) kar sakte hai aur kuch cheezein (jaise name, instructions) change kar sakte hai — bina naye agent ko scratch(change) kiye.

> Ye bilkul aise hi hai jaise ek file ka copy banao, aur usme kuch changes karo.

Forcing tool use:

1. Tool Kya Hota Hai?

Ek agent (chatbot) banaya hai, lekin usko kuch kaam khud se nahi aate — jaise:

- Calculation karna
- Currency convert karna
- Weather batana
- File read karna
- Code run karna

Toh hum uske liye Tool banate hain — yani functions jo woh agent use kar sakta hai.

2. Tool Use Karne Ke 3 Modes (`tool_choice`):

Hum agent ko bataty hy ke tool kaise use karein uske 4 options hote hain:

`tool_choice`	Matlab
`"auto"`	Agent khud decide kare: tool use kare ya khud jawab de
`"required"`	Agent tool lazmi use kare — bina tool ke jawab nahi de sakta
`"none"`	Agent tool bilkul bhi use nahi karega
`"calculator"` **(ya tool ka naam)**	Agent sirf wahi specific tool use karega

Example:

```
ModelSettings(tool_choice="calculator")
```

`4-tool_use_behavior?`

Ye batata hai ke tool ke use ke baad agent kya kare:

Option	Matlab
`"default"`	Tool ke result ke baad agent aage response continue kare
`"stop_on_first_tool"`	Tool ka result hi final answer ban jaye — no further reply