Dynamic aur context related hain, lekin same cheez nahi.

- **Context** → Pehle se di hui fixed info, jo har request me use hoti hai (jab tak tum change na karo).
- **Dynamic** → Info jo run-time pe badalti rahe, situation ya input ke hisaab se update hoti hai.

Example:

- Context: `"name = Akbar"` (fix jab tak tum change na karo)
- Dynamic: `"current_time"` ya `"user mood"` jo har request me naya ho sakta hai.

- • **Context** → Background data jo tools ya model ko saath-saath milta hai.
- • **Dynamic instructions** → Jab tum context + input ko mila ke, har request ke waqt **naya** instruction banate ho LLM ke liye.

Dynamic Instructions: Ye woh *hidayat* (instructions) hoti hain jo tum LLM ko run-time par dete ho, jo har request ke hisaab se badal sakti hain. Ye zyada *behavior control* hoti hain, data nahi.

> Example: `"Always answer in Urdu"` ya `"Act as a strict teacher"` — aur tum ye har request mein badal sakte ho.

Simple difference:

- **Context** → "Ye mera fixed background info hai" 🎒
- **Dynamic Instructions** → "Aaj tum is tareeqe se kaam karo" 🎭

Context aur Dynamic ka farq:

- **Context** → woh fixed data hota hai jo tum agent ko dete ho, jaise user ka naam, location, ID, etc.

  Ye run ke start me set hota hai aur jab tak tum change na karo, wahi rehta hai.
- **Dynamic Instructions** → yeh ek **function** hota hai jo har baar **agent ke chalne se pehle** call hota hai,

  taake tum instructions **on the fly** change kar sako, based on current context, user ka input, ya koi external halat.

Dynamic ka kaam:

Dynamic instructions ka kaam yeh hota hai ki:

- Tum har run ke waqt **naye instructions** bana sako.
- Instructions **runtime pe calculate** ho sakte hain.
- Agar user ka naam, mood, ya koi situation change ho gayi hai, to agent ko naye tareeke se behave karwa sako — bina puri agent definition ko dobara banaye.

Iska fayda ye hai ke tumko hardcode karna nahi padta — har naye input ka jawab alag hoga.

Static vs Dynamic Example:

Static Instructions	Dynamic Instructions
"Tum helpful assistant ho." (fixed)	Ek function jo kehta hai: `"Tum helpful assistant ho aur user ka naam {context.name} hai."`
Har run me same rahega	Har run me context ke data se update hoga

4Simple Example Without Code

Imagine tum ek teacher ho:

- **Static**: Tumne decide kiya tum hamesha Urdu mein padhana hai.
- **Dynamic**: Ek din ek student kehta hai — “Madam, is baar sirf English mein example do.”
  Tum ek class ke liye change karti ho, agle period mein phir Urdu mein ho jaati ho.

- **instructions** → fix hota hai, agent banate waqt likha jaata hai, har run me same rehta hai.

- **dynamic_instructions** → function hota hai, jo har run ke waqt **naya banega** based on current user info / conversation state.

- Tum is function ko **Agent** banate waqt parameter ke through pass nahi karte, balki **Runner.run()** ya **Runner.run_sync()** ke waqt dete ho.

---

1 Tracing kya hota hai?

- Tracing ka matlab hai **poora record rakhna** ki tumhara program kaunsa kaunsa kaam kar raha hai, kis order me, kitna time laga.
- Jaise ek *timeline* ya *story* jo tumhare code ki journey likh rahi ho.
- Isme har event ka start, end, time, aur result store hota hai.
- Example:
    - Agent start hua
    - LLM ko call gaya
    - LLM ne output diya
    - Tool call hua
    - Output return hua

📌 **Tracing = pura safar ka record.**

2️⃣ Span kya hota hai?

- Span tracing ka ek **chhota hissa** hota hai.
- Har span ek **specific kaam** represent karta hai.
- Ek trace me multiple spans hote hain.
- Har span ka apna **start time, end time, aur meta-data** hota hai.
- Example:
    - **Trace**: "Agent ne question ka jawab diya."
        - **Span 1**: "Prompt LLM ko bhejna"
        - **Span 2**: "Tool se weather data lena"
        - **Span 3**: "User ko final jawab banana"

📌 **Span = ek chhoti clip**, **Trace = pura movie**.