AGENTIC AI USING LANGGRAPH

→ Agentic AI characteristics :

1. Autonomous
2. Goal oriented
3. Planning
4. Reasoning
5. Adaptability
6. Context awareness.

- Autonomy can be controlled –

a) Permission Scope

What tools or actions the agent can perform independ.

Eg: Can screen candidates, but need approval before rejecting anyone.

b) Human-in-the-loop (HITL) –

Insert checkpoints where human approval is required before continuing

Eg: Can I post this JD?

c) Override controls –

~~Avoid~~ Allows users to stop, pause or change the agent's behaviour at any time.

d) Guardrails / Policies –

Defines ethical boundaries the agent must follow.

Eg: Never schedule interviews on weekends.

Dont use informal language

- **Reasoning**

During Planning

→ Tool selection.
→ Resource estimation. [time, dependencies etc]
→ Goal decomposition.

During Execution

→ HITL handling (knowing where human is needed, when to pause)
→ Error handling. (Tool / API failure. and recovering).

- **Adaptability**

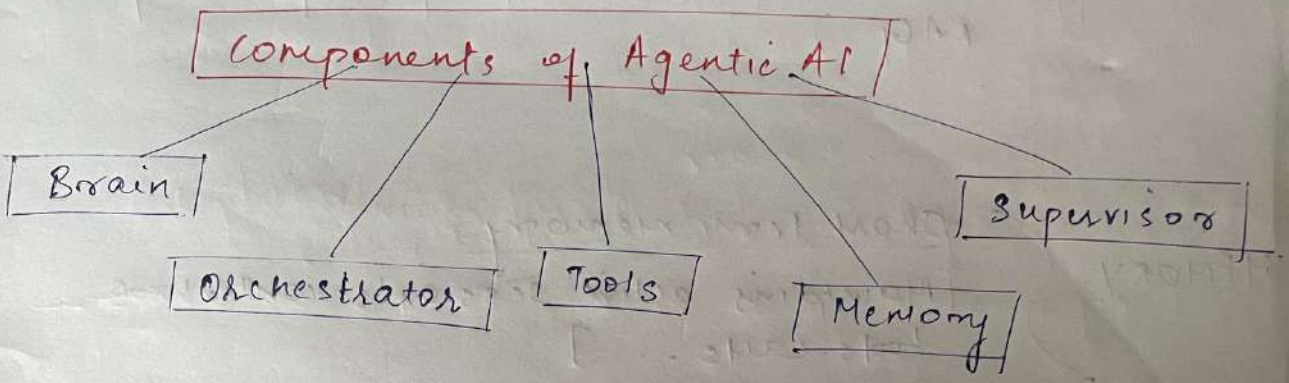
→ modify its plan / actions.
Situation:
- Failure (Calendar API)
- External feedback (less no. of application)
- Changing goals.

- **Context awareness.**

→ Types of content an agent have:
1. The ongoing goal.
2. Progress till now + Interaction history.
3. Environment state.
4. Tool response.
5. User specific preferences.
6. Policy or Guardians.

→ Context awareness is implemented through memory.

1. Short term memory
2. Long term memory



→ Brain is usually our LLM.

BRAIN
- Goal interpretation
- Planning
- Reasoning
- Tool selection
- Communication

ORCHESTRATOR (Executor)
- Task sequencing
- Conditional Routing
- Retry logic (handle failed tool calls / attempts)
- Looping & iteration
- Delegation → decides whether to hand off work to tools, LLM or humans.

TOOLS.

External actions.
(Perform API calls)

Knowledge base access.
(Retrieve domain-specific info using RAG)

MEMORY.

Short term memory
[Maintains active session's context → tools calls ...]

Long-term memory
[Past interactions & user preferences]

State tracking.
[Monitor progress — completed | Pending]

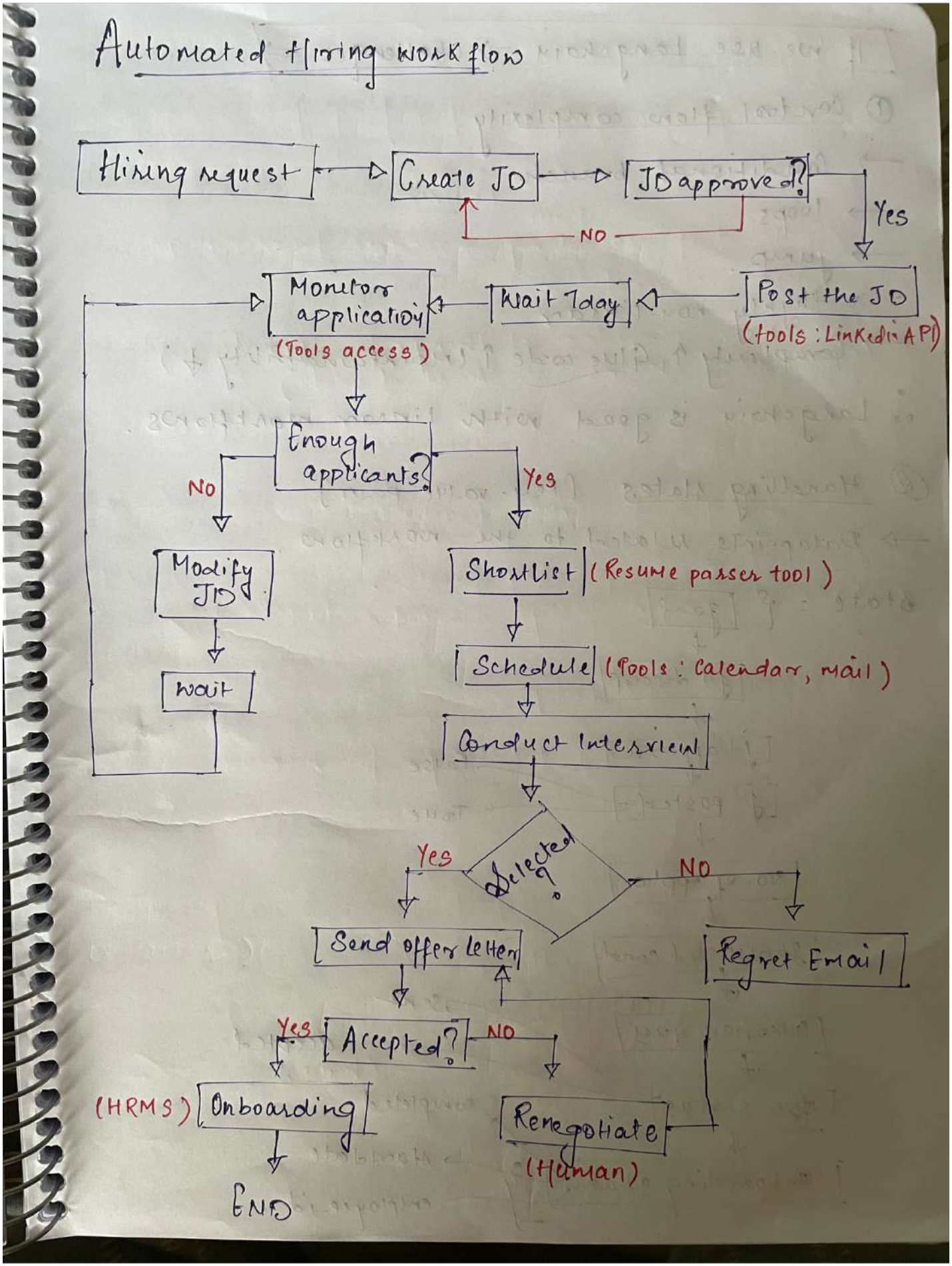
SUPERVISOR
(human-in-the loop)
Agent & Human.

Approval Requests.
(HITL)

Guardrails enforcement
Blocks unsafe behaviour.

Edge case escalation.
Alerts human when conflict arises / uncertainty.

Automated Hiring workflow



If we use Langchain ⇒ Challenges ⇒

① Control flow complexity

→ Conditional branch
→ loops
→ jump.
→ Highly non-linear
→ Complexity ↑ Glue code ↑ (Maintainability ↓)

∴ Langchain is good with Linear workflows.

② Handling states [Key-value pair]

→ Datapoints related to the workflow.

state = { [goal]
↓
[jd descrip]
↓
[jd approved]
↓
[jd posted] — False / True
↓
[No. of applic]
↓
[Shortlisted cand]
↓
[Interview Ques]
↓
[offer status] — sent / accepted
↓
[onboarding status] — completed / start date / employee_id.

→ Langchain doesn't provide option for tracking states of the workflow → manual code.

∴ LangGraph handles it

State object → Pydantic
→ Typed dict

(Langchain is stateless | LangGraph is stateful)

③ Event driven execution.

→ Langchain is used | made for sequential workflows [Once it has started executing the chain → it doesn't wait → executes till end goal is achieved].

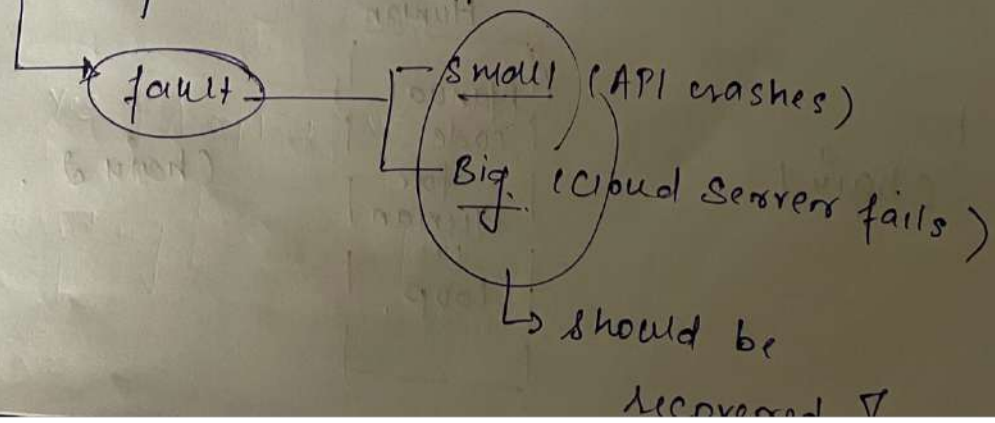
∴ Not reliable for event-driven execution.

"Glue code 1ses"

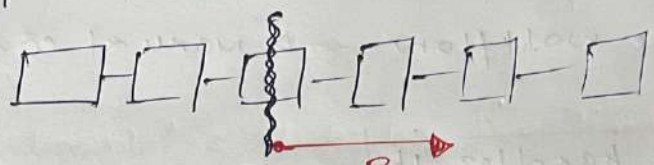
∴ LangGraph ✓
(checkpointer)

④ Fault Tolerance.

Long running workflows..



Ideally if

chain : 

<Instead of stalling → resume from where fault occurred ! >

∴ Langchain provides Nothing for this !
(Assumes chains are short-lived)

∴ LangGraph

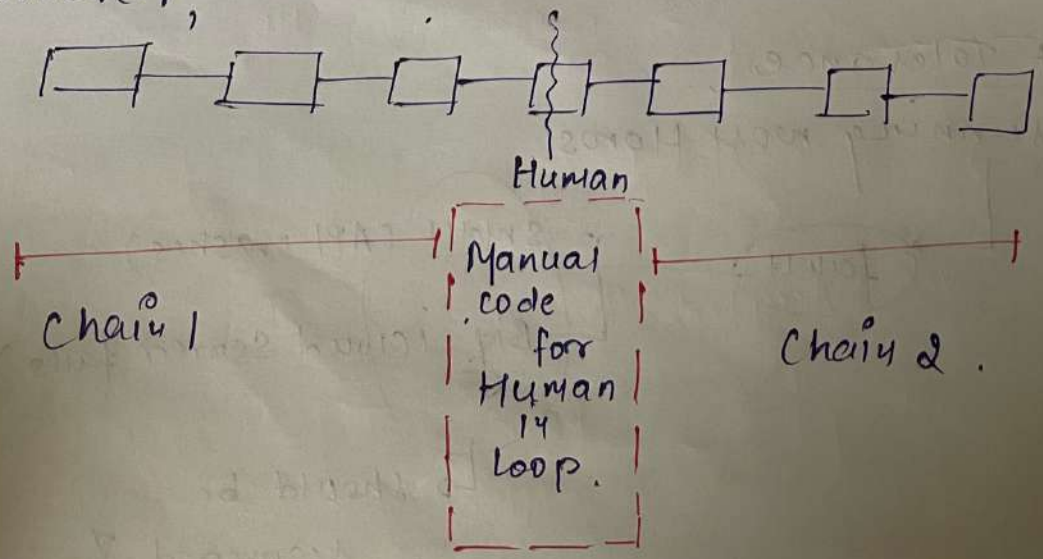
→ Retry Logic for small faults.

→ Recovery Logic for big faults.
<State is continuously stored & tracked >

(5) Human-in-the-Loop

No Logic for chain-pausing for human approval.

However,



∴ It is not reliable.

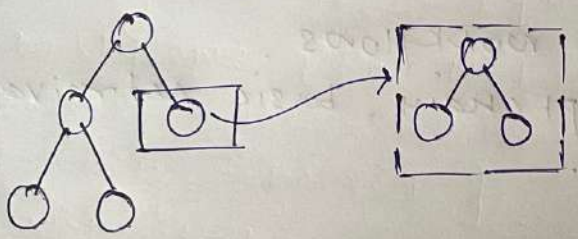
- **Lang Graph**

→ We will save the ~~step~~ state till Node in which human appr. is required & then can resume after that even in days.

→ Introduce checkpointers concept.

⑥ **Nested workflow**

Langgraph has concept of subgraphs to build complex systems with multiple components.



→ We can use multi-agentic systems.

→ Reusability ✓

∴ Langchain dont have ∀.

⑦ **Observability**

→ It refers to how easily you can monitor, debug & understand what your workflow is doing at runtime.

LangSmith is used ∀ → only monitor Langchain & can't monitor Glue code

So, we try to build complex system with Langchain → we will get partial observability

LangGraph provides whole timeline to Langsmith.

→ So we can backtrack.

WHEN TO USE WHAT ?

Langchain
- simple workflows
- Linear workflows like prompt chain, basic retreival system.

LangGraph
- Complex, Non-linear workflows
- Conditional paths
- Loops
- Human in the loop
- Multi agent coordination
- Event-driven execution

< LangGraph handles workflow orchestration while Langchain provide building blocks >