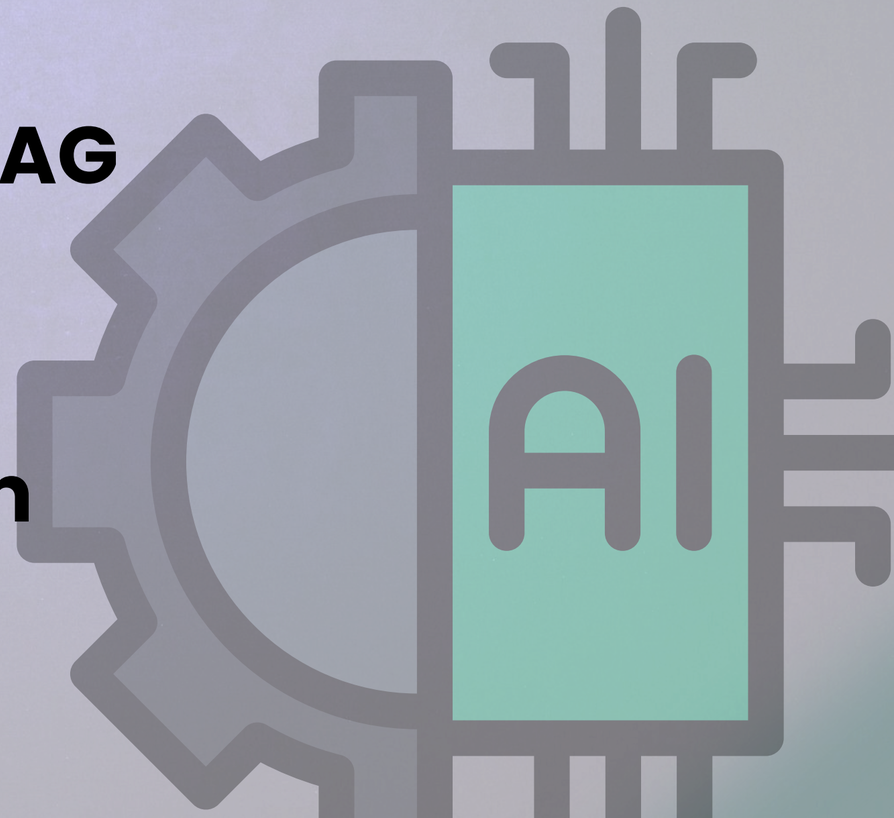
Introduction to Agentic RAG

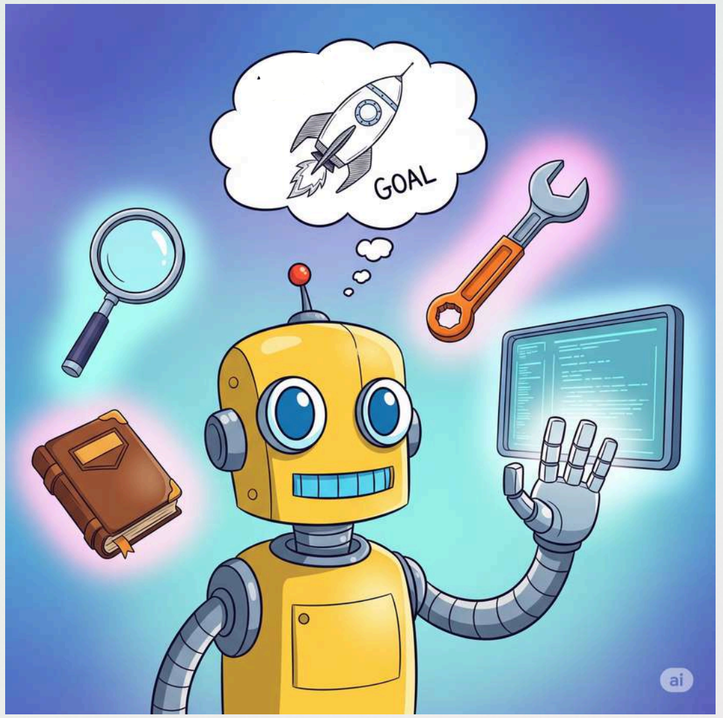
- **Introduction to Agents and Agentic RAG**
- **Traditional RAG vs Agentic RAG**
- **Introduction to LangGraph**
- **Workflows and Memory in LangGraph**
- **Hands on project**



WHAT IS AN AGENT?

edquest

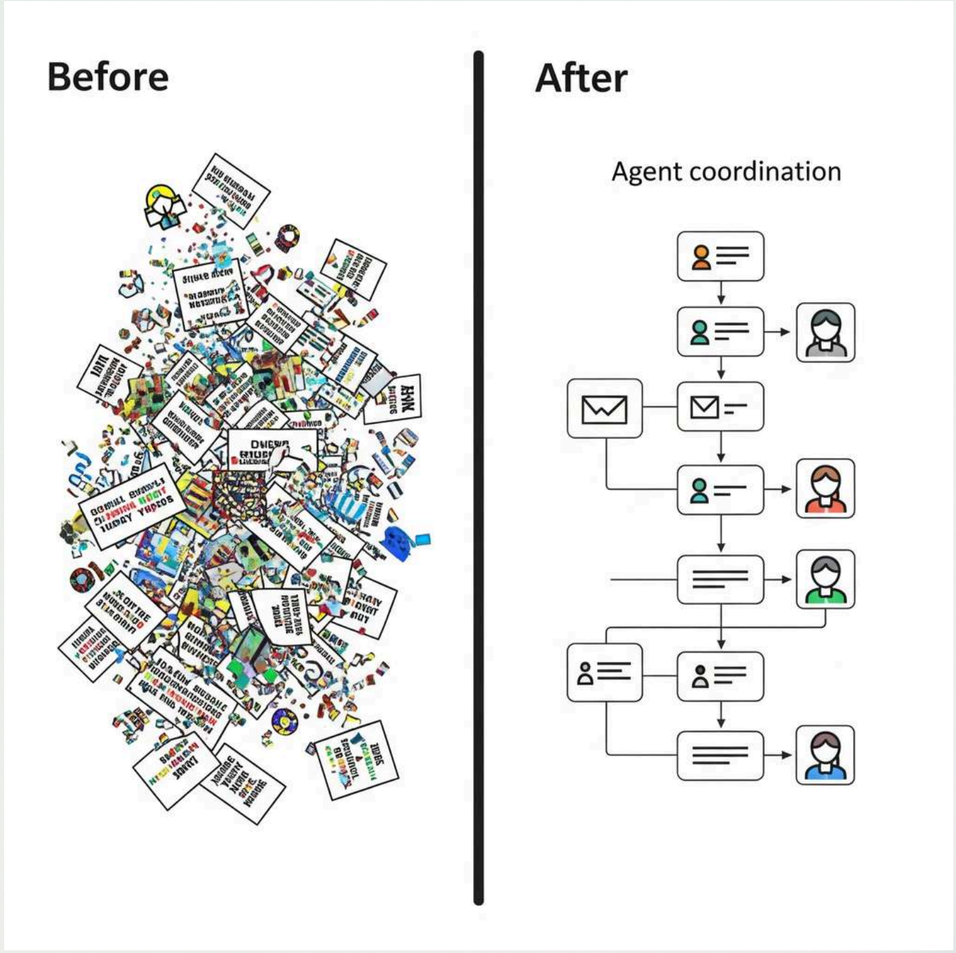
- Has a goal
- Makes decisions
- Can use tools or other agents



WHY DO WE NEED AGENTS?

edquest

- Complex tasks need structure
- One LLM prompt isn't enough
- Agents help organize the work



AGENTIC RAG

edquest

Agentic RAG combines autonomous AI agents with RAG to enable LLMs to **reason, plan, and execute.**

- Add intelligence to your RAG pipelines
- Brings structure and adaptability
- Use agents to solve complex tasks

info@edquest.co.in

+91-9467752177

edquest

TRADITIONAL RAG AND AGENTIC RAG

edquest

RAG

- One-way flow
- No memory or feedback
- No way to improve results

AGENTIC RAG

- Multi-hop questions
- Need for external tool use
- Dynamic information gathering

INTRODUCTION TO LANGGRAPH

edquest

LangGraph is an advanced AI framework to **build stateful, controllable AI workflows**

- Lets you define **how AI agent thinks and acts** step-by-step — like drawing a flowchart for your LLM logic.
- Easily build workflows that require **memory, control, and reliability**
- Go beyond prompt chaining → focus on **workflow orchestration**
- Add **structure and safety** → like a flowchart for AI applications

CORE COMPONENTS OF LANGGRAPH

edquest

- **Nodes** → Represent individual **tasks or functions** to be executed within the workflow.
- **Edges** → Define how nodes are connected and how **information flows**.
- **State** → Acts as **shared memory** across the workflow, storing variables and history.
- **Checkpointers** → Save the state of the workflow at defined points.

USE CASES OF LANGGRAPH

edquest

- **Conversational AI & Chatbots** → Stateful, contextual dialogue
- **Retrieval-Augmented Generation** (RAG) → Multi-step retrieval + reasoning
- **Business Process Automation** → Document review, approvals, workflows
- **Research & Analysis Pipelines** → Collect → Summarize → Compare results
- **Agent Systems** → Autonomous decision-making with tool use

WORKFLOWS IN LANGGRAPH

edquest

A workflow in LangGraph is a sequence of connected nodes that define how tasks are executed, controlling the flow of data and operations.

Types of Workflows:

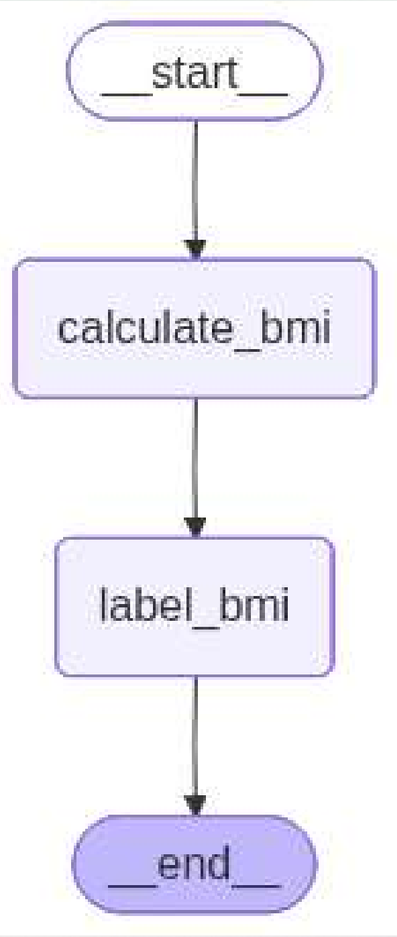
- **Sequential**: Nodes run one after another.
- **Parallel**: Nodes run simultaneously.
- **Conditional**: Nodes run based on conditions.
- **Iterative**: Nodes repeat until a condition is met.

SEQUENTIAL WORKFLOWS

edquest

Sequential Workflow executes tasks in a **strict, linear order**, where **each step begins only after the previous** one completes.

- Ensures predictable execution and **easy tracking** of process flow
- Suitable for tasks that require **step-by-step processing**

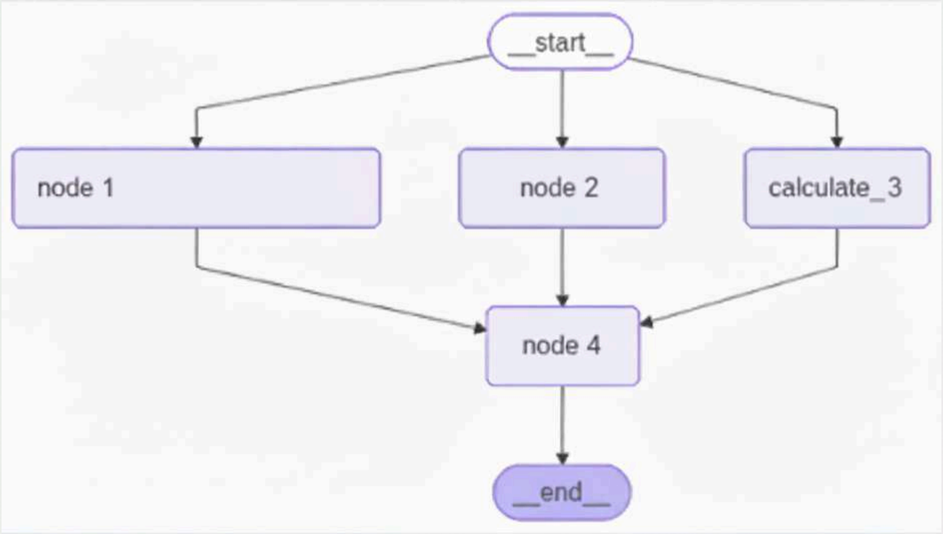


PARALLEL WORKFLOWS

edquest

Parallel Workflow executes **multiple tasks simultaneously**, without waiting for one to finish before starting the next.

- Improves efficiency by **handling tasks concurrently**
- Suitable for **speeding up processes** (e.g., data processing, notifications)

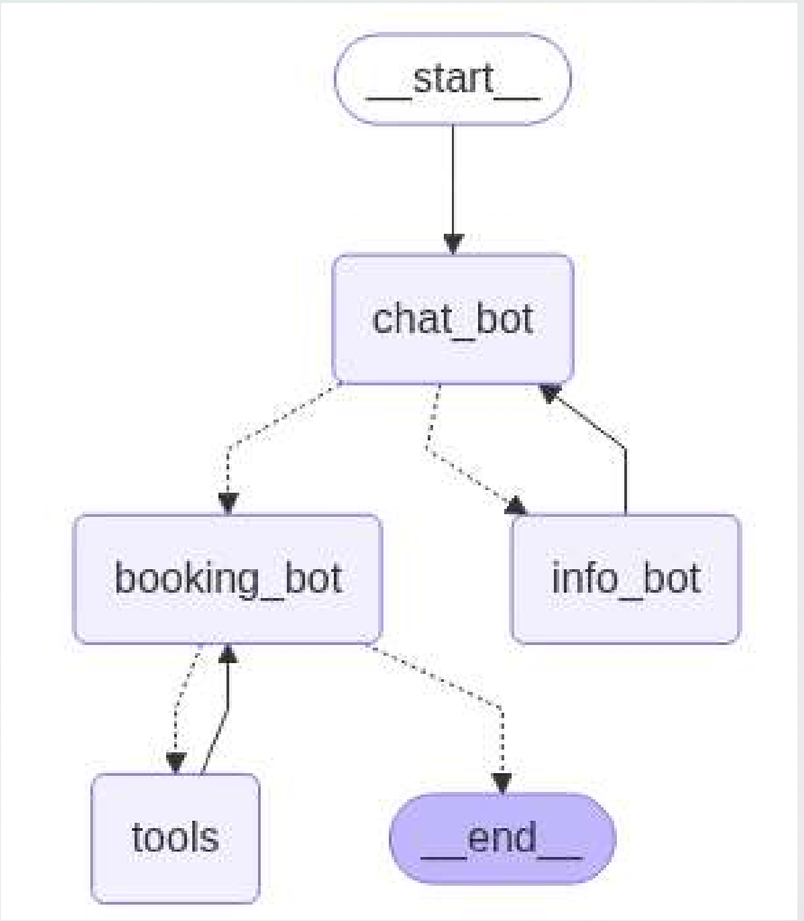


CONDITIONAL WORKFLOWS

edquest

Conditional Workflow executes tasks based on decision points, where the **next step depends on a condition** or rule being true or false.

- Allows branching paths **depending on specific criteria**
- Suitable for processes that **require if-else decisions** (e.g., approvals, validations)

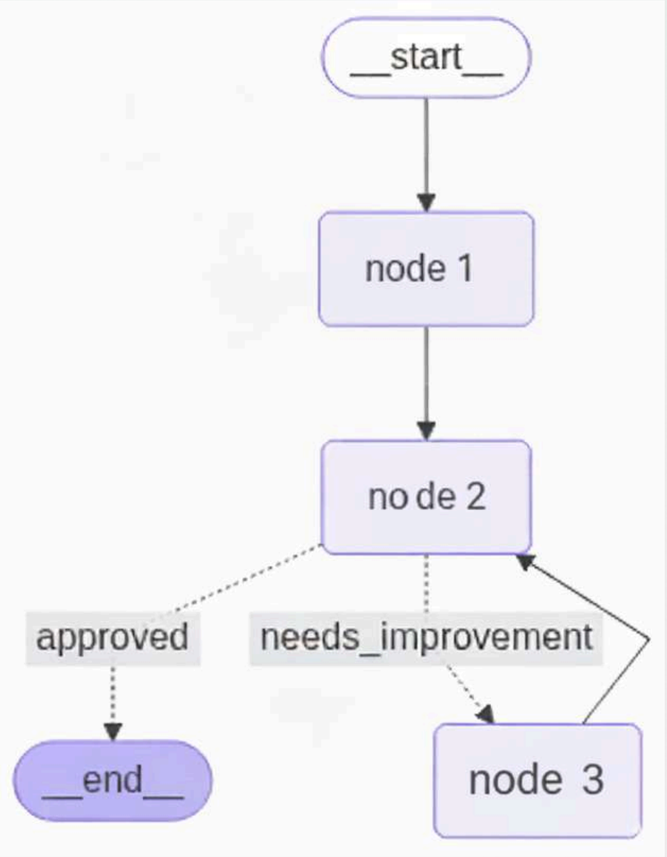


ITERATIVE WORKFLOWS

edquest

Iterative Workflow repeatedly **executes a set of tasks** in a loop **until a specific condition is met** or the desired result is achieved.

- Ensures continuous **refinement and improvement** of the output.
- Suitable for processes that require **repetition or re-evaluation**



MEMORY IN LANGGRAPH

edquest

Memory is used to **store and manage information across workflow** steps.

- Supports **short-term memory** (conversation context) & **long-term** memory (persistent data)
- Keeps track of **variables, entities, past decisions**

Example:

- User: “Book me a flight to New York”
- Later: “Change it to evening flight” → AI remembers prior request

CHECKPOINTERS IN LANGGRAPH

edquest

Checkpointers **saves** workflow progress at **defined checkpoints**.

Enables:

- Recovery after crash
- Debugging & replay
- Long-running tasks continuation

Example:

- **In multi-step research** → if crash occurs at step 4, resume from step 4, not from start

edquest

```python
from langgraph.checkpoint.memory import MemorySaver


# --- Use MemorySaver as checkpointer ---
checkpointer = MemorySaver()
app = app.compile(checkpointer=checkpointer)


# --- Run workflow ---
final_state = app.invoke({"results": []}, config={"configurable": {"thread_id": "demo"}})

print("Final State:", final_state)
```

HUMAN IN THE LOOP

edquest

Human in the Loop (HITL) **integrates human** oversight into AI workflows, allowing **humans to intervene, approve, or correct AI decisions**.

- Ensures safety, quality, and accountability
- Common in regulated or high-risk domains

Example:
AI drafts a contract → Human reviews → AI finalizes only after approval