1. Retrieval
2. Generation
   - Langgraph
3. API
4. UI

1. Scrapping
2. CSV
3. transformation
4. vector DB

ELT

5. Retrieval
6. Agentic RAG Flow
7. API
8. UI

9. CI/CD
10. Deploy

RAG

1. Data ingestion
2. Data Retrieval
3. Data Generation

FLIPKART ⟶ Datastax Astra DB ⟵ Retrieval ≈ Relevant Document

Generation → Prompt + question + Relevant Document

Advance Pool:
Agentic

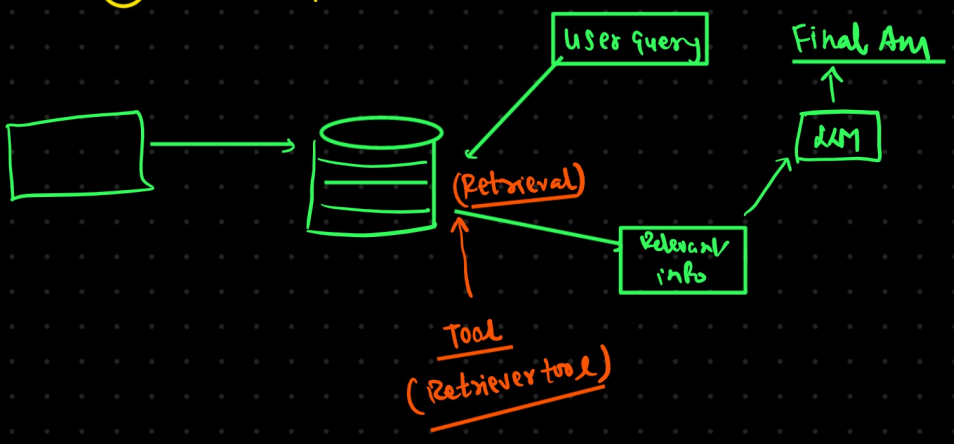
GenAI

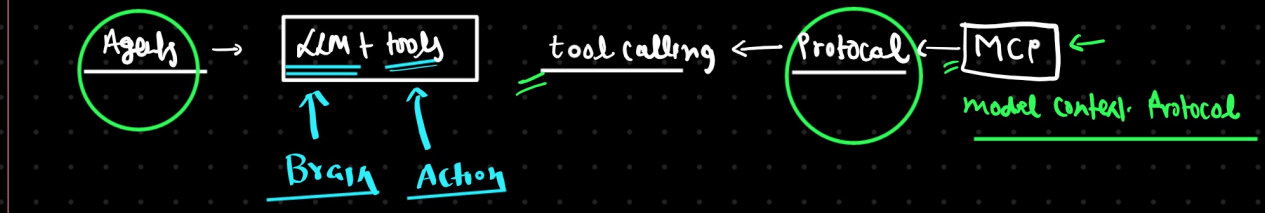
1. RAG
2. Agentic AI ⟵ AI Agent

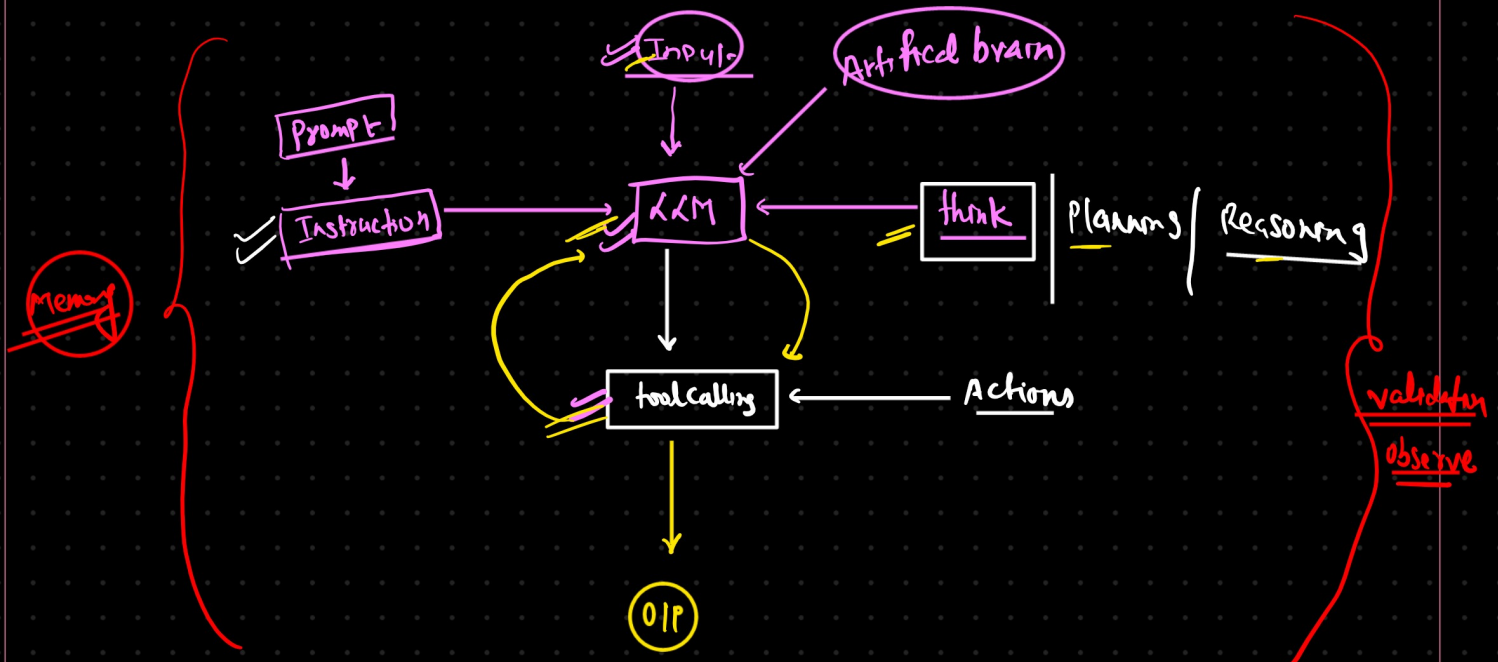
RAG

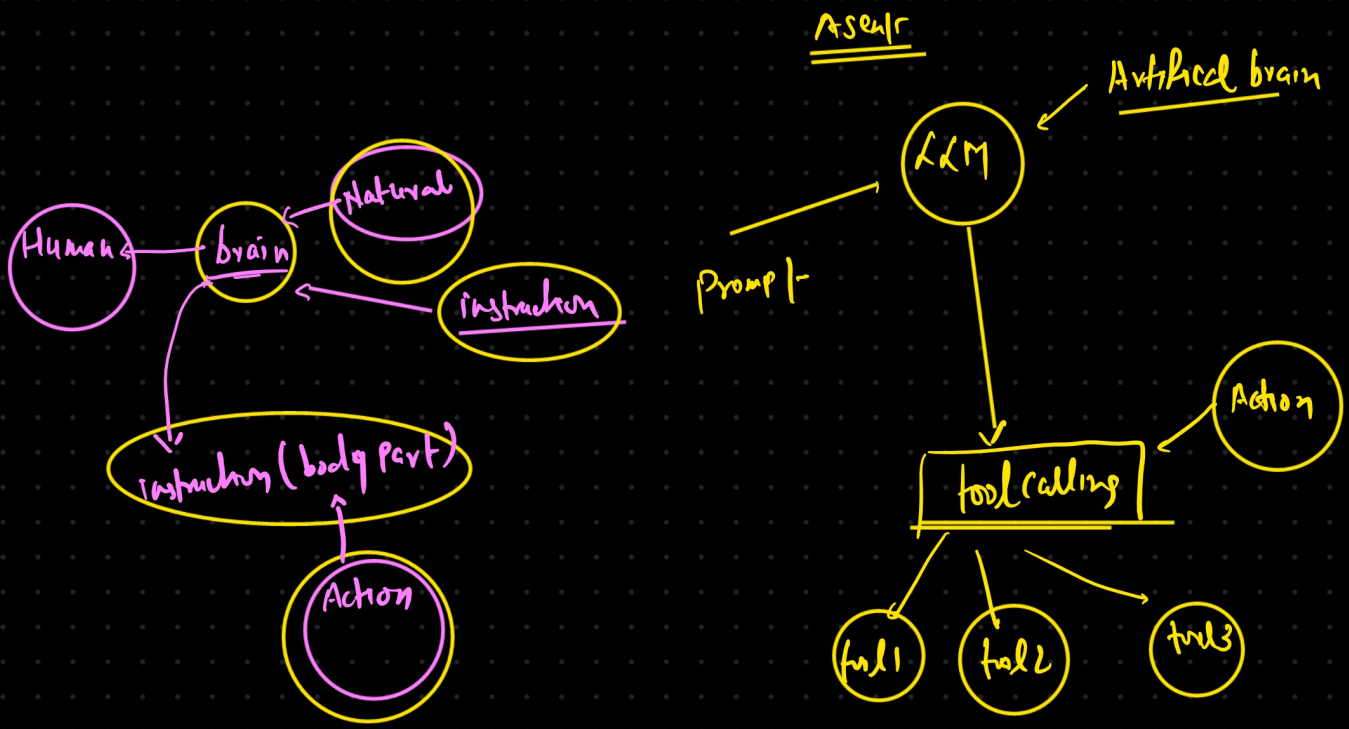
1. Data ingestion
2. Retrieval
3. Generation

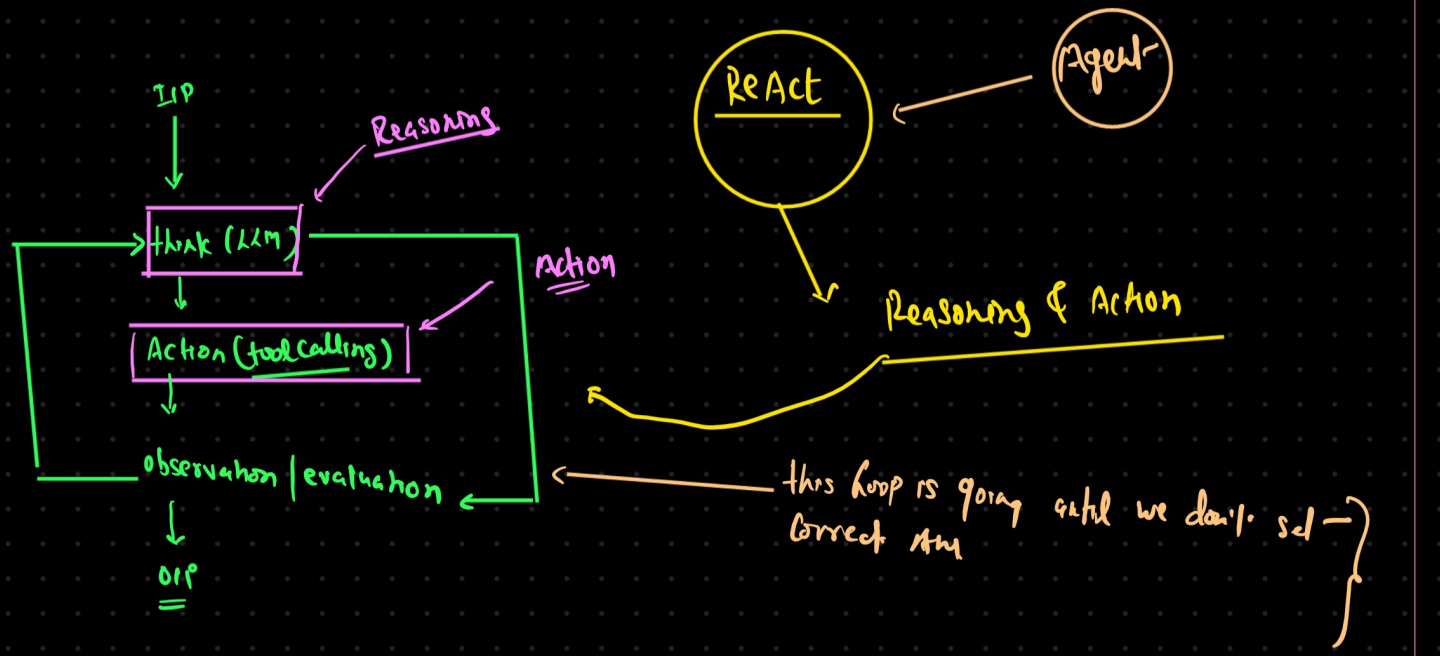
Agent

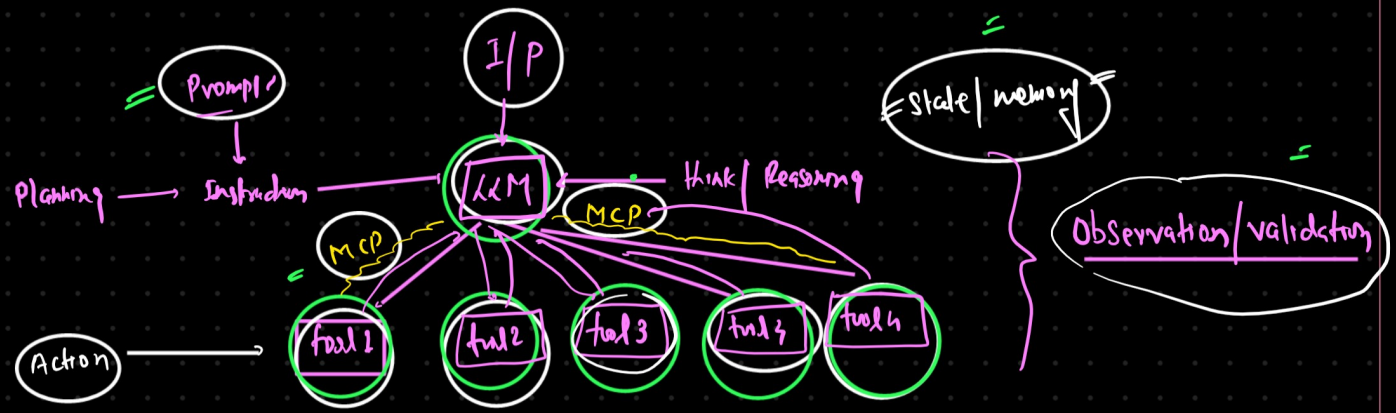


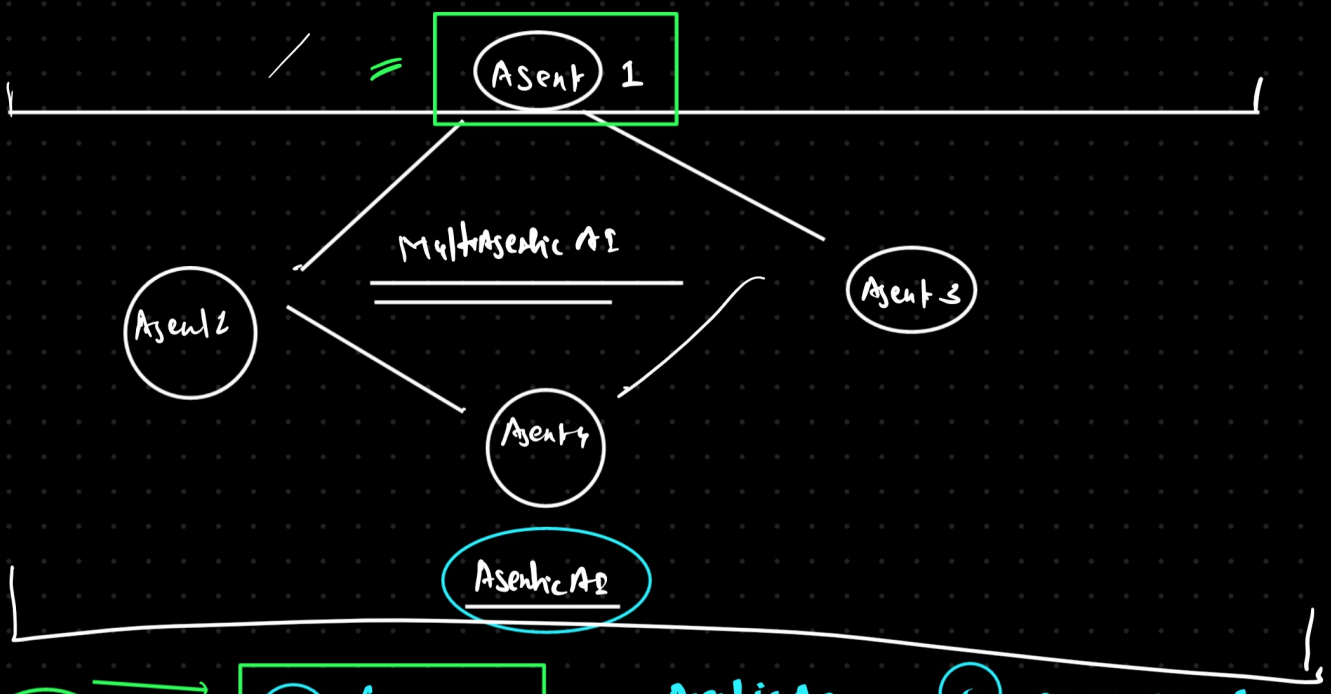

Agents
LLM + tools
Brain
Action
tool calling
Protocal
MCP
model context Protocal

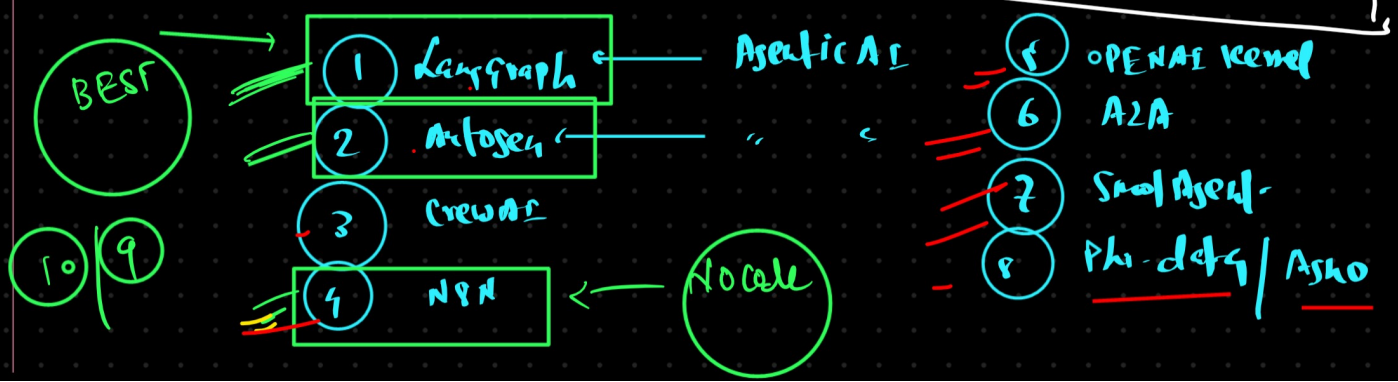

Input
Artificial brain
Prompt
Instruction
LLM
think
Planning
Reasoning
Memory
toolcalling
Actions
validation
observe
O/P

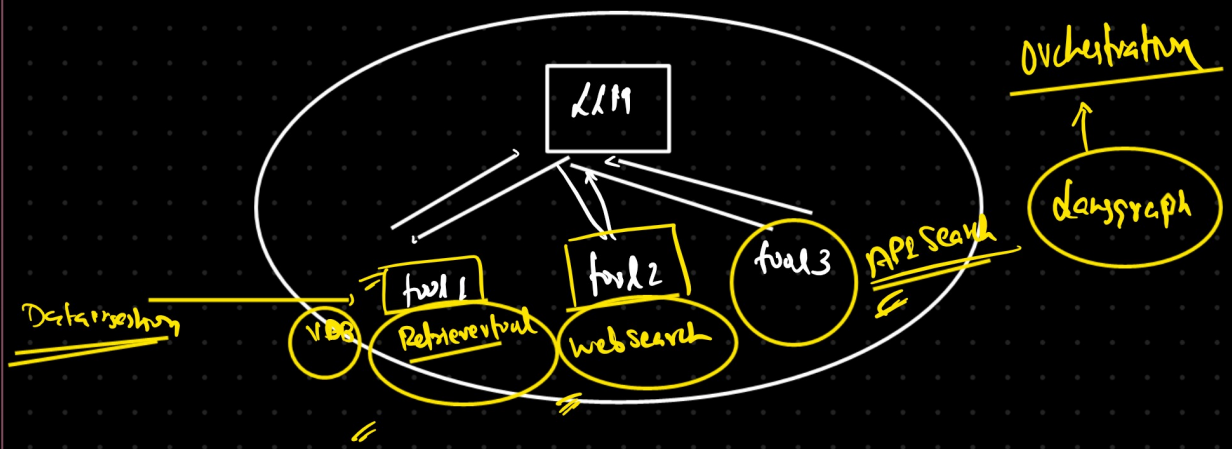

Agent
Artificial brain
LLM
Prompt
Natural
Human
brain
instruction
instruction (body part)
Action
toolcalling
Action
tool1
tool2
tool3

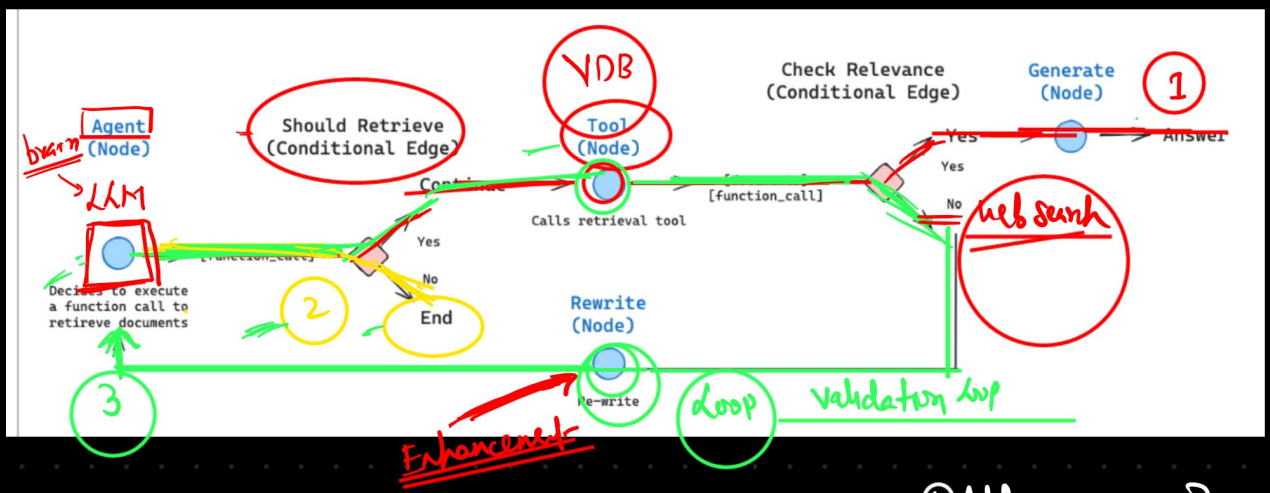

I/P
Reasoning
think (LLM)
Action
Action (tool calling)
observation / evaluation
O/P
ReAct
Agent
Reasoning & Action
this loop is going until we don't get correct result


I/P
Prompt
Planning
Instruction
LLM
Think / Reasoning
MCP
MCP
Action
tool 1
tool 2
tool 3
tool 4
tool 5
State / memory
Observation / validation


Agent 1
Multi Agentic AI
Agent 2
Agent 3
Agent 4
Agentic AI


BEST
1 Langgraph
2 Autogen
3 Crew AI
4 N8N
Agentic AI
Vocal
5 OPENAI Kernel
6 A2A
7 Small Agent
8 Phi-data / Agno
10
9





Langgraph ⟶ Functions ⟵ Nodes

Idea ← a=10, b=20 ① Add ② Factorial of it

Function 1 ← add ( )

Function 2 ← factorial ( )

node 1 (add) — State — (edge) (connection) (State) → node 2 (factorial) — State

info is flowing from node 1 → node 2 in the form of state

edge → ① normal add ② conditional add,

a, b

node 1 (add) → (Conditional) Connection if sum > 10 → node 2 (factorial)

Conditional edge

Problem statement.

1. addition
2. factorial.

Orchestration

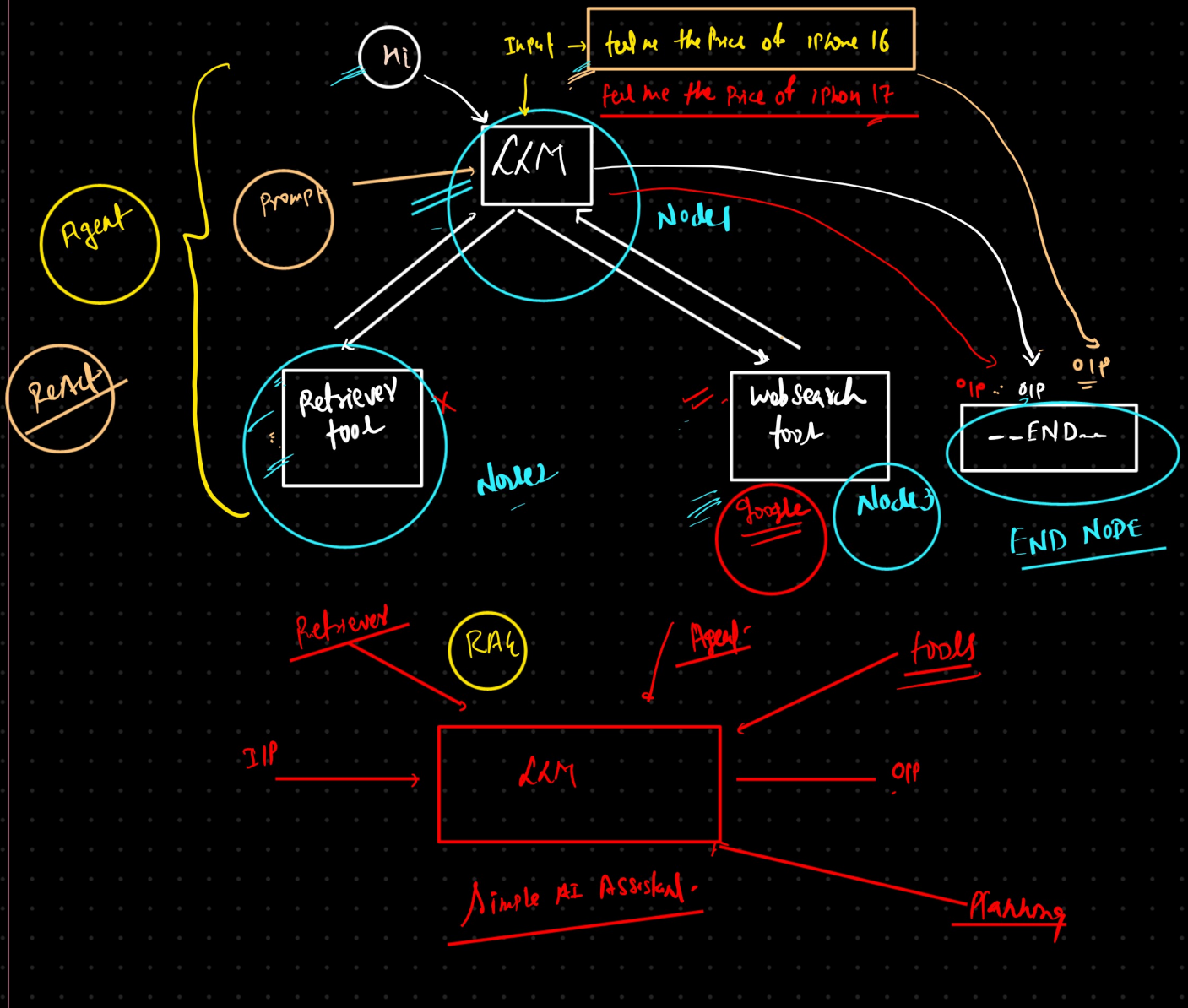
node 1

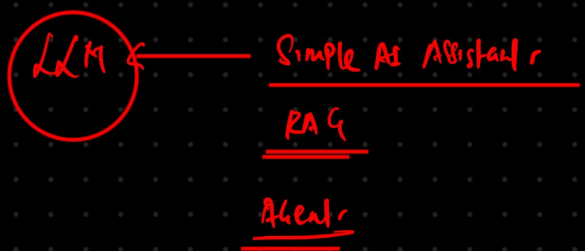
```
def add(a,b):
    c = a+b
    return c
```

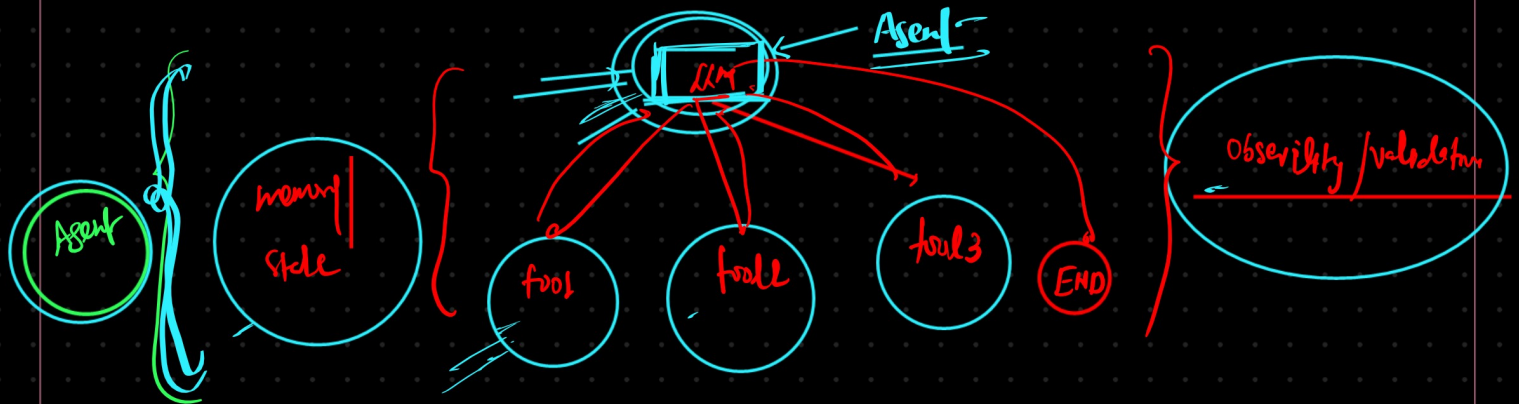
State (Pass this c to node 2)

node 2

```
def fact(n):
    o/p = n!
    print(o/p)
```



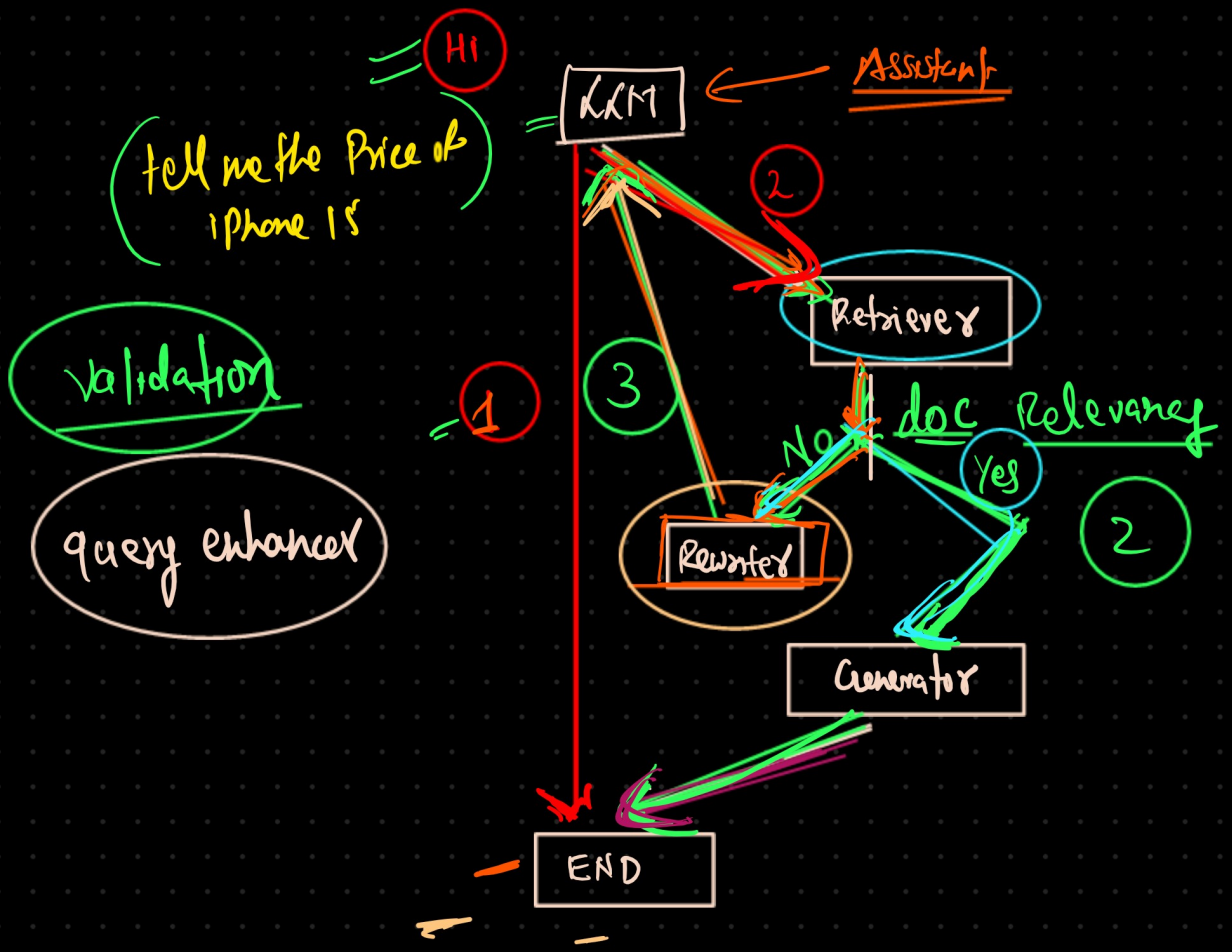




Assignment ← DuckDuckGo | tavily | GoogleSerper API |

In this flow you need to Add Websearch

Let's suppose if we are not able to Any Anything with retriever then will go for web search




```
workflow.add_edge("Rewriter", "Assistant")
```

```
workflow.add_conditional_edges(
    "Retriever",
    self._grade_documents,
    {"generator": "Generator",
     "rewriter": "Rewriter"},
)
```

Yes

NO

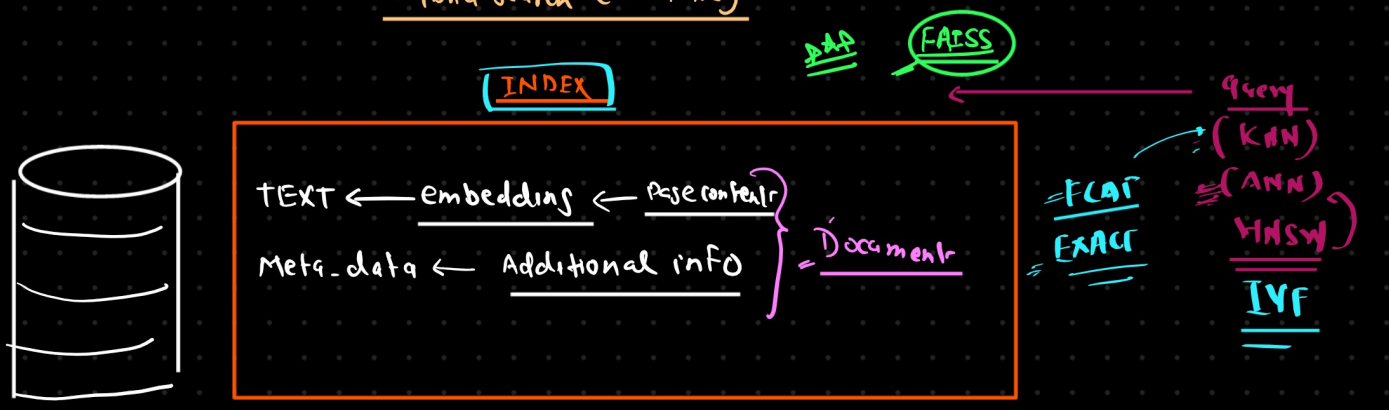
```
workflow.add_edge("Generator", END)
```

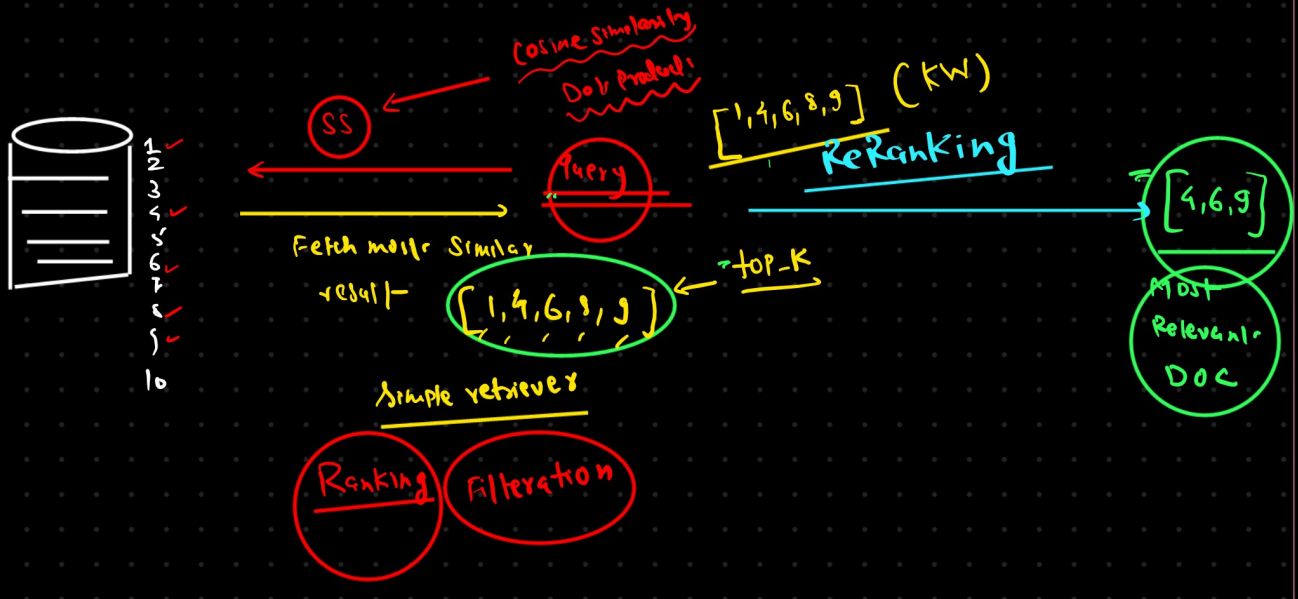
1. MMR
2. Contextual compression Retriever
3. RAGAs =
4. Memory =
5. Mcp server

AAM

1. Data ingestion ← Data Parsing
   = how to fetch the data and how to change the data

2. Data Retrieval ← Experimental
   - SS
   - MMR
   - hybrid search (kw + vec)

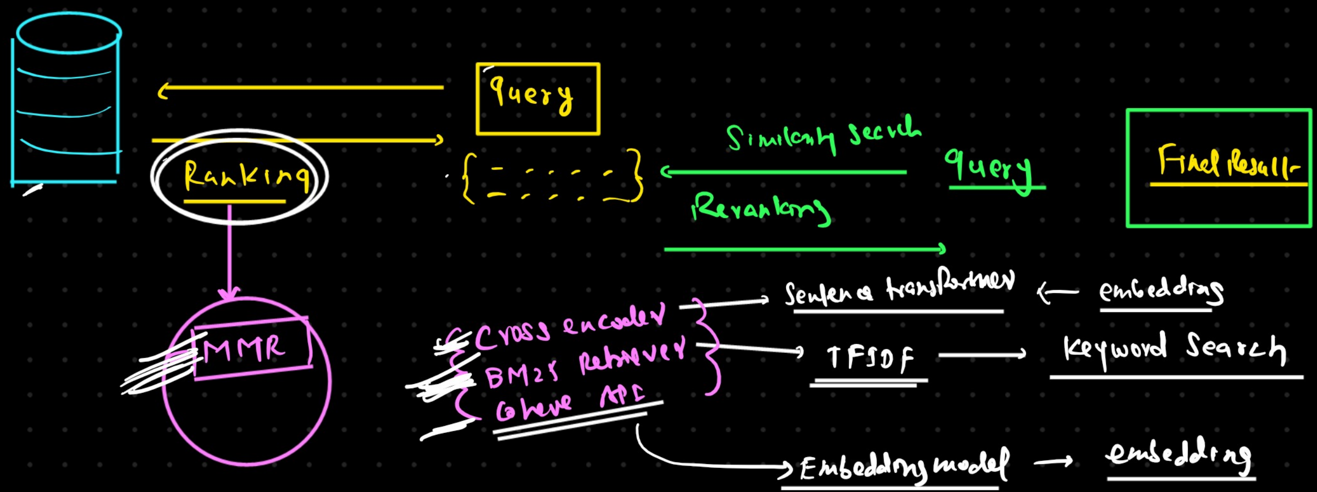




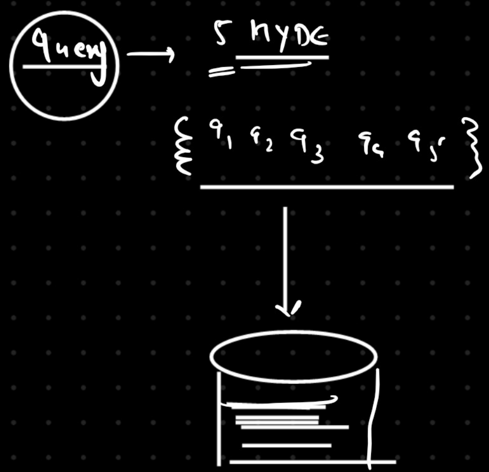
Simple retriever

Ranking | Filteration

MMR ← maximum marginal Relevance



1. HyDE -
2. window search -
3. Self query Retriever -
4. Rag fushion -
5. CCR -



2nd

1. execute the code end_to end
2. MCP server
3. Deployement ← Salrisan

explain MMR, Contextual Compression Retr, RAGAS

↑ mathematically (MMR)

↑ (Contextual Compression Retr)

↑ check with doc (RAGAS)

↑ use the appropriate metrics

3. AgenticAI ← MCP