Autonomous Cognitive Engine for Deep Research and Long-Horizon Tasks Enabling Long-Horizon Tasks with Memory, Planning, and Multi-Agent Collaboration

---

✅ Task 1 — Two Paragraphs Explaining the Project (Explore the Project)

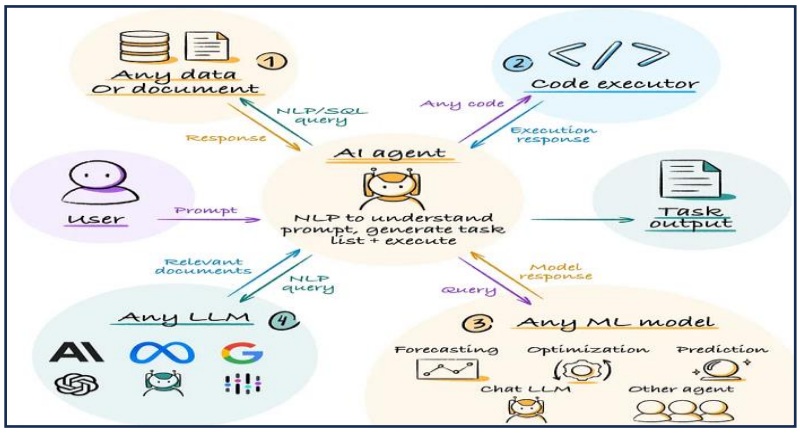
This project, **Autonomous Cognitive Engine for Deep Research and Long-Horizon Tasks**, aims to build an advanced AI agent using LangGraph that can handle complex tasks which require planning, memory management, and intelligent delegation. Unlike simple agents that respond in one step, this system behaves like a human researcher: it first plans the work by breaking a big goal into smaller TODO tasks, then executes each task step-by-step while storing intermediate results in a virtual file system. This solves the major limitation of LLMs — limited context window — by offloading memory externally. The agent continuously reasons about what to do next: whether to use tools (like web search), read/write from memory files, or delegate the task to a specialized sub-agent. All of this is managed through a stateful LangGraph workflow, where the agent's state (TODOs, files, results, messages) is preserved throughout the execution until the final output is generated.

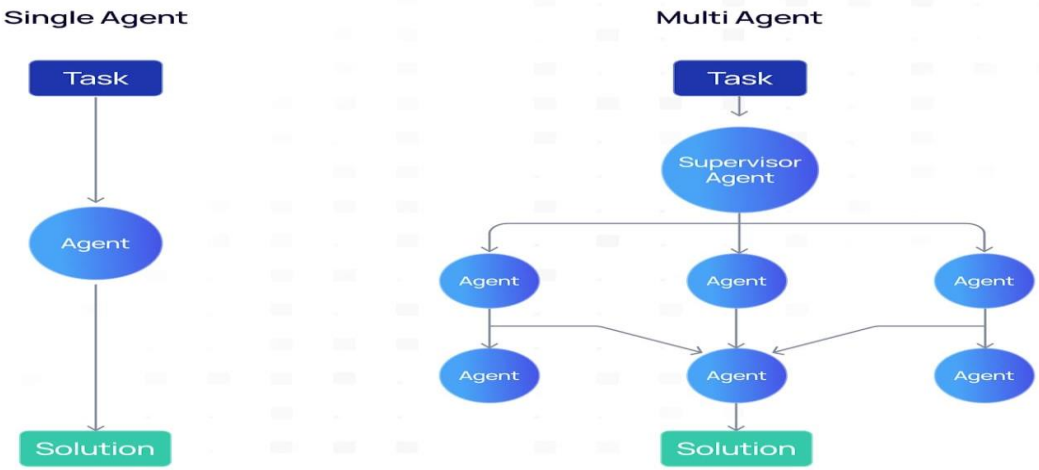
From start to end, the workflow begins when the user gives a complex request. The main agent analyzes the request and uses a planning tool (write_todos) to create a structured task list. Then, inside an execution loop, the agent selects each pending TODO, decides the best action, performs it (tool use, memory access, or delegation), observes the result, stores important information into the virtual file system, and marks the task as complete. If a task requires specialization, it is delegated to a sub-agent with its own focused context. After all TODOs are completed, the agent retrieves all stored notes from the file system, synthesizes them, and produces the final comprehensive output such as a research report or code solution. This architecture enables long-horizon task execution with planning, memory delegation, and modular intelligence.

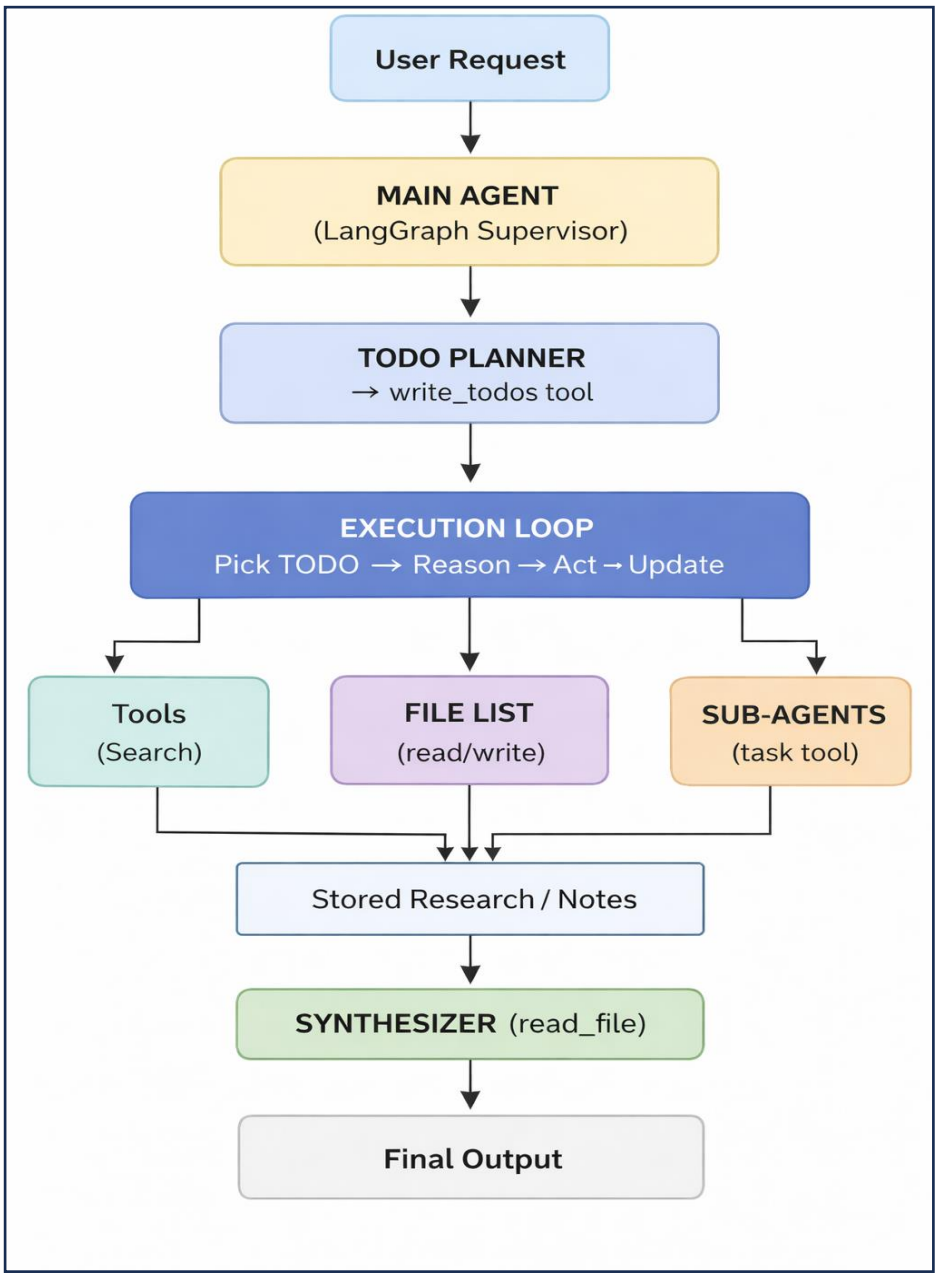
✅ Task 2 — Smart User Diagram (User Flow / Agent Diagram)

Below is the **clean user diagram** your guide wants (with Main Agent, TODO, File List, Sub-Agents, Final Output).

🧠 **User → Cognitive Agent Flow**




Single Agent
Task
Agent
Solution
Multi Agent
Task
Supervisor Agent
Agent
Agent
Agent
Agent
Agent
Agent
Solution


User Request
MAIN AGENT
(LangGraph Supervisor)
TODO PLANNER
→ write_todos tool
EXECUTION LOOP
Pick TODO → Reason → Act → Update
Tools
(Search)
FILE LIST
(read/write)
SUB-AGENTS
(task tool)
Stored Research / Notes
SYNTHESIZER (read_file)
Final Output

🔷 Stepwise Diagram Representation

🔷 1) User Request

Everything starts when the user gives a **complex request** (for example: “Generate a research report on cyberbullying in code-mixed Kannada–English chats”). This is not a one-step task, so the system cannot answer directly.

🧠 2) Main Agent (LangGraph Supervisor)

The **Main Agent** is the supervisor built using **LangGraph**.
It controls the whole process and maintains the **state** (TODOs, files, results, messages).

Its job is to:

- Understand the request
- Decide how to break it down
- Control which tool or sub-agent to use
- Track progress until completion

📝 3) TODO Planner → write_todos Tool

The agent does **planning before acting**.

Using the write_todos tool, it converts the big request into a **structured task list** like:

- Search articles
- Summarize findings
- Compare results
- Draft report
- Review and finalize

This TODO list is stored in the agent’s state.

🔁 4) Execution Loop (Core Thinking Cycle)

This is the most important part of the system.

For **each pending TODO**, the agent repeats:

Pick TODO → Reason → Act → Update

- **Pick TODO** → Select next unfinished task
- **Reason** → Decide what is needed (search? memory? sub-agent?)
- **Act** → Perform the action
- **Update** → Save result, mark task complete

This loop continues **until all TODOs are done**.

🧰 5) Three Possible Actions Inside the Loop

When acting, the agent has **three choices**:

🔷 Tools (Search)

Uses external tools like web search to collect information.

🔷 File List (read/write)

This is the **virtual memory**.

- Saves research notes
- Stores summaries
- Retrieves previous results

  This solves the LLM context limit.

🔷 Sub-Agents (task tool)

If a task needs specialization, the main agent delegates to a **sub-agent** (like a Summarizer or Search agent).

The sub-agent does the work and returns the result.

🗂️ 6) Stored Research / Notes

All outputs from tools and sub-agents are stored in the **file list** as research notes.
This becomes the knowledge base for the final step.

🩹 7) Synthesizer (read_file)

After all TODOs are completed, the agent:

- Reads all stored notes from files
- Combines them
- Organizes them into a meaningful structure

✅ 8) Final Output

Finally, the agent generates the **complete final output** (report / code / answer) based on everything it planned, executed, stored, and synthesized.

“The agent first plans using TODOs, then iteratively executes each task using tools, memory, or sub-agents, stores all intermediate results in a virtual file system, and finally synthesizes them into a complete output.”