Build a “Deep Research Agent for Memory”

1. Objective

Design an autonomous research agent that, given **only**:

- `user_id` – the identifier of the requesting user
- `prompt` – the user’s natural-language question

...automatically:

1. **Retrieves context**
    - Pulls the most relevant memories for that `user_id` from the long-term Memory store (Mem0).
    - Fetches past conversation messages that may help answer the prompt.
2. **Reasons over that context** to produce a grounded, well-structured answer.

---

2. Functional Requirements

Area	Requirement
Agent Entry-Point	A single function or CLI command accepting `user_id` and `prompt`.
Context Retrieval	Efficiently identify and rank relevant memories & message snippets (vector similarity, keyword heuristics, or both).
Answer Generation	Compose a final answer that:• Is factually aligned with the retrieved context.• Includes inline citations or footnotes pointing to memory IDs / message timestamps.• Provides a brief high-level rationale (chain-of-thought summary).
Memory Writing (bonus)	If new durable facts emerge, append them to the Memory store in the correct schema.
Model Choice	Any LLM is allowed. Briefly justify your selection in the README.

Local Demo	Provide an interactive way (CLI or simple web UI) to run the agent with sample data.

3. Deliverables

1. **GitHub repository**
   - Clean, well-commented code.
   - `README.md` with setup & usage (< 5 min copy-paste experience).
   - Unit/integration tests covering retrieval and generation.
2. **Video walk-through (5–10 min)**
   - Show installation, example queries, and how answers cite memories/messages.
3. **(Optional) Technical write-up**
   - Trade-offs, future improvements, what you'd tackle with more time.

4. Timeline

You have **3 calendar days** from the moment you receive this document. State your start and due dates in the PR or email.

5. Evaluation Rubric

Weight	Criterion
40 %	**Accuracy & grounding** — Answers correctly use relevant memories/messages and avoid hallucinations.
25 %	**Retrieval quality** — Approach reliably surfaces the right context, even as data scales.
15 %	**Code quality** — Readability, modularity, tests, and performance.
10 %	**Developer experience** — Smooth setup, clear documentation, easy to extend.
10 %	**Communication** — Video clarity, explanation of decisions, professionalism.

(Bonus points for robust memory-writing and thoughtful UX touches.)

---

6. Submission

- Push code to a public or private GitHub repo (invite **@deshraj** if private).
- Include the demo-video link in your `README`.