AI / Machine Learning Engineer – Technical Assignment

Estimated time: up to 2 working days

This assignment is designed to evaluate your ability to design, reason about, and implement an LLM-powered system end-to-end, with a strong focus on engineering quality, trade-offs, and clarity of thought.

We value pragmatism and reasoning over feature completeness.

Part 1 – Verbal Questions (Live)

Please answer verbally during the interview.

Core ML / AI Concepts

1. Difference between supervised, unsupervised, and reinforcement learning
2. What is self-supervised learning? Which category is it closer to?
3. What are regularization and dropout?
4. What is PCA?
5. Explain the bias–variance tradeoff
6. Relationship between:
    - model complexity
    - training accuracy
    - generalization on unseen data

LLM / Agent Systems

1. Are there LLMs designed for agent-based use? Provide examples.
2. How would you run LLMs locally? Mention tools and their use cases.
3. If you cannot use embedding APIs (OpenAI, Google, etc.), what alternatives exist?
4. What tools or frameworks can be used to build chatbots and agent-based systems?

Part 2 – Coding Assignment

Goal

Build a ReAct-style AI agent capable of:

- generating Python code for data analysis and visualization
- executing the code in a controlled sandbox (E2B)
- producing Plotly-based HTML visualizations

Phase A – Core System (Required)

A correct and well-structured Phase A is sufficient to pass.

Functional Requirements

1. ReAct-style agent loop (plan → act → observe → iterate)
2. Python code generation and execution

3. Data analysis and visualization support
4. Plot generation using Plotly
5. Output of plots as HTML files
6. Logging and error handling

Sandbox Requirements (Minimum)

The sandbox must:

- execute code in a separate process
- enforce an execution timeout
- restrict filesystem access to a temporary working directory
- prevent external network access

Full OS-level isolation (e.g., seccomp, VM-level sandboxing) is not required.

Phase B – Advanced Features (Optional / Bonus)

Implement one or more of the following.
Depth and quality matter more than quantity.

Option 1 – Chat Memory & Persistence

- Design a Chat Memory Port
- Implement a Neo4j adapter to store:
  - conversations
  - messages (role, content, timestamp, metadata)
- Support basic retrieval (e.g. resume conversation, retrieve past plots)

Option 2 – Agent Robustness

- Retry / recovery strategies
- Handling invalid code generation
- Graceful failure modes and fallback behaviors

Technical Stack

Required

- Python
- Plotly
- ReAct-style agent loop
- Docker or docker-compose setup
- Reproducible local execution

Recommended (not mandatory)

- LangGraph (or equivalent)
- Neo4j (for memory adapter)
- Docker / docker-compose

Equivalent tools are acceptable if clearly motivated.

Architecture Guidelines

If architecture is implemented, we expect:

- Ports layer – abstract interfaces
- Adapters layer – concrete implementations
- Use cases layer – application logic
- Clear dependency direction (inward)

Avoid over-engineering.

Evaluation Criteria

We evaluate how you think, not just what you build.

Area	What we look for
Agent design	Clear ReAct loop, reasoning steps
Code execution	Safe execution strategy, error handling
Architecture	Clear boundaries, testability
Trade-offs	Conscious scoping and decisions
Code quality	Readability, structure, naming
Robustness	Logging, failure handling
Bonus depth	Quality of optional features

Deliverables

1. Source code (Zip file)
2. README (mandatory) describing:
   - system architecture
   - key design decisions
   - known limitations
   - what you would improve with more time
3. Example HTML plot output

Notes

- The system does not need to infer the “best” chart type automatically.
- The agent must be able to generate and execute Plotly code when prompted.

- Partial implementations with clear reasoning are preferred over fragile completeness.

Final Note

This assignment is intentionally open-ended.
We are interested in engineering maturity, reasoning clarity, and decision-making.