Technical Assessment

NPC Dialogue System Using Model Context Protocol (MCP)

Objective:

You're tasked with building a dialogue engine for an NPC named **Adam**, providing useful suggestions in a Role Playing Game (RPG). Adam interacts with the player using GPT and remembers past interactions intelligently. Due to LLM token limits, you must build a **custom Model Context Protocol (MCP)** to manage context effectively.

This test focuses on:

- A **custom MCP server** exposing context-aware operations.
- An **MCP client** that generates dialogue with Adam.
- **LangGraph** for orchestration and routing.
- **Tool use** via a public API (e.g., Wikipedia search) to demonstrate tool calls.

Objectives

1. MCP Server

- Accept TCP/HTTP requests.
- Maintain conversation history and memory.
- Expose:
  - ADD_MESSAGE {role, content} – Append message.
  - GET_CONTEXT – Return summarized + recent messages within token limit.
  - SUMMARIZE_HISTORY – Condense old dialogue.
  - RESET – Clear conversation.
  - TOOL_CALL <query> – Search Wikipedia or any public data source and return top result (can use requests + parsing).

2. MCP Client with LangGraph Routing

- Client accepts user input and routes requests using a **LangGraph workflow**, which includes:
  - InputNode: Receives user message.
  - CheckContextNode: Calls MCP to get summarized context.
  - ToolDecisionNode: If user asks a factual question, route to TOOL_CALL node.

    - PromptAssemblyNode: Builds the full GPT prompt (persona + summary + recent turns).
    - LLMNode: Sends prompt to GPT and receives response.
    - OutputNode: Logs final output and updates server with new messages.
- Routing should be conditional (e.g., detect factual questions using keywords or a classifier and call external tools).

3. Token & Context Management

- Use tiktoken (or similar) to estimate tokens.
- Server must ensure total prompt fits within 4K tokens.
- MCP should summarize old messages or drop low-importance turns.

4. Dialogue Example & Persona

- Use a fixed persona, for example, *"Adam is a wise, centuries-old sage of the northern isles who guides with empathy and lore."*
- Provide at least **5 turns of conversation**.
- Include at least one factual lookup from a tool call (e.g., player asks "What are the different genres in gaming?").

Evaluation Criteria

- Correctness of MCP protocol & server implementation.
- Proper LangGraph workflow design and conditional routing.
- Context/token management and summarization logic.
- Use of tool calls and dynamic prompt building.
- Code structure, readability, and prompt quality.
- Bonus: extensibility for multiple NPCs.

Submission

- GitHub repo or zip with:
    - mcp_server.py
    - mcp_client.py
    - README.md (setup + sample dialogue run + persona definition)
- Include transcript of 5+ turns with at least one tool use.

- Include any helper scripts or API instructions.

Tips

- Use LangGraph for modular orchestration.
- For tool calls, you may use Wikipedia API via requests.
- Focus on logic and memory handling. Do not focus on building a GUI.

Please contact Karanjot Singh (karanjot.singh@razer.com) for any issues regarding the assignment.

Good luck and let’s bring Adam to life!