Interview Case - Data Fly Wheel Chatbot

Chatbot Challenge: Building an Adaptive AI Assistant

At Noxus, we're dedicated to pushing the boundaries of AI-powered automation. A key component of this vision is creating intelligent chatbots that continuously learn and improve from user interactions. Your mission in this challenge is to design and implement a chatbot platform that not only provides helpful responses but also actively learns from user feedback to enhance its performance.

The objective is to develop a chatbot platform capable of engaging in meaningful conversations, leveraging external knowledge sources, and, most importantly, adapting its behavior based on user feedback. The solution should include backend services for processing conversations and managing knowledge, a frontend interface for user interaction, and a robust mechanism for incorporating feedback into the chatbot's learning process. Containerization should be used for orchestration.

Core Requirements

- **Conversational Core:** The chatbot must be able to engage in multi-turn conversations, maintaining context and providing relevant responses.
- **Knowledge Integration:** The chatbot should be able to access and utilize external knowledge sources (e.g., a knowledge base, web search) to enhance its responses.
- **User Feedback Mechanism:** Implement a system for users to provide feedback on the chatbot's responses (e.g., thumbs up/down, free-form text).
- **CRUD Operations:** The system must have a set of CRUD operations for managing chatbot configurations, knowledge sources, and feedback data.
- **Dynamic Configuration:** The chatbot's configuration (e.g., system prompts, tool configurations) must be dynamic and serializable (e.g., JSON).
- **Persistence:** The system must store chatbot configurations, knowledge sources, conversation history, and feedback data in a database.
- **Technology Stack:** Use Python in the backend, Javascript/Typescript in the frontend, and Docker + Docker Compose for orchestration.
- **Documentation & Testing:** Include properly documented code and unit tests.
- **README:** Write a README detailing your thought process, decisions, and instructions for running the project.

Further Considerations (Advanced Topics)

If you meet the core criteria and want to expand on your work, consider exploring these advanced topics:

- **A/B Testing:** Implement A/B testing to compare different chatbot configurations and identify the most effective strategies.
- **Internal Performance Analytics:** Provide internal dashboards and reports to investigate chatbot performance, identify areas for improvement, and track the impact of changes over time.
- **Streaming:** Implement streaming of the chatbot's responses to the frontend for a more responsive user experience.
- **Advanced Rendered Data Types:** Support the rendering of complex data types in the chatbot's responses, such as document

references, images, or interactive elements.

- **Reasoning Support:** Integrate reasoning capabilities into the chatbot to enable it to answer more complex questions and solve problems that require logical inference.
- **Deep Research Support:** Enhance the chatbot's ability to conduct in-depth research on specific topics, synthesizing information from multiple sources to provide comprehensive answers.
- **Adaptive Learning:** The platform must incorporate a mechanism for using user feedback to improve the chatbot's performance. This could involve:
    - **Prompt Optimization:** Automatically adjusting prompts to elicit better responses.
    - **System Prompt Refinement:** Refining the chatbot's core instructions based on feedback.
    - **Tool Usage Improvement:** Optimizing how the chatbot utilizes external tools.
- **Evaluation Framework:** Implement an evaluation framework to automatically assess the quality of the chatbot's responses based on predefined metrics (e.g., relevance, accuracy, coherence).