Sarvam AI Agent Challenge: Use Case Template

1. Problem Statement:

Booking a restaurant reservation often involves navigating various websites, checking availability, and coordinating manually with restaurants. This can be time-consuming and inconvenient, especially when users simply want a quick and efficient way to reserve a table for a specific cuisine, date, and time.

2. User Persona:

Name: Kunal Mahadule
Age: 21
Profession: AI Engineer
Pain Points:
- Doesn't have time to call restaurants individually.
- Wants a fast, automated way to check availability and book.
- Prefers conversational interfaces like chatbots over browsing apps.

3. Use Case Title:

AI Agent for Restaurant Reservation

4. Description of the Use Case:

An AI agent that interacts with users to gather details like date, time, location, cuisine type, and number of people, then books a restaurant reservation using a function-calling model. It can either call a tool internally or interact with an external booking API.

5. Workflow of the Use Case (Step-by-step):

- 1. User opens the Streamlit frontend and enters a restaurant reservation query.
- 2. Input is passed to the LLM via the backend (using OpenRouter and a tool-calling model).
- 3. The model detects if a function call is needed and generates a tool-use request.
- 4. The function simulates or logs the booking with given parameters (cuisine, date, time, people).
- 5. The response is shown back to the user on the UI.
- 6. If something is missing (e.g., time), the AI asks follow-up questions.

6. Tool/LLM Used:

Model: meta-llama/llama-3-70b-instruct (via OpenRouter)
LLM Provider: DeepInfra
Tool-Calling: Custom tool integrated in backend to simulate booking process.

7. Deployment Plan / Running Instructions:

1. Clone the repository locally.
2. Create a `.env` file with your OpenRouter API Key.
3. Update the `model` value in `agent.py` with a valid model name.
4. Run `streamlit run frontend/app.py`.
5. Interact with the chatbot UI to test restaurant booking.

8. Challenges You Faced:

- Model ID errors from OpenRouter requiring correct provider/model match.
- Handling tool-calling schema properly.
- Ensuring proper .env file loading in Streamlit.
- Getting valid HTTP 200 responses from API.
- UI showing generic LLM responses instead of triggering tool use initially.

9. Screenshots:

Please add screenshots manually here of:
- Frontend UI
- CLI logs
- OpenRouter tool call responses.