Conversational AI Customer Service

Description

Build a simplified Python backend service for an AI agent that helps users with personal requests, such as:

- Tracking their orders
- Managing their profile
- Search Products

Requirements & Instructions

1. Backend API

- **Framework:** Any Python web framework (Flask, FastAPI, Django, etc.)
- **Endpoints:** You decide, but at minimum you should expose a single webhook or message-handling endpoint that accepts user messages and returns AI responses.

2. Function-Calling

- **Example functions:**
  - `get_my_orders(user_id: str) -> dict`
  - `update_profile(user_id: str, updates: dict) -> dict`
  - `get_order(order_id: str) -> dict`

3. Database (PostgreSQL/Supabase)

Define a schema that supports your features. Suggested tables:

- **users** (user_id, name, email, …)
- **conversations** (conv_id, user_id, timestamp, message, direction)
- **products** (product_id, name, price, …)
- **orders** (order_id, user_id, product_id, quantity, status, …)

Feel free to add or remove tables as needed.

4. Dockerization

- Provide a **docker-compose.yml** that brings up:
    1. Your API service
    2. The mock external-API service
    3. A PostgreSQL (or Supabase) database

5. AI Integration

- You may use any AI client or SDK for natural language understanding.

Deliverables

1. **Source Code**
    - Hosted in a GitHub repository or as a zipped archive.
2. **README.md**
    - Setup & run instructions (including environment variables)
    - API documentation (endpoints, request/response examples)
3. **Sample Requests/Responses**
    - cURL or Postman snippets for each endpoint

Evaluation Criteria

- **Code Quality:** Readability, best practices, modularity
- **API Design:** RESTful conventions, input validation, error handling
- **Function-Calling:** Accurate intent detection & invocation
- **External Integration:** Robust mock client with API-key auth and retries
- **Database Design:** Clear schema and efficient queries
- **Docker Setup:** Containers build and run smoothly
- **Documentation:** Clarity, completeness, and ease of onboarding
- **Problem Solving:** Thoughtful handling of edge cases and extensibility

Good luck! Let me know if anything needs clarification.