Systango Interview Coding Assessment

TASK: Multi-Agent Ordering Chatbot

Objective

Build a multi-agent chatbot system to streamline user order requests. The chatbot should interact with users via a simple UI (Streamlit or similar), classify the type of order (bulk or generic), collect the required details, and store interactions and final data in a database.

🛠️ System Components

1. Main Agent

- **Role**: Ask description of what user wants to order, then directs the user to the correct respective agent.
- **Asks**:
  - Description of what user wants to buy.
  - Is this request for personal use or for reselling.
- **Actions**:
  - Uses `category_finder_tool(description, type_of_request)` to classify the request.
  - Routes the request to either Generic Order Agent or Bulk Order Agent.
  - Logs interactions.

---

2. Generic Order Agent

- **Role**: Gathers information for general/single orders.
- **Asks**:
  - Product name.
  - Quantity required.
  - Brand preference (if any).
- **Actions**:
  - Validates and confirms details.

- Allows detail updates.
- Supports agent handoffs.
- Logs interactions.

---

3. Bulk Order Agent

- **Role**: Gathers information for bulk/mass orders.
- **Asks**:
  - Type of products being ordered in bulk.
  - Total quantity or units needed.
  - Any supplier preference or constraints.
- **Actions**:
  - Validates and confirms details.
  - Allows detail updates.
  - Supports agent handoffs.
  - Logs interactions.

---

4. Category Finder Tool

A Python function that classifies a request as "generic" or "bulk":

```
def category_finder_tool(description: str = None, type_of_request: str =
None) -> str:
    # Use LLM to analyze title and description
    return "generic"  # or "bulk"
```

---

5. User Interface

- **Framework**: Streamlit (preferred), Gradio, or simple HTML.
- **Features**:
  - Chatbox for user inputs and responses.
  - Clear transitions between agents (e.g., "Handing off to Bulk Order Agent...").
  - Auto-reset session on refresh.

- Optionally display final order in JSON/table view.

---

6. Database

- **Use**: SQLite (or PostgreSQL or MySQL or CSV File).
- **Tables**:

NOTE: You can modify table schema as per as your need

`conversation`

Column	Type	Description
id	Integer (PK)	Unique ID
session_id	String	Unique chat session
timestamp	Datetime	When message occurred
user_input	Text	User's message
chatbot_response	Text	Chatbot's reply
agent	String	Agent name ("Orchestrator", "Generic" or "Bulk")

`order`

Column	Type	Description
id	Integer (PK)	Unique ID
session_id	String	Links to conversation
title	Text	Order title
description	Text	Order description
product_name	Text	Product name
quantity	Integer	Requested quantity
brand_preference	Text	Brand preference
additional_details	Jsonb	Any additional information provdided by user
created_at	Datetime	Submission timestamp

✅ Final Output Format (JSON)

```
{
  "title": "<order_title>",
  "description": "<order_description>",
  "product_name": "<product_name>",
  "quantity": <quantity>,
  "brand_preference": "<brand_preference>"
}
```

Example conversation

Below are the example conversation of bot and user, this is expected flow from this coding task. Each individual agent should be able to hand off task to any other agent.

🧪 Example-1 Conversation

```
[Running Orchestrator Agent]

User: I need 10 wooden desks.
Bot: Please provide a title for this order.
User: Wooden Desk Order
Bot: Describe the request.
User: 10 wooden desks for the new conference room.

[Orchestrator Agent calls category_finder_tool]
[category_finder_tool tool's output -> generic_order_agent]
[Orchestrator Agent hands off to Generic Order Agent]

Bot: Confirming order for 10 wooden desks. Any brand or vendor preference?
User: Yes, from UrbanCraft.
Bot: Here’s the summary...
[Summary shown, details saved to database]
```

🧪 Example-2 Conversation

```
[Running Orchestrator Agent]
```

```
User: I need 500 water bottles.
Bot: Please provide a title for this order.
User: Bottle Order
Bot: Describe the request.
User: Need 500 reusable water bottles for our marathon event.

[Orchestrator Agent calls category_finder_tool]
[category_finder_tool tool's output -> bulk_order_agent]
[Orchestrator Agent hands off to Bulk Order Agent]

Bot: Confirming bulk order of 500 reusable water bottles. Any supplier
preference?
User: Yes, prefer BottlePro or any eco-friendly vendor.
Bot: Here’s the summary...
[Summary shown, details saved to database]
```

🧪 Example-3 Conversation

```
[Running Orchestrator Agent]

User: I need 300 USB drives.
Bot: Please provide a title for this order.
User: USB Drive Bulk Order
Bot: Describe the request.
User: 300 USB drives for employee onboarding kits.

[Orchestrator Agent calls category_finder_tool]
[category_finder_tool tool's output -> bulk_order_agent]
[Orchestrator Agent hands off to Bulk Order Agent]

Bot: Confirming bulk order of 300 USB drives. Any supplier preference?
User: No, actually I need 20 desk lamps instead.
[Bulk Order Agent hands off to Orchestrator Agent]

Bot: Please provide a title for desk lamp order.
User: Desk Lamp Order

[Orchestrator Agent calls category_finder_tool]
[category_finder_tool tool's output -> generic_order_agent]
[Orchestrator Agent hands off to Generic Order Agent]
```

```
Bot: Confirming order for 20 desk lamps. Any brand or vendor preference?
User: No
Bot: Here's the summary...
[Summary shown, details saved to database]
```

📋 Requirements Summary

Feature	Required
Multi-agent logic	✅
UI (Streamlit/Gradio)	✅
Chat history with session management	✅
Input validation (e.g., quantity > 0)	✅
Handoff between agents	✅
Database storage using any: SQLITE / CSV / MYSQL / POSTGRES	✅
Final output structure	✅

🧰 Tech Stack

- **Language**: Python
- **UI**: Streamlit (preferred), Gradio, HTML
- **Database**: SQLite, PostgreSQL (preferred), MySQL, or simple CSV files
- **Agent Frameworks**: OpenAI SDK, LangChain, LangGraph, Pydantic AI, (anything else if you know) etc.
- **LLMs**: Google AI Studio, Groq, OpenAI

You can get free API keys from-

- https://aistudio.google.com
- https://grok.com
- Use any open-source model from Ollama or vLLM or Transformers (https://ollama.com)

🧠 Evaluation Criteria

- Correctness of chatbot workflow
- Smooth agent handoffs & classification
- Proper logging & session handling
- Clean, readable code
- UI usability and completeness
- Input validation and error handling