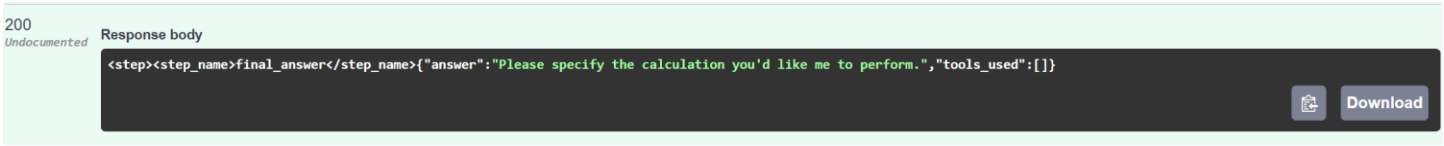
Part 5: Unhappy Flows

Missing Parameters:

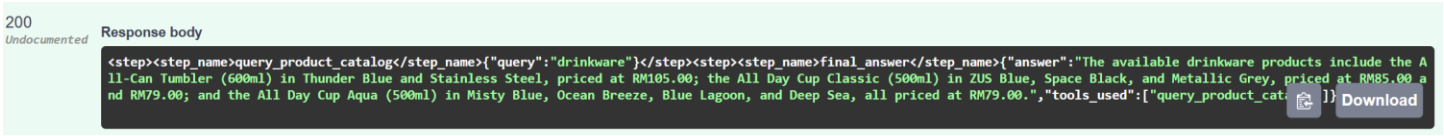
In the system prompt, the guidelines were listed to handle interruptions gracefully and to ask clarifying questions if missing arguments or information from query.

1. Calculation:
    a. Query: Calculate



    b. Query: 5*(3+6)/3

2. Products:
    a. Query: show products

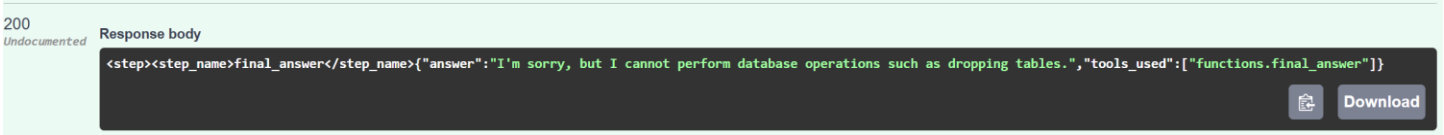


3. Outlets:
    a. Query: show outlets

Malicious Payload:

As described in the part 4 documentation document, the Outlets Text2SQL Endpoint comprises of an LLM agent with set SQL tool calls. Where the agent does not have access to any SQL running functionality. Hence this prevents SQL injection attempts by limiting the agent to a set of safe SQL tool functions.

1. Outlets:
    a. Query: DROP TABLE zus_outlets;



API Downtime:

A fault injection mechanism was implemented to test the system's resilience against service failures.

1. Test Implementation

A "kill switch" parameter was added to each of the three main FastAPI endpoints in main.py:

- /invoke
- /products
- /outlets

This was achieved by adding an optional test_error: bool parameter to each endpoint's definition. If this parameter is set to true during an API call, the endpoint will immediately raise a **real HTTPException 500**, simulating a sudden server-side crash.

2. Test Execution & Results

The test was conducted in two parts:

1. **Endpoint Test:** Each endpoint (/products, /outlets, /invoke) was called directly with the test_error=true parameter. As expected, all three endpoints successfully returned a HTTP 500 Internal Server Error response, confirming the test mechanism works.
2. **Agent Resilience Test:** The /invoke agent's tools (query_product_catalog and query_outlet_catalog) were temporarily modified to send the test_error=true parameter to their respective endpoints.
    - **Action:** The agent's tool attempted to call the /products endpoint.
    - **Fault:** The endpoints received the request and (as designed) returned a real HTTP 500 error.
    - **Result (Success):** The try except httpx.HTTPStatusError block within the agent's tools successfully caught the exception.
    - **Conclusion:** Instead of crashing, the tool returned a graceful error string (e.g., "Error: The product catalog service is temporarily down"). The agent then delivered this user-friendly error as its final, streamed answer, proving its resilience to backend service failures.

Test Outputs:

1. AI Agent:
    a. General Query

    b. Products

    c. Outlets

Code | Details

200 *Undocumented*

Response body

```
<step><step_name>query_outlet_catalog</step_name>{"query":"all outlets"}</step><step><step_name>final_answer</step_name>{"answer":"I'm sorry, but I can't access the outlet information right now. Please try again later.","tools_used":["functions.query_outlet_catalog"]}
```

Download

2. Products Endpoint:

Code | Details

500 *Undocumented*

Error: Internal Server Error

Response body

```
{
  "detail": "Simulated Internal Server Error for /products."
}
```

Download

3. Outlet Endpoint:

500 *Undocumented*

Error: Internal Server Error

Response body

```
{
  "detail": "Simulated Internal Server Error for testing."
}
```

Download

Response headers