

AI-Powered Ecommerce Support Assistant

Technical Design & Architecture

Project Documentation & Evaluation

Ecommerce Support Assistant - Technical Documentation

1. Overview

This document describes the design, architecture, and evaluation of an AI-powered Ecommerce Support Assistant. The system combines Retrieval-Augmented Generation (RAG), an agentic workflow with intent-based routing, deterministic tools (orders, returns, refunds, warranty, products, troubleshooting), guardrails for safety, and in-memory caching for performance.

The assistant acts as a smart support layer for an ecommerce platform, handling queries about order status, returns, warranty coverage, product discovery, and troubleshooting.

2. System Goals

2.1 Functional Goals:

- * Support transaction-like queries (orders, returns, warranty).
- * Make product suggestions using structured catalog data.
- * Provide troubleshooting steps and answer policy FAQs.

2.2 Non-Functional Goals:

- * Low hallucination: Answers grounded in policies/tools.
- * Safety: Guardrails preventing harmful content.
- * Performance: In-memory caching for rapid responses.

3. Tech Stack

- * Backend: FastAPI (Async Python framework).
- * LLM: Gemini 2.5 Flash (Reasoning and generation).
- * Embeddings: Sentence Transformers (all-MiniLM-L6-v2).
- * Vector Store: ChromaDB (Local, file-based).
- * Orchestration: Custom Router + LangChain.

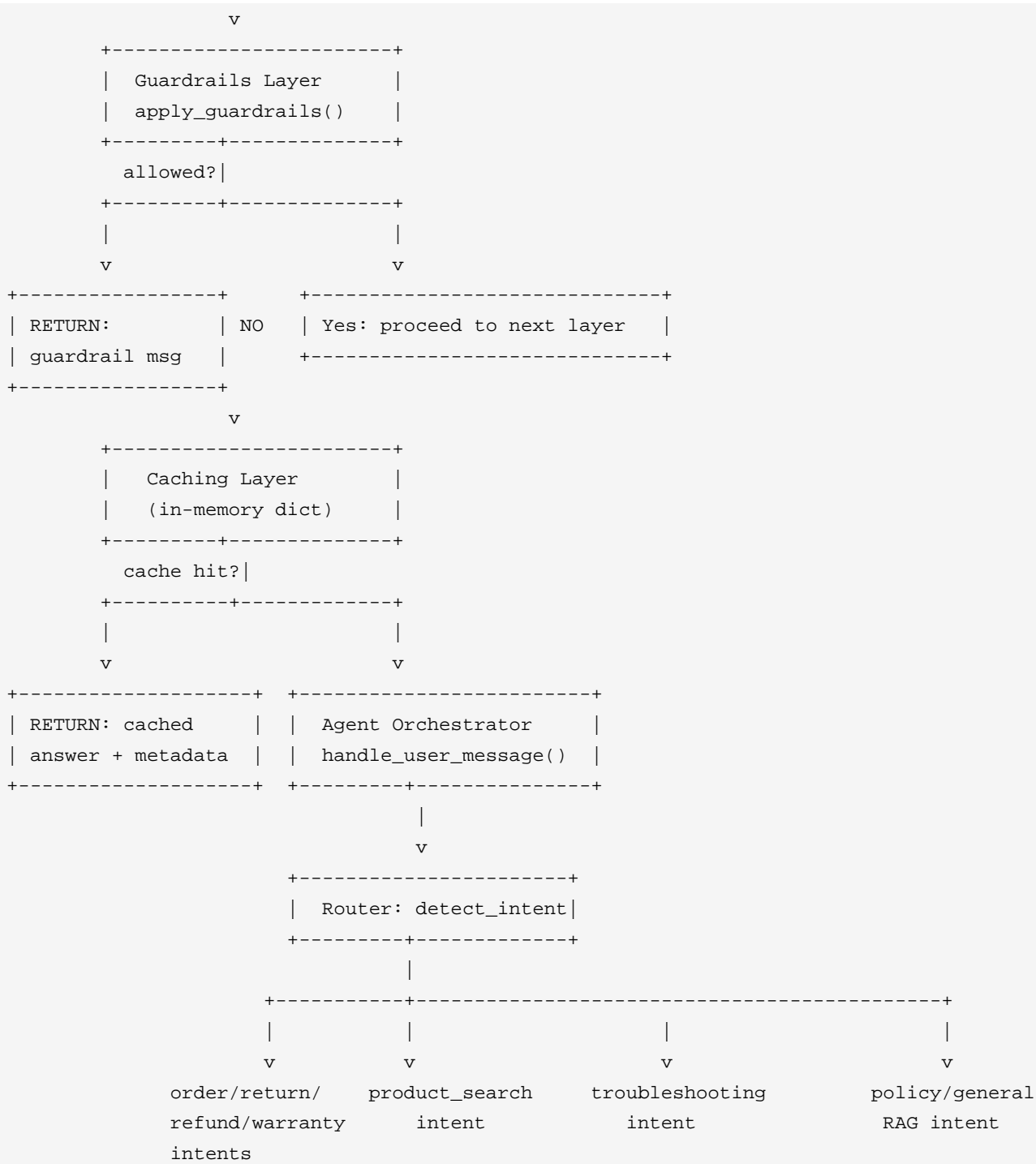
4. System Workflow & Architecture

The system follows a strict decision flow to ensure accuracy and safety. Below are the detailed diagrams for the backend and specific query branches.

4.1 High-Level Backend Flow

```
+-----+
|  User /chat POST  |
+-----+
| raw_message      |
```

Ecommerce Support Assistant - Technical Documentation



Ecommerce Support Assistant - Technical Documentation

4.2 Branch 1: Tool-based Queries (Orders/Returns/Warranty)

These queries are handled deterministically using pure Python logic on JSON data.

```
Intent IN {order_status, return_eligibility, refund, warranty_status}
    |
    v
+-----+
| Call corresponding tool: |
| - get_order_status_tool |
| - check_return_eligibility_tool
| - check_refund_possibility_tool
| - check_warranty_status_tool
+-----+
    | (Python logic on JSON data)
    v
Build natural language answer -> Save result in cache -> RETURN
```

4.3 Branch 2: Product Search (Tools + LLM)

A hybrid approach: Tools filter the catalog, and the LLM phrases the recommendation.

```
Intent == product_search
    |
    v
+-----+
| Tool: search_products_tool.invoke |
| - Filter by category, max_price |
| - Filter by tags |
+-----+
    |
    v
+-----+
| LLM (Gemini via get_llm) |
| Prompt: |
| - "Here are products from catalog..." |
| - Explain, compare, recommend |
+-----+
    |
    v
Final recommended answer -> Cache -> Return
```

4.4 Branch 3: Troubleshooting (Tool-First, RAG-Second)

```
Intent == troubleshooting
    |
```

Ecommerce Support Assistant - Technical Documentation

```

      v
+-----+
| Tool: get_troubleshooting_steps_tool |
|   - Normalize product_type           |
|   - Infer issue_key                  |
+-----+
      |
      | steps found?
      +----- Yes -----> Return tool message (no RAG)
      |
      v
    No
      |
      v
+-----+
| Fallback to RAG: answer_with_rag()   |
|   - Retrieve from manuals/policies    |
|   - LLM (Gemini) generates answer    |
+-----+

```

4.5 Branch 4: Policy & FAQ (RAG-Only)

```

Intent IN {policy_question, general_rag}
      |
      v
+-----+
| RAG: answer_with_rag()               |
|   1. Embed query                     |
|   2. Chroma similarity                |
|   3. Retrieve top-k chunks           |
|   4. Prompt Gemini                    |
+-----+

```

5. User Interface & Capabilities

5.1 Order Tracking & Greeting

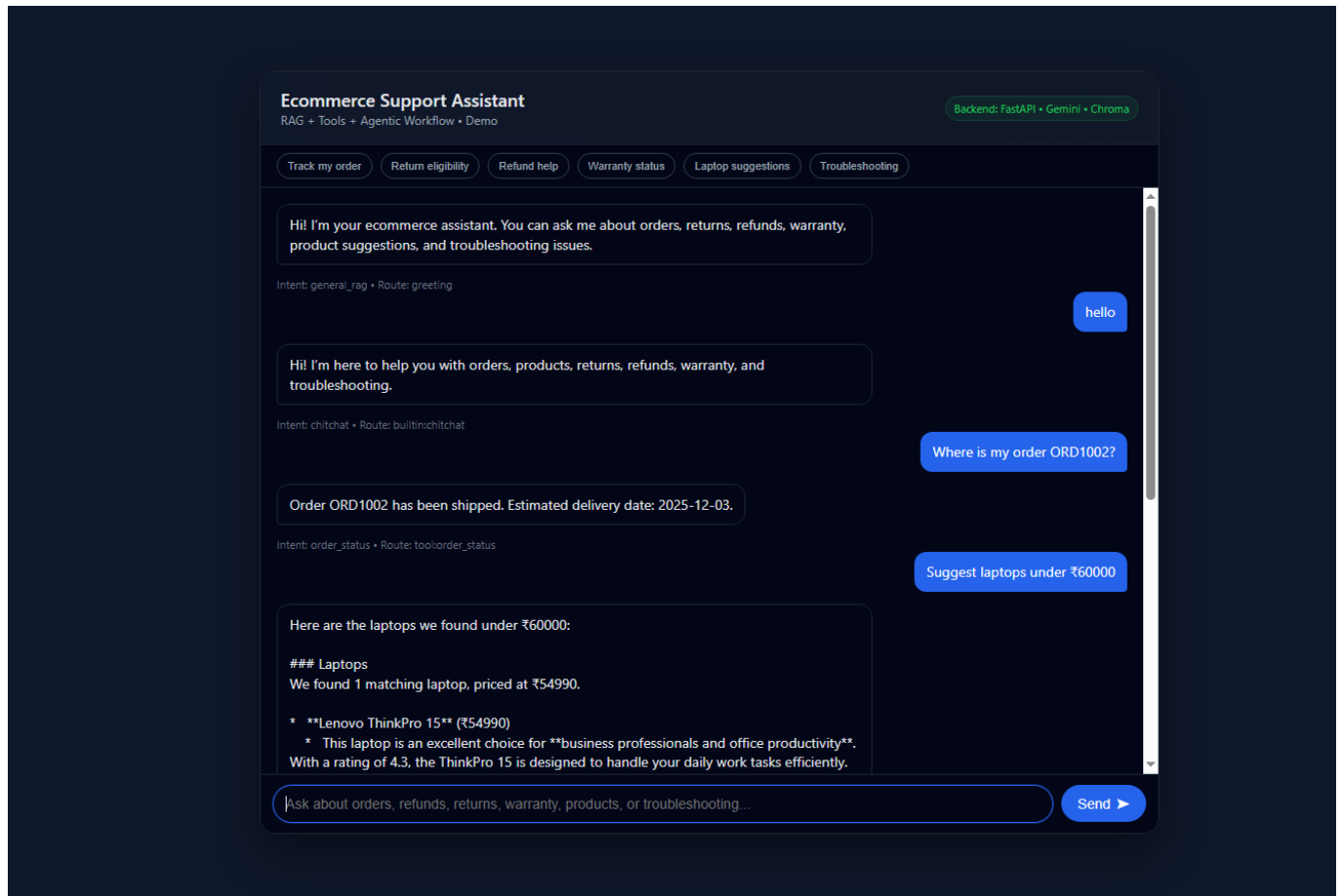


Figure 1: Greeting, Order Tracking, and Search Intents

5.2 Product Recommendations & Troubleshooting

Ecommerce Support Assistant - Technical Documentation

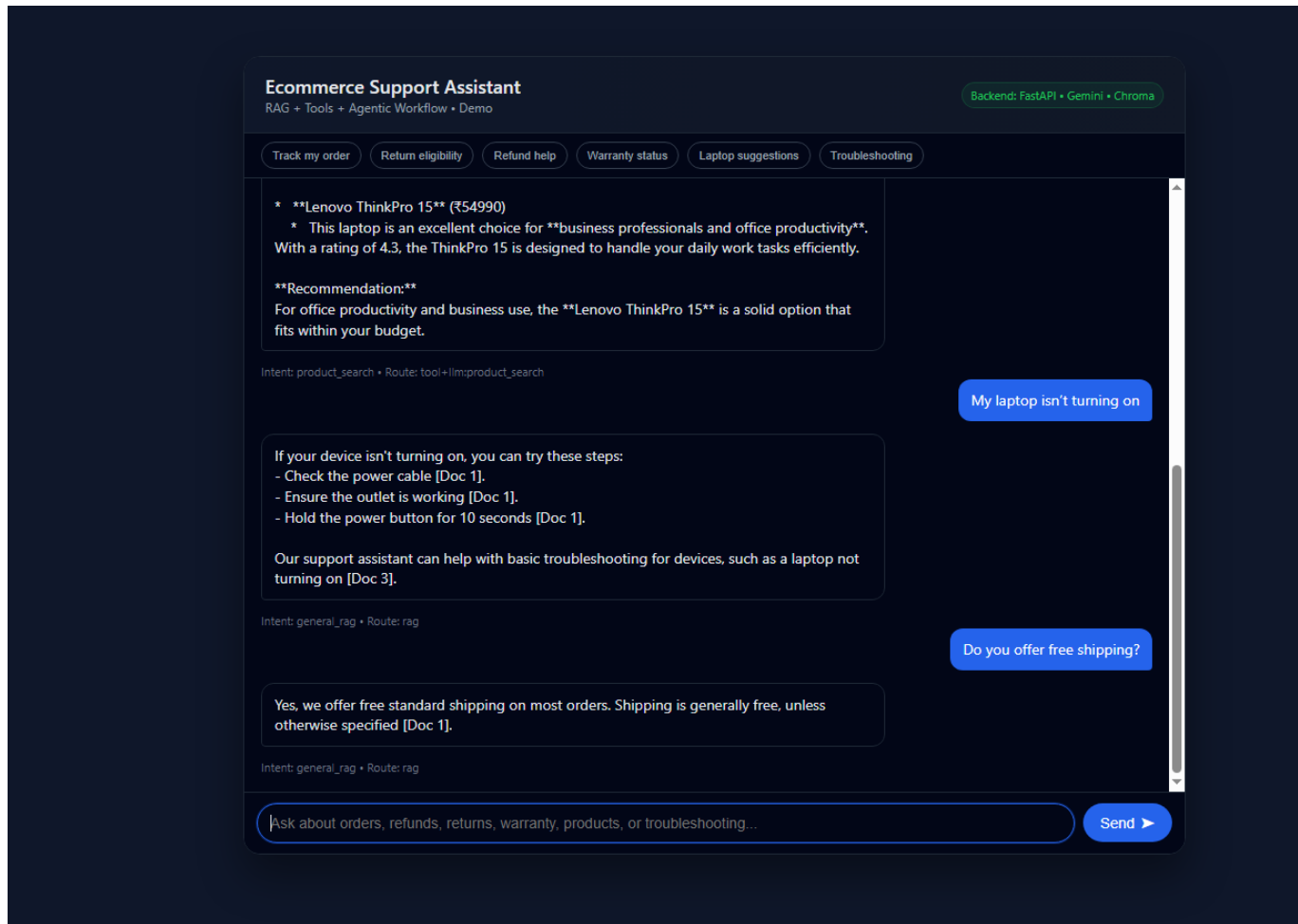


Figure 2: Product Suggestion and Troubleshooting Flow

5.3 RAG Support & Guardrails

Ecommerce Support Assistant - Technical Documentation

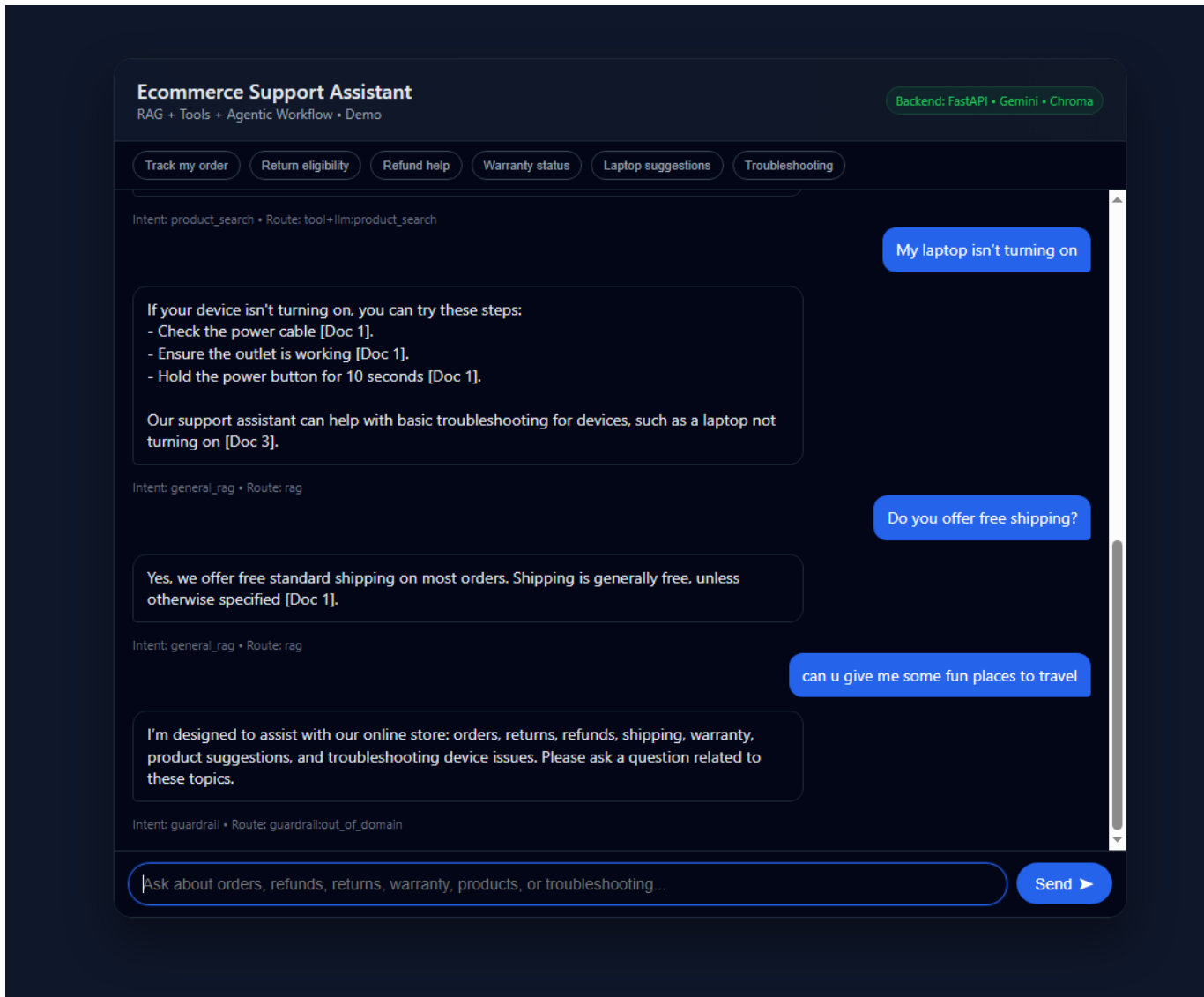


Figure 3: Policy RAG and Guardrail Blocking

6. Evaluation

We performed a structured evaluation on 10 queries across policy retrieval, tool-based logic, and troubleshooting.

6.1 Retrieval Evaluation Results

Metric	Score	Remark
-----	-----	-----
Average Recall@k	1.00 (100%)	Perfect document retrieval
Average Precision@k	0.57 (57%)	Some extra chunks retrieved (expected)

Interpretation: The system achieved Perfect Recall, meaning the relevant reference document was always retrieved within the Top-3 results. Precision is moderate due to the inclusion of adjacent context chunks, which is typical for small vector stores.

6.2 Answer Quality Evaluation Results

Metric	Score	Remark
-----	-----	-----
Routing Accuracy	90%	Strong agent workflow
Hallucination Rate	10%	Good safety behavior
ROUGE-L F1	0.24	Answers factually correct but vary in phrasing

Interpretation: Routing is very strong (9/10 correct). Hallucinations are low due to tool-first logic. The ROUGE score reflects that while the answers are correct, the LLM often generates more natural/verbose responses than the ground truth references.

6.3 Insights per Query Type

Query Type	Performance	Notes
-----	-----	-----
Policy & Company FAQs	Very High	RAG works accurately
Order/Return/Warranty	Perfect	Tools are 100% deterministic
Troubleshooting	Good	Structure is good, needs more detail
Product Discovery	Moderate	Limited by small demo catalog

6.4 Final Evaluation Summary

We evaluated the Antigravity AI Commerce Assistant using a structured evaluation set. Retrieval performance was excellent (100% Recall), ensuring correct knowledge access. The agent demonstrated 90% routing accuracy, confirming that the intent classifier successfully activates the correct tool or RAG pipeline.

Hallucination was low (10%), showing that the safeguards, RAG grounding, and tool-first logic significantly reduce fabricated responses. Overall, the system delivers reliable ecommerce support automation with room for further enhancements in product discovery.

7. Future Improvements

- 1. Reranking: Implement LLM scoring to boost Precision@k.
- 2. Expanded Catalog: Add more products and attributes to improve recommendations.
- 3. Prompt Reinforcement: Further reduce hallucinations with stricter system prompts.
- 4. Fine-grained Evaluation: Add BERTScore for stronger NLP benchmarking.