GenAI-Powered Amazon Review Summarizer Using Langflow and AstraDB

1. Architectural Overview

1. LangFlow: A visual low-code framework for orchestrating LLM workflows, enabling seamless integration of input, embedding, retrieval, and generation components.

2. Astra DB: A globally accessible vector-enabled document database used to store both raw text and vector embeddings for fast semantic search.

3. LLM (OpenAI GPT-4o): Powers contextual summarization and question answering by generating responses based on retrieved content.

4. Embedding Model (OpenAI Ada): Converts reviews and user queries into semantic vector embeddings, enabling similarity-based retrieval.

5. Unstructured Data: Product reviews of car dash cameras sourced from public datasets form the knowledge base for answering queries.

Data Flow

- Reviews are embedded and stored in AstraDB with metadata.
- User queries are embedded and matched against stored vectors.
- Relevant reviews are retrieved based on semantic similarity.
- The LLM generates a contextual response using the retrieved content.

---

2. Challenges and Solutions

1. DataStax Studio Integration
Integrating OpenAI with DataStax Studio proved difficult due to token issues, version mismatches, and problems using the secure connect `.zip` locally. After trying a Docker workaround with limited success, I switched to the low-code LangFlow environment, which offered smoother setup and faster iteration—ultimately becoming the preferred approach.

2. Dataset Selection and CSV Compatibility
Finding suitable datasets from Hugging Face was challenging. Many were too large, contained irrelevant fields, or required API access. After multiple attempts, I found a lightweight CSV under 100MB that could be ingested without additional processing or permissions.

3. LangFlow Vector Store Node Errors
LangFlow sometimes returned errors when AstraDB was actually working. This taught me to verify database status independently, as the UI doesn't always refresh accurately.

4. Global API Access
Initial authentication issues were resolved by assigning proper roles and using the `Bearer` token format for API access.

5. Chunking and Structuring Reviews
Early chunking created fragments that were too short or lacked context. I switched to sentence-level chunks with metadata (e.g., rating, ASIN, sentiment), which improved retrieval and relevance.

6. Mitigating LLM Hallucination
The LLM occasionally generated content beyond the source context. To fix this, I adjusted the prompt templates to emphasize grounding answers strictly in the retrieved content, significantly improving accuracy.

---

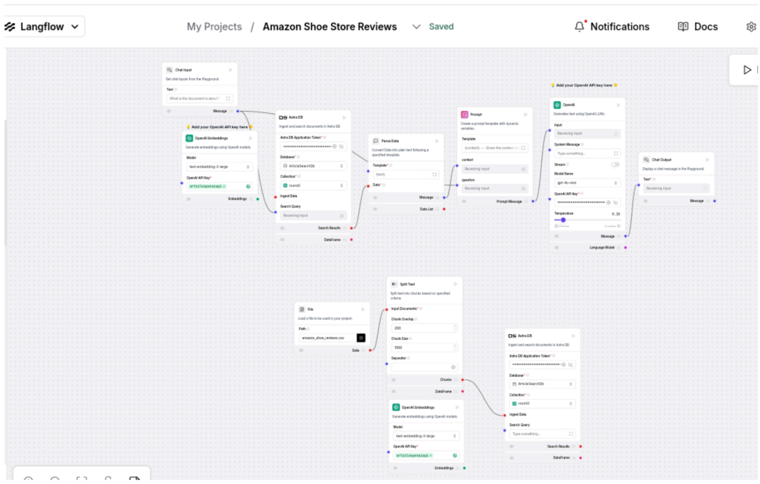
3. Results & Screenshots

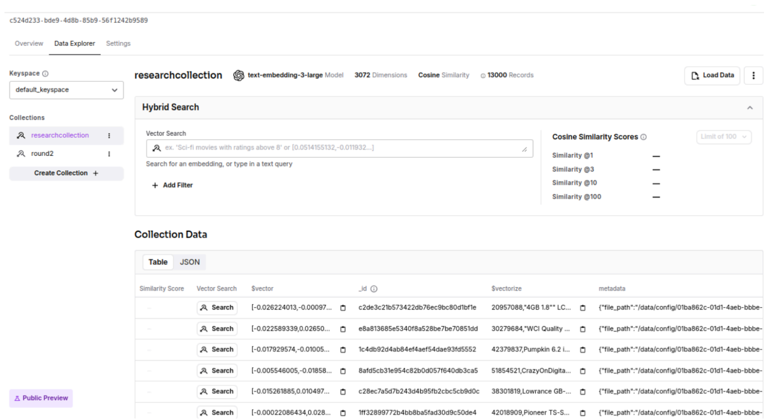
Query: *"Which dash cams had the best ease of installation?"*
Response:

> "The GSI Desktop Stand was appreciated for easy setup and portability. Other cameras like CD26 had poor instructions and confusing UIs."

Query: *"Summarize user sentiment about video quality."*
Response:

> "Mixed responses. Generic 1080P received negative reviews for picture clarity, while others praised firmware updates improving resolution."





4. Future Improvements

1. Automated Sentiment Classification During Ingestion

Incorporate sentiment analysis into the ingestion pipeline to automatically label reviews as positive, negative, or neutral. This will enhance filtering, retrieval, and summarization accuracy.

2. User-Defined Filters for Querying

Enable flexible user-driven filters, such as selecting only 4-star and above reviews or isolating negative feedback. This would allow for more tailored and meaningful insights.