E-Commerce Chatbot using GenAI

Fractal-Nexus Hackathon (March 2025)
Prince Kumar Singh (prince.singh@fractal.ai)

Overview

Customers often face difficulties due to complicated return policies, long processing times, and poor customer service experiences. Leverage GenAI to build an intelligent chatbot that:

- Guides users through the return/exchange process based on product type, policy, and purchase history.
- Understands customer intent (return, exchange, refund, replacement) and provides instant solutions.
- Checks order eligibility for returns/exchanges based on provided policy documents.
- Offers personalized recommendations for alternative products.
- Estimates customer mood and escalates issues accordingly.

Data Understanding

Two main source of data coming from other systems:

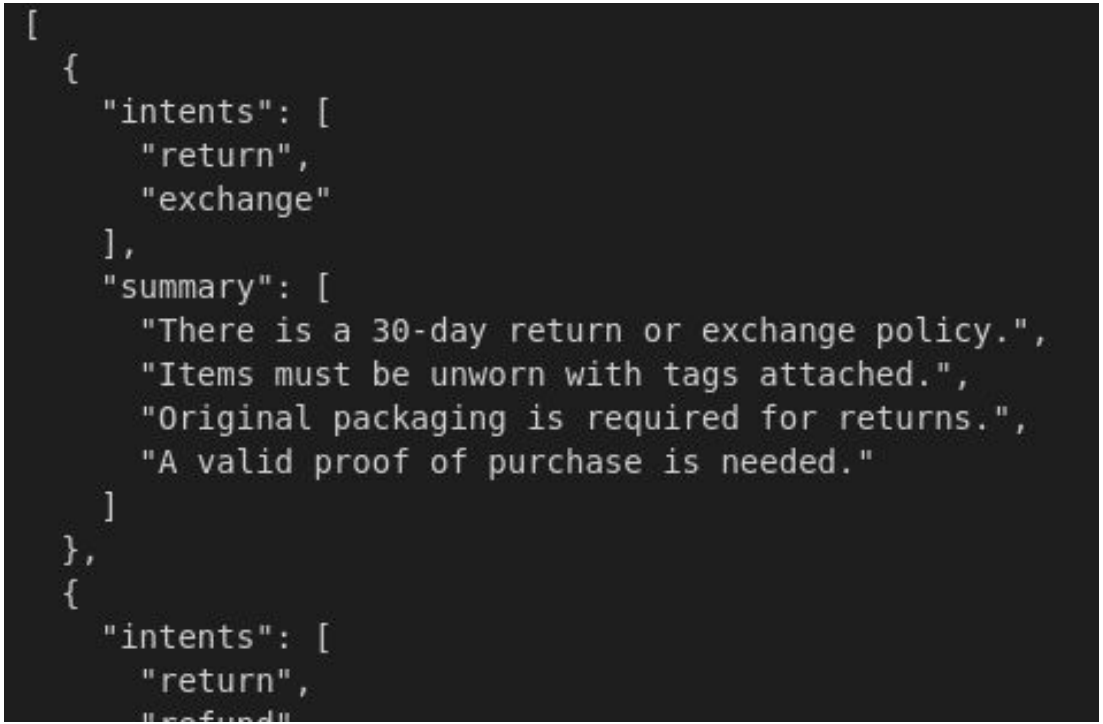
- Policy and rules related documents.
- Customer orders data.

Data Sources

Policy Info	Customer Orders Data
● **Return Window**: 30-day return period from the delivery date. ● **Eligibility Conditions**: Items must be in their original condition with tags and packaging. ● **Non-Returnable Items**: Final sale, custom-made, and used items. ● **Process Steps**: Initiation via chatbot, Return Authorization (RA) number, and trackable shipping.	● **Order ID**: Unique identifier for each purchase. ● **Product Category**: Specifies the type of product purchased. ● **Purchase Date**: Helps determine return eligibility based on policy timelines. ● **Return/Exchange Status**: Indicates whether an order has been returned, exchanged, or refunded. ● **Customer ID**: Links orders to specific users for personalized interactions.

Parsing Policy Data

- The documents are broken down into sections and subsections.
- Each of the subsections are summarized into key points and their main intents are derived to create a better structured format.



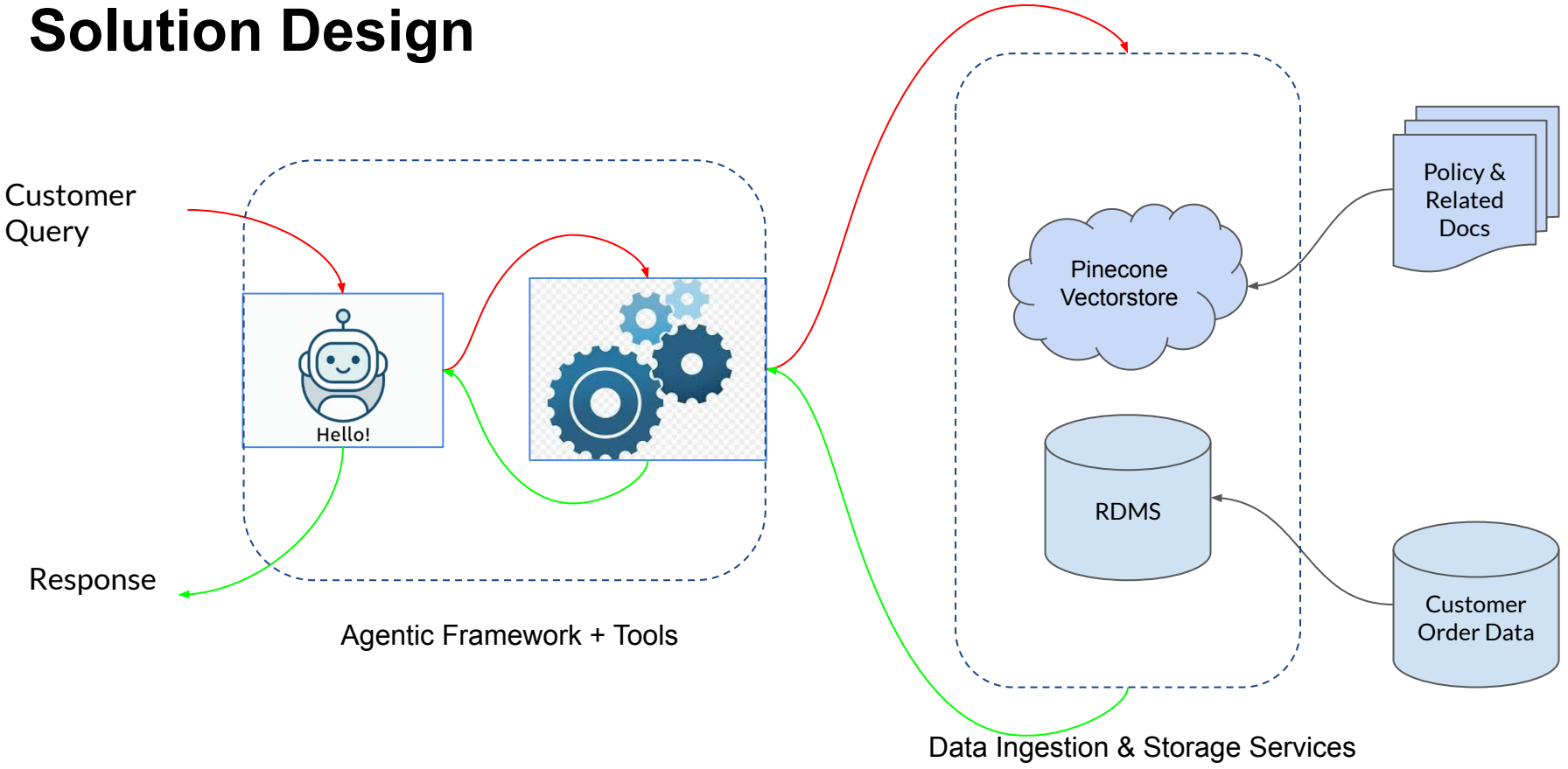
Policy Data Ingestion Service

- The following fields are created for each sentence in the JSON file:
  - id: This a unique ID for each of the unique sentence
  - metadata: This field stores the text and list of intents applicable to the text
  - values: Embedding vector of each sentence. Here **all-mpnet-base-v2** is used.
- All the records are saved in Pinecone vector store
- This service is always live and checks for any new document to process every hour.
- Once the document is processed it saves the filename in database.

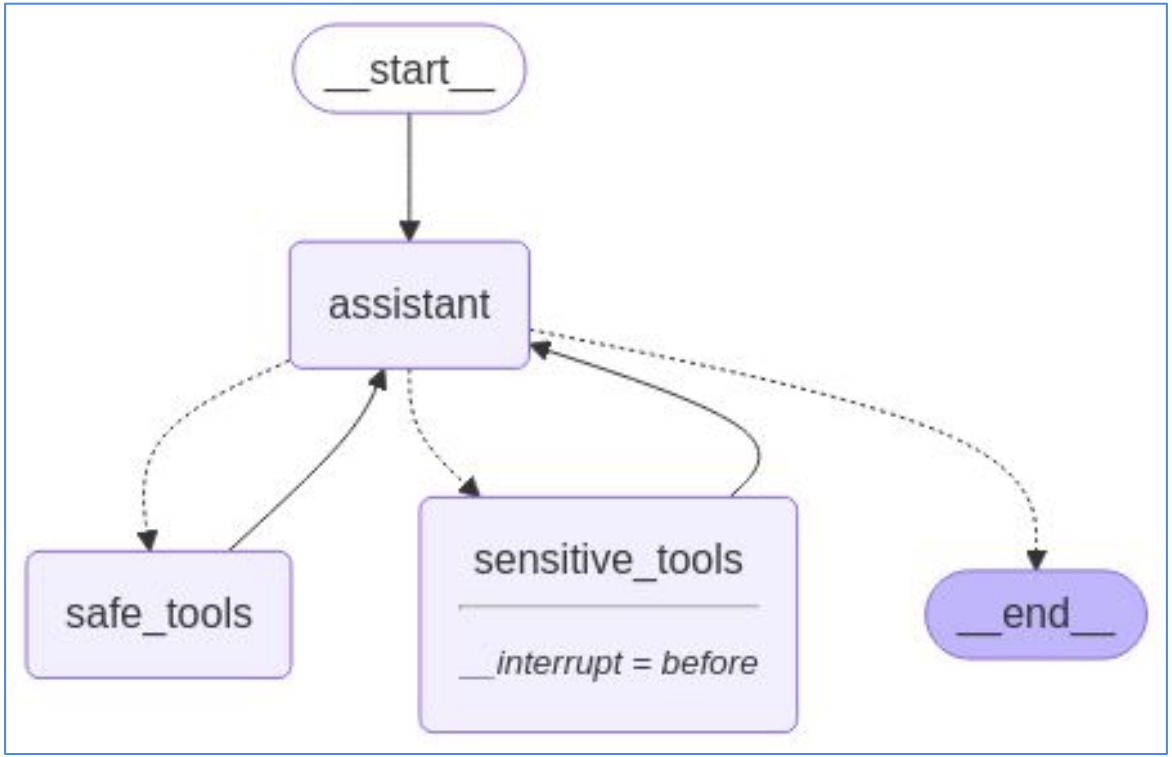
Orders Data Ingestion Service

- This service checks for new records in the given physical file location.
- It uses Order ID as unique key to identify new orders data.
- It standardizes the columns names and date format and saves data to database.
- This service checks for new records every hour.

Solution Design



Bot Graph


__start__
assistant
safe_tools
sensitive_tools
__interrupt = before
__end__

Tools Used

- Product-Recommendor-By-OrderID
  - Gets products similar to the product given any order ID.
  - Checks for same product category, size and gender (or Unisex)
- Generate-Return-Authorization
  - Generates a unique return authorization number for a given order ID.
  - Considered as “Sensitive” tool in the model.
- Get-Order-Details
  - Fetches order details from the database given order ID.
- Get-Relevant-Policies-By-Query
  - Retrieves relevant policy details given an order ID.
  - Fetches policies from vector store by similarity and intents metadata
  - Re-ranks the policies using bge-reranker-v2-m3 reranker model and filters all having score lower than threshold
- Days-Since-Date
  - Calculates the number of days passed since the given order date.

Tech Stack Used

- UI/UX
- Chatbot and Tools for RAG framework
  - Langgraph - very granular control of data flow, flexible, less black-box and more developer friendly
- LLM
  - gemma2-9b-it from Google - lightweight, state-of-the-art open model
  - llama3-70b-8192 from Meta - state-of-the-art open LLM model
  - Groq API - free API
- Vectorstore
  - Pinecone - free, serverless, low latency, reranking and metadata filtering available
  - Embedding - sentence-transformers/all-mpnet-base-v2 through Huggingface - well-established, pre-trained, suitable for similarity tasks for RAG like applications

Next Steps

- Fine-tuning and further prompt-engineering for better accuracy and response.
- Add a gateway node which can act as filter for all kind of rules.
- Add multi-user support and session handling.
- Design multi-agent system rather than can enable all the models to run with smaller scope of work and lightweight models thus decrease running cost.
- Make the agent interact with other systems like - recording return requests, informing inventory and shipping partners in automated way.

Thank You