Nestor - SKEZI's Artificial Intelligence

Technical Architecture & Algorithmic Overview

The RAG Pipeline, Semantic Caching, and Data Processing Workflows behind SKEZI's Support AI

"Je pense, donc je suis." — René Descartes

Executive Summary

Nestor is SKEZI's AI-powered support chatbot, designed to provide immediate, accurate, and context-aware answers to product questions. It leverages a modern Retrieval-Augmented Generation (RAG) pipeline to ensure all responses are grounded in SKEZI's official documentation and website data.

The system's architecture is inspired by human cognition. The persistent vector database (ChromaDB) acts as the **"Hippocampus"**—our long-term memory for all product knowledge. A semantic cache (FAISS) functions as **"Working Memory"** for rapid, efficient recall of common, identical, or similar questions.

When a query is received, Nestor first checks its "Working Memory" (cache). If a high-similarity answer isn't found, it queries its "Long-Term Memory" (ChromaDB) to retrieve the most relevant context. This context is then passed to the "Frontal Lobe" (the LLM via OVH AI Endpoints), which reasons, synthesizes, and generates a coherent, human-like answer.

This "cache-first" approach significantly reduces API costs and latency, providing a fast user experience. Prompt templates are further engineered with specific persona and **"emotional" cues**, guiding the LLM to respond in SKEZI's helpful and professional brand voice. Unanswered questions are logged, creating a data-driven feedback loop for continuous knowledge base improvement and data analysis.

Résumé Exécutif

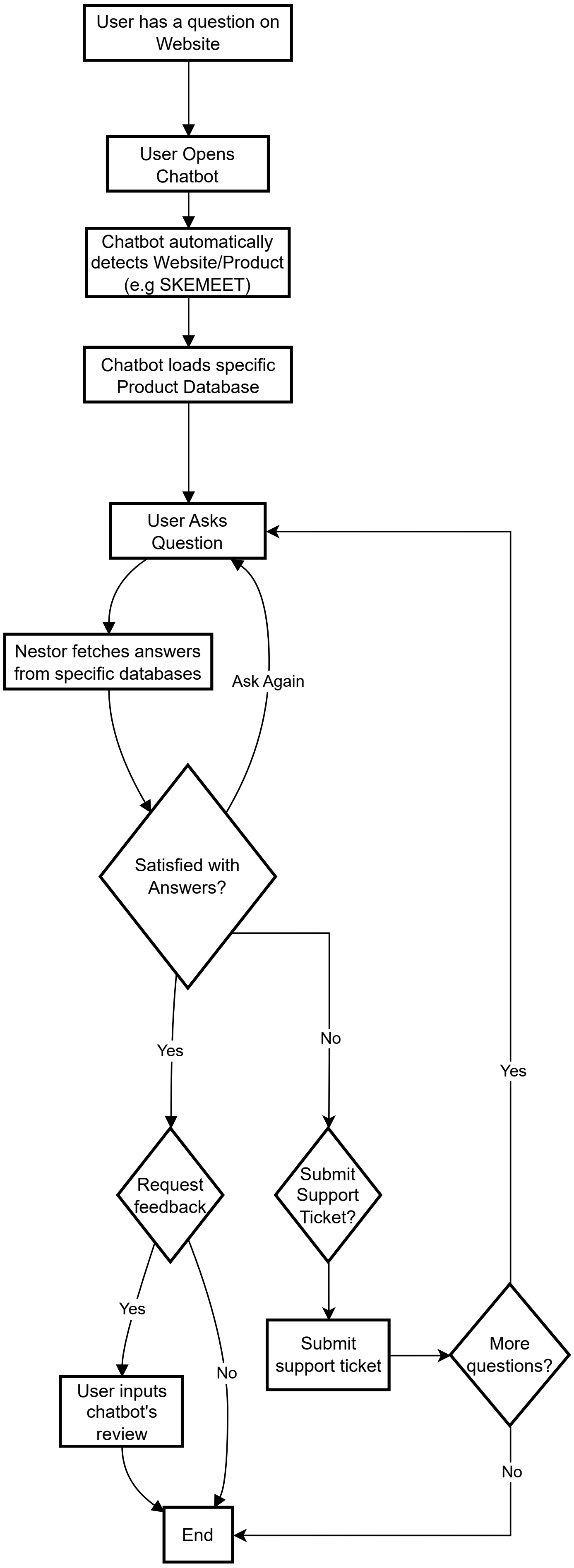
Nestor est le chatbot d'assistance de SKEZI, alimenté par l'IA et conçu pour fournir des réponses immédiates, précises et contextuelles aux questions sur les produits. Il s'appuie sur un pipeline moderne de Génération Augmentée par Récupération (RAG) pour garantir que toutes les réponses sont basées sur la documentation officielle et les données du site web de SKEZI.

L'architecture du système est inspirée de la cognition humaine. La base de données vectorielle persistante (ChromaDB) agit comme notre **"Hippocampe"** – notre mémoire à long terme pour toute la connaissance produit. Un cache sémantique (FAISS) fonctionne comme une **"Mémoire de Travail"** pour un rappel rapide et efficace des questions courantes, identiques ou similaires.

Lorsqu'une requête est reçue, Nestor vérifie d'abord sa "Mémoire de Travail" (le cache). Si aucune réponse à haute similarité n'est trouvée, il interroge sa "Mémoire à Long Terme" (ChromaDB) pour récupérer le contexte le plus pertinent. Ce contexte est ensuite transmis au **"Lobe Frontal"** (le LLM via OVH AI Endpoints), qui raisonne, synthétise et génère une réponse cohérente et naturelle.

Cette approche "cache-first" (cache en priorité) réduit considérablement les coûts d'API et la latence, offrant une expérience utilisateur rapide. Les modèles de prompt (prompt templates) sont également conçus avec un persona spécifique et des **"indices émotionnels"**, guidant le LLM pour qu'il réponde avec le ton serviable et professionnel de SKEZI. Les questions sans réponse sont enregistrées, créant une boucle de rétroaction basée sur les données pour l'amélioration continue de la base de connaissances et l'analyse des données.

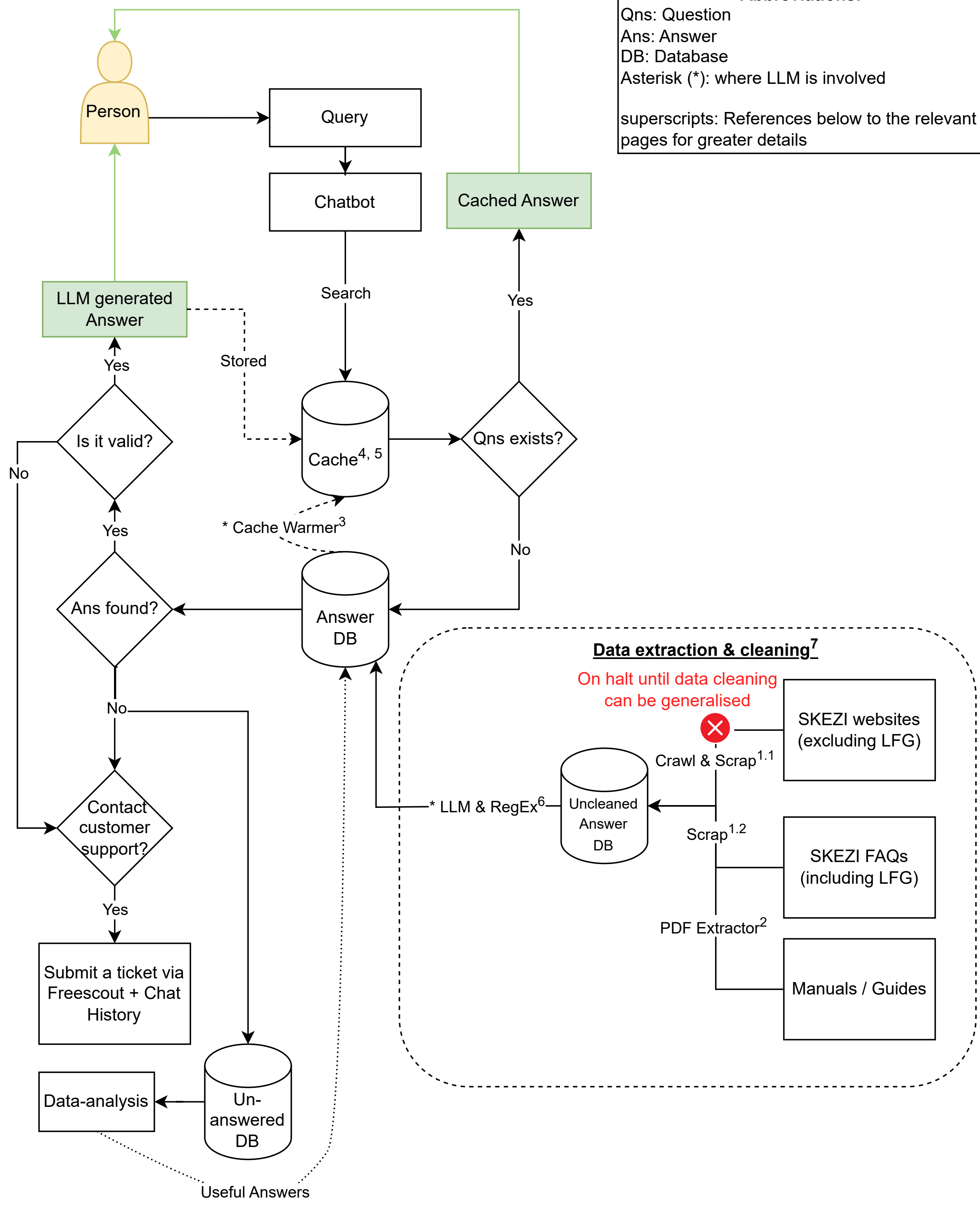
Nestor's UX Flow


User has a question on Website
User Opens Chatbot
Chatbot automatically detects Website/Product (e.g SKEMEET)
Chatbot loads specific Product Database
User Asks Question
Nestor fetches answers from specific databases
Ask Again
Satisfied with Answers?
Yes
No
Yes
Request feedback
Submit Support Ticket?
Yes
No
User inputs chatbot's review
Submit support ticket
More questions?
No
End

Nestor's backend Flow (Chatbot.py)

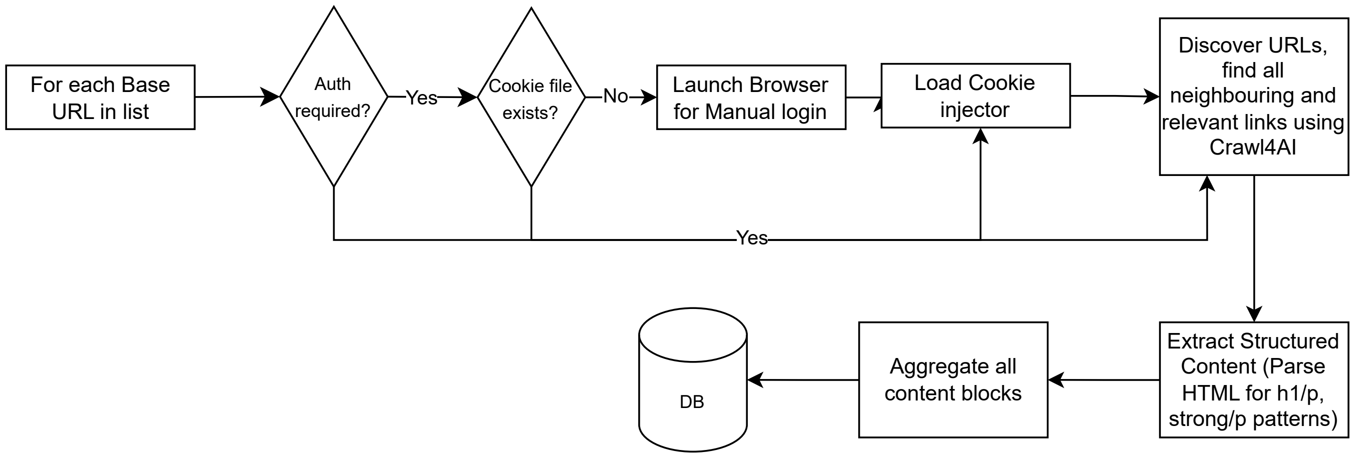
Abbreviations:
Qns: Question
Ans: Answer
DB: Database
Asterisk (*): where LLM is involved

superscripts: References below to the relevant pages for greater details

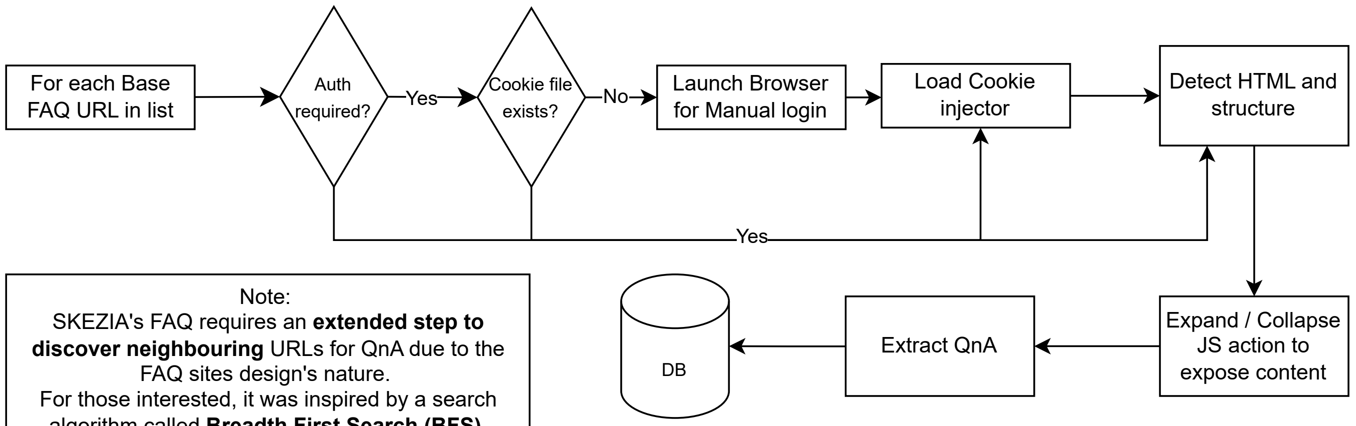


Document retrieval process

1.1 Web Crawling and Scrapping (Crawler.py)


For each Base URL in list
Auth required?
Yes
Cookie file exists?
No
Launch Browser for Manual login
Load Cookie injector
Discover URLs, find all neighbouring and relevant links using Crawl4AI
Yes
Extract Structured Content (Parse HTML for h1/p, strong/p patterns)
Aggregate all content blocks
DB

1.2 Scrapper for FAQ sites (FAQ_scrapper.py)

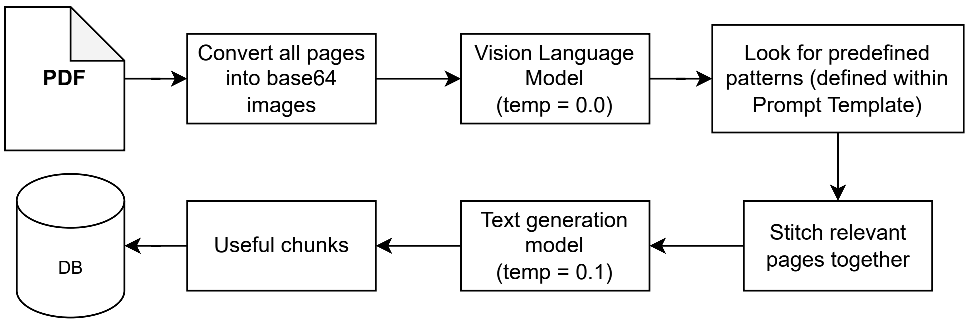

For each Base FAQ URL in list
Auth required?
Yes
Cookie file exists?
No
Launch Browser for Manual login
Load Cookie injector
Detect HTML and structure
Yes
Expand / Collapse JS action to expose content
Extract QnA
DB

Note:
SKEZIA's FAQ requires an **extended step to discover neighbouring** URLs for QnA due to the FAQ sites design's nature.
For those interested, it was inspired by a search algorithm called **Breadth First Search (BFS).**

Although, one can also use Depth First Search (DFS) as well albeit with slight modifications to the code.

2. Extracting information from PDF guides (PDF_Extractor.py)

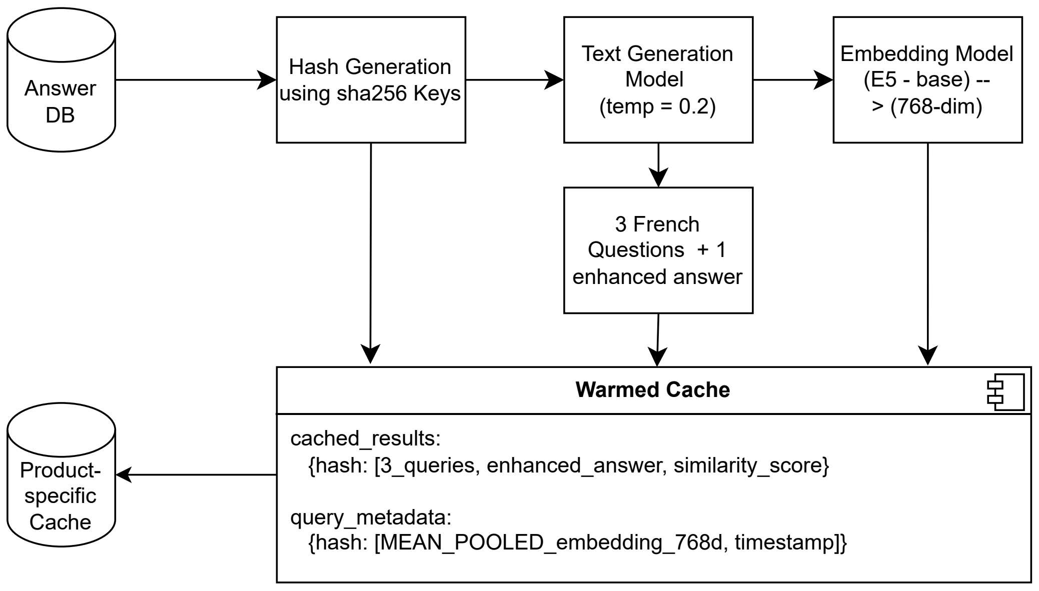
REQUIRES WORK FOR GENERALISATION


PDF
Convert all pages into base64 images
Vision Language Model (temp = 0.0)
Look for predefined patterns (defined within Prompt Template)
Stitch relevant pages together
Text generation model (temp = 0.1)
Useful chunks
DB

Note:
Temp controls the randomness of an LLM's output.
0 = Deterministic: Focused, predictable responses.
1 = Creative: Highly variable responses, which can lead to hallucinations.

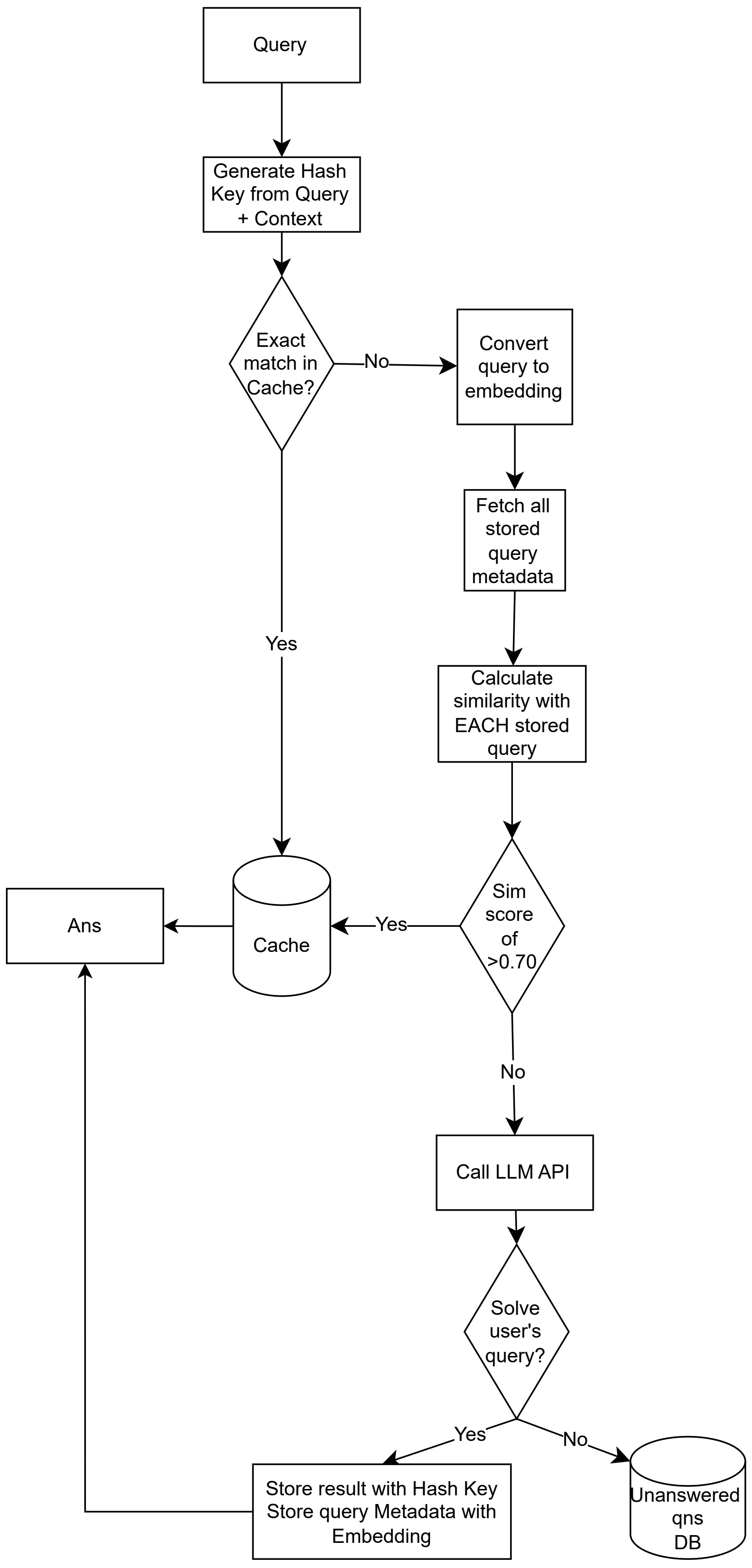
Cache system

3. Cache warming ON HOLD FOR NOW UNTIL THERE IS SUFFICIENT DATA FROM REAL USERS


Answer DB
Hash Generation using sha256 Keys
Text Generation Model (temp = 0.2)
Embedding Model (E5 - base) --> (768-dim)
3 French Questions + 1 enhanced answer
Warmed Cache
cached_results:
{hash: [3_queries, enhanced_answer, similarity_score}
query_metadata:
{hash: [MEAN_POOLED_embedding_768d, timestamp]}
Product-specific Cache

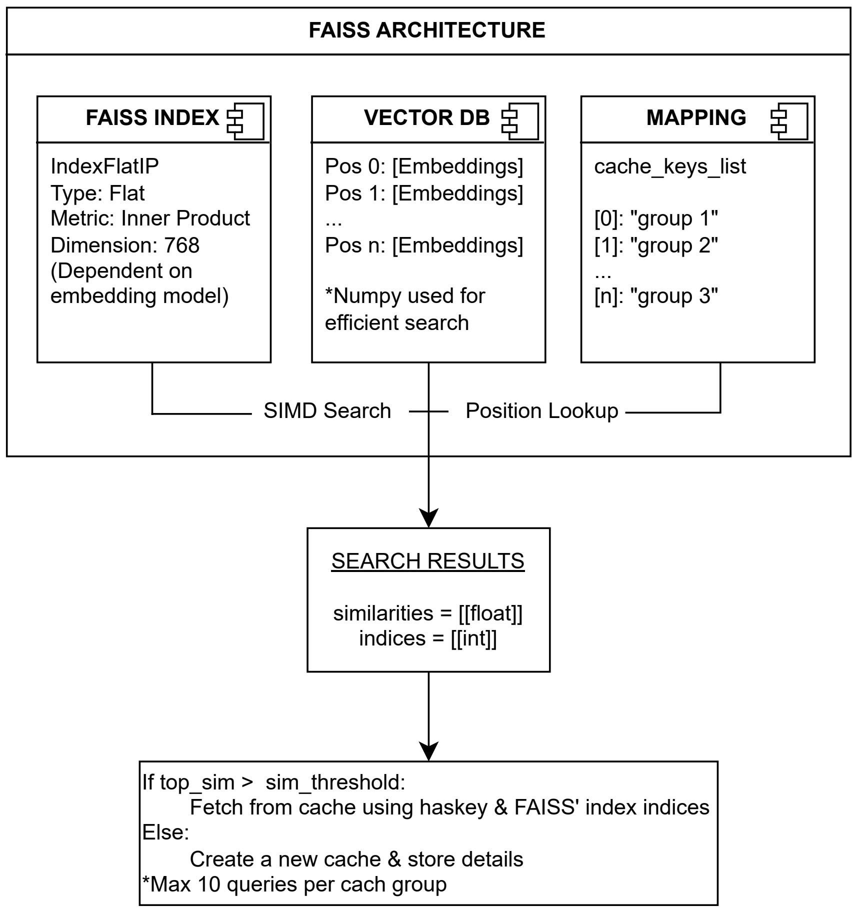
Cache system

4. Cache data flow


Query
Generate Hash Key from Query + Context
Exact match in Cache?
No
Convert query to embedding
Fetch all stored query metadata
Calculate similarity with EACH stored query
Yes
Sim score of >0.70
Yes
Cache
Ans
No
Call LLM API
Solve user's query?
Yes
No
Store result with Hash Key Store query Metadata with Embedding
Unanswered qns DB

Cache system

5. Improved Cache Retrieval System (FAISS)



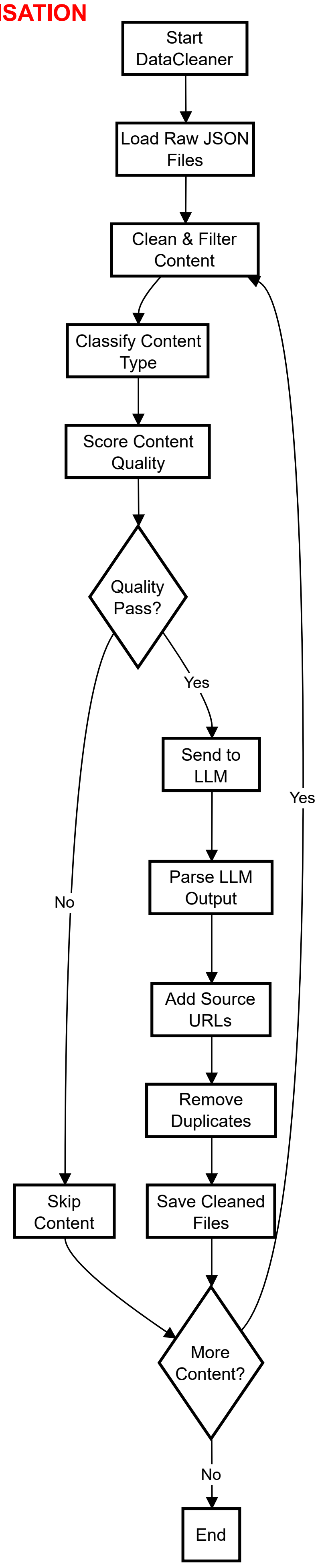
```
If top_sim >  sim_threshold:
        Fetch from cache using haskey & FAISS' index indices
Else:
        Create a new cache & store details
*Max 10 queries per cach group
```

Note:
The cache databases utilises *custom* FAISS (for its product quantisation and inverted file index structures) for log(n) retrieval. This is because the implementation of the cache DB requires a more customary approach due to the usage of mean-pooled embeddings.

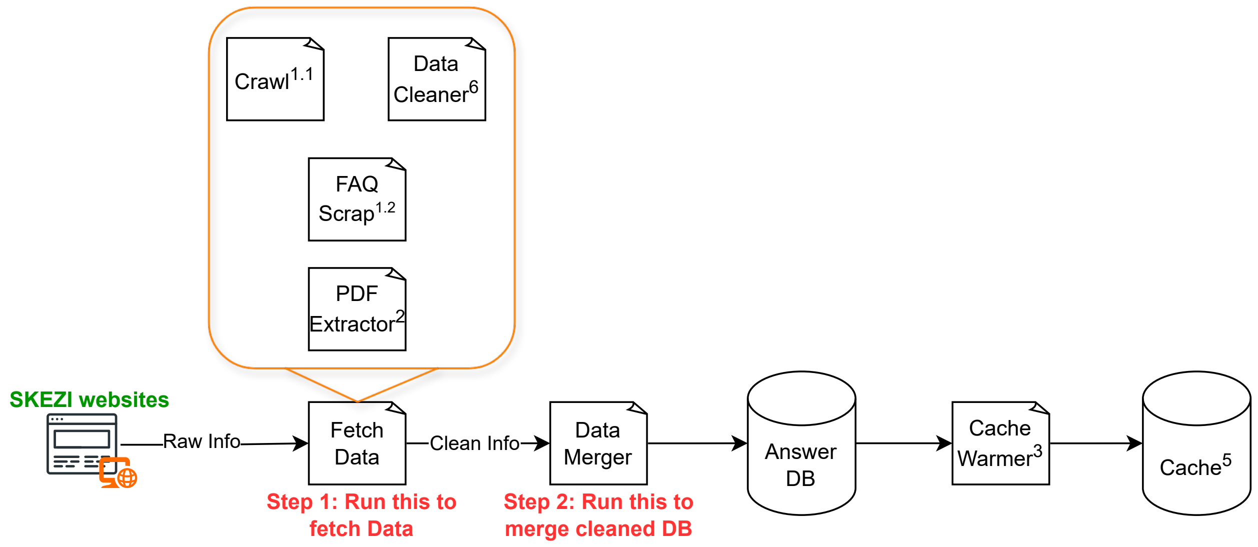
Data cleaning

6. Cleaning unstructured data scrapped and crawled from websites (DataCleaner.py)

REQUIRES WORK FOR GENERALISATION

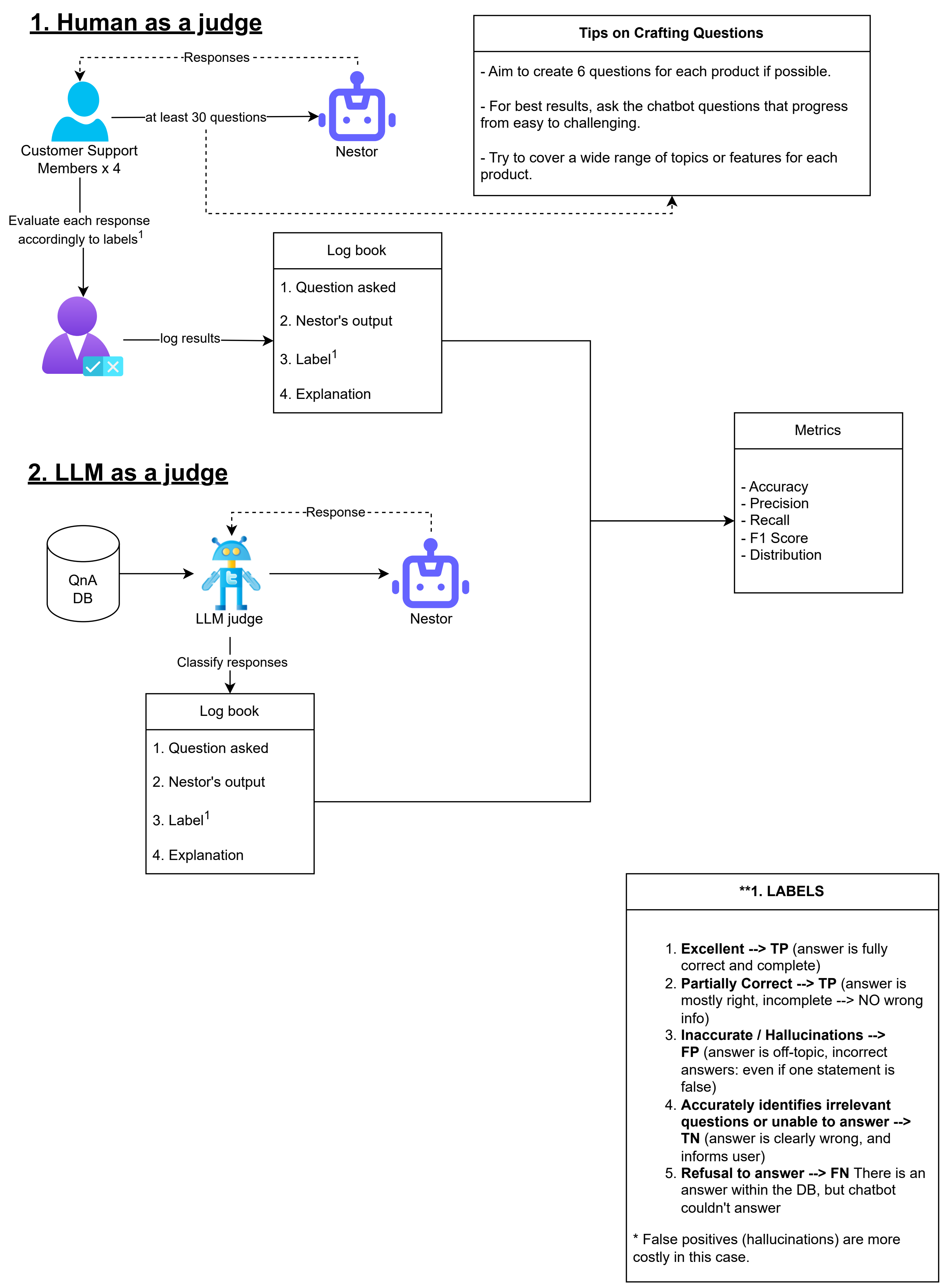


7. How to fetch data automatically (Scripts under Data Processing)



CAUTION:
There are some scripts that still require your attention for improvement due to its instability for generalisation. As a result, requires more work to extract meaningful and non-hallucinated information. **I have noted them respectively in their titles.**

Chatbot Evaluation



Tips on Crafting Questions

- Aim to create 6 questions for each product if possible.

- For best results, ask the chatbot questions that progress from easy to challenging.

- Try to cover a wide range of topics or features for each product.

Log book

1. Question asked
2. Nestor's output
3. Label[1]
4. Explanation

Metrics

- Accuracy
- Precision
- Recall
- F1 Score
- Distribution

Log book

1. Question asked
2. Nestor's output
3. Label[1]
4. Explanation

****1. LABELS**

1. **Excellent --> TP** (answer is fully correct and complete)
2. **Partially Correct --> TP** (answer is mostly right, incomplete --> NO wrong info)
3. **Inaccurate / Hallucinations --> FP** (answer is off-topic, incorrect answers: even if one statement is false)
4. **Accurately identifies irrelevant questions or unable to answer --> TN** (answer is clearly wrong, and informs user)
5. **Refusal to answer --> FN** There is an answer within the DB, but chatbot couldn't answer

* False positives (hallucinations) are more costly in this case.

Reference: https://www.evidentlyai.com/llm-guide/llm-as-a-judge#reference-based-evaluation