**Technical Design Document**

**1. Introduction**

This document provides a technical overview of a **Retrieval-Augmented Generation (RAG)** pipeline that integrates structured restaurant data from CSV files and unstructured Wikipedia knowledge to power a conversational AI chatbot. The chatbot enables users to query restaurant-related information efficiently by retrieving relevant data and generating responses using a language model.

**2. System Overview**

**Architecture Overview**

The system follows a **modular pipeline** comprising the following stages:

1. **Data Ingestion**
   * Load structured restaurant data (CSV).
   * Load unstructured Wikipedia articles (JSON).
2. **Text Processing & Chunking**
   * Convert structured CSV data into a textual format suitable for retrieval.
   * Chunk Wikipedia data for better retrieval efficiency.
3. **Embedding & Indexing**
   * Convert processed text into vector embeddings using Hugging Face's all-MiniLM-L6-v2 model.
   * Store embeddings in a FAISS index for fast similarity search.
4. **Query Processing & Retrieval**
   * Extract metadata-based filters (location, category, rating) from user queries.
   * Retrieve the most relevant chunks from FAISS.
   * Boost results based on metadata relevance.
5. **Response Generation**
   * Generate responses using Mistral-7B-Instruct with a structured prompt strategy.
6. **User Interaction via Streamlit UI**
   * Users submit queries through an interactive interface.
   * Processed results and responses are displayed dynamically.

**3. Solution Process**

**CSV Data Ingestion**

* Made changes to the Restaurant data file:
  + Grouped by item ID—structured data at the menu item level instead of ingredient level.
  + Removed confidence column.
  + Removed item ID (but later realized it is needed for fact-checking citations, so can be added back).
  + Combined location.
  + Made rating, review count, and price more descriptive.
* Columns used for metadata: source: "csv", restaurant\_name, location, rating, categories.
* Convert each row into a LangChain document (each menu item).

**Build Wikipedia Knowledge Base**

* Generated 100 sample questions—10 different categories that the bot should handle.
* Based on those questions, downloaded relevant Wikipedia articles for the knowledge base.
* Used RecursiveCharacterTextSplitter to break long Wikipedia articles into smaller chunks (~500 characters).
* Each chunk is converted into a LangChain Document with metadata (title, url).
* Combined both structured and unstructured data.

**4. Embedding and Indexing Data**

Convert text-based restaurant and Wikipedia documents into **vector representations** and store them in FAISS for fast retrieval.

**Process:**

* **Load Hugging Face Sentence Transformer (all-MiniLM-L6-v2)**
  + Embeds all documents (both CSV and Wikipedia).
* **Store embeddings in FAISS (vector\_store = FAISS.from\_documents(...))**
  + Each document’s embedding and metadata are stored in FAISS.
* **Save FAISS Index (vector\_store.save\_local(...))**
  + The FAISS index is stored locally for future use.

**5. Query Processing & Retrieval Approach**

**Extracting Metadata Filters from the User Query**

* **Restaurant categories:** Matches categories in metadata (e.g., "Italian") and boosts matching documents in retrieval.
* **Wikipedia-specific keywords:**
  + "history"
  + "origin"
  + "traditional"
  + "popular"
  + "famous"
* If a Wikipedia keyword is detected, Wikipedia content is preferred.

**Search & Ranking Mechanism**

* Uses FAISS to retrieve the **k most relevant** documents based on query similarity.
* If the query contains restaurant-related keywords (**e.g., "menu", "dish", "restaurant"**), it **filters out Wikipedia documents** to prioritize structured CSV data.
* The boost\_faiss\_results function enhances ranking using metadata.

**6. LLM Integration (Mistral-7B-Instruct)**

* Uses Hugging Face’s hf\_hub\_download() to download the GGUF model.
* The model is loaded via Hugging Face Hub (hf\_hub\_download).

**Structured Prompt for Accuracy**

The prompt template:

### 🍽️ AI Assistant for Restaurant & Culinary Queries

You are an expert AI assistant specializing in restaurant-related queries. Your responses \*\*must be strictly based on the retrieved information\*\*.

---

### \*\*📖 Context Details\*\*

1️⃣ \*\*Structured Restaurant Database (CSV-based)\*\* – Includes restaurant names, menu items, pricing, reviews, and ingredients.

2️⃣ \*\*Unstructured Wikipedia Knowledge (JSON-based)\*\* – Includes restaurant history, food trends, Michelin Guide information, and general culinary knowledge.

### \*\*🔎 Retrieved Context:\*\*

{context\_text}

### \*\*❓ User Query:\*\*

{query}

### \*\*📝 Answer:\*\*

**7. User Interaction via Streamlit**

* **Text Input Box (st.text\_input)** for users to enter queries.
* **Submit Button (st.button)** triggers the retrieval and response pipeline.
* **Displays AI-generated responses (st.write)**.

**8. Summary**

| **Stage** | **Process** | **Output** |
| --- | --- | --- |
| **Data Ingestion** | Load CSV & Wikipedia JSON | LangChain Documents |
| **Text Chunking** | Split Wikipedia & format CSV | Text Chunks with Metadata |
| **Embedding & Indexing** | Convert to vectors & store in FAISS | FAISS Index |
| **Query Processing** | Extract filters from query | Metadata-based retrieval |
| **Retrieval & Ranking** | Search FAISS, apply boosting | Ranked Document Chunks |
| **Response Generation** | Use LLM with structured prompt | AI-generated answer |
| **Streamlit UI** | Interactive chatbot | User-friendly interface |

**9. Future Improvements**

* **Real-time data ingestion** for dynamic restaurant updates.
* **Improved metadata filtering** for personalized recommendations.
* **Fine-tuning the LLM** for better response accuracy.

This technical design document outlines the **end-to-end pipeline** for a RAG-based chatbot integrating restaurant data with Wikipedia knowledge, ensuring efficient retrieval and accurate AI-powered responses.