RAG System Architecture (Task 1.1)

1. System Overview

This document outlines the production-ready Retrieval-Augmented Generation (RAG) architecture designed for the Restaurant Discovery System. It leverages a cost-optimized, high-performance stack combining **Groq** for inference and **HuggingFace** for local embeddings.

2. Architecture Diagram

```
┌─────────────────────────────────────────────────────────────────┐
│                   Restaurant Discovery RAG System                │
│             (Groq LLM + FREE HuggingFace Embeddings)             │
└─────────────────────────────────────────────────────────────────┘

┌─────────────────┐
│  User Query     │  "Find Italian restaurants in downtown Dubai
│  (Natural Lang) │   with outdoor seating under AED 200"
└────────┬────────┘
         │
         ▼
┌─────────────────────────────────────────────────────────────────┐
│                   Query Processing Layer                         │
│  • Natural language understanding                                │
│  • Entity extraction (cuisine, location, price, amenities)       │
└─────────────────────────────────────────────────────────────────┘
         │
         ▼
┌─────────────────────────────────────────────────────────────────┐
│                   Retrieval Layer                                │
│                                                                  │
│  ┌──────────────┐         ┌──────────────┐                      │
│  │  Semantic    │         │  Metadata    │                      │
│  │  Search      │────────▶│  Filtering   │                      │
│  │  (HuggingFace│         │ (Structured) │                      │
│  │   Embeddings)│         │              │                      │
│  └──────────────┘         └──────────────┘                      │
│        │                        │                               │
│        └────────────┬───────────┘                               │
│                     ▼                                            │
│              ┌──────────────┐                                    │
│              │   Hybrid     │                                    │
│              │   Search     │                                    │
│              └──────┬───────┘                                    │
└─────────────────────┼───────────────────────────────────────────┘
```

```
                       │
                       ▼
┌─────────────────────────────────────────────────────────────┐
│                  Vector Database (ChromaDB)                  │
│                                                             │
│  ┌───────────────────────────────────────────────────────┐  │
│  │ Restaurant Documents (50 restaurants)                  │  │
│  │ • Text: Name, cuisine, location, description, amenities │  │
│  │ • Metadata: ID, price_range, rating, coordinates        │  │
│  │ • Embeddings: HuggingFace all-MiniLM-L6-v2 (FREE!)      │  │
│  └───────────────────────────────────────────────────────┘  │
│                                                             │
│  ┌───────────────────────────────────────────────────────┐  │
│  │ Embedding Model: HuggingFace all-MiniLM-L6-v2          │  │
│  │ • Dimension: 384                                       │  │
│  │ • FREE - No API key needed!                            │  │
│  │ • Runs locally on CPU                                  │  │
│  └───────────────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────┘
                       │
                       ▼
┌─────────────────────────────────────────────────────────────┐
│                  Generation Layer (Groq LLM)                 │
│                                                             │
│  ┌───────────────────────────────────────────────────────┐  │
│  │ LLM: Groq (llama-3.1-8b-instant)                       │  │
│  │ • Context: Retrieved restaurant documents              │  │
│  │ • Prompt: Custom template for restaurant recommendations│  │
│  │ • Output: Natural language response with matches        │  │
│  │ • Speed: 10x faster than OpenAI                         │  │
│  └───────────────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────┘
                       │
                       ▼
┌────────────────┐
│ Contextual     │  "Based on your criteria, I found 3 Italian
│ Response       │   restaurants in Downtown Dubai with outdoor
│ (Natural Lang) │   seating under AED 200: [restaurant details]"
└────────────────┘
```

3. Technology Stack & Rationale

Component	Choice	Rationale
Orchestration	**LangChain 1.0**	Industry standard for RAG; provides robust chains, retrievers, and LCEL (LangChain Expression Language) for modern composition.
Vector DB	**ChromaDB**	Lightweight and embedded. It requires no separate server, supports persistent storage, and offers efficient similarity search for the dataset size.
Embeddings	**HuggingFace** *(sentence-transformers)*	**100% Free & Local.** Runs on CPU without API keys. Provides 384-dimensional vectors that balance quality with speed.
LLM	**Groq** *(llama-3.1-8b)*	**Speed.** Inference is ~10x faster than OpenAI, which is critical for real-time search. The free tier is generous for development.

4. Data Pipeline Implementation

Step 1: Data Ingestion

- Parses `restaurant.json`.
- Converts raw data into LangChain `Documents`.
- Enriches text with metadata fields (Cuisine, Location, Price) for hybrid filtering.

Step 2: Embedding Generation

- **Model:** Local HuggingFace (`all-MiniLM-L6-v2`).
- **Process:** Converts text to vectors on the CPU.
- **Benefit:** Zero cost and full data privacy.

Step 3: Retrieval Strategy (Hybrid Search)

The system combines two search methods to ensure accuracy:

1. **Semantic Search:** Finds matches based on meaning (e.g., "cozy" matches description text).
2. **Metadata Filtering:** Filters based on hard constraints (e.g., "Cuisine = Italian").
3. **Thresholding:** Only returns results with a similarity score > 0.3 to reduce hallucinations.

Step 4: Generation

- Constructs a prompt with the retrieved context.
- Sends the prompt to Groq (Llama 3.1).
- Returns a natural language response formatted as a helpful concierge.

5. Edge Case Handling & Production Readiness

- **No Results Found:** System detects empty retrieval sets and returns a helpful fallback message ("I couldn't find exact matches, but here are some alternatives...").
- **API Reliability:** Includes error handling for Groq API timeouts or limits.
- **Cost Optimization:** The architecture incurs **$0 operational costs** for the embedding layer by using local models.
- **Scalability:** ChromaDB is configured for persistence, allowing the system to restart without rebuilding the index.

6. Future Enhancements

- **Agent Integration:** Connect RAG to LangGraph agents for multi-step reasoning (Task 1.2).
- **Caching:** Implement Redis to cache frequent queries (e.g., "Italian food") to reduce LLM latency.
- **Re-ranking:** Add a Cross-Encoder step to re-rank retrieved results for higher precision.