Technical Documentation

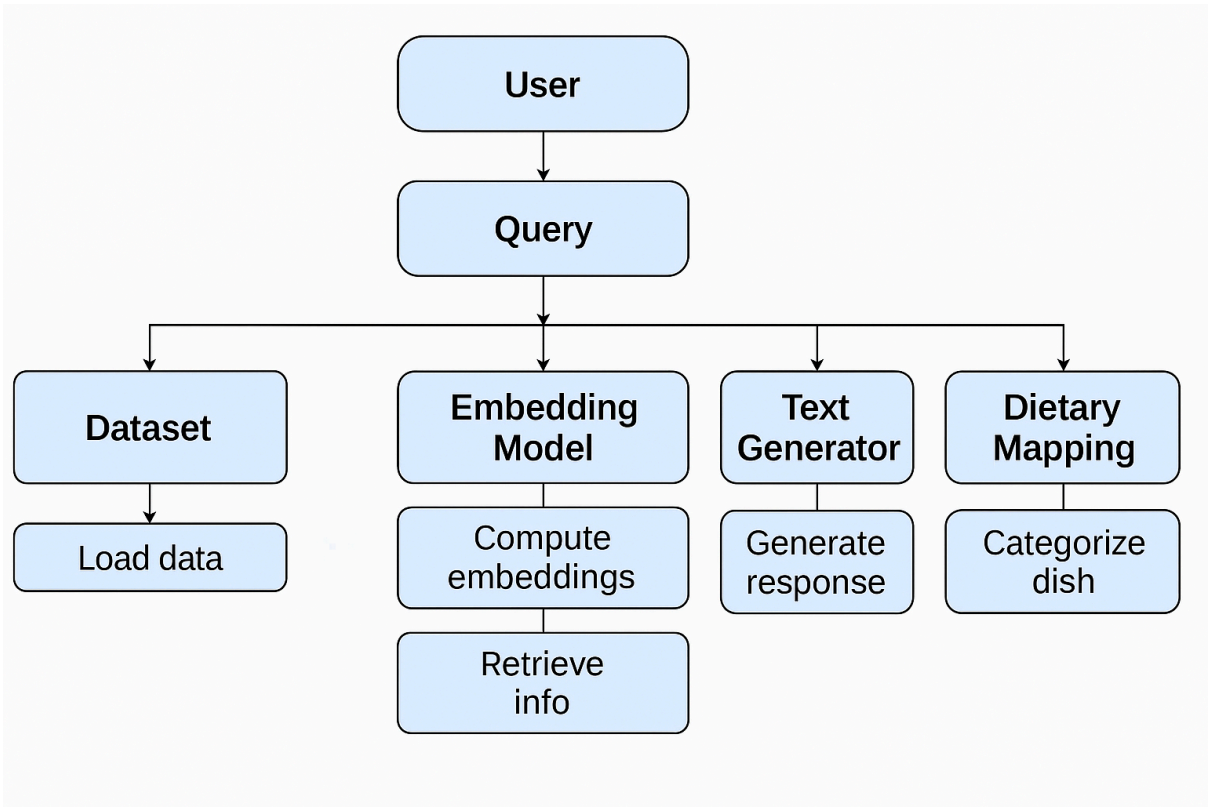
Name – Kritnandan

This documentation for the file zomato_chatbot.py

System Architecture

a. Streamlit Interface: Offers a user-friendly web UI for querying.
b. Rule-Based Query Handler: Manages well-defined queries such as contact information, dish availability, price comparisons, and dietary filtering through pattern matching and keyword logic.
c. Embedding and Vector Store (FAISS): Used to index and retrieve pertinent data for intricate searches.
d. LLM (OpenAI): Handles natural language answers to questions that rules can not answer.
e. Data Loader and Classifier: Loads menu data and classifies dishes as vegetarian or non-vegetarian using keyword-based confidence scoring.
f. JSON-based Dietary Rules: Supplies mappings for spice levels, preparation styles, allergen contents, and more.

Flowchart



---

1. User Inputs a Query

The user enters a natural language query like:

- *"Is the Chicken Tikka Masala available?"*
- *"Show me vegetarian options."*
- *"Compare prices of Margherita Pizza between restaurants."*

---

2. Query Handling Pipeline

a. **Query Classification**

- The system classifies the query into types like:
  - Contact Info
  - Dish Availability
  - Dietary Restrictions
  - Veg/Non-Veg Classification
  - Menu Listing
  - Price Comparison
  - Out-of-Scope

b. Knowledge Retrieval (RAG)

If the query needs external data (like a menu or dietary info), the RAG module retrieves relevant chunks from the database.

c. Intent-Specific Handler

Each query type is handled by a specific function:

- E.g., `handle_contact_info(query)`, `handle_price_comparison(query)`, etc.

---

3. Veg/Non-Veg Classification Subprocess

- A dish or restaurant is classified using:
  - Ingredient keywords

    - Confidence score
    - Optional manual overrides

---

4. Data Sources

- CSV files for restaurant menus
- JSON mapping for:
    - Allergens
    - Spice levels
    - Dietary types
    - Combo types
    - Pizza base
    - Preparation style

---

5. Response Generation

- The system combines the processed data into a natural language response and returns it to the user.

Design Decisions

- Hybrid Approach: Rule-based logic gives deterministic control and faster performance, while RAG provides flexibility for unhandled queries.
- Keyword Classification: Simple and explainable veg/non-veg logic based on common terms like "paneer", "chicken", and mapping the data, etc.
- Streamlit UI: Chosen for rapid prototyping and ease of use.
- JSON Rules: Decouples logic from code, making dietary mappings easy to extend.

Challenges Faced

- Ambiguous Dish Names: Some dishes (e.g., "Veg Biryani") contain terms that could be misclassified.
- Easing case handling is the most challenging part of the assignment.
- Diverse Query Wording: Variations in how users ask the same thing ("Is X available?", "Can I get X?").
- Missing Metadata: Scraped data may lack allergens or preparation methods.

Solutions Implemented

- Added confidence scores and fallback classification for Veg/Non-Veg.

- Used regex and normalisation to identify multiple query phrasings.
- Augmented scraped data with JSON rule-based enhancements.
- Structured query routing to separate rule-based and RAG logic.

Future Improvements

- Replace keyword-based classification with ML-based classifiers.
- Expand dietary mapping to include nutrition data.
- Enable real-time scraping or API-based updates.
- Add personalisation (e.g., user profiles for allergies).
- Improve the Streamlit UI with filtering options and query history.
- Create a chatbot-based website and host it.
