Chefmate AI

Backend API Documentation

1 Overview

The Chefmate AI backend API enables intelligent food-related conversations by integrating Retrieval-Augmented Generation (RAG) and a local Mistral 7B language model. It dynamically retrieves relevant recipe data and streams personalized responses to user prompts.

2 POST `/chat`

Description

Streams an AI-generated response based on user input, retrieved context (recipes), and chat history using a local LLM.

Request Details

- **Method:** `POST`
- **Endpoint:** `/chat`
- **Content-Type:** `application/json`

Request Body

```
{
  "chat_history": [
    {"role": "user", "content": "What can I cook with flour, eggs,
       salt, onion and garlic"}
  ],
  "top_k": 3
}
```

Listing 1: Example JSON Request

Parameters:

- `chat_history`: **(Required)** Array of messages with fields:
  - `role`: "user" or "assistant"
  - `content`: Text of the message
- `top_k`: **(Optional)** Integer number of relevant recipes to retrieve (default: 3)

Response

- **Content-Type:** `text/plain`
- **Encoding:** Streamed token-by-token
- **Format:** Markdown-formatted plain text

Listing 2: Example Streamed Response

```
Sure! Here are some recipes you can make with flour, eggs, salt, onion,
and garlic:

**1. Onion-Garlic Omelette**
- **Ingredients:** eggs, onion, garlic, salt
- **Instructions:**
  1. Beat eggs with salt.
  2. Saute chopped onion and garlic until golden.
  3. Pour eggs in and cook until firm.
```

Error Responses

- `400 Bad Request`: Missing or malformed `chat_history`
- `500 Internal Server Error`: Issues during embedding, retrieval, or model generation

Example cURL

Listing 3: Example Request Using cURL

```
curl -X POST http://localhost:8000/chat/ \
  -H "Content-Type: application/json" \
  --data-raw '{
    "chat_history": [
      {"role": "user", "content": "What can I cook with flour, eggs,
          salt, onion and garlic"}
    ]
  }'
```

3 Pipeline Flow

1. **Intent Detection:** Classifies the latest user message.
2. **Semantic Retrieval:** Uses FAISS to embed and retrieve top-`k` relevant recipes.
3. **Prompt Construction:** Combines retrieved context and chat history to build an LLM prompt.
4. **LLM Streaming:** Streams responses from the Mistral 7B model in Markdown format.

4 System Architecture Diagram

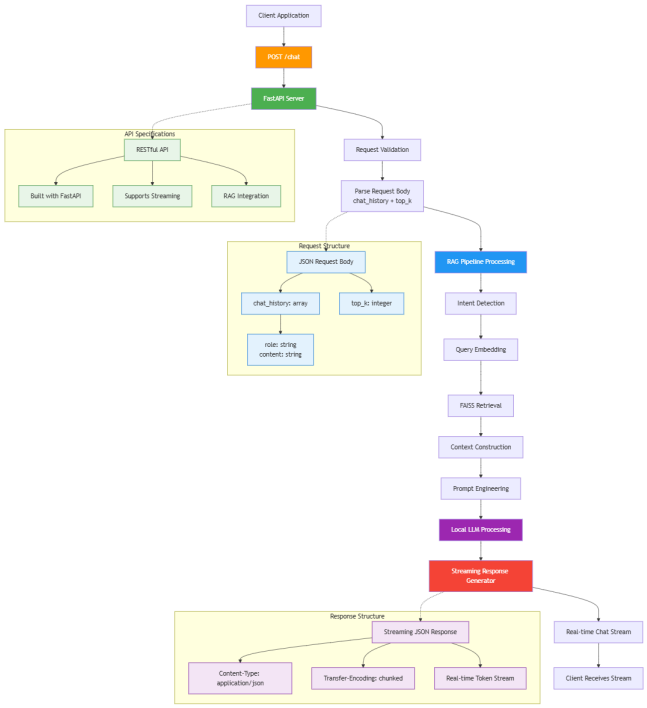

Figure 1: Chefmate API Request Processing Flow

5 Source Files

- `app/api/chat.py` – Chat endpoint implementation
- `app/utils/prompt.py` – Builds contextual prompts
- `app/utils/llm_model.py` – Handles LLM response streaming
- `app/utils/faiss.py` – Embedding and FAISS-based recipe retrieval
- `app/utils/intent.py` – Intent classification logic