Weather Chatbot System Documentation

Overview

This document provides a detailed overview of the Weather Chatbot system built using FastAPI. It outlines the architecture, API endpoints, setup instructions, and provides suggestions for future enhancements. The main goal of this chatbot is to provide weather-related information via a conversational interface.

1. System Architecture

The system consists of several components, each responsible for specific tasks:

1. FastAPI Application: This serves as the main web application framework.

2. Database Manager: Manages SQLite database operations for storing chat queries and responses.

3. Chat Router: Handles HTTP requests related to chat functionality.

4. Weather Service: Interacts with third-party APIs (OpenWeather) to fetch weather data.

5. Geolocation Service: Uses OpenCage Geocode API to resolve location names into latitude and longitude.

6. Query Processor: Utilizes a language model for processing general queries.

2. API Endpoints

- Base URL: `http://<host>:8000/api`
- Endpoints: `POST /chat`
- Description: Processes incoming chat queries and returns weather information.
- Request Body:

```
{
  "user_query": "What's the current weather in New York?"
}
```

- Response:

```
{
  "What's the current weather in New York?": "The current weather in New York is clear sky with a temperature of 18°C."
}
```

- Errors: 500 Internal Server Error: When the query processing fails.

3. Setup Instructions

Follow these steps to set up the FastAPI Weather Chatbot:

1. Environment Setup

   - Ensure you have Python 3.8 or later installed.
   - Clone the repository.
   - Navigate to the project directory.

2. Install Dependencies

```
pip install -r requirements.txt
```

3. Environment Variables

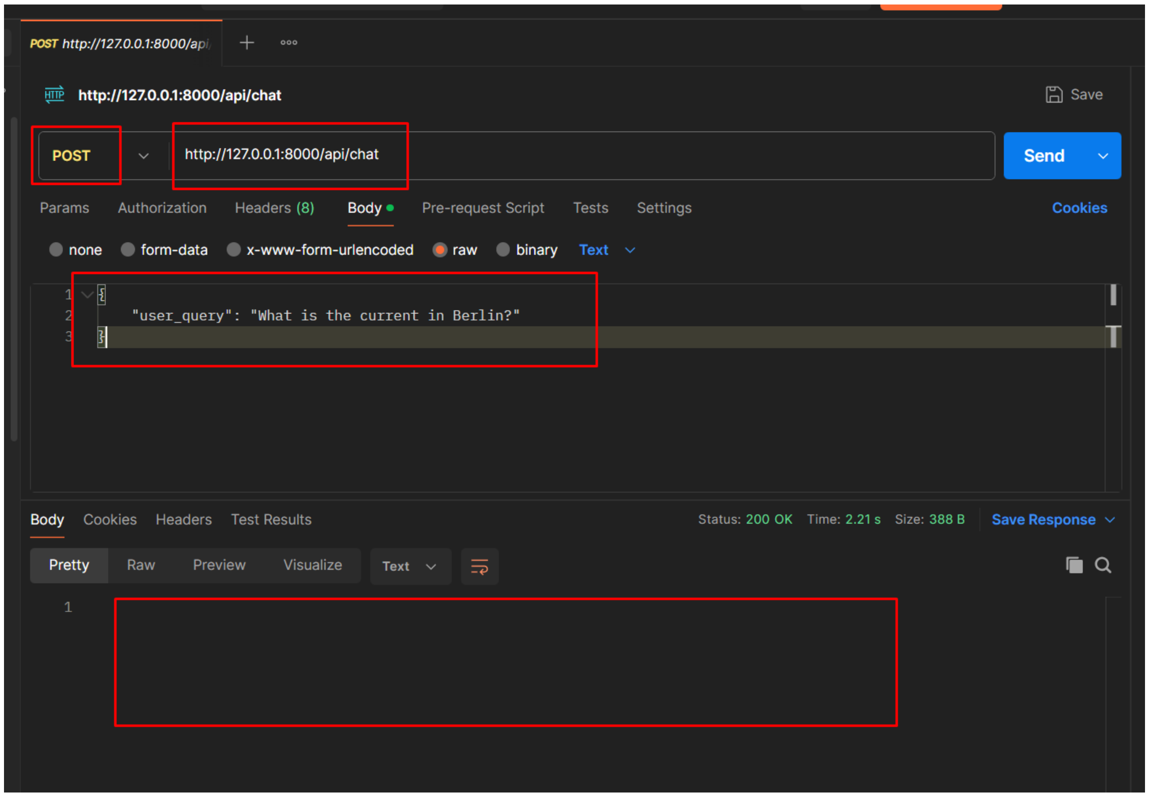
   - Create a `.env` file in the root directory.
   - Define the following variables:

```
OPENWEATHER_API_KEY=<your_openweather_api_key>
OPENCAGE_API_KEY=<your_opencage_api_key>
LANGSMITH_TRACING=true
LANGSMITH_ENDPOINT=https://api.smith.langchain.com
LANGSMITH_API_KEY=<your_langsmith_api_key>
LANGSMITH_PROJECT=<your_langsmith_project_name>
OPENAI_API_KEY=<your_openai_api_key>
CURRENT_WEATHER_URL=https://api.openweathermap.org/data/2.5/weather
FORECAST_WEATHER_URL=https://pro.openweathermap.org/data/2.5/forecast/hourly
HISTORY_WEATHER_URL=https://history.openweathermap.org/data/2.5/history/city
```

4. Run the Application

```
python main.py
```

5. Testing with Postman Collection



POST http://127.0.0.1:8000/api/chat

Body: raw, Text

```
{
    "user_query": "What is the current in Berlin?"
}
```

Status: 200 OK Time: 2.21 s Size: 388 B

4. Assumptions & Design Decisions

- SQLite Database: Chosen for its simplicity and ease of setup. Suitable for light-to-moderate workloads.

- Asynchronous Calls: Async features used in FastAPI to improve performance under concurrent request loads.

- Third-Party Libraries:

  - `requests`: For HTTP requests.

  - `pydantic`: For data validation.

  - `langchain_openai`: For natural language query processing.

  - `opencage.geocoder`: For geolocation services.

5. Proposed Innovative Features

1. Natural Language Processing Enhancements

   - Use advanced NLP models like GPT for more complex query understanding.

2. Voice Interaction

   - Integrate voice recognition to allow users to interact with the chatbot via speech.

3. Advanced Analytics

- Implement analytics to track user interactions, improving response accuracy over time.

4. Scalability Improvements

- Transition to a cloud-based database and use containerization (Docker) for easier scalability.

5. Security Enhancements

- Implement JWT authentication for secure access control.

- Use HTTPS through a reverse proxy for encrypted communication.

These enhancements aim to increase the utility, accessibility, and robustness of the AI-powered weather chatbot system.