EIEI GENERIC MODULAR AI RESPONSE API

The idea is that a webservice that given a text prompt/audio/image and some other params like channel/user\_id/chat\_id for logging puorpuse will reply in a chatbot manner.

The key task of this script will be to have modular ways to process each type of file

And the avility to easyly swap api calls to the other services so that for example if needed to change the provider or audio transcriptions, or the model of the llm used to respond it can be done just changing a few params in the global env file.

**Application Workflow Documentation**

**Overview**

This document describes the workflow of a messaging application integrating with a generic AI service. The application uses a webhook to receive messages and employs a series of scripts and database interactions to process and respond to user inputs.

**Workflow Steps**

**Message Reception**

1. **Bot Receives Message via Webhook to API**: The application's bot is configured to receive new messages from the messaging platform (Telegram) via a webhook that pushes the message data to the application's API.

**Message Processing**

1. **Telegram Message Script Converter to Generic AI Script**: Upon reception, the message is passed through a conversion script which translates the Telegram-specific message format into a generic format compatible with the AI service.
2. **Generic AI Python Script**: The converted message is then processed by a generic AI Python script, which has the capability to handle different types of inputs and perform various actions based on the input type.

**Input Type Handling**

1. **Text**: If the input is text, it is sent directly to the database for storage.
2. **Audio**: If the input is audio, the script transcribes the audio to text. And again stores it

**Image**: If the input is an image, the script generates a description and converts it to text. And again stores it

**Database Operations**

1. **Send Plain Text to DB**: The plain text, either received directly or derived from audio/image, is stored in the database.
2. **Store Type and Text/Transcript/Description in DB**: The type of input (text, audio, image) and the corresponding text or metadata is stored in the database for record-keeping and further processing.

**Outgoing Message Processing**

1. **Procedure Checks New Messages in DB Every 5 Seconds**: A background procedure runs at a 5-second interval to check for new messages in the database that exceed a 3-second threshold from the last message.
2. **Creates JSON Payload to be Sent**: Once a new message is detected, a

JSON payload is created, which includes information such as chat history, the latest message, the master prompt, the model name, and other settings necessary for the AI service.

1. **Receives Payload and Resend to OpenRouter AI API**: This payload is sent to the OpenRouter llm AI API, which is responsible for determining the appropriate response based on the input and the context provided.

**Response Generation and Dispatch**

1. **Gets Response from AI, Sends to Telegram**: The OpenRouter AI API processes the request and generates a response. This response is then sent back to the messaging platform (Telegram) to be delivered to the user.
2. **Telegram Message Script Converter from Generic AI Script**: Before sending the response to the user, the AI's response is converted from the generic AI format back into the Telegram-specific format using a script converter.
3. **Bot Replies Message to User**: Finally, the bot sends the converted message as a reply to the user on Telegram, completing the cycle of receiving, processing, and responding to user input.

Database type: postgrest

Data model:

1. Message Table

tbl\_200\_messages (

pk\_messages\_ID NUMBER NOT NULL,

channel VARCHAR2(100),

bot\_id NUMBER NOT NULL,

chat\_id NUMBER,

type VARCHAR2(100),

role VARCHAR2(100),

content\_text VARCHAR2(4000),

file\_id VARCHAR2(4000),

message\_timestamp TIMESTAMP,

update\_id NUMBER,

message\_id NUMBER,

is\_processed VARCHAR2(1),

created\_by VARCHAR2(1000),

created\_on DATE,

updated\_by VARCHAR2(1000),

updated\_on DATE

);

pk\_ messages\_ID -> will cointain a primary key that will be sequential

channel -> Where is the message coming from (“TELEGRAM”,”WHATSAPP”,”WEBAPP” ETC)

BOT\_ID -> bOT ID Coming from the channel system i.e.:Telegram Bot ID

chat\_id -> bOT ID Coming from the channel system i.e.:Telegram Chat ID

type -> Type of original message (AUDIO/TEXT/IMAGE)

role -> Who is the original creator of the record (HUMAN=USER / BOT= ASSISTANT)

content\_text -> The reply-msg/transcription of the audio/description of image

file\_id -> file id Coming from the channel system i.e.:Telegram Chat ID

message\_timestamp -> timestamp of when the message was received/sent

update\_ -> related to telegram id

message\_-> related to telegram id

is\_processed -> is the message being replied to?

created\_by ,created\_on , updated\_by , updated\_on -> this are audit fields filled automatically by the system

Stack:

FASTAPI

POSTGREST

Running on Render.com

Workflow inmage:

A diagram of a software program

Description automatically generated