Detailed Documentation for GPT Web Application

# Overview

gpt is a web application designed to manage interactions between a CRM system ,ChatGpt and Superchat using various APIs. This documentation provides an in-depth guide to the application, covering setup, configuration, and code functionalities.

# Project Structure

- package.json  
- .env  
- .gitignore  
- fetch.js  
- generateZohoToken.js  
- gptFilter.js  
- index.js  
- sendMessagefromAttribute.js  
- superchatFunctions.js  
- zohoFunctions.js

# File Descriptions and Functionalities

## 1. package.json

This file contains metadata about the project and its dependencies.

{  
 "name": "gpt",  
 "version": "1.0.0",  
 "description": "this is the first version if the easy menu web app",  
 "main": "index.js",  
 "type": "module",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1",  
 "start": "nodemon index.js"  
 },  
 "author": "Bassem",  
 "license": "ISC",  
 "dependencies": {  
 "@ngrok/ngrok": "^1.3.0",  
 "@zohocrm/nodejs-sdk-6.0": "^2.0.0",  
 "axios": "^1.7.2",  
 "body-parser": "^1.20.2",  
 "dirname": "^0.1.0",  
 "dotenv": "^16.4.5",  
 "express": "^4.19.2",  
 "openai": "^4.49.1"  
 },  
 "devDependencies": {  
 "nodemon": "^3.1.2"  
 }  
}

## 2. .env

Contains environment variables necessary for the application to run. They are in the digtalocean or with Bassem or Robert. The env should keep stored in a save way.

Example:  
CLIENT\_ID=your\_client\_id  
CLIENT\_SECRET=your\_client\_secret  
REDIRECT\_URI=your\_redirect\_uri  
REFRESH\_TOKEN=your\_refresh\_token  
ZOHO\_OAUTH\_TOKEN=your\_zoho\_oauth\_token  
SUPERCHAT\_API\_KEY=your\_superchat\_api\_key  
SUPERCHATCHANNEL\_ID=your\_superchat\_channel\_id  
OPENAI\_API\_KEY=your\_openai\_api\_key  
OPENAI\_ASSISTANT=your\_openai\_assistant  
NGROK\_TOKEN=your\_ngrok\_token  
GPT\_ATTRIBUTE=your\_gpt\_attribute  
GPT\_LEBEL=your\_gpt\_label  
PORT=3000

## 3. .gitignore

Specifies files and directories to be ignored by Git.

Example:  
node\_modules  
.env

## 4. fetch.js

Handles interactions with the OpenAI API and Zoho CRM. It check if the customer who sent the message has already a thread id in zoho leads, if it has it, then it take it the process it in the same thread and send the message to Superchat.

import axios from 'axios';  
import dotenv from 'dotenv';  
import { generateZohoOauthToken } from './generateZohoToken.js';  
import OpenAI from "openai";  
import { updateRecord, createRecord } from './zohoFunctions.js';  
import { getSuperchatRecord, sendMessage } from './superchatFunctions.js';  
  
dotenv.config();  
  
const openai = new OpenAI();  
const OPENAI\_API\_URL = 'https://api.openai.com/v1/';  
const ZOHO\_CRM\_API\_URL = 'https://www.zohoapis.eu/crm/v6/';  
const OPENAI\_API\_KEY = process.env.OPENAI\_API\_KEY;  
const OPENAI\_ASSISTANT = process.env.OPENAI\_ASSISTANT;  
const SUPERCHAT\_API\_KEY = process.env.SUPERCHAT\_API\_KEY;  
  
const headers = {  
 'Authorization': `Bearer ${OPENAI\_API\_KEY}`,  
 'Content-Type': 'application/json',  
 'OpenAI-Beta': 'assistants=v2'  
};  
  
export async function call\_in\_OpenAi(mg, phone, superchat\_contact\_id, checker) {  
 // Function implementation  
}  
  
export async function putMessageInThreadAssistant(template\_id, quickReplayBody, phone) {  
 // Function implementation  
}  
  
async function getContentTemplateFromSuperchat(template\_id) {  
 // Function implementation  
}

## 5. generateZohoToken.js

Generates and refreshes Zoho OAuth tokens.

import axios from 'axios';  
import dotenv from 'dotenv';  
  
dotenv.config();  
  
const ZOHO\_TOKEN\_URL = 'https://accounts.zoho.eu/oauth/v2/token';  
const CLIENT\_ID = process.env.CLIENT\_ID;  
const CLIENT\_SECRET = process.env.CLIENT\_SECRET;  
const REDIRECT\_URI = process.env.REDIRECT\_URI;  
const REFRESH\_TOKEN = process.env.REFRESH\_TOKEN;  
  
export async function generateZohoOauthToken() {  
 try {  
 const response = await axios.post(ZOHO\_TOKEN\_URL, null, {  
 params: {  
 client\_id: CLIENT\_ID,  
 client\_secret: CLIENT\_SECRET,  
 redirect\_uri: REDIRECT\_URI,  
 refresh\_token: REFRESH\_TOKEN,  
 grant\_type: 'refresh\_token'  
 },  
 headers: {  
 'Content-Type': 'application/x-www-form-urlencoded'  
 }  
 });  
  
 const { access\_token, expires\_in } = response.data;  
 process.env.ZOHO\_OAUTH\_TOKEN = access\_token;  
  
 return access\_token;  
 } catch (error) {  
 console.error('Error generating Zoho OAuth token:', error.response ? error.response.data : error.message);  
 throw error;  
 }  
}

## 6. gptFilter.js

Filters and processes messages based on certain labels and attributes. If client hat one of the lebels , it should just store the message in the thread and do nothing.

import { getSuperchatRecord, getSuperchatConveration } from './superchatFunctions.js';  
import { config } from 'dotenv';  
import { call\_in\_OpenAi } from './fetch.js';  
import { putMessageInThreadAssistant } from './fetch.js'  
  
config();  
  
const blockingLebels = [  
 process.env.BLOCK\_AI\_LEBEL,  
 process.env.VIP\_KUNDE\_LEBEL,  
 process.env.BESTANDSKUNDE\_LEBEL,  
];  
const gbt\_attribute\_id = process.env.GPT\_ATTRIBUTE;  
const gpt\_lebel = process.env.GPT\_LEBEL;  
  
export async function runGpt(contact\_id, mg , phone) {  
 // Function implementation  
}

## 7. index.js

Main entry point of the application. Sets up Express server and handles routes. It start with create a list of client, every client has more information like the client id, phone…. After a message arrives and it wait 20 second to process all the message those came in 20 second together. If a client has a blocker lebel then it should watch the out bound messages if a message is arrive, that’s mean the automation has replayed.

import express from 'express';  
import path from 'path';  
import { fileURLToPath } from 'url';  
import { dirname } from 'path';  
import bodyParser from 'body-parser';  
import { config } from 'dotenv';  
import { call\_in\_OpenAi, putMessageInThreadAssistant } from './fetch.js';  
import { runGpt } from './gptFilter.js';  
import { runThreadAndSend } from './sendMessagefromAttribute.js';  
  
config();  
  
const \_\_filename = fileURLToPath(import.meta.url);  
const \_\_dirname = dirname(\_\_filename);  
  
const app = express();  
const PORT = process.env.PORT || 3000;  
const NGROK\_AUTHTOKEN = process.env.NGROK\_TOKEN;  
  
app.use(bodyParser.json());   
app.use(bodyParser.urlencoded({ extended: true }));  
app.use(express.static(path.join(\_\_dirname, 'public')));  
  
app.get('/', (req, res) => {  
 res.send('Hello World!');  
});  
  
let userInfo = {}; // To store user messages and timeouts  
  
app.post('/webhook', (req, res) => {  
 // Route implementation  
});  
  
app.post('/outboundWebhook', (req, res) => {  
 // Route implementation  
});  
  
app.post('/runchatgpt', (req, res) => {  
 // Route implementation  
});  
  
app.listen(PORT, () => {  
 console.log('Server is running on http://localhost:' + PORT);  
});

## 8. sendMessagefromAttribute.js

Handles sending messages based on specific attributes. If the gpt has been updated to 1, then it run the thread and send the message to the client.

import { config } from 'dotenv';  
import { generateZohoOauthToken } from './generateZohoToken.js';  
import OpenAI from "openai";  
import axios from 'axios';  
import { sendMessage } from './superchatFunctions.js';  
  
config();  
  
export async function runThreadAndSend(contact) {  
 // Function implementation  
}

## 9. superchatFunctions.js

Functions to interact with the Superchat API.

import { config } from 'dotenv';  
  
config();  
  
const SUPERCHAT\_API\_KEY = process.env.SUPERCHAT\_API\_KEY;  
const SUPERCHATCHANNEL\_ID = process.env.SUPERCHATCHANNEL\_ID;  
  
export async function getSuperchatRecord(contact\_id) {  
 // Function implementation  
}  
  
export async function sendMessage(message, contact\_id) {  
 // Function implementation  
}

export async function getSuperchatConveration(contact\_id) {  
 // Function implementation  
}

## 10. zohoFunctions.js

Functions to interact with the Zoho CRM.

import \* as ZOHOCRMSDK from "@zohocrm/nodejs-sdk-6.0";  
import { config } from 'dotenv';  
import { getSuperchatRecord } from './superchatFunctions.js';  
  
config();  
  
const client\_id = process.env.CLIENT\_ID;  
const client\_secret = process.env.CLIENT\_SECRET;  
const redirect\_url = process.env.REDIRECT\_URI;  
const refresh\_token = process.env.REFRESH\_TOKEN;  
  
export async function initializeZohoCRM() {  
 // Function implementation  
}  
  
export async function updateRecord(recordId, thread\_id) {  
 // Function implementation  
}  
  
export async function createRecord(phone, thread\_id, superchat\_contact\_id) {  
 // Function implementation  
}

# Setting Up the Application

## Prerequisites

1. Node.js and npm installed on your system.

2. Necessary environment variables configured in a `.env` file.

## Installation

1. Clone the repository.

2. Navigate to the project directory.

3. Run the following command to install dependencies:

```  
npm install  
```

4. if you want to run the program locally you need to set up ngrok. And add the webhook to superchat.

## Running the Application

1. Start the server using the following command:

```  
npm start  
```

The server will start on the specified port (default is 8080).

## Hosting

The program is hosted in DigtalOcean with Abrahairadabra gmail account, and connected to the digital-ecosystem github account, pushing to the main branch will update the program in the server.

# API Endpoints

## Webhook Endpoint

\*\*URL:\*\* `/webhook`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Handles incoming messages and processes them.

## Outbound Webhook Endpoint

\*\*URL:\*\* `/outboundWebhook`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Handles outbound messages.

## Run ChatGPT Endpoint

\*\*URL:\*\* `/runchatgpt`

\*\*Method:\*\* `POST`

\*\*Description:\*\* Runs ChatGPT for specific contacts based on attributes “GPT”.

# Key Functions

## Generating Zoho OAuth Token

\*\*File:\*\* `generateZohoToken.js`

\*\*Function:\*\* `generateZohoOauthToken`

Generates and refreshes the Zoho OAuth token for API authentication.

## Fetch Functions

\*\*File:\*\* `fetch.js`

\*\*Functions:\*\*

- `call\_in\_OpenAi`

- `putMessageInThreadAssistant`

- `getContentTemplateFromSuperchat`

Handles interactions with OpenAI and Zoho CRM.

## GPT Filter

\*\*File:\*\* `gptFilter.js`

\*\*Function:\*\* `runGpt`

Filters messages based on specific labels and attributes.

## Superchat Functions

\*\*File:\*\* `superchatFunctions.js`

\*\*Functions:\*\*

- `getSuperchatRecord`

- `sendMessage`

- `getSuperchatConveration`

Handles interactions with the Superchat API.

## Zoho Functions

\*\*File:\*\* `zohoFunctions.js`

\*\*Functions:\*\*

- `initializeZohoCRM`

- `updateRecord`

- `createRecord`

Handles interactions with Zoho CRM.

# Conclusion

This documentation provides a comprehensive guide to understanding and working with the Abrahairadabra Gpt web application. Each file and function has been described in detail to assist developers in maintaining and extending the application.