**### ChatBot using OpenAI API ###**

*# Introduction:*

This chatbot uses OpenAI API which is based on the cutting-edge GPT-3.5 architecture, which enables it to generate human-like responses to a wide variety of questions and prompts.

The bot is designed to interact with users through a chat interface and provide assistance in various areas, such as customer support, e-commerce, and more. The target audience for this bot is anyone who interacts with the website or app where the bot is deployed.

The capabilities of these bot include answering questions, generating text, completing sentences, and even creating original content like stories or poetry. It is designed to be flexible and adaptable, so It can adjust its responses to suit the user's needs and preferences.

*# Requirements:*

1.Web Browser: The bot requires a web browser that supports ReactJS, such as Google Chrome, Mozilla Firefox, or Safari.

2. Node.js: This is a JavaScript runtime environment that allows you to run JavaScript code outside of a web browser. React is built on top of Node.js, so you'll need to install it on your machine from https://nodejs.org/.

3. Package manager: npm or yarn. npm is included with Node.js, while yarn is an alternative package manager that provides some additional features.

4. Vite (a frontend tool that is used for building fast and optimized web applications).

5. Chatscope ui-kit for the chatbot User Interface.

6. An OpenAI API key.

*# Architecture:*

The bot's architecture consists of two main components: the ReactJS front-end and the OpenAI API back-end. The front-end is responsible for rendering the chat interface, handling user input, and sending requests to the OpenAI API. The back-end processes the requests from the front-end and generates responses based on the input data.

*# Design:*

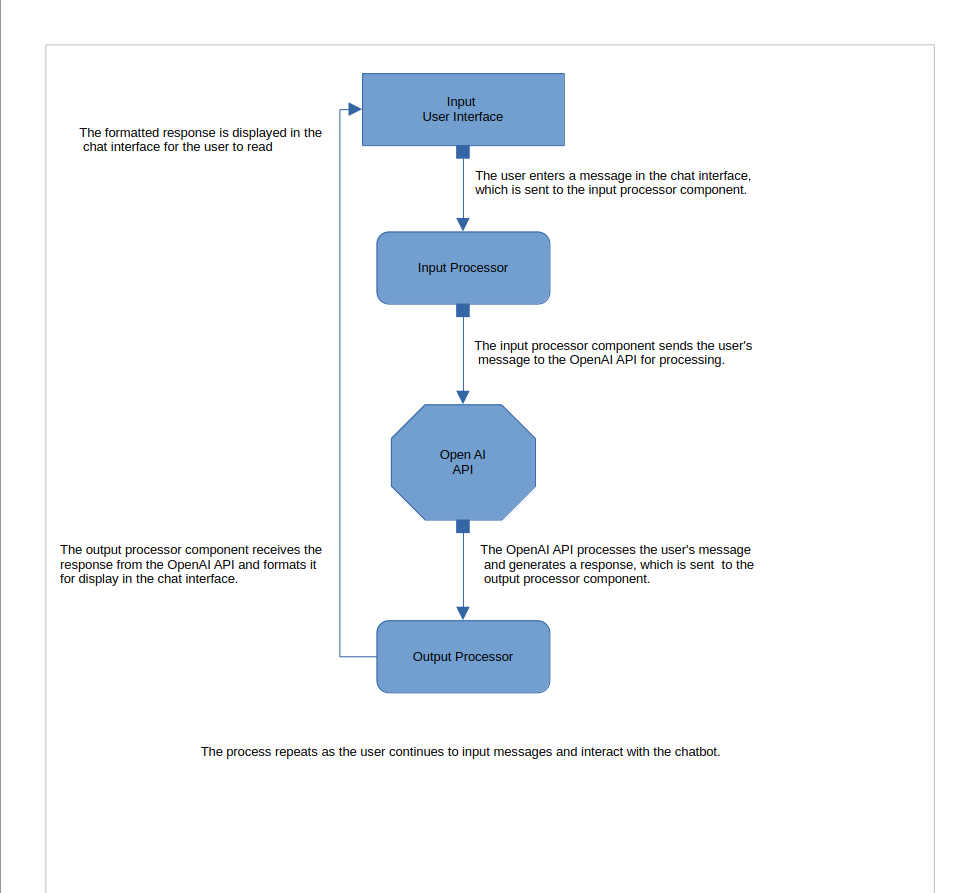
1. User Interface: The user interface component of the chatbot includes the chat interface where users can input text messages and send them to the bot.

2. Input Processor: The input processor component receives the user's input from the user interface and processes it by sending it to the OpenAI API for analysis and processing.

3. OpenAI API: The OpenAI API component processes the user's input data and generates an appropriate response, which is sent back to the input processor component.

4. Output Processor: The output processor component receives the response from the OpenAI API and processes it by formatting it and displaying it in the chat interface.

5. User: The user component represents the end user who interacts with the chatbot through the user interface component.



*#Implementation:*

*App.jsx*

import { useState } from "react";

import "./App.css";

import "@chatscope/chat-ui-kit-styles/dist/default/styles.min.css";

import {

MainContainer,

ChatContainer,

MessageList,

Message,

MessageInput,

TypingIndicator,

} from "@chatscope/chat-ui-kit-react";

const API\_KEY = "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";

const systemMessage = {

role: "system",

content: "Explain things like you're talking to a software professional.",

};

function App() {

const [messages, setMessages] = useState([

{

message: "Hello, I'm ChatGPT! Ask me anything!",

sentTime: "just now",

sender: "ChatGPT",

},

]);

const [isTyping, setIsTyping] = useState(false);

const handleSend = async (message) => {

const newMessage = {

message,

direction: "outgoing",

sender: "user",

};

const newMessages = [...messages, newMessage]; // All the previous messages + the new messages

setMessages(newMessages);

//set typing indicator

setIsTyping(true);

//process message to chatgpt (send it over and see the response)

await processMessageToChatGPT(newMessages);//

};

// Formatting messages for ChatGPT API

async function processMessageToChatGPT(chatMessages) {

let apiMessages = chatMessages.map((messageObject) => {

let role = "";

if (messageObject.sender === "ChatGPT") {

role = "assistant";

} else {

role = "user";

}

return { role: role, content: messageObject.message };

});

const apiRequestBody = {

model: "gpt-3.5-turbo",

messages: [systemMessage, ...apiMessages],

};

await fetch("https://api.openai.com/v1/chat/completions", {

method: "POST",

headers: {

Authorization: "Bearer " + API\_KEY,

"Content-Type": "application/json",

},

body: JSON.stringify(apiRequestBody),

})

.then((data) => {

return data.json();

})

.then((data) => {

console.log(data);

setMessages([

...chatMessages,

{

message: data.choices[0].message.content,

sender: "ChatGPT",

},

]);

setIsTyping(false);

});

}

return (

<div className="App">

<div style={{ position: "relative", height: "800px", width: "700px" }}>

<MainContainer>

<ChatContainer>

<MessageList

scrollBehavior="smooth"

typingIndicator={

isTyping ? (

<TypingIndicator content="ChatGPT is typing" />

) : null

}

>

{messages.map((message, i) => {

console.log(message);

return <Message key={i} model={message} />;

})}

</MessageList>

<MessageInput placeholder="Type message here" onSend={handleSend} />

</ChatContainer>

</MainContainer>

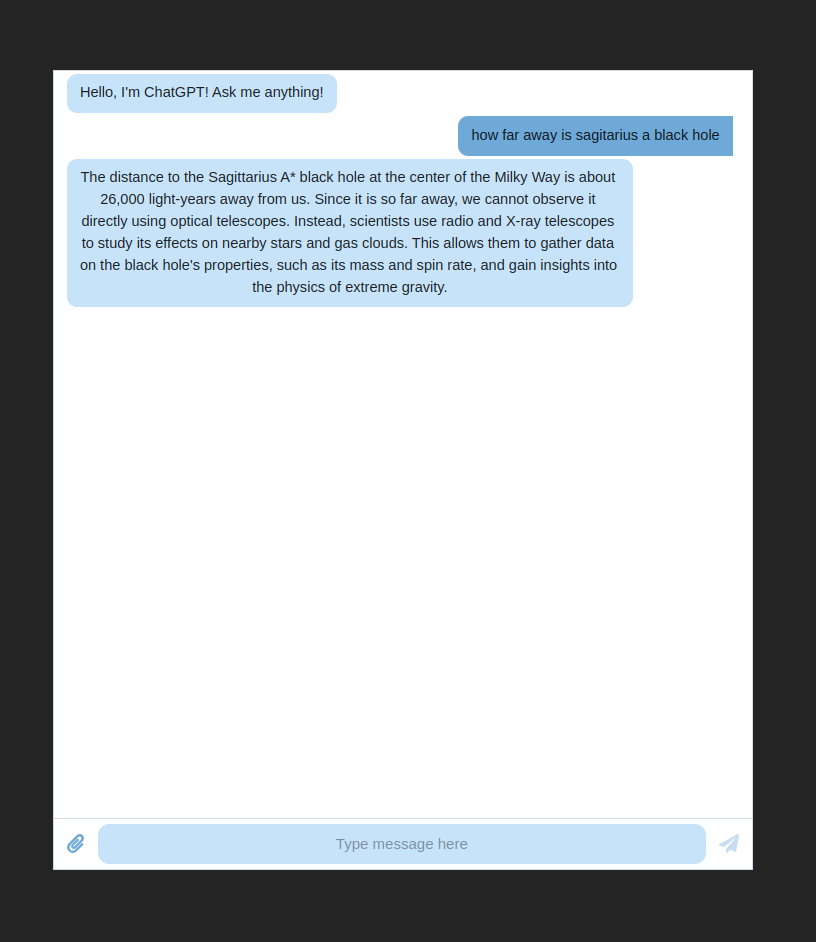
</div>

</div>

);

}

export default App;

**

*# Testing:*

The bot was tested using various testing methodologies, such as unit testing, integration testing, and end-to-end testing. The tests were performed to ensure that the bot functioned properly, responded appropriately to user input, and did not have any bugs or errors.

*# Maintenance:*

The bot will require maintenance to ensure that it continues to function properly. This may include updating the code to fix any bugs or errors, updating Node (npm), adding new features, or updating the OpenAI API key if necessary.

*# User guide*:

1. You'll need to install nodejs it on your machine from https://nodejs.org/.

2. Package manager: npm or yarn. npm is included with Node.js, while yarn is an alternative package manager that provides some additional features.

3. You will also need to install vite (a frontend tool that is used for building fast and optimized web applications).

Open your terminal or command prompt and run the following command to install Vite globally:

npm install -g vite

Once Vite is installed, create a new directory for your app and navigate to it in your terminal:

mkdir my-vite-app

cd my-vite-app

Run the following command to initialize a new Vite app in your current directory:

vite init

This command will prompt you to choose a framework and a template for your app.

You can choose from several options, including React, Vue.js, and vanilla JavaScript.

Once you've selected your framework and template, Vite will generate a basic project

structure for you. Navigate to the new directory that was created, and run the

following command to install the dependencies:

npm install

4. You will also need the chatscope ui-kit for the chatbot User Interface.

For that go into you app directory and run:

npm install @chatscope/chat-ui-kit-react

5. After that you will also need an OpenAI API key. In order to do that follow these steps:

Go to the OpenAI website and sign up for an account at https://openai.com/signup/.

Once you have signed up and logged in, navigate to the "API" section of your account dashboard.

Click the "Create API Key" button to generate a new API key.

Choose a name for your API key and select the API model that you want to use. OpenAI currently offers several models, including GPT-3 and Codex.

Review and agree to the API terms of service, then click the "Create API Key" button to generate your key.

Your new API key will be displayed on the screen. Make sure to copy and save it in a secure location, as you will need it to access the OpenAI API.

6. After you have done all the things mentioned above copy my App.jsx file in your src folder replace it with you existing file.

7. In the App.jsx file go to line no. 14 and there will be a **const API\_KEY** variable in this variable remove my key and paste your key that you have generated in the 5th step.

8. Finally, start the development server by running the following command:

npm run dev

This will start the development server, and your app will be accessible at [http://localhost:5173](http://localhost:5173/).

*# Conclusion:*

In conclusion, this chatbot built with ReactJS and integrated with OpenAI API provides a valuable tool for interacting with website or app visitors. Future enhancements or improvements could include expanding the bot's functionality to support additional use cases, improving the accuracy of the bot's responses, or integrating with other APIs or tools.