**Open AI API Documentation:**

OpenAI API is a general-purpose text-in, text-out interface that empowers developers to experiment with various English language tasks. It's a powerful tool for natural language processing (NLP) applications. The OpenAI API provides a simple interface for developers to create an intelligence layer in their applications, powered by OpenAI's state of the art models. The Chat Completions endpoint powers ChatGPT and provides a simple way to take text as input and use a model like GPT-4 to generate an output.

**Account setup**

First, create an OpenAI account or sign in. Next, navigate to the API key page and "Create new secret key", optionally naming the key. Make sure to save this somewhere safe and do not share it with anyone.

URL used for the Application

<https://api.openai.com/v1/chat/completions>

**Request Parameters**

 {

                model: "gpt-3.5-turbo",

                max\_tokens: 1024,

                messages:  [ { role: "system", content: "You are a customer support chatbot for WiseWallet." },{ role: "assistant", content: 'Hi, how can I help you today?' },]

,

                n: 1,

                stop: null,

                temperature: 0.7,

            }

**model:** (string)

* This specifies the specific language model you want to use for generating text. In this case, "gpt-3.5-turbo" refers to a fine-tuned version of GPT-3 known for its speed and improved performance. OpenAI offers different models with varying capabilities, so you can choose the one that best suits your needs.

**max\_tokens:** (integer)

* This parameter sets the maximum number of tokens (words or subwords) the model will generate in its response. Here, 1024 tells the model to keep the response concise and within 1024 tokens. Adjusting this value can control the length and detail of the generated text.

**messages:** (list of objects)

* This list defines the conversation history that the model should consider when generating its response. Each object in the list represents a single message:
  + role: (string) This indicates whether the message is from the "system" (chatbot) or the "assistant" (user).
  + content: (string) This contains the actual text of the message.

In your example, the messages list sets the context for a customer support interaction, where the system (WiseWallet chatbot) initiates the conversation with a greeting, and the assistant (user) is yet to respond.

**n:** (integer)

* This parameter specifies the number of different response completions you want the model to generate. Here, n: 1 means you'll receive only one response from the model. You can increase n to get multiple variations, but this might incur additional costs depending on the pricing plan you're using.

**stop:** (optional, string or list of strings)

* This setting allows you to define specific phrases or tokens that should act as stop signals for the model. When the model encounters any of these stop words/phrases in its generated text, it will terminate the response. This can be helpful for controlling the direction and focus of the conversation. In your example, stop: null means no specific stop signals are defined.

**temperature:** (float, between 0 and 1)

* The temperature parameter controls the randomness or creativity of the model's response. Here, temperature: 0.7 indicates a balance between following the prompt and introducing some creative elements.
  + A lower temperature (closer to 0) makes the model's response more likely to be factual and follow the established conversation flow.
  + A higher temperature (closer to 1) increases the randomness and can lead to more surprising or imaginative responses, but it might also deviate from the intended context.