**Chatbot Software Document**

**Introduction**

The Chatbot created using ChatGPT and Python is an intelligent conversational agent powered by OpenAI's GPT (Generative Pre-trained Transformer) model. It utilizes advanced natural language processing techniques to engage in interactive and meaningful conversations with users. This cutting-edge technology allows the chatbot to understand user inputs, process them contextually, and generate human-like responses, making it an invaluable tool for various applications.

**Key Features:**

1. **Natural Language Understanding (NLU):**

The Chatbot leverages the power of GPT, which has been pre-trained on a vast corpus of text, enabling it to grasp the nuances of human language. This ensures that it can comprehend user queries accurately and respond contextually.

1. **Real-Time Response:**

With low latency and high responsiveness, the chatbot can deliver real-time replies, providing an interactive and seamless conversational experience to users.

1. **Adaptability and Learning:**

The chatbot continually learns from user interactions, improving its responses over time. This adaptability allows it to cater to various user inputs and become more efficient in addressing user needs.

1. **Customizability:**

The Chatbot's behavior can be tailored to specific use cases and domains, making it adaptable for a wide range of applications, from customer support to knowledge sharing.

1. **Integration with Gradio Interface:**

The Chatbot is integrated with Gradio, a user-friendly interface, providing an interactive platform for users to communicate with the chatbot effortlessly.

**Use Cases:**

* Customer Support: The chatbot can assist users with common queries and provide timely support, reducing the workload on human agents.
* Information Retrieval: Users can engage with the chatbot to retrieve relevant information on various topics, such as news, facts, and more.
* Language Translation: The chatbot can facilitate language translation tasks, enabling users to communicate across linguistic barriers.
* Education and Learning: It can serve as a virtual tutor, answering questions and explaining concepts to learners.
* Personal Assistance: Users can use the chatbot for scheduling tasks, setting reminders, and other personal assistance tasks.

**Getting Started:**

To interact with the Chatbot, users simply enter their queries through the Gradio interface, and the Chatbot responds promptly with contextually appropriate answers. The Chatbot's capabilities can be extended by integrating it with different applications and platforms.

**Note:**

The Chatbot is designed to assist users effectively but may have limitations in understanding complex or ambiguous queries. Continuous improvements and enhancements are being made to provide users with a more enriching experience.

Experience the power of the Chatbot created using ChatGPT and Python, and explore a new dimension of interactive and intelligent conversations.

**Code Description**

**CustomChatGPT Function:**

The CustomChatGPT function is the core component responsible for interacting with the OpenAI GPT-3.5 model and generating responses. Below is the code snippet with comments to illustrate its implementation:

Github link: <https://github.com/Swaroopkr7/Chatbot-using-ChatGPT>

**Communication with OpenAI GPT-3.5 Model:**

The chatbot communicates with the OpenAI GPT-3.5 model using the OpenAI Python library. Here's an explanation of the steps involved:

* The CustomChatGPT function takes the user input as a parameter.
* It creates a list of messages, starting with a system message and appending the user input to it. The messages list is structured as a conversation history to provide context to the model.
* The OpenAI API key is set using openai.api\_key to authenticate API requests.
* The function then calls openai.ChatCompletion.create() to interact with the GPT-3.5 model. It passes the model name as "gpt-3.5-turbo" and the messages list as input to the model.
* The API response contains the chatbot's generated response in the form of a completion. The response is extracted from the API response using the response["choices"][0]["message"]["content"] syntax.
* The chatbot's response is appended to the messages list.
* Finally, the function returns the chatbot's response.

By following these steps, the chatbot effectively communicates with the GPT-3.5 model through the OpenAI Python library, generating contextually appropriate responses based on the user input and conversation history. The GPT-3.5 model's language capabilities and the chatbot's context management combine to deliver a powerful and interactive conversational experience for users.

**CustomChatGPT Function**

**Purpose:**

The CustomChatGPT function serves as the core component of the chatbot, responsible for processing user inputs and generating appropriate responses using the OpenAI GPT-3.5 model. Its purpose is to facilitate interactive and contextual conversations with users, making the chatbot capable of understanding natural language queries and providing meaningful replies.

**Functionality:**

Let's break down the functionality of the CustomChatGPT function step by step:

* **Authentication:** Before interacting with the GPT-3.5 model, the function sets the OpenAI API key using openai.api\_key. This step is crucial for authenticating API requests and ensuring that the chatbot can access the OpenAI GPT-3.5 model.
* **Creating Conversation History:** The function starts by creating a list of messages representing the conversation history. It begins with a system message, setting the context for the chatbot, which is that of a "financial expert specializing in real estate investment and negotiation." This system message serves as the initial context for the GPT-3.5 model.
* **User Input:** The function takes the user input as a parameter, which is the text entered by the user in the chatbot interface. This user input is appended to the messages list as the next item in the conversation history, along with the role "user."
* **Calling the GPT-3.5 Model:** Once the conversation history (messages list) is prepared, the function interacts with the GPT-3.5 model through the OpenAI API. It sends the conversation history to the model using openai.ChatCompletion.create(). The API call includes the model name ("gpt-3.5-turbo") and the messages list as the input to the model.
* **Handling the API Response:** The API response contains the GPT-3.5 model's generated completion as the chatbot's response. The function extracts this response from the API response using response["choices"][0]["message"]["content"].
* **Updating the Conversation History:** The chatbot's response is appended to the messages list as the next item in the conversation history, with the role "assistant." This updated conversation history ensures that the chatbot maintains context and provides coherent responses in subsequent interactions.
* **Returning the Chatbot's Response:** Finally, the function returns the chatbot's response, allowing it to be displayed to the user in the chatbot interface.

**Interacting with the GPT-3.5 Model:**

The CustomChatGPT function interacts with the GPT-3.5 model through the OpenAI API, which facilitates communication between the chatbot and the model. The OpenAI API key is essential for authenticating the API requests and gaining access to the GPT-3.5 model.

The conversation history (messages list) is provided to the GPT-3.5 model as a context, allowing it to understand user queries in the context of previous interactions. This context is vital for generating relevant and contextually appropriate responses from the model.

Once the GPT-3.5 model receives the conversation history, it processes the user input and generates a completion that serves as the chatbot's response. The model's language capabilities and training on vast amounts of text data enable it to generate coherent and human-like responses to user queries.

The chatbot's ability to communicate effectively with the GPT-3.5 model through the OpenAI Python library empowers it to engage in dynamic and interactive conversations with users, providing a natural and seamless conversational experience.

**Gradio Interface Implementation:**

The Gradio interface implementation in the code allows users to interact with the chatbot through a web-based interface. The Chatbot block displays the conversation history, enabling users to input queries in the Textbox. The ClearButton resets the chat. The respond function processes user input, communicates with the GPT-3.5 model, and updates the chat history. This ensures the chatbot provides real-time responses, maintaining context throughout the conversation.

**References:**

**OpenAI GPT-3.5 API**: <https://beta.openai.com/docs/api-reference/generate-chat>

**Gradio Documentation**: https://gradio.app/docs

**Conclusions:**

The chatbot created using ChatGPT and Python harnesses the power of OpenAI's GPT-3.5 model to engage in interactive and meaningful conversations with users. Through the Gradio interface, users can easily input their queries and receive real-time responses. The CustomChatGPT function communicates with the GPT-3.5 model, enabling the chatbot to understand natural language and generate contextually relevant replies. This implementation provides an efficient and dynamic conversational experience, making the chatbot adaptable for various applications such as customer support, information retrieval, and language translation. With continuous improvements and user feedback, the chatbot has the potential to become an increasingly valuable conversational tool in a wide range of domains.