NAAN MUDHALVAN

PROJECT PHASE -2

CHATGPT USING PYTHON

NAME : C.J.ANUROOPA

ROLL NO : 2021105505

COMPLETE STEPS FOR DESIGNING CHATGPT USING PYTHON :

Designing a chatbot using the python requires the use of the OpenAI API and some Python code. Here are the steps:

1. Get access to the OpenAI API: Visit the OpenAI website and sign up for an API key to access the GPT-3 model. Select the appropriate subscription plan to meet the requirements.

2. Setting up the Python environment: Installing Python on the system and setting up a virtual environment using tools like ‘venv’ or ‘conda’ to isolate your project's dependencies.

3. Install the OpenAI package: Use the following command to install the OpenAI Python package: bash pip install openai

4. Importing the required libraries: In Python script, importing the necessary libraries such as openai and any additional libraries needed for the chatbot's functionality. python import openai

5. Authenticating with OpenAI API key: Set up your API key for authentication to access the GPT-3 model.we can either set it as an environment variable or directly use it within your code. python openai.api_key ='YOUR_API_KEY'

6. Defining chatbot prompt: Creating a prompt that will serve as the input for the chatbot model. To start with a simple greeting or any conversational context you want for your chatbot. python prompt = "Hello, how can I assist you today?"

7. Generate responses: using a loop to have an interactive conversation with the chatbot, where the prompt will be updated with each user input. python while True: user_input = input("User: ") prompt += '\nUser: ' + user_input response = openai.Completion.create(engine='text-davinci-003', prompt=prompt, max_tokens=50, temperature=0.7) chatbot_response = response.choices[0].text.strip().split('\n')[0] prompt += '\nChatBot: ' +chatbot_response print("ChatBot:", chatbot_response)

8. Customize the chatbot behavior: doing experiment with different parameters like max_tokens (to limit the response length), temperature (to control the randomness of the output), and the choice of GPT-3 model (e.g., davinci, curie, etc.) to influence the chatbot's responses.

9. Error handling and termination: To Handle any possible exceptions or errors that may occur during the conversation. And add a termination condition when a user says "exit" or any other keyword you choose. python if user_input.lower() == "exit": break

10. Run the script: Save your Python script and run it using a command prompt or an integrated development environment (IDE) to start interacting with your chatbot. Note: The provided code snippet is a basic example, and there are many possibilities to improve and extend the functionality
of your chatbot.