

# PROJECT TITLE

## CREATE A CHATBOT IN PYTHON

### PHASE 4: Development Part 2

Continue building the chatbot by integrating it into a web app using Flask.

### INTRODUCTION:

- ✓ Flask is commonly used to create web-based chatbot applications.
- ✓ It provides the necessary tools and libraries to build a web-based interface for chatbots, making it easier to interact with users through a web-based messaging platform.
- ✓ Flask allows developers to create routes and handle HTTP requests, which can be essential for receiving user input and providing responses from a chatbot.
- ✓ You can integrate a chatbot developed with natural language processing capabilities into a Flask-based web application to create a user-friendly interface for users to interact with the chatbot via a web browser.

## FLASK INTEGRATION:

```
from flask import Flask, render_template, request

from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
```

- ✓ This line imports necessary modules from the Flask web framework.
- ✓ Flask is used to create a web application, and
  - **Flask** is the main class for that purpose.
  - **Render\_template** is used for rendering HTML templates,
  - **request** is used for handling HTTP requests, and **jsonify** is used for returning JSON responses.
  - **AutoModelForCausalLM** and **AutoTokenizer** are commonly used for loading pre-trained language models and their corresponding tokenizers.
  - **AutoModel** is also imported, although it's not used in your provided code.
  - **import torch**: PyTorch is often used in conjunction with Hugging Face Transformers to work with neural networks and machine learning models.

```
app = Flask(__name__)
tokenizer = AutoTokenizer.from_pretrained("microsoft/DialogPT-medium")
model1 = AutoModelForCausalLM.from_pretrained("chatbot_model")

@app.route("/")
def index():
    return render_template('chatty.html')
```

- ✓ By calling **Flask(\_\_name\_\_)**, you are creating an instance of this class, which represents your web application.
- ✓ This line initializes a tokenizer using the Hugging Face Transformers library.
- ✓ **The AutoTokenizer:**
  - It is a convenient way to load a pre-trained tokenizer for a specific language model.
  - Tokenizers are used to break down text into smaller units (tokens) that can be processed by a language model.
- ✓ **AutoModelForCausalLM:**
  - This class is used to load a pre-trained language model for causal language modeling.
  - Causal language models are often used for text generation tasks where the order of tokens is important, such as chatbots and text completion.
- ✓ **@app.route("/")**: This is a decorator in Flask that associates the URL route "/" (the root URL) with the following function, **index()**.
- ✓ **def index()**: This is a Python function definition. It defines a function named index that will be called when a user accesses the root URL.
- ✓ **return render\_template('chatty.html')**: Inside the index function, it returns the result of rendering a template called 'chatty.html'.

```
@app.route("/get", methods=["GET", "POST"])
def chat():
    msg = request.form["msg"]
    input = msg
    return get_chat_response(input)
```

- ✓ **@app.route("/get", methods=["GET", "POST"]):**
  - This is a decorator in Flask.
  - It associates the function below it (**chat**) with a specific URL route. In this case, it associates the **/get** route with the **chat** function, and it specifies that this route can handle both GET and POST requests.
- ✓ **def chat():**
  - This is a Python function definition. It defines a function named **chat** that will be executed when a user accesses the **/get** route of your web application.
- ✓ **msg = request.form["msg"]:**
  - This line retrieves the value of the "msg" parameter from the form data in the request.
  - It assumes that a user has submitted a form with a field named "msg" containing their input message.
  - The value is stored in the **msg** variable.
- ✓ **input = msg:**
  - The value of **msg** is assigned to a variable named **input**.

- This step is optional and not necessary; you could use **msg** directly in the following line.
- ✓ **return get\_Chat\_response(input):**
- This line calls the **get\_Chat\_response** function with the user's input (stored in the **input** variable) as an argument.
  - The **get\_Chat\_response** function generates a response based on the user's input and returns it as an HTTP response.

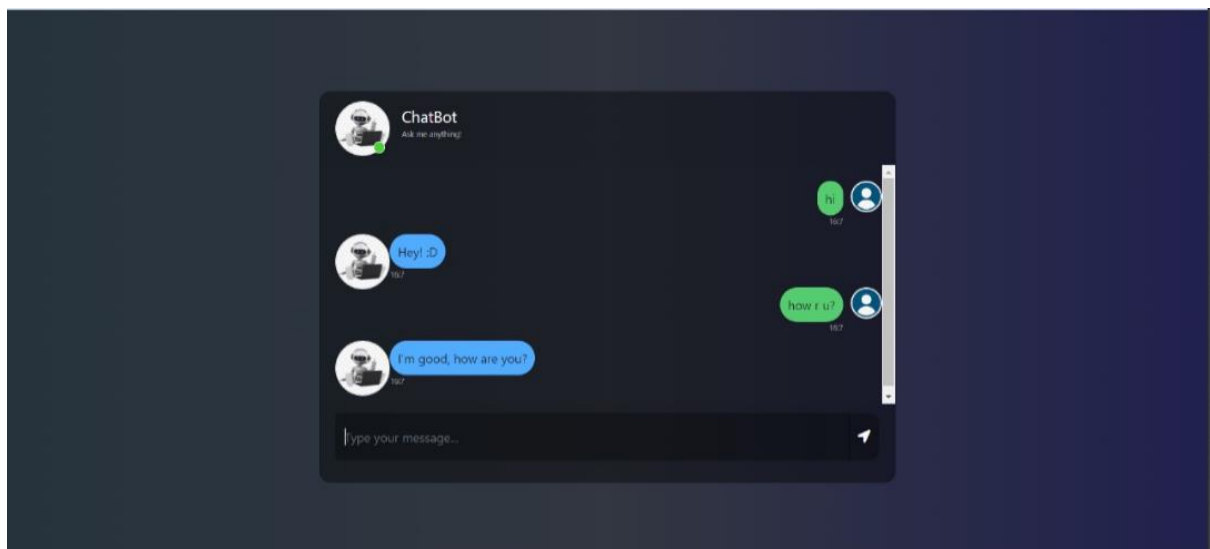
```
def get_Chat_response(text):  
    # Let's chat for 5 lines  
    for step in range(3):  
        # encode the new user input, add the eos_token and return a tensor in Pytorch  
        new_user_input_ids = tokenizer.encode(str(text) + tokenizer.eos_token, return_tensors='pt')  
  
        # append the new user input tokens to the chat history  
        bot_input_ids = torch.cat([chat_history_ids, new_user_input_ids], dim=-1) if step > 0 else new_user_input_ids  
  
        # generated a response while limiting the total chat history to 1000 tokens,  
        chat_history_ids = model.generate(bot_input_ids, max_length=1000, pad_token_id=tokenizer.eos_token_id)  
  
        # pretty print last output tokens from bot  
        return tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)  
  
if __name__ == '__main__':  
    app.run()
```

- ✓ **def get\_Chat\_response(text):** This is a Python function definition. It takes a **text** parameter, which presumably represents the user's input.
- For each iteration, it encodes the user's input, adds an end-of-sequence token (**eos\_token**), and returns a PyTorch tensor.
  - It appends the new user input tokens to the chat history.

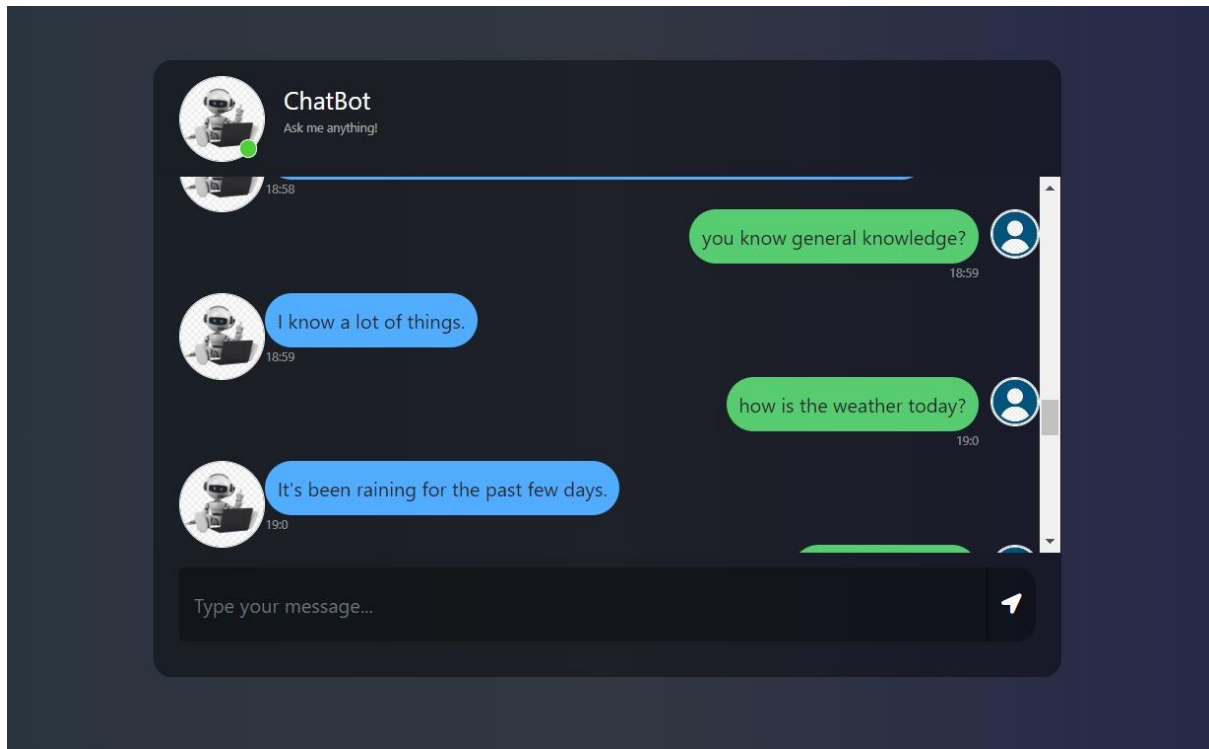
- It generates a response while limiting the total chat history to 1000 tokens.
  - Finally, it decodes the response and returns it.
- ✓ **if `__name__ == '__main__':`**
- This is a common Python idiom used to check if the script is being run as the main program.
  - If it's the main program (not imported as a module in another script), the following code block is executed.
- ✓ **app.run():** It starts the Flask development server, which allows your web application to listen for incoming HTTP requests and serve web pages.

## APPERANCE OF CHATBOT:

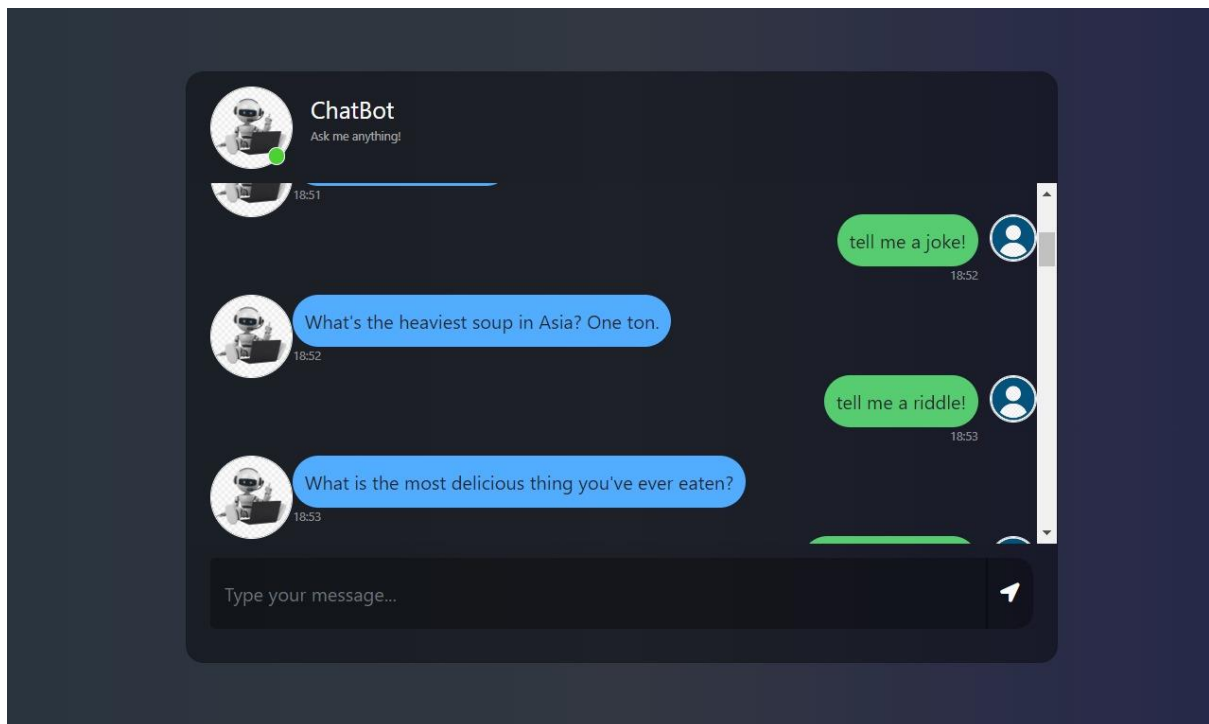
### 1.GREETINGS:



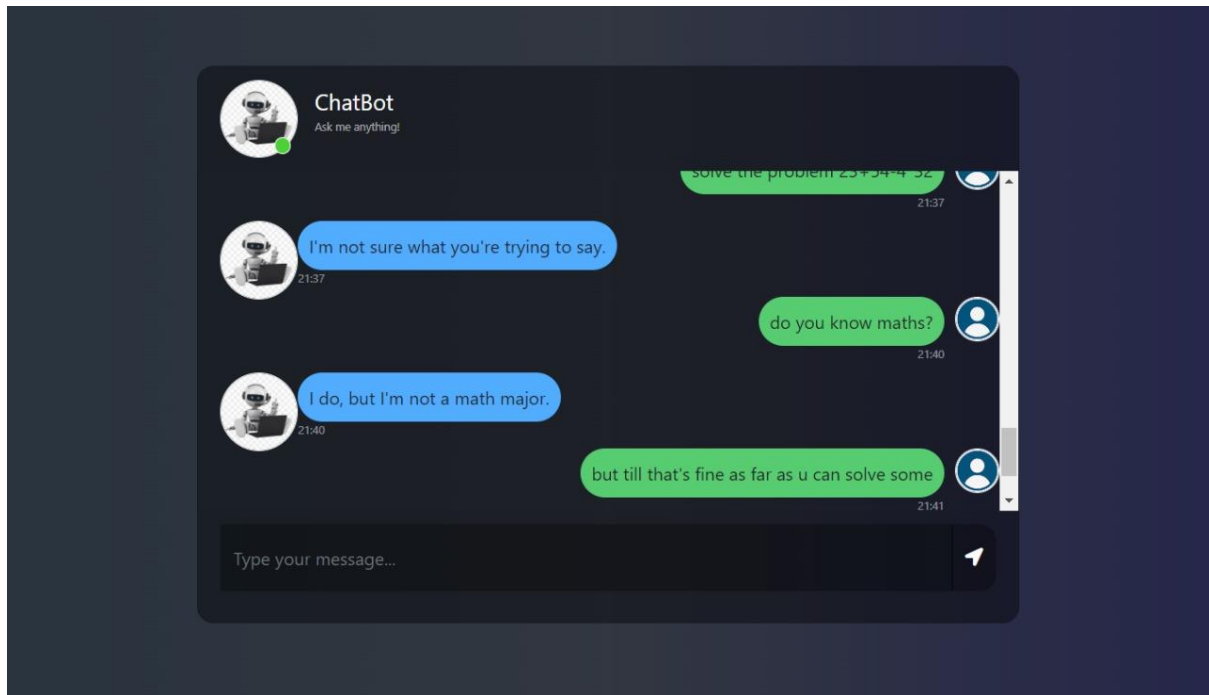
## 2. INFORMATION SHARING:



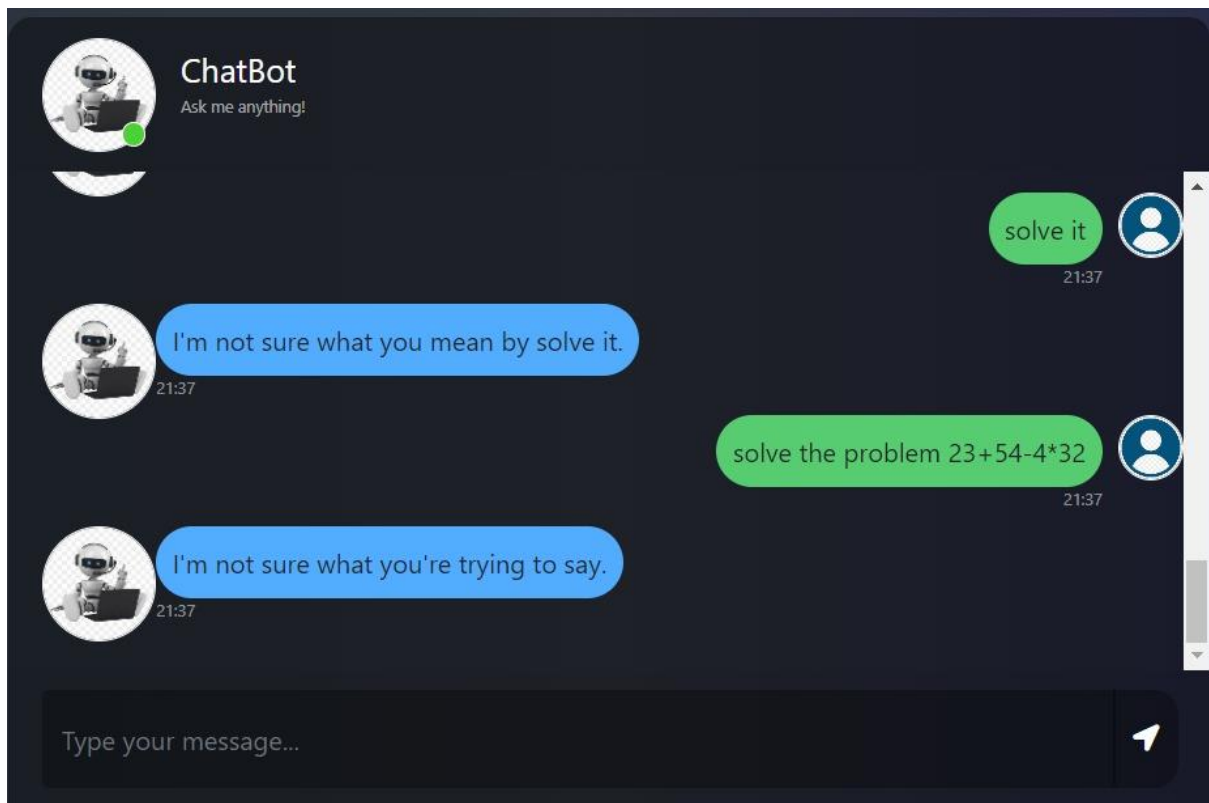
## 3. ENTERTAINMENT:



## 4.CAN ACT AS HUMAN INTELLIGENCE:

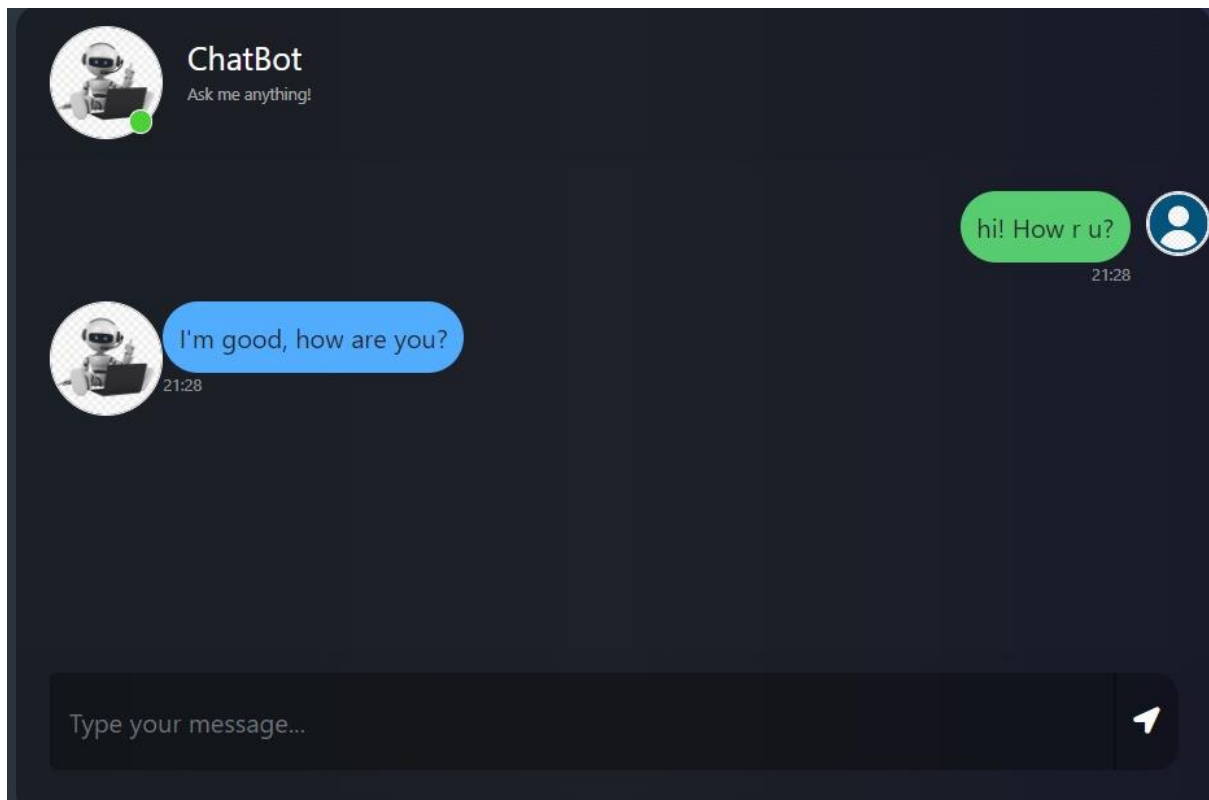


## 5.ERROR HANDLING:





## **6.INNOVATION:**



## **CONCLUSION:**

The response provided earlier is a result of the capabilities of the pre-training model, which is a sophisticated artificial intelligence language model developed by Dialo-GPT. By utilizing this pre-training model, the chatbot can analyze and interpret natural language input, generating appropriate and contextually relevant responses based on the patterns and information it has learned during its training.