CREATE A CHATBOT IN PYTHON

Phase 4: Development Part 2

In this part you will continue building your project.

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

Dataset

Link: *https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot*

Building a chatbot and integrating it into a web app using Flask is a great way to provide interactive and personalized experiences to your users. Here's a step-by-step guide on how to do this:

Set Up Your Development Environment: Make sure you have Python installed on your system. You'll also need Flask and any necessary dependencies, such as a natural language processing library like NLTK or spaCy if your chatbot needs to process text.

Create a Flask Project: Start a new Flask project by creating a directory and setting up a basic Flask application.

Create HTML Templates: Create HTML templates for your web app. You can use these templates to display the chat interface.

Implement the Chatbot Logic: You need to implement the chatbot's logic in Python. You can use libraries like NLTK or spaCy for natural language processing, or you can use pre-built chatbot frameworks like Rasa or ChatterBot. In your Flask app, import your chatbot logic and integrate it with your route.

Add User Interface Elements: In your HTML templates, create user interface elements for input and chat display.

Test and Deploy: Test your chatbot locally to ensure it's working as expected. Once it's ready, you can deploy your Flask app to a web server.

Improve and Expand: You can continue to improve your chatbot by adding more features, training it on more data, and enhancing its natural language understanding capabilities.

Remember to secure your web app and validate user inputs, as this will be important in a production environment. Additionally, you can make your chatbot more interactive by integrating it with external services, databases, or APIs to provide more dynamic responses and functionality.

In this phase, we continue developing the chatbot

1.Alter the Application for Flask:

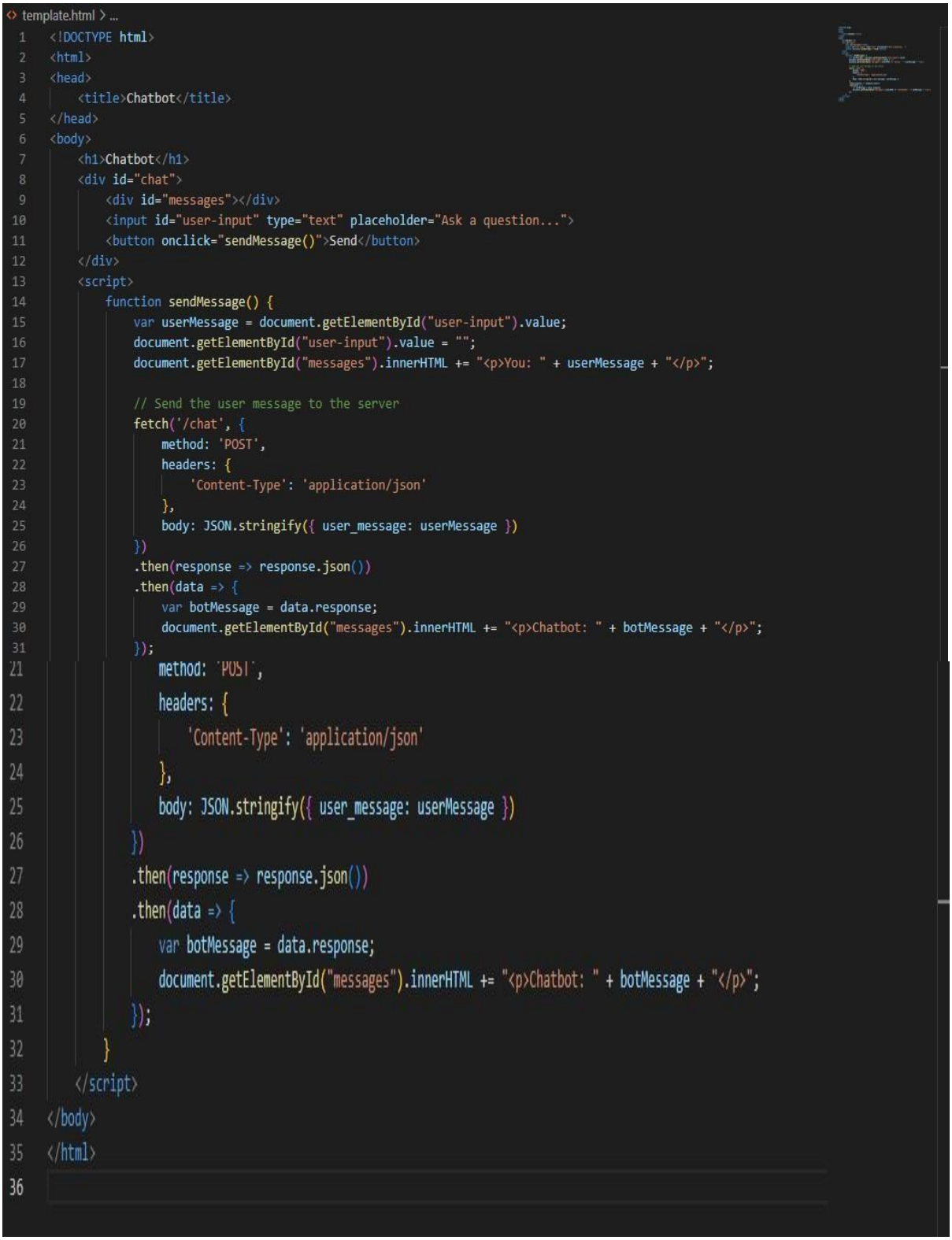
We created an application for flask in phase 3, now we run and debug the code and alter it if needed.

app.py > ...

```python
from flask import Flask, request, jsonify
from transformers import GPT3Tokenizer, GPT2LMHeadModel
import openai  # Make sure you install the 'openai' library and set your API key

app = Flask(__name__)

# Set your OpenAI GPT-3 API key
openai.api_key =' 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 '

#import dataset
dataset="dialogs.txt"

# Initialize the GPT-3 model
tokenizer = GPT3Tokenizer.from_pretrained(dataset)
model = GPT2LMHeadModel.from_pretrained(dataset)

@app.route('/chat', methods=['POST'])
def chat():
    user_message = request.json['user_message']

    # Generate a response using GPT-3
    input_ids = tokenizer.encode("User: " + user_message, return_tensors='pt')
    response = model.generate(input_ids, max_length=100, num_return_sequences=1, no_repeat_ngram_size=2, top_p=0
    chat_response = tokenizer.decode(response[0], skip_special_tokens=True)

    return jsonify({"response": chat_response})

if __name__ == '__main__':
    app.run(debug=True)
```

2. HTML Template:

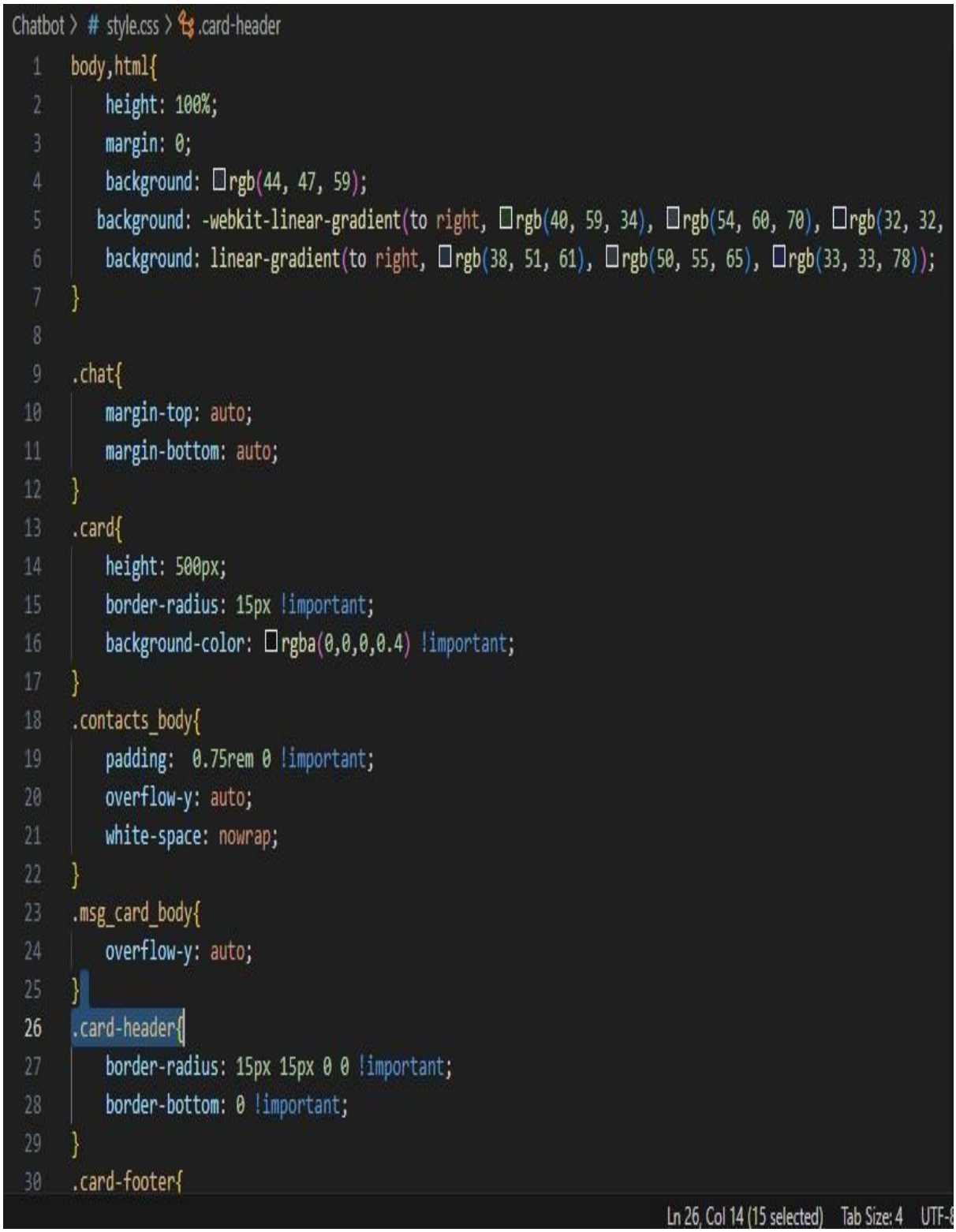
In this step, we alter the HTML template to a simple chat interface.



```
<!DOCTYPE html>
<html>
<head>
    <title>Chatbot</title>
</head>
<body>
    <h1>Chatbot</h1>
    <div id="chat">
        <div id="messages"></div>
        <input id="user-input" type="text" placeholder="Ask a question...">
        <button onclick="sendMessage()">Send</button>
    </div>
    <script>
        function sendMessage() {
            var userMessage = document.getElementById("user-input").value;
            document.getElementById("user-input").value = "";
            document.getElementById("messages").innerHTML += "<p>You: " + userMessage + "</p>";

            // Send the user message to the server
            fetch('/chat', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ user_message: userMessage })
            })
            .then(response => response.json())
            .then(data => {
                var botMessage = data.response;
                document.getElementById("messages").innerHTML += "<p>Chatbot: " + botMessage + "</p>";
            });
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ user_message: userMessage })
            })
            .then(response => response.json())
            .then(data => {
                var botMessage = data.response;
                document.getElementById("messages").innerHTML += "<p>Chatbot: " + botMessage + "</p>";
            });
        }
    </script>
</body>
</html>
```

3.Create A CSS:

Here we create a CSS style sheet for chatbot’s html interface.



```
}
.user_info{
    margin-top: auto;
    margin-bottom: auto;
    margin-left: 15px;
}
.user_info span{
    font-size: 20px;
    color: white;
}
.user_info p{
    font-size: 10px;
    color: rgba(255,255,255,0.6);
}
.video_cam{
    margin-left: 50px;
    margin-top: 5px;
}
.video_cam span{
    color: white;
    font-size: 20px;
    cursor: pointer;
    margin-right: 20px;
}
.msg_cotainer{
    margin-top: auto;
    margin-bottom: auto;
    margin-left: 10px;
    border-radius: 25px;
    background-color: rgb(82, 172, 255);
```