1. Set Up Flask Project:
    1. Create a new directory for your project.
    2. Set up a virtual environment for your Flask application.
2. Install Required Dependencies:
    1. Install Flask and any other necessary libraries.
3. Create Flask App:
    1. Create a Python file (e.g., `app.py`) to define your Flask application.
    2. Import necessary modules like Flask, render_template, and request.
4. Create HTML Templates:
    1. Create a folder (e.g., `templates`) to store your HTML templates.
    2. Create an HTML file (e.g., `index.html`) where you'll embed the chatbot interface.
5. Integrate Chatbot API:
    1. In your Flask app, integrate the ChatGPT API using HTTP requests. You'll need to make POST requests to OpenAI's API endpoint.
6. Handle User Input:
    1. Set up routes in your Flask app to handle user input from the chat interface.
    2. Extract user messages from the POST request and send them to the ChatGPT API.
7. Receive and Display Bot Responses:
    1. Retrieve the bot's responses from the API and pass them back to the chat interface.
8. Render Templates:
    1. Use the `render_template` function in Flask to render your HTML templates.
9. CSS and Styling:
    1. Apply CSS styles to your HTML templates to make the chat interface look appealing.
10. Test Your App Locally:
    1. Start your Flask app locally and test it in your web browser to ensure everything is working as expected.

Set Up a Flask Project:

Create a new directory for your project and set up a basic Flask application. If you haven't already, install Flask using `pip install Flask`.

Create HTML Templates:

Create HTML templates for your web pages. For this example, you'll need at least two templates: one for the main chat interface and another for displaying responses.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ChatGPT Web App</title>
</head>
<body>
    <div id="chatbox">
        <div id="chat"></div>
        <input type="text" id="userInput" placeholder="Type a message...">
        <button onclick="sendMessage()">Send</button>
    </div>
<script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

(for displaying bot responses):

```
<div class="message">{{ response }}</div>
```

Create Static Files:

Create a directory named `static` inside your project folder. Inside the `static` directory, create a JavaScript file `script.js`.

```
function sendMessage() {
```

```
    var userInput = document.getElementById("userInput").value;
    document.getElementById("chat").innerHTML += "<div class='message'>" + userInput + "</div>";
    document.getElementById("userInput").value = "";

    // Send the user input to the Flask backend
    fetch("/get_response", {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify({ message: userInput }),
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById("chat").innerHTML += data.response;
    });
}
```

Set Up Flask Routes

: In your Flask app, set up routes for rendering the templates and handling the chat interactions.

```
from flask import Flask, render_template, request, jsonify
import openai

app = Flask(__name__)
openai.api_key = 'YOUR_OPENAI_API_KEY'  # Replace with your OpenAI API key

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

```
@app.route('/get_response', methods=['POST'])
def get_response():
    user_message = request.json['message']
    response = openai.Completion.create(
        engine="davinci", prompt=user_message, max_tokens=50
    )
    bot_response = response.choices[0].text.strip()

    return jsonify({'response': "<div class='message bot'>" + bot_response + "</div>"})



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

Make sure to replace `'YOUR_OPENAI_API_KEY'` with your actual OpenAI API key.

Run the Flask App:

Run your Flask application by executing `python app.py` in your project directory.