Phase 4 Development :

Integrating a chatbot into a web app using Flask is a great way to provide a user-friendly interface for interacting with your chatbot. Here's a step-by-step guide on how to do it:

Step 1: Set Up Your Development Environment
Before you start, make sure you have Python and Flask installed. You can install Flask using

pip install Flask

Step 2: Create a Flask Project
Create a new directory for your Flask project and set up a basic project structure. Here's a simple example structure:

```
my_chatbot_app/
   app.py
   templates/
      index.html
```

Step 3: Create the Flask App (app.py)
In your app-py' file, you'll create a Flask web application and define routes for your chatbot.
Here's a basic

```
from flask import Flask, request, render_template
from chatbot import generate_response  # Import your chatbot logic

app = Flask(__name)

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

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

```
    user_message = request.form['user_message']
    bot_response = generate_response(user_message)  # Use your chatbot's
logic to generate a response
    return bot_response

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

In this example, we have an / route that renders an HTML template and a " /chat' route that receives user input and sends it to the chatbot logic.

Step 4: Create the HTML Template (index.html)
Create an HTML template in the templates' directory. This template will contain the chat interface where users can enter messages and receive responses. Here's a simple

```
<html>
<head>
    <title>Chatbot App</title>
</head>
<body>
    <h1>Chatbot</h1>
    <div id="chat-container">
        <div id="chat-log"></div>
        <input type="text" id="user-input" placeholder="Type your
message...">
        <button id="send-button">Send</button>
    </div>

    <script>
        document.getElementById("send-button").addEventListener("click",
function() {
            var userMessage = document.getElementById("user-input").value;
            document.getElementById("chat-log").innerHTML += '<div
class="user-message">' + userMessage + '</div>';
            document.getElementById("user-input").value = '';
```

```
            // Send the user's message to the server
            fetch('/chat', {
                method: 'POST',
                body: new URLSearchParams({ 'user_message': userMessage }),
                headers: { 'Content-Type': 'application/x-www-form-urlencoded' }
            })
            .then(response => response.text())
            .then(data => {
                document.getElementById("chat-log").innerHTML += '<div
class="bot-message">' + data + '</div>';
            });
        });
    </script>
</body>
</html>
```

This HTML template provides a basic chat interface where user messages are displayed in the chat log, and bot responses are appended as well.

Step 5: Implement Your Chatbot Logic
In the app-py file, you'll need to import your chatbot's logic, which can be implemented using a separate Python module or class. The 'generate _response ' function in the example represents your chatbot logic. Replace it with your own logic for generating chatbot
Responses.

Step 6: Run Your Flask App
Now, you can run your Flask, using the following cmd.

python app.py

Your chatbot web app should be accessible at “http://127.0.0.1/5000” in your web browser.

This is a basic example of integrating a chatbot into a Flask web app. Depending on your specific chatbot and application requirements, you can enhance the user interface and add more features.