# PHASE 4 : DEVELOPING PART 2

# CHATBOT USING IBM CLOUD WATSON ASSISTANT

**Problem Definition:** The project involves creating a chatbot using IBM Cloud Watson Assistant. The goal is to develop a virtual guide that assists users on messaging platforms like Facebook Messenger and Slack. The chatbot should provide helpful information, answer frequently asked questions (FAQs), and offer a friendly conversational experience. The project includes designing the chatbot's persona, configuring responses, integrating with messaging platforms, and ensuring a seamless user experience.

In this Phase, we will see simple chatbot program using Flask and integrate with mobile API2's

Creating a chatbot using Flask involves setting up a web server to handle incoming user requests and generating responses based on the input.

Here's a simple example of basic chatbot using Flask:

**1.Install Flask:**

- pip install Flask

**2.Create a Flask Application:**

```
# Import the necessary modules

from flask import Flask, request, jsonify

app = Flask(__name__)

# Simple dictionary-based chatbot responses

chatbot_responses = {

    "hi": "Hello! How can I assist you?",
```

```
    "how are you": "I'm just a chatbot, but thanks for asking!",
    "bye": "Goodbye! Have a great day!",
}
# Route for the chatbot
@app.route('/chatbot', methods=['POST'])
def chatbot():
    data = request.get_json()
    user_message = data['message'].lower()  # Convert the input to lowercase for case insensitivity
    if user_message in chatbot_responses:
        response = chatbot_responses[user_message]
    else:
        response = "I'm not sure how to respond to that."
    return jsonify({"message": response})
if __name__ == '__main__':
    app.run(debug=True)
```

**In this code:**

- We import the necessary modules, including Flask.
- We create a simple dictionary of chatbot responses. This can expand this dictionary with more responses as needed.
- We define a route /chatbot that listens for POST requests. This route expects a JSON payload with a message field.
- The user's message is converted to lowercase to make it case-insensitive when looking up responses in the dictionary.

- If the user's message is found in the dictionary, the corresponding response is returned. If not, a default response is given.
- The chatbot responds with the chosen message in JSON format.

**INTEGRATION PART :**

**Step 1: Set Up Your Watson Assistant:**

**1.Build Your Chatbot:**

- Access the Watson Assistant tool for your newly created service.

- Create a new skill or import an existing one that contains the dialog flow and responses for your chatbot.

- Train your chatbot with relevant data, define intents, entities, and build your conversational logic.

**2.Test Your Chatbot:**

- Test your chatbot within the Watson Assistant tool to ensure it understands and responds to user queries correctly.

**Step 2: Create a Facebook Page:**

**1.Create a Facebook Page:**

- If you don't have a Facebook Page for your chatbot, create one by visiting [https://www.facebook.com/pages/create](https://www.facebook.com/pages/create).

**Step 3: Set Up Facebook Developer Account:**

**1.Create a Facebook App:**

- Go to the Facebook for Developers website: [https://developers.facebook.com/](https://developers.facebook.com/).

- Click on "Get Started" and create a new Facebook App.

**2.Configure Your App:**

- In your Facebook App's dashboard, configure the settings:

  - Add a Facebook Page that you created in the previous step to your app.

  - Under "Products," add the "Messenger" product to your app.

  - Configure the "Messenger" product settings, including generating an App Secret and a Page Access Token.

**Step 4: Connect Facebook and Watson Assistant:**

**1.Create a Webhook:**

- In your Facebook App's dashboard under "Messenger," locate the "Webhooks" section.

- Click on "Setup Webhooks."

**2.Configure Webhook:**

- Provide your Watson Assistant webhook URL as the Callback URL. This is the endpoint where Facebook will send incoming messages.

- Enter the verification token, which you'll need to verify your webhook from your chatbot server.

**3.Subscribe to Events:**

- Subscribe to the necessary events (e.g., `messages`, `messaging_postbacks`) in the "Webhooks" settings.

**Step 5: Develop a Chatbot Server:**

**1.Create a Server:**

- Develop a server that listens for incoming POST requests from Facebook Messenger. You can use Node.js, Python, or any programming language of your choice.

- Use a web framework (e.g., Express.js for Node.js) to handle incoming requests.

**2.Handle Webhook Events:**

- Create logic to handle incoming webhook events from Facebook, including user messages.

**3.Integrate with Watson Assistant:**

- Use the Watson Assistant SDK to send user messages to your Watson Assistant service and receive responses.

**4.Send Responses to Facebook:**

- Send chatbot responses back to Facebook Messenger using the Page Access Token.

**Step 6: Test and Deploy:**

**1.Test Your Chatbot:**

- Test your chatbot by sending messages to the associated Facebook Page to ensure it works as expected.

**2.Deploy Your Server:**

- Deploy your server to a production environment that is publicly accessible over HTTPS.

**Step 7: Go Live:**

**1.Submit for Review:**

- If you want to make your chatbot publicly available to a wider audience, you may need to submit your Facebook App and chatbot for review by Facebook.

**2.Approval and Go Live:**

- Once Facebook approves your chatbot, it can go live, and users can start interacting with it on Facebook Messenger.

******