Phase-4 Submission Document

Project Title:*Create a chatbot using python*

Phase-4:Development part-2

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

CREATE A CHATBOT USING PYTHON

INTRODUCTION:

A chatbot is a computer program or application that is designed to simulate human conversation through text or voice interactions. It uses artificial intelligence (AI) and natural language processing (NLP) to understand and respond to user queries and requests. Chatbots are increasingly popular in various industries and applications, from customer support and e-commerce to healthcare and entertainment.

Key Components of a Chatbot

- **Machine Learning**: Chatbots often use machine learning algorithms to continuously improve their responses based on user interactions and feedback. This allows them to become more effective over time.
- **Knowledge Base:** Chatbots may rely on a database of information or a knowledge base to provide accurate and relevant responses. This knowledge base can be pre-defined or constantly updated.
- **Integration:** Many chatbots are integrated with external systems and APIs to perform specific tasks, such as checking the weather, making restaurant reservations, or retrieving information from a database.
- **User Authentication:** In some cases, chatbots require user authentication to access personalised information or perform certain tasks securely.

Given Data set:

hi, how are you doing?	I'm fine. How about yourself?
I'm fine. How about yourself?	I'm pretty good. Thanks for asking.
I'm pretty good. Thanks for asking.	no problem. So how have you been?
no problem. So how have you been?	I've been great. What about you?
I've been great. What about you?I've been good. I'm in school right now.
I've been good. I'm in school right now.	What school do you go to?
What school do you go to? I go to pcc.
I go to pcc.	Do you like it there?
Do you like it there?	It's okay. It's a really big campus.
It's okay. It's a really big campus.good luck with school.
So how have you been lately?
So how have you been lately?	I've actually been pretty good. you?
I've actually been pretty good. you?	I'm actually in school right now.
I'm actually in school right now.	Which school do you attend?
Which school do you attend?	I'm attending pcc right now.
I'm attending pcc right now.	Are you enjoying it there?

Overview of the process:

Integrating a chatbot into a web app using Flask is a popular use case and can enhance user engagement and provide real-time assistance. Flask is a lightweight Python web framework that is well-suited for building web applications, and you can easily incorporate a chatbot using this framework. Here's an overview of how you can do this:

Set Up Your Development Environment:

- ❖ Install Python and Flask if you haven't already.
- ❖ Choose a chatbot platform or library. You can use frameworks like Rasa, Dialog Flow, or build a custom chatbot using libraries like ChatterBot.

Create a Flask Application:

- ❖ Start by creating a new Flask project or use an existing one. You can initialise a Flask project using `flask init`.

Design the Chatbot:

- ❖ Design your chatbot's functionality, including its responses, intents, and dialogue flow. If you're using a platform like Dialog Flow, create your agent and train it to understand user input.

Integrate Chatbot Logic:

- ❖ Write Python code in your Flask application to interact with your chatbot framework or library. This code should handle incoming user messages, send them to the chatbot for processing, and retrieve and display the chatbot's responses.

Handle User Input and Chatbot Responses:

- ❖ Use JavaScript to capture user input from the web interface and send it to the Flask server via the API endpoint you created. The Flask server should then send this input to the chatbot for processing and return the chatbot's response to be displayed on the web interface.

Real-time Communication:

- ❖ To create a real-time chat experience, you can use WebSocket or long polling techniques. Libraries like Socket.IO or

Flask-SocketIO can help you achieve real-time communication between the web interface and the Flask server.

Testing and Deployment:

- Test your chatbot within the Flask web app to ensure it functions correctly. Make any necessary adjustments to improve the user experience.
- Deploy your Flask web app to a hosting platform or server. You can use services like Heroku, AWS, or a VPS to make your chatbot accessible to users.

Security and Authentication:

- Implement security measures to protect the chatbot and user data. This may include user authentication, input validation, and securing the communication between the web app and the chatbot server.

Maintenance and Improvements:

- Regularly maintain and update your chatbot and web app to enhance its functionality, fix bugs, and improve the user experience.
- By integrating a chatbot into a web app using Flask, you can provide users with a dynamic and interactive experience, whether it's for customer support, information retrieval, or any other use case. The specific implementation details will

depend on your chosen chatbot framework and the complexity of your web app.

PROCEDURE:

Building a web app using Flask to integrate the chatbot is a great idea. Flask is a lightweight Python web framework that's well-suited for such projects. Below, I'll outline the steps to create a simple web-based chatbot using Flask.

Prerequisites:

You should have Python and Flask installed. You can install Flask using pip:

```
pip install Flask
```

1. Create a Flask App:

First, create a directory for your project and a Python file (e.g., `app.py`) **for your Flask application.**

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

import chatbot_function  # This is where your chatbot logic resides

app = Flask(__name__)

@app.route('/')

def index():

    return render_template('index.html')
```

```
@app.route('/ask', methods=['POST'])

def ask():

    user_message = request.form['user_message']

    bot_response =
chatbot_function.generate_response(user_message)

    return jsonify({'bot_response': bot_response})

if __name__ == '__main__':

    app.run(debug=True)
```

2. Create HTML Templates:

Create HTML templates for your web app, especially for the chat interface. Save these templates in a folder named `templates`. For example, you can create `index.html`.

```
<!DOCTYPE html>

<html>

<head>

    <title>Chatbot Web App</title>

</head>

<body>

    <h1>Chatbot</h1>

    <div id="chat-box">

        <div id="chat-container"></div>
        <input type="text" id="user-input" placeholder="Type a
message..." onkeypress="handleKeyPress(event)">

        <button onclick="sendMessage()">Send</button>

    </div>

</body>

<script>

    function appendUserMessage(message) {

        var chatContainer =
document.getElementById('chat-container');

        chatContainer.innerHTML += '<div class="user-message">' +
message + '</div>';

    }

 function appendBotMessage(message) {

        var chatContainer =
document.getElementById('chat-container');

        chatContainer.innerHTML += '<div class="bot-message">' +
message + '</div>';

    }
```

This is a basic example of how to integrate a chatbot into a Flask web app. You can expand on this by using more advanced chatbot models, implementing natural language processing, and enhancing the user interface.

PROGRAM:

Creating a chatbot program with output involves various components and features. Below is a simplified example of a Python-based chatbot program using the ChatterBot library. Make sure to install the ChatterBot library using “`pip install chatterbot`”.

```
from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer

# Create a ChatBot instance

chatbot = ChatBot('MyBot')

# Create a new instance of the ListTrainer

trainer = ListTrainer(chatbot)

# Custom training data

custom_training_data = [

    'Hello',

    'Hi there!',

    'How are you?',
    'I am a chatbot. How can I assist you?',

    'What is your name?',

    'I am called MyBot.',

    'Goodbye',

    'Farewell!',

]

# Train the chatbot on custom data

trainer.train(custom_training_data)

# Function to get a response from the chatbot

def get_response(user_input):

    response = chatbot.get_response(user_input)

    return response

# Main loop for the chatbot

while True:

    user_input = input("You: ")
    if user_input.lower() == 'exit':
        print("Chatbot: Goodbye!")
        break
response = get_response(user_input)
    print("Chatbot:", response)
```

OUTPUT:

```
You: Hello
Chatbot: Hi there!
You: How are you?
Chatbot: I am a chatbot. How can I assist you?
You: What is your name?
Chatbot: I am called MyBot.
```

CONCLUSION

- In conclusion, the chatbot development phase has been an exciting journey, characterised by innovation and collaboration. As we transition to the next phase, which may involve testing, deployment, and further refinement, we look forward to seeing the chatbot in action, serving its intended purpose, and continuing to evolve based on user feedback and changing requirements. The dedication and hard work of the development team have brought us to this crucial juncture, and we are eager to see the chatbot's positive impact on users and the project's overall success.