PHASE 4:DEVELOPMENT PART II

TITLE: CREATE A CHATBOT USING PYTHON

PROBLEM STATEMENT: CONTINUE BUILDING THE CHATBOT BY INTEGRATING IT INTO A WEB APP USING FLASK.

INTRODUCTION:

Integrating a chatbot into a web app using Flask is a great way to create interactive and dynamic web applications. Here's a step-by-step guide to help you get started:

Step 1: Set Up Your Development Environment

Step 2: Create a Flask Project Structure

- `app.py` will contain your Flask application.
- `templates` will store your HTML templates.
- `static` is where you can store CSS, JavaScript, and other static files.
- `chatbot.py` will be used to implement your chatbot logic.

Step 3: Create the Flask Application

Step 4: Create the Chatbot Logic

Step 5: Create the HTML Template

Step 6: Add JavaScript for Real-time Interaction

Step 7: Run Your Flask Application

How to Make Chatbot in Python?

Now we are going to build the chatbot using Flask framework but first, let us see the file structure and the type of files we will be creating:

- **data.json –** The data file which has predefined patterns and responses.

- **trainning.py –** In this Python file, we wrote a script to build the model and train our chatbot.
- **Texts.pkl –** This is a pickle file in which we store the words Python object using Nltk that contains a list of our vocabulary.
- **Labels.pkl –** The classes pickle file contains the list of categories(Labels).
- model.h5 – This is the trained model that contains information about the model and has weights of the neurons.
- app.py – This is the flask Python script in which we implemented web-based GUI for our chatbot. Users can easily interact with the bot.

Here are the 5 steps to create a chatbot in Flask from scratch:

1. Import and load the data file
2. Preprocess data
3. split the data into training and test
4. Build the ANN model using keras
5. Predict the outcomes
6. Deploy the model in the Flask app

PROJECT CODE:

app.py

```
import nltk

nltk.download('popular')

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np



from keras.models import load_model
```

```
model = load_model('model.h5')

import json

import random

intents = json.loads(open('data.json').read())

words = pickle.load(open('texts.pkl','rb'))

classes = pickle.load(open('labels.pkl','rb'))



def clean_up_sentence(sentence):

    # tokenize the pattern - split words into array

    sentence_words = nltk.word_tokenize(sentence)

    # stem each word - create short form for word

    sentence_words = [lemmatizer.lemmatize(word.lower()) for
word in sentence_words]

    return sentence_words



# return bag of words array: 0 or 1 for each word in the bag
that exists in the sentence



def bow(sentence, words, show_details=True):

    # tokenize the pattern

    sentence_words = clean_up_sentence(sentence)

    # bag of words - matrix of N words, vocabulary matrix

    bag = [0]*len(words)
```

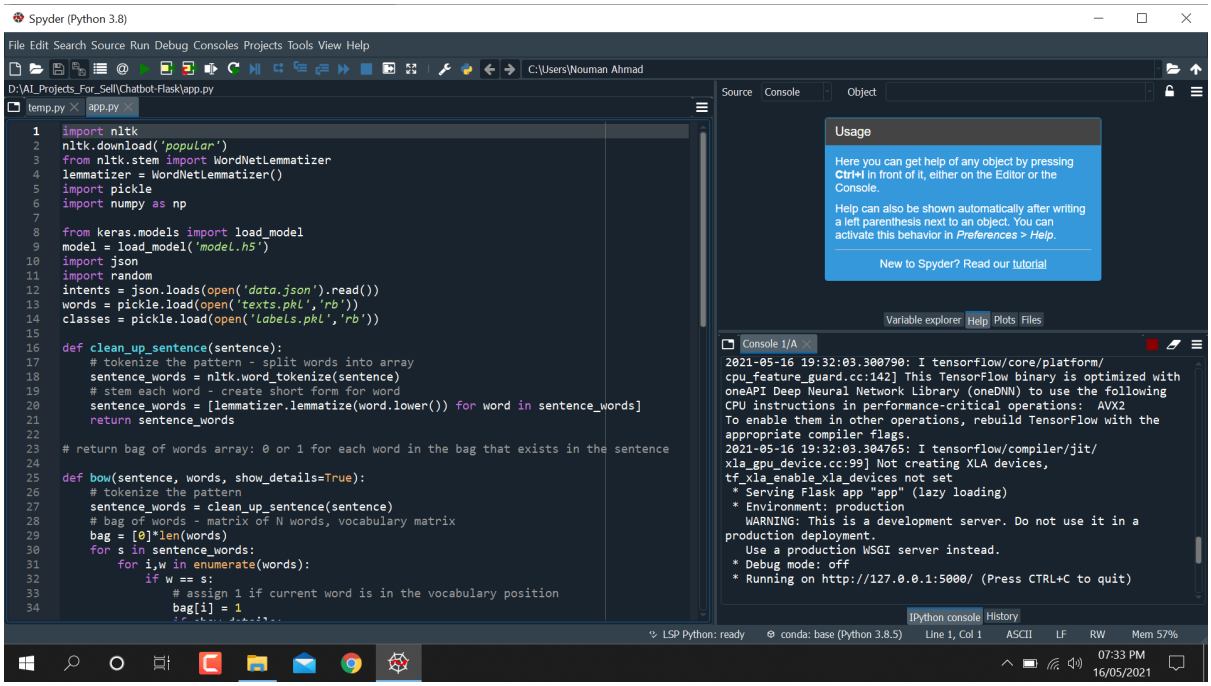
```
    for s in sentence_words:

        for i,w in enumerate(words):

            if w == s:

                # assign 1 if current word is in the
vocabulary position

                bag[i] = 1

                if show_details:

                    print ("found in bag: %s" % w)

    return(np.array(bag))



def predict_class(sentence, model):

    # filter out predictions below a threshold

    p = bow(sentence, words,show_details=False)

    res = model.predict(np.array([p]))[0]

    ERROR_THRESHOLD = 0.25

    results = [[i,r] for i,r in enumerate(res) if
r>ERROR_THRESHOLD]

    # sort by strength of probability

    results.sort(key=lambda x: x[1], reverse=True)

    return_list = []

    for r in results:

        return_list.append({"intent": classes[r[0]],
"probability": str(r[1])})
```

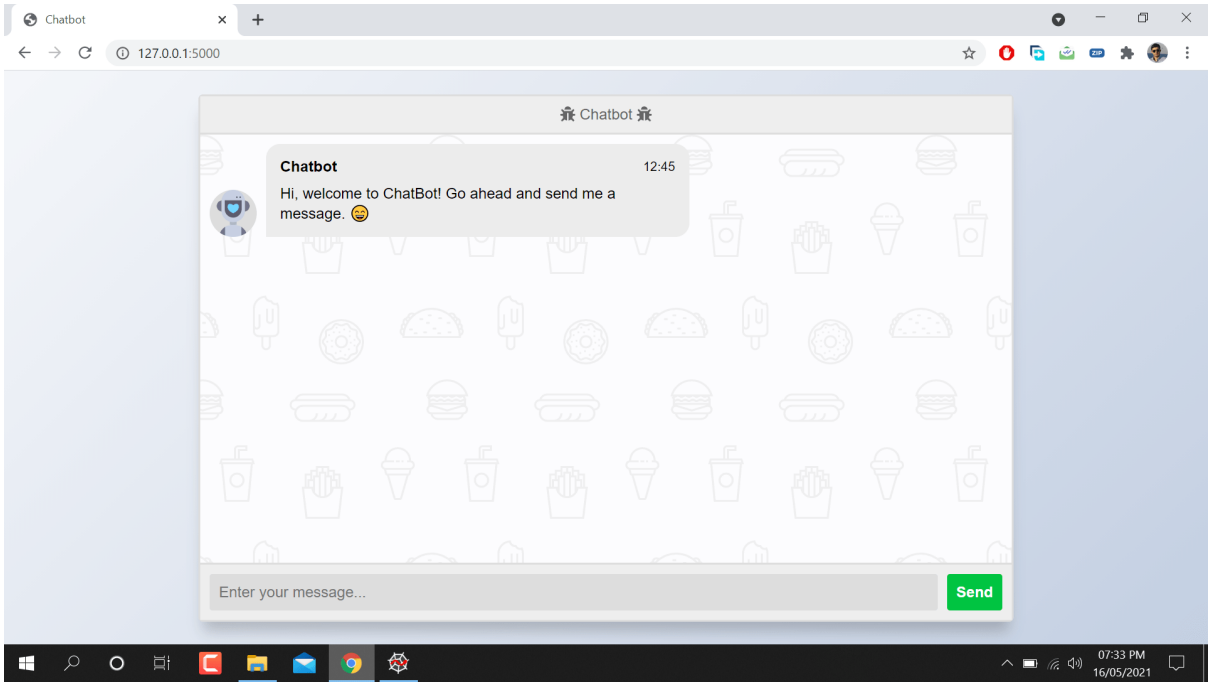
```
    return return_list


def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result


def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res




from flask import Flask, render_template, request


app = Flask(__name__)
app.static_folder = 'static'
```

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


@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)




if __name__ == "__main__":
    app.run()
```

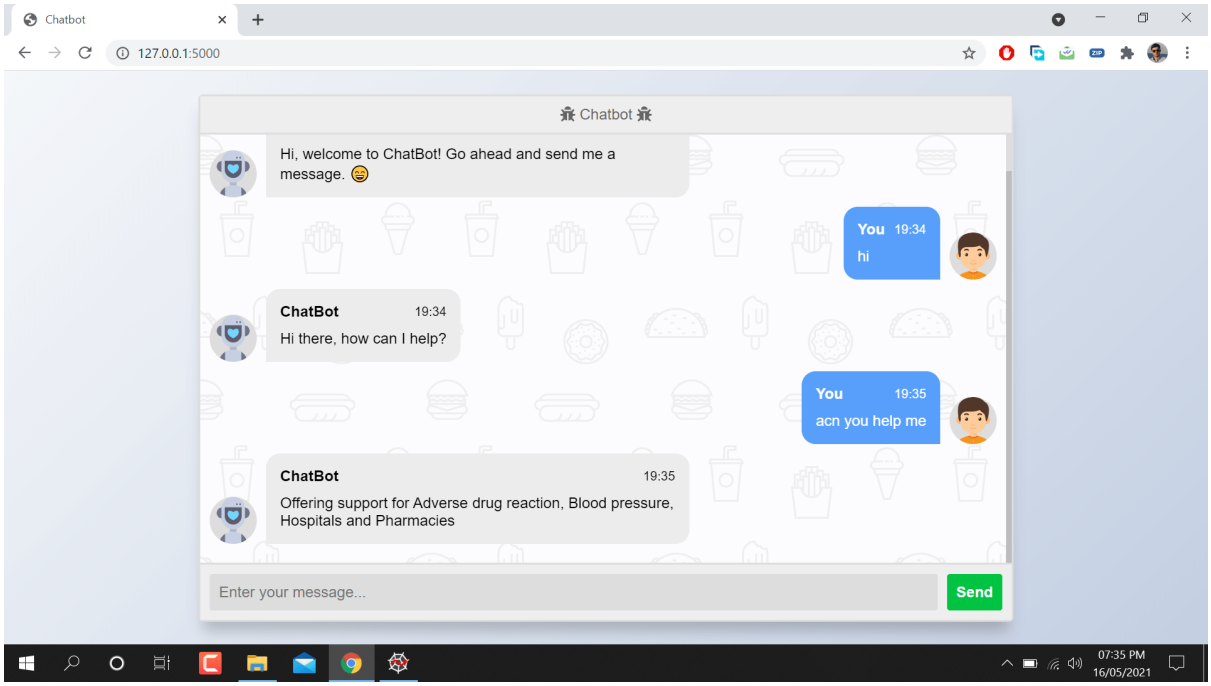
OUTPUT:

Run Flask App





BOT RESPONSE


Chatbot
127.0.0.1:5000
Chatbot
Hi, welcome to ChatBot! Go ahead and send me a message.
You 19:34
hi
ChatBot 19:34
Hi there, how can I help?
You 19:35
acn you help me
ChatBot 19:35
Offering support for Adverse drug reaction, Blood pressure, Hospitals and Pharmacies
Enter your message...
Send
07:35 PM
16/05/2021